Particle Filters for Partially-Observed Boolean Dynamical Systems

Mahdi Imani [a], Ulisses Braga-Neto [a]

[a] *Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA*

Abstract

Partially-observed Boolean dynamical systems (POBDS) are a general class of nonlinear models with application in estimation and control of Boolean processes based on noisy and incomplete measurements. The optimal minimum mean square error (MMSE) algorithms for POBDS state estimation, namely, the Boolean Kalman filter (BKF) and Boolean Kalman smoother (BKS), are intractable in the case of large systems, due to computational and memory requirements. To address this, we introduce approximate MMSE filtering and smoothing algorithms based on the auxiliary particle filter (APF) method, which are called APF-BKF and APF-BKS respectively. For joint state and parameter estimation, the APF-BKF is used jointly with maximum-likelihood (ML) methods for simultaneous state and parameter estimation in POBDS models. In the case the unknown parameters are discrete, the proposed ML adaptive filter consists of multiple APF-BKFs running in parallel, in a manner reminiscent of the Multiple Model Adaptive Estimation (MMAE) method in classical linear filtering theory. In the presence of continuous parameters, the proposed ML adaptive filter is based on an efficient particle-based expectation maximization (EM) algorithm for the POBDS model, which is based on a modified Forward Filter Backward Simulation (FFBSi) in combination with the APF-BKS. The performance of the proposed particle-based adaptive filters are assessed through numerical experiments using a POBDS model of the well-known cell cycle gene regulatory network observed through noisy RNA-Seq time series data.

Key words: Adaptive Filtering, Partially-Observed Boolean Dynamical Systems, Boolean Kalman Filter, Auxiliary Particle-Filter, Fixed-Interval Smoother, Maximum-Likelihood Estimation, Expectation Maximization, Gene Regulatory Networks, RNA-Seq data.

1 Introduction

Partially-observed Boolean dynamical systems consist of a Boolean state process, also known as a Boolean network, observed through an arbitrary noisy mapping to a measurement space [1,2,3,4]. Instances of POBDSs abound in fields such as genomics [5], robotics [6], digital communication systems [7], and more. The optimal recursive minimum mean-square error (MMSE) state estimators for this model are called the Boolean Kalman Filter (BKF) [1] and the Boolean Kalman Smoother (BKS) [3]. These filters have many desirable properties; in particular, it can be shown that the MMSE estimate of the state vector provides both the MMSE and the maximum-a-posteriori (MAP) estimates of each state vector component.

Email addresses: m.imani88@tamu.edu (Mahdi Imani), ulisses@ece.tamu.edu (Ulisses Braga-Neto).

The BKF and BKS are exact algorithms, which is unusual in the class of general partially-observable nonlinear dynamical systems, of which POBDS is a special case. However, for large systems with large number of state variables, exact computation of the BKF and BKS becomes impractical due to large computational and memory requirements. In the general case, various approximations to the optimal estimator have been developed. The classical approach is to apply linearization at each time point through a first-order Taylor series expansion, and then apply the traditional Kalman filter solution; such an approach is called the Extended Kalman Filter (EKF) [8]. The EKF cannot be applied to Boolean dynamical systems, the reason bwing that the Boolean transition functions are not differentiable. There is a class of schemes that can be applied to system functions without derivatives, collectively called derivative-less filters [9], which include the Unscented Kalman Filter (UKF) [10], the Central Difference Filter (CDF) [11], and the Divided Difference Filter (DDF) [12]. These

derivativeless filters are special cases of the general class of Sigma-Point Kalman Filters (SPKF) in [9]. However, SPKF theory has not been developed for discrete distributions such as the ones involved in Boolean dynamical systems. The only well-known approximation that can also be applied to POBDS is the class of Sequential Monte-Carlo (SMC) methods [13,14]. In [15], an approximate sequential Monte-Carlo algorithm was proposed to compute the BKF using *sequential importance resampling* (SIR). By contrast, we propose here SMC algorithms for both the BKF and fixed-interval BKS using the more efficient *auxiliary particle filter* (APF) algorithm [16], which are called the APF-BKF and APF-BKS algorithms, respectively.

The BKF and BKS require for their application that all system parameters be known. In the case where noise intensities, the network topology, or observational parameters are not known or only partially known, an adaptive scheme to simultaneously estimate the state and parameters of the system is required. An exact adaptive filtering framework to accomplish that task was proposed recently in [17], which is based on the BKF and BKS in conjunction with maximum-likelihood estimation of the parameters. In this paper, we develop an accurate and efficient particle filtering implementation of the adaptive filtering framework in [17], which is suitable for large systems.

Several exact and approximate adaptive filters for dual and joint state and parameter estimation for general Hidden Markov Models (HMM) [14,18,19] have been proposed. However, these methods are quite sensitive to initialization and do not necessarily result in maximum likelihood estimates of the parameters. Another class of methods are provided by gradient-based maximization techniques [20,21,22], which try to directly maximize the log-likelihood function using particle-based techniques. However, most of these methods suffer from unreliable approximation of the likelihood function or high computational complexity. The expectation maximization (EM) algorithm is a very popular alternative procedure for likelihood maximization, originally introduced by [23]. Applications of EM to linear/Gaussian state space models [24], general nonlinear/non-Gaussian state space models [25,26,27] and the POBDS model [17] have been proposed. Gaussian smoothing and sigma-point based approximations in an EM context have also been discussed in [28]. Particle-based implementation of EM filters have shown more numerical stability than gradient-based techniques [14]. For more information about inference methods of HMM, reader is referred to [14].

For POBDS with discrete (finite) parameter space, the adaptive filter we propose in this paper uses a bank of particle filters (APF-BKFs) in parallel, which is reminiscent of the Multiple Model Adaptive Estimation (MMAE) procedure in classical linear filtering [29]. The log-likelihood approximated by the APF-BKFs is used to obtain the ML estimate of the parameters, and the approximated MMSE estimate of the state of the selected filter yields the estimate of the state. On the other hand, if the parameter space is continuous, the available particle-based EM methods for general HMM with continuous state-space do not result in efficient estimation in POBDS models. To address this, we propose an efficient particle-based EM method for POBDS based on a modified Forward Filter Backward Simulation (FFBSi) algorithm [30]. The proposed filter yields the following advantages in comparison to the original EM method introduced in [17]:

(1) Smaller computational complexity of smoothing at the E-step.
(2) Smaller memory requirement to store the required matrices and vectors (e.g. the posterior probability vectors) from the E-Step to the M-Step.
(3) Reduced complexity of each iteration in the M-step, in which several function evaluations are required.

The application of interest in this paper is to model Boolean gene regulatory networks [5,31] observed through a single time series of RNA-seq data [32]. Using the POBDS model, we employ the proposed approximate adaptive ML algorithm to estimate the gene expression state simultaneously to the inference of the network topology and noise and expression parameters. Performance is assessed through a series of numerical experiments using the well-known cell-cycle gene regulatory model [33]. The influence of transition noise, expression parameters, and RNA-seq measurement noise (data dispersion) on performance is studied, and the consistency of the adaptive ML filter (i.e., convergence to true parameter values) is empirically established.

The article is organized as follows. In Section 2, the POBDS signal model and the Boolean Kalman Filter and Boolean Kalman Smoother are reviewed, while in Section 3, a detailed description of the APF-based filtering and smoothing algorithms proposed in this paper is provided. In Section 4, the particle-based ML adaptive filter is developed for discrete and continuous parameter spaces. A POBDS model for gene regulatory networks observed though RNA-seq measurements is reviewed in Section 5. Results for the numerical experiments with the cell-cycle network are presented in Section 6. Finally, Section 7 contains concluding remarks.

2 Optimal State Estimators for POBDS

In this section, we review the POBDS model and exact algorithms for computation of its optimal state estimators. For more details see [1,15,34,35,36]. For a proof of optimality of the BKF, see [17].

We 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. The sequence of states is observed indirectly through the *observation process* $\{\mathbf{Y}_k; k = 1, 2, \ldots\}$, where $\mathbf{Y}_k$ is a vector of (typically non-Boolean) measurements. The size of vector $\mathbf{Y}_k$ depends on the number of sensors (or outputs) of the system. The states are assumed to be updated and observed at each discrete time through the following nonlinear signal model:

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

for $k = 1, 2, \ldots$. Here, $\mathbf{n}_k \in \{0,1\}^d$ is Boolean transition noise, "$\oplus$" indicates componentwise modulo-2 addition, $\mathbf{f}_k : \{0,1\}^d \to \{0,1\}^d$ is a Boolean function, called the *network function*, whereas $\mathbf{h}_k$ is a general function mapping the current state and observation noise $\mathbf{v}_k$ (which can be discrete or continuous depending on the distribution of observation model) into the measurement space, for $k = 1, 2, \ldots$. 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 independent random variables (i.e., $\mathbf{n}_k$ and $\mathbf{n}_l$, and similarly $\mathbf{v}_k$ and $\mathbf{v}_l$, are indpendent for $k \neq l$). It is also assumed that the noise processes are independent from each other and from the initial state $\mathbf{X}_0$; their distribution is otherwise arbitrary.

We will assume further that the Boolean process noise $\mathbf{n}_k$ is **zero-mode**, i.e., $\mathbf{n}_k = \mathbf{0}$ is the most probable value of the noise vector at each time k. This implies that the most likely value of $\mathbf{X}_k$ at each time k is $\mathbf{f}_k(\mathbf{X}_{k-1})$ — this could be seen as the counterpart of the zero-mean noise assumption in continuous state-space models. As is the case with nonzero mean noise, nonzero mode noise introduces a systematic error component, which can always be removed by moving it into the function $\mathbf{f}_k$. Hence, the state model in (1) is a general model for a first-order Markov Boolean stochastic process. For a specific example, which will be adopted in Section 6, one has $P(\mathbf{n}_k(i) = 1) = p$, for $i = 1, \ldots, d$, and $k = 1, 2, \ldots$ independently for $i \neq j$. In this case, $\mathbf{n}_k$ is zero-mode if and only if $p \leq 1/2$. The systematic bias introduced in the case $p > 1/2$ can be removed by considering the equivalent state process with $\mathbf{f}'_k = \mathbf{1} - \mathbf{f}_k$, where $\mathbf{1}$ is the vector with all components equal to 1, and $p' = 1 - p$, i.e., by moving the systematic bias into the model. Therefore, one effectively needs only to consider the case $p \leq 1/2$.

A Boolean estimator $\hat{\mathbf{X}}_{k|r}$ predicts the Boolean state $\mathbf{X}_k$ based on the sequence of observations $\mathbf{Y}_{1:r} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_r)$. The estimator $\hat{\mathbf{X}}_{k|r}$ is called a *filter* or *smoother* according to whether $k = r$ or $k < r$, respectively. The set of all Boolean estimators for a given k and r shall be denoted by $\mathcal{X}_{k|r}$. The (conditional) mean-square error (MSE) of $\hat{\mathbf{X}}_{k|r}$ given $\mathbf{Y}_{1:r}$ is the expected square L_2-norm of the difference between the estimated and true states at time k:

$$\mathrm{MSE}(\hat{\mathbf{X}}_{k|r} \mid \mathbf{Y}_{1:r}) = E\left[||\hat{\mathbf{X}}_{k|r} - \mathbf{X}_k||_2^2 \mid \mathbf{Y}_{1:r}\right]. \tag{2}$$

where $||\mathbf{v}||_2^2 = \sum_{i=1}^d |\mathbf{v}(i)|^2$ (Note that using the L_1-norm $||\mathbf{v}||_1 = \sum_{i=1}^d |\mathbf{v}(i)|$ would yield the same result, since all vectors are Boolean.) We would like to obtain the *Boolean MMSE estimator*, i.e., a Boolean estimator $\hat{\mathbf{X}}^{\mathrm{MS}}_{k|r}$ such that

$$\hat{\mathbf{X}}^{\mathrm{MS}}_{k|r} = \arg \min_{\hat{\mathbf{X}}_{k|r} \in \mathcal{X}_{k|r}} \mathrm{MSE}(\hat{\mathbf{X}}_{k|r} \mid \mathbf{Y}_{1:r}), \tag{3}$$

at each value of $\mathbf{Y}_{1:r}$ (so that it also minimizes the frequentist expected MMSE over all possible realizations of $\mathbf{Y}_{1:r}$). For a vector $\mathbf{v} \in [0,1]^d$, define the *binarized vector* $\overline{\mathbf{v}}$ and the *complement vector* $\mathbf{v}^c$, via $\overline{\mathbf{v}}(i) = 1$ if $\mathbf{v}(i) > 1/2$ and $\overline{\mathbf{v}}(i) = 0$ otherwise, and $\mathbf{v}^c(i) = 1 - \mathbf{v}(i)$, for $i = 1, \ldots, d$. It has been proved [17] that the solution to (3) is given by

$$\hat{\mathbf{X}}^{\mathrm{MS}}_{k|r} = \overline{E\left[\mathbf{X}_k \mid \mathbf{Y}_{1:r}\right]}, \tag{4}$$

with optimal MMSE

$$\begin{aligned} &\mathrm{MSE}(\hat{\mathbf{X}}^{\mathrm{MS}}_{k|r} \mid \mathbf{Y}_{1:r}) \\ &\quad = \left|\left| \min\left\{E\left[\mathbf{X}_k \mid \mathbf{Y}_{1:k}\right], E\left[\mathbf{X}_k \mid \mathbf{Y}_{1:k}\right]^c\right\} \right|\right|_1, \end{aligned} \tag{5}$$

where the minimum is computed componentwise. Here, the L_1 norm is a notational device to indicate that the MMSE is equal to the sum of MMSEs of the individual vector components.

The optimal Boolean MMSE estimator will be called a *Boolean Kalman Filter* (BKF) or a *Boolean Kalman Smoother* (BKS), according to whether $k = r$ or $k < r$, respectively. The name "Kalman" for the BKF and BKS is used in agreement with the similar usage in the Extend Kalman Filter (EKF), Unscented Kalman Filter (UKF), and Sigma-Point Kalman Filter (SPKF), which are all established in the literature, even though none were designed for linear Gaussian dynamical systems. All of these algorithms share with the classical Kalman filter the property of being recursive MMSE filters for nonstationary processes. Notice that the Maximum-A-Posteriori (MAP) estimator is commonly used for state estimation in discrete state-spaces, while the MMSE is the criterion used by the BKF and BKS. Interestingly, we can show that each component of the MMSE estimator $\hat{\mathbf{X}}^{\mathrm{MS}}_{k|r}(i)$ is *both* the MMSE and the MAP estimator of the corresponding state variable $\mathbf{X}_k(i)$, for $i = 1, \ldots, d$. Perhaps surprisingly, the MAP estimator $\hat{\mathbf{X}}^{\mathrm{MAP}}_{k|r}$ does not enjoy in general the property that $\hat{\mathbf{X}}^{\mathrm{MAP}}_{k|r}(i)$ is the MAP estimator of $\mathbf{X}_k(i)$, for $i = 1, \ldots, d$. In cases where

optimal estimation performance for each component of $\hat{\mathbf{X}}_{k|r}$ is required (e.g., estimating the state of each gene in a gene regulatory network), then this could be an important distinction.

Let $(\mathbf{x}^1, \ldots, \mathbf{x}^{2^d})$ be an arbitrary enumeration of the possible state vectors, define the state conditional probability distribution vector $\mathbf{\Pi}_{k|r}$ of length 2^d via

$$\mathbf{\Pi}_{k|r}(i) \;=\; P\left(\mathbf{X}_k = \mathbf{x}^i \mid \mathbf{Y}_{1:r}\right)\;,\; i = 1, \ldots, 2^d, \quad (6)$$

and let $A \;=\; \left[\mathbf{x}^1 \cdots \mathbf{x}^{2^d}\right]$ be a matrix of size $d \times 2^d$. Then it is clear that $E\left[\mathbf{X}_k \mid \mathbf{Y}_{1:r}\right] = A\mathbf{\Pi}_{k|r}$, so it follows from (4) and (5) that

$$\hat{\mathbf{X}}^{\mathrm{MS}}_{k|r} \;=\; \overline{A\mathbf{\Pi}_{k|r}}\,, \quad (7)$$

with optimal MSE

$$\mathrm{MSE}(\hat{\mathbf{X}}^{\mathrm{MS}}_{k|r} \mid \mathbf{Y}_{1:r}) \;=\; ||\min\{A\mathbf{\Pi}_{k|r}, (A\mathbf{\Pi}_{k|r})^c\}||_1. \quad (8)$$

The distribution vector $\mathbf{\Pi}_{k|r}$ can be computed by a recursive matrix-based procedure, which is similar to the well-known "forward-backward" algorithm [37]. Briefly, let M_k of size $2^d \times 2^d$ be the transition matrix of the Markov chain defined by the state model:

$$\begin{aligned}(M_k)_{ij} \;&=\; P(\mathbf{X}_k = \mathbf{x}^i \mid \mathbf{X}_{k-1} = \mathbf{x}^j) \\ &=\; P\left(\mathbf{n}_k \;=\; \mathbf{f}(\mathbf{x}^j) \oplus \mathbf{x}^i\right),\; i, j = 1, \ldots, 2^d.\end{aligned} \quad (9)$$

Additionally, given a value of the observation vector $\mathbf{Y}_k$ at time k, let $T_k(\mathbf{Y}_k)$ be a diagonal matrix of size $2^d \times 2^d$ defined by:

$$\left(T_k(\mathbf{Y}_k)\right)_{ii} \;=\; p\left(\mathbf{Y}_k \mid \mathbf{X}_k = \mathbf{x}^i\right),\; i = 1, \ldots, 2^d\,, \quad (10)$$

where p is either a probability density or a mass function, according to the nature of the measurement $\mathbf{Y}_k$. Then it is easy to show that $\mathbf{\Pi}_{k|k} \;\propto\; T_k(\mathbf{Y}_k) M_k\, \mathbf{\Pi}_{k-1|k-1}$. This relation forms the basis for the iterative computation of $\mathbf{\Pi}_{k|r}$, and therefore of the BKF and BKS. We need to distinguish two cases. The first is a recursive implementation of the Boolean Kalman Filter (BKF), which does not need a backward iteration, and can be iterated forward as new observations arrive, for as long as desired. In this case, we use (7) and (8) with $r = k$ to get the optimal filter estimator and its minimum MSE [1,17]. The entire procedure is given in Algorithm 1.

The second case is a *fixed-interval* Boolean Kalman Smoother, where a fixed batch of observations $\mathbf{Y}_{1:T} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_T)$ of length T is available, and it is desired

Algorithm 1 BKF: Boolean Kalman Filter

1: Initialization: $(\mathbf{\Pi}_{0|0})_i \;=\; P\left(\mathbf{X}_0 = \mathbf{x}^i\right)$, for $i = 1, \ldots, 2^d$.
For $k = 1, 2, \ldots$, do:
2: Prediction: $\mathbf{\Pi}_{k|k-1} \;=\; M_k\, \mathbf{\Pi}_{k-1|k-1}$.
3: Update: $\boldsymbol{\beta}_k \;=\; T_k(\mathbf{Y}_k)\, \mathbf{\Pi}_{k|k-1}$.
4: Normalization: $\mathbf{\Pi}_{k|k} \;=\; \boldsymbol{\beta}_k / ||\boldsymbol{\beta}_k||_1$.
5: MMSE Estimator Computation: $\hat{\mathbf{X}}^{\mathrm{MS}}_{k|k} \;=\; \overline{A\mathbf{\Pi}_{k|k}}\,$.
6: $\mathrm{MSE}(\hat{\mathbf{X}}^{\mathrm{MS}}_{k|k} \mid \mathbf{Y}_{1:k}) \;=\; ||\min\{A\mathbf{\Pi}_{k|k}, (A\mathbf{\Pi}_{k|k})^c\}||_1$.

to obtain estimates of the state at all points in the interval $k = 1, \ldots, T$. In this case, a backward iteration will be needed (unless $k = T$). Define the probability distribution vector $\mathbf{\Delta}_{k|s}$ of length 2^d via

$$\mathbf{\Delta}_{k|s}(i) \;=\; p\left(\mathbf{Y}_{s+1}, \ldots, \mathbf{Y}_T \mid \mathbf{X}_k = \mathbf{x}^i\right)\;,\; i = 1, \ldots, 2^d, \quad (11)$$

for $s = 0, \ldots, T$, where $\mathbf{\Delta}_{T|T}$ is defined to be $\mathbf{1}_{d\times 1}$, the vector with all components equal to 1. It can be shown that

$$\mathbf{\Pi}_{k|T} \;=\; \frac{\mathbf{\Pi}_{k|k-1} \;\circ\; \mathbf{\Delta}_{k|k-1}}{||\mathbf{\Pi}_{k|k-1} \;\circ\; \mathbf{\Delta}_{k|k-1}||_1}\,, \quad (12)$$

where "$\circ$" denotes the "Hadamard" product. We then use (7) and (8) with $r = T$ to get the optimal smoothed estimator and its minimum MSE [3]. The entire procedure is given in Algorithm 2.

Algorithm 2 BKS: Fixed-Interval Boolean Kalman Smoother

1: Initialization: $(\mathbf{\Pi}_{0|0})_i \;=\; P\left(\mathbf{X}_0 = \mathbf{x}^i\right)$, for $i = 1, \ldots, 2^d$.
Forward Probabilities: For $s = 1, \ldots, T$, do:
2: Prediction: $\mathbf{\Pi}_{s|s-1} \;=\; M_s\, \mathbf{\Pi}_{s-1|s-1}$.
3: Update: $\boldsymbol{\beta}_s \;=\; T_s(\mathbf{Y}_s)\, \mathbf{\Pi}_{s|s-1}$.
4: Normalization: $\mathbf{\Pi}_{s|s} \;=\; \boldsymbol{\beta}_s / ||\boldsymbol{\beta}_s||_1$.
Backward Probabilities: For $s = T, T-1, \ldots, 1$, do:
5: Update: $\mathbf{\Delta}_{s|s-1} \;=\; T_s(\mathbf{Y}_s)\, \mathbf{\Delta}_{s|s}$ (with $\mathbf{\Delta}_{T|T} = \mathbf{1}$).
6: Prediction: $\mathbf{\Delta}_{s-1|s-1} \;=\; {M_s}^T\, \mathbf{\Delta}_{s|s-1}$.
MMSE Estimator Computation: For $k = 1, \ldots, T$, do:
7: $\mathbf{\Pi}_{k|T} \;=\; (\mathbf{\Pi}_{k|k-1} \circ \mathbf{\Delta}_{k|k-1}) / ||\mathbf{\Pi}_{k|k-1} \circ \mathbf{\Delta}_{k|k-1}||_1$.
8: $\hat{\mathbf{X}}^{\mathrm{MS}}_{k|T} \;=\; \overline{A\mathbf{\Pi}_{k|T}}\,$.
9: $\mathrm{MSE}(\hat{\mathbf{X}}^{\mathrm{MS}}_{k|T} \mid \mathbf{Y}_{1:T}) \;=\; ||\min\{A\mathbf{\Pi}_{k|T}, (A\mathbf{\Pi}_{k|T})^c\}||_1$.

3 Particle Filters for State Estimation

When the number of states is large, the exact computation of the BKF and the BKS becomes intractable, due to the large size of the matrices involved, which each contain 2^{2d} elements, and approximate methods must be used, such as sequential Monte-Carlo methods, also

known as *particle filter algorithms*. In the next subsections, we describe particle filter implementations of the BKF and BKS.

3.1 Auxiliary Particle Filter Implementation of the BKF (APF-BKF)

The basic algorithm to perform particle filtering is called *sequential importance resampling* (SIR). Importance sampling is used when direct sampling of the target distribution is difficult. The idea is to approximate the target distribution $p(\mathbf{x})$ using sample points ("particles") $\{\mathbf{x}_i\}_{i=1}^N$ drawn from a proposal distribution $q(\mathbf{x})$, which is easier to sample than the target distribution. The discrepancy created by sampling from $q(\mathbf{x})$ instead of $p(\mathbf{x})$ is compensated by weighting each particle. After a few iterations of the algorithm, a condition in usually reached where only few of the particles have significant weights, whereas most particles have negligible weight. To address this degeneracy problem, SIR performs resampling of the particles, whereby a fresh set of particles is drawn (with replacement) from the approximate current posterior distribution.

The original particle filtering implementation of the BKF in [15] was based on the SIR algorithm; we therefore call it the SIR-BKF algorithm. We present here a more sophisticated implementation based on the *Auxiliary Particle Filter* (APF) of [16]. The APF algorithm can be seen as a variation of SIR, and is thus also known as auxiliary SIR (ASIR). Basically, APF is a look-ahead method that at time step $k-1$ tries to predict the location of particles with high probability at time k, with the purpose of making the subsequent resampling step more efficient. Without the look-ahead, the basic SIR algorithm blindly propagates all particles, even those in low probability regions. As put in [38], "it is natural to ask whether it is possible to employ knowledge about the next observation *before* resampling to ensure that particles which are likely to be compatible with that observation have a good chance of surviving."

The APF algorithm augments the state vector to $(\mathbf{X}_k, \zeta_k)$, where ζ_k is an auxiliary variable. Particles are drawn from the filtering distribution $P(\mathbf{X}_k, \zeta_k \mid \mathbf{Y}_{1:k})$ (to be specified below), and the auxiliary variable is simply dropped to obtain particles from $P(\mathbf{X}_k \mid \mathbf{Y}_{1:k})$. Given particles $\{\mathbf{x}_{k-1,i}\}_{i=1}^N$ at time $k-1$, with associated weights $\{W_{k-1,i}\}_{i=1}^N$, the APF algorithm defines

$$\begin{aligned} &P(\mathbf{X}_k, \zeta_k \,|\, \mathbf{Y}_{1:k}) \\ &\quad \propto p(\mathbf{Y}_k \,|\, \mathbf{X}_k)\, P(\mathbf{X}_k \,|\, \mathbf{X}_{k-1}, \zeta_k)\, P(\mathbf{X}_{k-1}, \zeta_k \mid \mathbf{Y}_{1:k-1}) \\ &\quad \propto p(\mathbf{Y}_k \,|\, \mathbf{X}_k)\, P(\mathbf{X}_k \,|\, \mathbf{x}_{k-1,\zeta_k})\, W_{k-1,\zeta_k}\,, \end{aligned} \tag{13}$$

for $\zeta_k = 1, \ldots, N$. The auxiliary variable functions thus as an index for the particles at the previous time point. As will be seen below, sampling from (13) will have the effect of "selecting" the particles that are compatible with the observation at time k.

One can sample from (13) by using SIR on the following approximation:

$$P(\mathbf{X}_k, \zeta_k \,|\, \mathbf{Y}_{1:k}) \;\propto\; p(\mathbf{Y}_k \,|\, \mu_{k,\zeta_k})\, P(\mathbf{X}_k \,|\, \mathbf{x}_{k-1,\zeta_k})\, W_{k-1,\zeta_k}\,, \tag{14}$$

for $\zeta_k = 1, \ldots, N$, where $\mu_{k,i}$ is a characteristic of $\mathbf{X}_k$ given $\mathbf{x}_{k-1,i}$, which can be the mean, the mode or even a sample from $P(\mathbf{X}_k \mid \mathbf{x}_{k,i})$ [16]. In our implementation, we use the mode:

$$\begin{aligned} \mu_{k,i} \;&=\; \text{Mode}[\mathbf{X}_k \mid \mathbf{x}_{k-1,i}] \\ &=\; \text{Mode}[\mathbf{f}(\mathbf{x}_{k-1,i}) \oplus \mathbf{n}_{k,i}] \;=\; \mathbf{f}(\mathbf{x}_{k-1,i})\,, \end{aligned} \tag{15}$$

for $i = 1, \ldots, N$, where we used (1) and the fact that the noise is zero-mode. Notice that the subscript i in $\mathbf{n}_{k,i}$ denotes that the process noise can be different for the various particles.

Sampling from (13) is done in two steps. In the first step, $\{\mu_{k,i}\}_{i=1}^N$ is obtained from the particles $\{\mathbf{x}_{k-1,i}\}_{i=1}^N$ using (15) and the *first-stage* weights $\{V_{k,i}\}_{i=1}^N$ are computed as:

$$V_{k,i} \;=\; p(\mathbf{Y}_k \mid \mu_{k,i}) W_{k-1,i}\,, \tag{16}$$

for $i = 1, \ldots, N$. In the second step, the auxiliary variables $\{\zeta_{k,i}\}_{i=1}^N$ (i.e., the indices of the selected particles) are obtained as a sample from the discrete distribution defined by $\{V_{k,i}\}_{i=1}^N$ (after proper normalization). For example, if $N = 4$ and $V_{k,1} = V_{k,2}$, $V_{k,3} = V_{k,4}$, and $V_{k,1} = 2V_{k,3}$, then the indices $\zeta_{k,0}, \ldots, \zeta_{k,4}$ will be independent and each will be twice as likely to be 1 or 2 than 3 or 4. We denote this by $\{\zeta_{k,i}\}_{i=1}^N \sim \text{Cat}(\{V_{k,i}\}_{i=1}^N)$, where "Cat" stands for the categorical (discrete) distribution.

Finally, the new particles $\{\mathbf{x}_{k,i}\}_{i=1}^N$ and associated *second-stage* weights $\{\tilde{W}_{k,i}\}_{i=1}^N$ can be obtained as follows:

$$\mathbf{x}_{k,i} \;=\; \mu_{k,\zeta_{k,i}} \oplus \mathbf{n}_{k,i} \;\sim\; P(\mathbf{X}_k \mid \mathbf{x}_{k-1,\zeta_{k,i}})\,, \tag{17}$$

$$\tilde{W}_{k,i} \;=\; \frac{p(\mathbf{Y}_k \mid \mathbf{x}_{k,i})}{p(\mathbf{Y}_k \mid \mu_{k,\zeta_{k,i}})}\,. \tag{18}$$

It can be shown that the unbiased estimator of the unnormalized posterior probability at each time step can be obtained by [39]

$$||\hat{\boldsymbol{\beta}}_k||_1 \;=\; \left(\frac{1}{N}\sum_{i=1}^N V_{k,i}\right)\left(\frac{1}{N}\sum_{i=1}^N \tilde{W}_{k,i}\right)\,. \tag{19}$$

This quantity will be needed in Section 4 when the particle filter for maximum-likelihood adaptive estimation is discussed.

Given the normalized second-stage weights $W_{k,i} = \tilde{W}_{k,i} / \sum_{j=1}^{N} \tilde{W}_{k,j}$, $i = 1, \ldots, N$, one can write

$$E[\mathbf{X}_k \mid \mathbf{Y}_{1:k}] \approx \mathbf{z}_k = \sum_{i=1}^{N} W_{k,i}\, \mathbf{x}_{k,i}\,. \tag{20}$$

From (4) and (5), it follows that the MMSE state estimate and conditional MSE at time step k are approximated as:

$$\hat{\mathbf{X}}_{k|k}^{\mathrm{MS}} = \overline{\mathbf{z}_k}\,, \tag{21}$$

with optimal MMSE

$$\mathrm{MSE}(\hat{\mathbf{X}}_{k|k}^{\mathrm{MS}} \mid \mathbf{Y}_{1:k}) = ||\min\{\mathbf{z}_k, \mathbf{z}_k^c\}||_1. \tag{22}$$

The entire procedure of APF-BKF is summarized in Algorithm 3. Notice that the computational complexity of this algorithm at each time point is $O(N)$ which can be much smaller than $O(2^{2d})$ that is the complexity of the BKF.

Algorithm 3 APF-BKF: Auxiliary Particle Filter implementation of the Boolean Kalman Filter

1: $\mathbf{x}_{0,i} \sim \mathbf{\Pi}_{0|0}, W_{0,i} = 1/N$, for $i = 1, \ldots, N$.
2: **for** $k = 1, 2, \ldots,$ **do**
3: **for** $i = 1$ to N **do**
4: $\mu_{k,i} = \mathbf{f}(\mathbf{x}_{k-1,i})$.
5: $V_{k,i} = p(\mathbf{Y}_k \mid \mu_{k,i})\, W_{k-1,i}$.
6: **end for**
7: $\{\zeta_{k,i}\}_{i=1}^{N} \sim \mathrm{Cat}(\{V_{k,i}\}_{i=1}^{N})$.
8: **for** $i = 1$ to N **do**
9: $\mathbf{x}_{k,i} = \mu_{k,\zeta_{k,i}} \oplus \mathbf{n}_{k,i}$.
10: $\tilde{W}_{k,i} = \frac{p(\mathbf{Y}_k|\mathbf{x}_{k,i})}{p(\mathbf{Y}_k|\mu_{k,\zeta_{k,i}})}$.
11: **end for**
12: $\|\hat{\boldsymbol{\beta}}_k\|_1 = \left(\frac{1}{N}\sum_{i=1}^{N} V_{k,i}\right)\left(\frac{1}{N}\sum_{i=1}^{N} \tilde{W}_{k,i}\right)$.
13: $W_{k,i} = \tilde{W}_{k,i} / \sum_{j=1}^{N} \tilde{W}_{k,j}$, $i = 1, \ldots, N$.
14: $\mathbf{z}_k = \sum_{i=1}^{N} W_{k,i}\, \mathbf{x}_{k,i}$.
15: $\hat{\mathbf{X}}_{k|k}^{\mathrm{MS}} = \overline{\mathbf{z}_k}$.
16: $\mathrm{MSE}(\hat{\mathbf{X}}_{k|k}^{\mathrm{MS}} \mid \mathbf{Y}_{1:k}) = ||\min\{\mathbf{z}_k, \mathbf{z}_k^c\}||_1$.
17: **end for**

3.2 Auxiliary Particle Filter Implementation of the BKS (APF-BKS)

There are a few different approximate Monte-Carlo smoothing methods in the literature of nonlinear and non-Gaussian systems [40,41,42,30]. It should be noted that some of these particle smoother methods suffer from degeneracy problems or can only be applied in a few special conditions (such as MC with good forgetting properties). We follow an approach similar to the well-known fixed-interval smoother of [43] to approximate the Boolean Kalman Smoother.

As described in Section 2, a fixed-interval smoother is a forward-backward method, such that the filtering distributions $\mathbf{\Pi}_{k|k}$, for $k = 0, 1, \ldots, T$, are computed in the forward step, and the smoothed distributions $\mathbf{\Pi}_{k|T}$ are found in a backward step. The forward process is obtained here by running the APF-BKF algorithm of Section 3.1, while the backward process is performed by correcting the filtering weights in the backward iteration. We explain next how the backward step is applied efficiently.

First, assume $\{\mathbf{x}_{k,i}, W_{k,i}\}$, $k = 0, \ldots, T$ are the forward particles and weights obtained by the APF-BKF algorithm for the sequence of measurements $\mathbf{Y}_{1:T}$. Due to the finite number of states in the POBDS, one can compute unique particles and their associated weights at different time steps as:

$$\{\mathbf{x}_{k,i}^u, W_{k,i}^u\}_{i=1}^{F_k} \xleftarrow{\text{Unique}} \{\mathbf{x}_{k,i}, W_{k,i}\}_{i=1}^{N}\,,\ k = 0, \ldots, T. \tag{23}$$

where F_k is the number of unique forward particles, and $\mathbf{x}_{k,i}^u$ is the ith unique particle with aggregated weight $W_{k,i}^u$, both at time step k.

The backward process is based on the following equation:

$$\begin{aligned} &P(\mathbf{X}_s, \mathbf{X}_{s+1} \mid \mathbf{Y}_{1:T}) \\ &\quad = P(\mathbf{X}_s \mid \mathbf{X}_{s+1}, \mathbf{Y}_{1:T})\, P(\mathbf{X}_{s+1} \mid \mathbf{Y}_{1:T}) \\ &\quad = P(\mathbf{X}_s \mid \mathbf{X}_{s+1}, \mathbf{Y}_{1:s})\, P(\mathbf{X}_{s+1} \mid \mathbf{Y}_{1:T}) \\ &\quad = \frac{P(\mathbf{X}_{s+1} \mid \mathbf{X}_s) P(\mathbf{X}_s \mid \mathbf{Y}_{1:s}) P(\mathbf{X}_{s+1} \mid \mathbf{Y}_{1:T})}{P(\mathbf{X}_{s+1} \mid \mathbf{Y}_{1:s})}\,, \end{aligned} \tag{24}$$

where $s < T$ and $P(\mathbf{X}_{s+1} \mid \mathbf{Y}_{1:T})$ is the smoothed distribution at time step $s+1$. The summation over $\mathbf{X}_{s+1}$ in both sides of equation (24) results in

$$\begin{aligned} P(\mathbf{X}_s \mid \mathbf{Y}_{1:T}) &= P(\mathbf{X}_s \mid \mathbf{Y}_{1:s}) \\ &\quad \times \sum_{\mathbf{X}_{s+1}} \frac{P(\mathbf{X}_{s+1} \mid \mathbf{X}_s)\, P(\mathbf{X}_{s+1} \mid \mathbf{Y}_{1:T})}{P(\mathbf{X}_{s+1} \mid \mathbf{Y}_{1:s})}\,. \end{aligned} \tag{25}$$

As we mentioned before, the filter and smoother estimate at final time T are the same. Therefore, the smoothed weights $W_{T|T,i}$ are defined in the same way as the forward unique weights $W_{T,i}^u$, so that

$$P(\mathbf{X}_T \mid \mathbf{Y}_{1:T}) \approx \sum_{i=1}^{F_T} W_{T|T,i}\, \delta_{\mathbf{x}_{T,i}^u}\,. \tag{26}$$

Now, using equation (25), the the smoothed weights at time $s < T$ can be obtained as:

$$W_{s|T,j} \;=\; W^u_{s,j} \sum_{i=1}^{F_{s+1}} \frac{P(\mathbf{x}^u_{s+1,i} \mid \mathbf{x}^u_{s,j})\, W_{s+1|T,i}}{\sum_l^{F_s} P(\mathbf{x}^u_{s+1,i} \mid \mathbf{x}^u_{s,l})\, W^u_{s,l}}\,. \quad (27)$$

The smoothed weights are obtained by solving equation (27) in a backward fashion using the terminal condition $W_{T|T,j} = W^u_{T,j}$, $j = 1, \ldots, F_T$. The computational complexities of finding unique particles in (23) and computation of weights in (27) are from the order $O(N \log(N))$ and $O(F_s F_{s+1})$ respectively. Thus, the complexity of the developed smoother at time point s is $\max\{O(N \log(N)), O(F_s F_{s+1})\}$ which can be much smaller than $O(N^2)$ in practice.

Using the smoothed weights, we can write

$$E[\mathbf{X}_s \mid \mathbf{Y}_{1:T}] \;\approx\; \mathbf{z}_s \;=\; \sum_{i=1}^{F_s} W_{s|T,i}\, \mathbf{x}^u_{s,i}\,. \quad (28)$$

From (4) and (5), it follows that the MMSE state estimate and conditional MSE at time step k are approximated as:

$$\hat{\mathbf{X}}^{\mathrm{MS}}_{s|T} \;=\; \overline{\mathbf{z}_s}\,, \quad (29)$$

with optimal conditional MSE

$$\mathrm{MSE}(\hat{\mathbf{X}}^{\mathrm{MS}}_{s|T} \mid \mathbf{Y}_{1:T}) \;=\; ||\min\{\mathbf{z}_s, \mathbf{z}^c_s\}||_1. \quad (30)$$

The entire procedure is summarized in Algorithm 4.

Algorithm 4 APF-BKS: Auxiliary Particle Filter implementation of the fixed-interval Boolean Kalman Smoother

1: Run the APF-BKF for the sequence of measurements $\mathbf{Y}_{1:T}$ to obtain $\{\mathbf{x}_{k,i}, W_{k,i}\}_{i=1}^N$, $k = 0, \ldots, T$.
2: $\{\mathbf{x}^u_{k,i}, W^u_{k,i}\}_{i=1}^{F_k} \xleftarrow{\text{Unique}} \{\mathbf{x}_{k,i}, W_{k,i}\}_{i=1}^N, k = 0, ..., T.$
3: Set $W_{T|T,i} = W^u_{T,i}$, for $i = 1, \ldots, F_T$.
4: **for** $s = T - 1$ to 0 **do**
5: **for** $j = 1$ to F_s **do**
6: $$W_{s|T,j} \;=\; W^u_{s,j} \sum_{i=1}^{F_{s+1}} \frac{P(\mathbf{x}^u_{s+1,i} \mid \mathbf{x}^u_{s,j})\, W_{s+1|T,i}}{\sum_l^{F_s} P(\mathbf{x}^u_{s+1,i} \mid \mathbf{x}^u_{s,l})\, W^u_{s,l}}$$
7: **end for**
8: $\mathbf{z}_s \;=\; \sum_{i=1}^{F_s} W_{s|T,i}\, \mathbf{x}^u_{s,i}\,.$
9: $\hat{\mathbf{X}}^{\mathrm{MS}}_{s|T} \;=\; \overline{\mathbf{z}_s}.$
10: $\mathrm{MSE}(\hat{\mathbf{X}}^{\mathrm{MS}}_{s|T} \mid \mathbf{Y}_{1:T}) \;=\; ||\min\{\mathbf{z}_s, \mathbf{z}^c_s\}||_1.$
11: **end for**

This particle smoother is an efficient method for state estimation, as will be shown in Section 6, but it is not appropriate for parameter estimation, as we will argue in the next section. A different particle smoother will be used in the next section to perform continuous parameter estimation.

4 Particle Filters For Maximum-Likelihood Adaptive Estimation

Suppose that the nonlinear signal model in (1) is incompletely specified. For example, the deterministic functions $\mathbf{f}_k$ and $\mathbf{h}_k$ may be only partially known, or the statistics of the noise processes $\mathbf{n}_k$ and $\mathbf{v}_k$ may need to be estimated. By assuming that the missing information can be coded into a finite-dimensional vector parameter $\theta \in \Theta$, where Θ is the parameter space, we propose next particle filtering approaches for simultaneous state and parameter estimation for POBDS. For simplicity and conciseness, we consider two cases: a Boolean Kalman Filter algorithm with discrete (finite) Θ and a Boolean Kalman Smoother algorithm with $\Theta \subseteq R^m$, but the algorithms can be modified and even combined to perform other estimation tasks. Exact algorithms for such filters can be found in [17].

4.1 APF Implementation of the Discrete-Parameter ML Adaptive BKF (APF-DPMLA-BKF)

Discrete (finite) parameter spaces are common in applications of the POBDS model; e.g., the topology of gene regulatory networks, c.f. Section 5, is specified by the presence or absence of interactions (edges) between pairs of genes, so that the total number of possibilities is finite. In this case, $\Theta = \{\theta_1, \theta_2, \ldots, \theta_M\}$. Given the observations $\mathbf{Y}_{1:k} = \{\mathbf{Y}_1, \ldots, \mathbf{Y}_k\}$ up to time k, the log-likelihood function can be written as

$$\begin{aligned} L_k(\theta_i) \;&=\; \log p_{\theta_i}(\mathbf{Y}_{1:k}) \\ &=\; \log p_{\theta_i}(\mathbf{Y}_k \mid \mathbf{Y}_{1:k-1}) \;+\; \log p_{\theta_i}(\mathbf{Y}_{1:k-1}) \\ &=\; \log p_{\theta_i}(\mathbf{Y}_k \mid \mathbf{Y}_{1:k-1}) \;+\; L_{k-1}(\theta_i)\,, \end{aligned} \quad (31)$$

for $i = 1, \ldots, M$, where $\|\boldsymbol{\beta}^{\theta_i}_k\|_1 = p_{\theta_i}(\mathbf{Y}_k \mid \mathbf{Y}_{1:k-1})$ can be approximated by running the APF-BKF algorithms discussed in the previous section tuned to parameter vector θ_i.

The approximate log-likelihood is updated via

$$\hat{L}_k(\theta_i) \;=\; \hat{L}_{k-1}(\theta_i) + \log \|\hat{\boldsymbol{\beta}}^{\theta_i}_k\|_1\,, \quad i = 1, \ldots, M\,, \quad (32)$$

with $\hat{L}_0(\theta_i) = 0$, for $i = 1, \ldots, M$, and the ML estimator for both parameter and state at time k can be directly obtained by running M particle filters in parallel, each tuned to a candidate parameter θ_i, for $i = 1, \ldots, M$:

$$\hat{\theta}^{\mathrm{ML}}_k \;=\; \underset{\theta \in \{\theta_1, \ldots, \theta_M\}}{\operatorname{argmax}} \hat{L}_k(\theta)\,, \quad (33)$$

$$\hat{\mathbf{X}}^{\mathrm{ML}}_{k|k} = \hat{\mathbf{X}}^{\mathrm{MS}}_{k|k}(\hat{\theta}^{\mathrm{ML}}_k)\,, \quad (34)$$

for $k = 1, 2, \ldots$.

The computation in (32)–(34) is parallelized, on-line, and entirely recursive: as a new observation at time $k+1$

arrives, the ML estimator can be updated easily without restarting the computation from the beginning. The procedure is summarized in Fig. 1 and Algorithm 5.

Algorithm 5 APF-DPMLA-BKF: APF implementation of the discrete-parameter ML Adaptive BKF

1: $\hat{L}_0(\theta_i) = 0$, for $i = 1, \ldots, M$.
2: **for** $k = 1, 2, \ldots,$ **do**
3: Run M APF-BKFs tuned to $\theta_1, \ldots \theta_M$.
4: $\hat{L}_k(\theta_i) = \hat{L}_{k-1}(\theta_i) + \log \|\hat{\boldsymbol{\beta}}_k^{\theta_i}\|_1$, $i = 1, \ldots, M$.
5: $\hat{\theta}_k^{\mathrm{ML}} = \arg\max_{\theta_1,\ldots,\theta_M} \hat{L}_k(\theta_i)$.
6: $\hat{\mathbf{X}}_{k|k}^{\mathrm{ML}} = \hat{\mathbf{X}}_{k|k}^{\mathrm{MS}}(\hat{\theta}_k^{\mathrm{ML}})$.
7: **end for**

4.2 APF Implementation of the Continuous-Parameter ML Adaptive BKS (APF-CPMLA-BKS)

Here, $\Theta \subseteq R^m$, and the approach developed in the last subsection for discrete parameter spaces is not directly applicable. There are two options: 1) discretize the parameters using a suitable quantization grid and apply an approach similar to the one in the last section; 2) attempt to obtain a good approximation of the MLE in the continuous parameter space directly. In this section, we describe how to implement the second option using the expectation-maximization (EM) algorithm for a particle filter implementation of a fixed-interval Boolean Kalman Smoother.

In our case, maximum likelihood estimation attempts to find the value of θ that maximizes the "incomplete" log-likelihood function $L_k(\theta) = \log p_\theta(\mathbf{Y}_{1:T})$. The EM algorithm considers instead the "complete" log-likelihood function $\log p_\theta(\mathbf{X}_{0:T}, \mathbf{Y}_{1:T})$, which includes the unknown state sequence, the assumption being that maximising the complete log-likelihood is easier than maximising the incomplete one (the reader is referred to [17] for more information on the EM algorithm for POBDS).

The EM algorithm obtains a sequence of parameter estimates $\{\theta^{(n)}; n = 0, 1, \ldots\}$. Given the current estimate $\theta^{(n)}$, the algorithm obtain the next estimate $\theta^{(n+1)}$ in the sequence by computing (E-step) the function (see [17]):

$$\begin{aligned} Q(\theta, \theta^{(n)}) &= \sum_{\mathbf{X}_{0:T}} \log p_\theta(\mathbf{X}_{0:T}, \mathbf{Y}_{1:T})\, p_{\theta^{(n)}}(\mathbf{X}_{0:T} \mid \mathbf{Y}_{1:T}) \\ &= I_1(\theta, \theta^{(n)}) + I_2(\theta, \theta^{(n)}) + I_3(\theta, \theta^{(n)}), \end{aligned} \tag{35}$$

where

$$I_1(\theta, \theta^{(n)}) = \sum_{i=1}^{2^d} \log P_\theta(\mathbf{X}_0 = \mathbf{x}^i)\, P_{\theta^{(n)}}(\mathbf{X}_0 = \mathbf{x}^i \mid \mathbf{Y}_{1:T}), \tag{36}$$

$$\begin{aligned} I_2(\theta, \theta^{(n)}) = \sum_{s=1}^{T} \sum_{i=1}^{2^d} \sum_{j=1}^{2^d} &\log P_\theta(\mathbf{X}_s = \mathbf{x}^i \mid \mathbf{X}_{s-1} = \mathbf{x}^j) \\ &\times P_{\theta^{(n)}}(\mathbf{X}_s = \mathbf{x}^i, \mathbf{X}_{s-1} = \mathbf{x}^j \mid \mathbf{Y}_{1:T}), \end{aligned} \tag{37}$$

$$\begin{aligned} I_3(\theta, \theta^{(n)}) = \sum_{s=1}^{T} \sum_{i=1}^{2^d} &\log p_\theta(\mathbf{Y}_s \mid \mathbf{X}_s = \mathbf{x}^i) \\ &\times P_{\theta^{(n)}}(\mathbf{X}_s = \mathbf{x}^i \mid \mathbf{Y}_{1:T}), \end{aligned} \tag{38}$$

and then maximizing (M-step) this function:

$$\theta^{(n+1)} = \operatorname*{argmax}_\theta Q(\theta, \theta^{(n)}). \tag{39}$$

In [17] this computation is carried out exactly. For large systems, this is impractical, for the following reasons:

(1) The E-Step (computing the Q function) requires performing a Boolean Kalman Smoother, which is too expensive computationally.
(2) The transition matrix and filtered and smoothed posterior probability vectors at all time steps must be stored, demanding large amounts of memory.
(3) In certain cases, such as when $\mathbf{f}_k$ and $\mathbf{h}_k$ are linear in the parameter vector θ, it is possible to maximize $Q(\theta, \theta^{(n)})$ using closed-form expressions (e.g. [42]). However, in general, one needs to resort to gradient-based optimization methods in the M-step. This requires evaluating $Q(\theta, \theta^{(n)})$ and computing its derivatives, which is analytically intractable.

Several particle-based EM methodologies were developed in recent years (e.g. [25,26,27]). However, due to the fact that these methods were designed for HMMs with continuous state-spaces, direct application of these methods to the POBDS model results in computationally expensive algorithms.

To address the aforementioned issues, we develop our efficient EM algorithm based on the *forward filter backward simulation* (FFBSi) [30]. The FFBSi method tries to capture the most probable state trajectories and use them to find the smoothing particles. The method contains two main steps: 1) Forward Step: the APF-BKF algorithm is employed to obtain the particles and their weights from time 0 to T ($\{\mathbf{x}_{1:T,i}, W_{1:T,i}\}_{i=1}^N$). 2) Backward Step: the backward simulation procedure, which is explained in detail in the sequel, computes N trajecto-

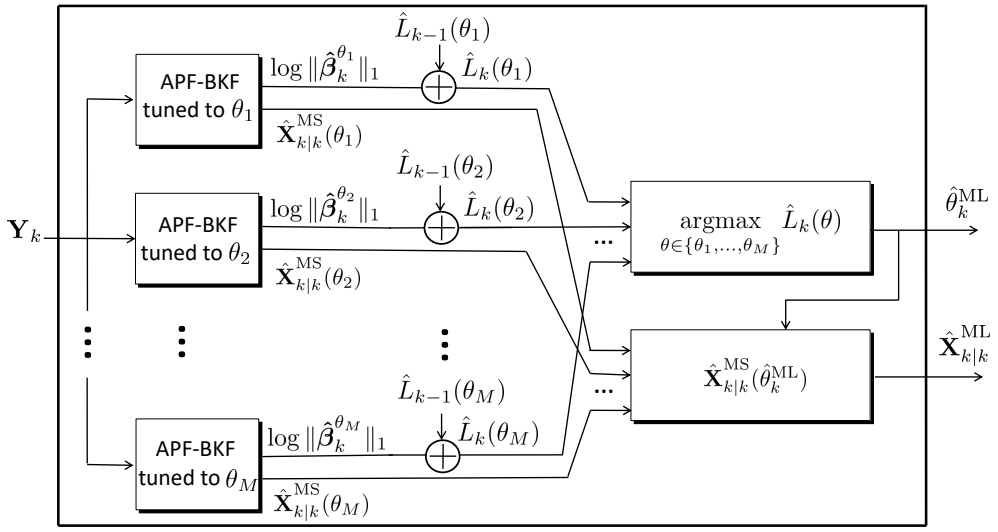

Fig. 1. Schematic diagram of particle-filter implementation of the discrete-parameter ML adaptive Boolean Kalman Filter.

ries $\{\tilde{\mathbf{x}}_{0:T,i}\}_{i=1}^N \sim P(\mathbf{X}_{0:T} \mid \mathbf{Y}_{1:T})$, where

$$\begin{aligned} & P(\mathbf{X}_{0:T} \mid \mathbf{Y}_{1:T}) \\ &= P(\mathbf{X}_T \mid \mathbf{Y}_{1:T}) \prod_{s=0}^{T-1} P(\mathbf{X}_s \mid \mathbf{X}_{s+1:T}, \mathbf{Y}_{1:T}) \\ &= P(\mathbf{X}_T \mid \mathbf{Y}_{1:T}) \prod_{s=0}^{T-1} P(\mathbf{X}_s \mid \mathbf{X}_{s+1}, \mathbf{Y}_{1:s}) \,. \end{aligned} \tag{40}$$

Based on equation (40), smoothed particles can be obtained using the FFBSi method, by means of the following backward procedure:

$$\begin{aligned} \tilde{\mathbf{x}}_{T,i} &\sim P(\mathbf{X}_T \mid \mathbf{Y}_{1:T}) \,, \\ \tilde{\mathbf{x}}_{s,i} &\sim P(\mathbf{X}_s \mid \tilde{\mathbf{x}}_{s+1,i}, \mathbf{Y}_{1:T}) \,, \end{aligned} \tag{41}$$

for $i = 1, \ldots, N$ and $s = T-1, \ldots, 0$, where $\{\tilde{\mathbf{x}}_{s,i}\}_{i=1}^N$ are the smoothed particles at time step s.

The backward process starts by resampling N particles $\{\tilde{\mathbf{x}}_{T,i}\}_{i=1}^N$ from the unique forward particles $\{\mathbf{x}_{T,i}^u\}_{i=1}^{F_T}$ at time step T using the forward weights $\{W_{T,i}^u\}_{i=1}^{F_T}$ as:

$$\begin{aligned} \{\eta_T(i)\}_{i=1}^N &\sim \mathrm{Cat}\left(\{W_{T,j}^u\}_{j=1}^{F_T}\right), \\ \tilde{\mathbf{x}}_{T,i} &= \mathbf{x}_{T,\eta_T(i)}^u, \;\; i = 1, \ldots, N \,. \end{aligned} \tag{42}$$

Now, to obtain N smoothed particles at time step $s < T$, let $\{\tilde{\mathbf{x}}_{s+1,i}\}_{i=1}^N$ be the smoothed particles at time $s+1$, and let

$$\{\tilde{\mathbf{x}}_{s+1,j}^u, \xi_{s+1}^j\}_{j=1}^{S_{s+1}} \xleftarrow{\text{Unique}} \{\tilde{\mathbf{x}}_{s+1,i}\}_{i=1}^N \tag{43}$$

where S_{s+1} specifies the number of unique smoothed particles at time $s+1$, and ξ_{s+1}^j is a vector containing the indexes of the j-th unique smoothed particles before shrinkage and reordering. Notice that $N = \sum_{j=1}^{S_{s+1}} |\xi_{s+1}^j|$. For the j-th unique smoothed particle at time step $s+1$, one can use the fact that $P(\mathbf{X}_s \mid \mathbf{X}_{s+1}, \mathbf{Y}_{1:s}) \propto P(\mathbf{X}_{s+1} \mid \mathbf{X}_s)\, P(\mathbf{X}_s \mid \mathbf{Y}_{1:s})$ to compute the following weights;

$$D_{s,i}^j = W_{s,i}^u \, P(\tilde{\mathbf{x}}_{s+1,j}^u \mid \mathbf{x}_{s,i}^u) \,, \tag{44}$$

for $i = 1, \ldots, F_s$, and draw $|\xi_{s+1}^j|$ particles as:

$$\begin{aligned} \{\eta_s(t)\}_{t=1}^{|\xi_{s+1}^j|} &\sim \mathrm{Cat}(\{D_{s,i}^j\}_{i=1}^{F_s}) \,, \\ \tilde{\mathbf{x}}_{s,\xi_{s+1}^j(t)} &= \mathbf{x}_{s,\eta_s(t)}^u, \quad \text{for } t = 1, \ldots, |\xi_{s+1}^j|. \end{aligned} \tag{45}$$

Repeating the above process for $j = 1, \ldots, S_{s+1}$ and $s = T-1, \ldots, 0$ results in N trajectories from the joint smoothed distribution $\{\tilde{\mathbf{x}}_{0:T,i}\}_{i=1}^N$. Notice that the computational complexity of finding unique particles in equation (43) is of order $O(N \log(N))$. Therefore, the overall complexity of our proposed methodology is $\max\{O(\{N \log(N)), O(S_s F_s)\}$ at time step s, which can be much smaller than the computational complexity of the optimal smoother, which is $O(N^2)$.

Given the N trajectories $\{\tilde{\mathbf{x}}_{0:T,i}\}_{i=1}^N$ obtained by running the forward filter backward simulation tuned to parameter $\theta^{(n)}$, equations (36)-(38) can be approximated

as:

$$\hat{I}_1(\theta, \theta^{(n)}) \;=\; \frac{1}{N}\sum_{i=1}^{N} \log P_\theta(\tilde{\mathbf{x}}_{0,i})\,, \tag{46}$$

$$\hat{I}_2(\theta, \theta^{(n)}) \;=\; \frac{1}{N}\sum_{s=1}^{T}\sum_{i=1}^{N} \log P_\theta(\tilde{\mathbf{x}}_{s,i} \mid \tilde{\mathbf{x}}_{s-1,i})\,, \tag{47}$$

$$\hat{I}_3(\theta, \theta^{(n)}) \;=\; \sum_{s=1}^{T}\sum_{i=1}^{N} \log p_\theta(\mathbf{Y}_s \mid \tilde{\mathbf{x}}_{s,i})\,. \tag{48}$$

Thus, one can approximate the Q function in equation (35) as:

$$\hat{Q}(\theta, \theta^{(n)}) \;=\; \frac{1}{N}\sum_{i=1}^{N}\Bigg[\log P_\theta(\tilde{\mathbf{x}}_{0,i}) + \sum_{s=1}^{T} \log P_\theta(\tilde{\mathbf{x}}_{s,i} \mid \tilde{\mathbf{x}}_{s-1,i}) + \sum_{s=1}^{T} \log\, p_\theta(\mathbf{Y}_s \mid \tilde{\mathbf{x}}_{s,i})\Bigg]. \tag{49}$$

In [26], similar equations are derived for the Hammerstein-Wiener model structure. The computational complexity of evaluating the $\hat{Q}$ function in (49) is only of order $O(N \times k)$, which results in large savings in the computation of the gradient of $\hat{Q}$ in the M-Step of the EM algorithm. In Section 5, expressions for the gradients are given in the special case where the observations consist of RNA sequencing data. Finally, as regards to memory, the only values that must be stored from the E-Step to be used in the M-Step are the N smoothed trajectories (storing filter weights or particles is not necessary). In Section 6, the effect of the value of N on performance will be discussed.

The steps of the EM adaptive filter are as follows. Initially, N smoothed trajectories are obtained using the developed FFBSi method tuned to a initial parameter guess $\theta^{(0)}$ to compute $\hat{Q}(\theta, \theta^{(0)})$ (E-Step). The aforementioned gradient-descent procedure is applied to find the best parameter $\theta^{(1)}$ that maximizes $\hat{Q}(\theta, \theta^{(0)})$ with $\theta^{(0)}$ fixed (M-Step). The obtained parameter vector is set as the parameter for the particle smoother for the next run, and the process continues until there is no significant change in parameter estimates between two consecutive steps, yielding the final parameter estimate θ^{ML}. Then the smoothed state estimates can be obtained by performing an APF-BKS tuned to parameter θ^{ML}. The procedure is summarized in Fig. 2 and Algorithm 6.

Algorithm 6 APF-CPMLA-BKS: APF implementation of the continuous-parameter ML Adaptive BKS.

Specify $\theta^{(0)}$ (initial guess) and tolerance $\varepsilon > 0$.
$n \leftarrow -1$.
repeat
  $n \leftarrow n + 1$.
  $\{\mathbf{x}_{0:T,i}, W_{0:T,i}\}_{i=1}^{N} \leftarrow$ Run APF-BKF tuned to $\theta^{(n)}$.
  $\{\mathbf{x}^u_{k,j}, W^u_{k,j}\}_{j=1}^{F_k} \xleftarrow{\text{Unique}} \{\mathbf{x}_{k,i}, W_{k,i}\}_{i=1}^{N}$, $k = 0, \ldots, T$.
  Sample $\{\eta_T(i)\}_{i=1}^{N} \sim \text{Cat}\,(\{W^u_{T,j}\}_{j=1}^{F_T})$.
  Set $\tilde{\mathbf{x}}_{T,i} = \mathbf{x}^u_{T,\eta_T(i)}$, for $i = 1, \ldots, N$.
  for $s = T - 1$ to 0 **do**
    $\{\tilde{\mathbf{x}}^u_{s+1,j}, \xi^j_{s+1}\}_{j=1}^{S_{s+1}} \xleftarrow{\text{Unique}} \{\tilde{\mathbf{x}}_{s+1,i}\}_{i=1}^{N}$.
    for $j = 1$ to S_{s+1} **do**
      $D^j_{s,i} \;=\; W^u_{s,i}\, P(\tilde{\mathbf{x}}^u_{s+1,j} \mid \mathbf{x}^u_{s,i})$, $i = 1, \ldots, F_s$.
      $\{\eta_s(t)\}_{t=1}^{|\xi^j_{s+1}|} \sim \text{Cat}\,(\{D^j_{s,i}\}_{i=1}^{F_s})$.
      $\tilde{\mathbf{x}}_{s,\xi^j_{s+1}(t)} \;=\; \mathbf{x}^u_{s,\eta_s(t)}$, for $t = 1, \ldots, |\xi^j_{s+1}|$.
    end for
  end for
  Find $\hat{Q}(\theta, \theta^{(n)})$ using equation (49).
  Find $\theta^{(n+1)} \;=\; \text{argmax}_\theta\, \hat{Q}\left(\theta, \theta^{(n)}\right)$.
until $|\theta^{(n+1)} - \theta^{(n)}| > \varepsilon$
$\hat{\theta}^{\mathrm{ML}} = \theta^{(n+1)}$.
$\hat{\mathbf{X}}^{\mathrm{MS}}_{1:T|T}(\hat{\theta}^{\mathrm{ML}}) \leftarrow$ Run APF-BKS tuned to $\hat{\theta}^{\mathrm{ML}}$.
$\hat{\mathbf{X}}^{\mathrm{ML}}_{1:T|T} = \hat{\mathbf{X}}^{\mathrm{MS}}_{1:T|T}(\hat{\theta}^{\mathrm{ML}})$.

5 Gene Regulatory Network and RNA-Seq Measurement Models

The algorithms developed in the previous section apply to the general POBDS signal model in (1). In this section, we describe a specific instance of that model, which allows the application of the methodology to Boolean gene regulatory networks observed through noisy gene-expression data. There are several gene-expression measurement technologies currently in use, such as cDNA microarrays [44] and live cell imaging-based assays [45], in which gene expression measurements are continuous and unimodal (within a single population of interest), and next-generation sequencing measurements (RNA-seq) which produce noisy integer-valued measurements of bimodal gene expression, which is the case considered in this paper.

5.1 Gene Regulatory Network Model

This model is motivated by gene pathway diagrams commonly encountered in biomedical research. The network function in (1) is assumed to be time-invariant and expressed as $\mathbf{f} = (f_1, \ldots, f_d)$, where each component $f_i : \{0,1\}^d \to \{0,1\}$ is a Boolean function given by

$$f_i(\mathbf{x}) \;=\; \begin{cases} 1, & \sum_{j=1}^{d} a_{ij}\mathbf{x}(j) + b_i > 0\,, \\ 0, & \sum_{j=1}^{d} a_{ij}\mathbf{x}(j) + b_i \le 0\,, \end{cases} \tag{50}$$

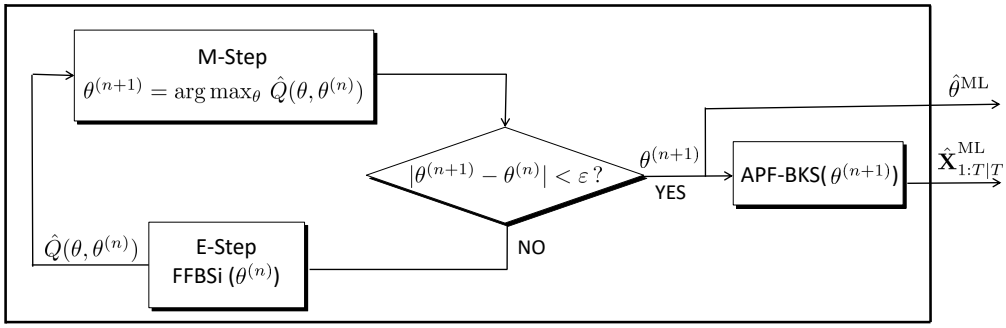

Fig. 2. Schematic diagram of particle-filter implementation of the continuous-parameter ML adaptive BKS.

where a_{ij} and b_i are system parameters. The former can take three values: $a_{ij} = +1$ if there is positive regulation (activation) from gene j to gene i; $a_{ij} = -1$ if there is negative regulation (inhibition) from gene j to gene i; and $a_{ij} = 0$ if gene j is not an input to gene i. The latter specifies regulation *biases* and can take two values: $b_i = +1/2$ or $b_i = -1/2$. The network function is depicted in Fig. 3, where the threshold units are step functions that output 1 if the input is nonnegative, and 0, otherwise.

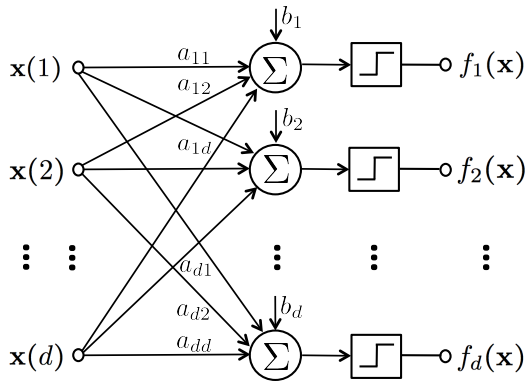

Fig. 3. Gene regulatory network model.

The process noise $\mathbf{n}_k$ in (1) is assumed to have independent components with $P(\mathbf{n}_k(i) = 1) = p$, for $i = 1, \ldots, d$, and $k = 1, 2, \ldots$ The noise parameter $0 \leq p \leq 0.5$ gives the amount of "perturbation" to the Boolean state process; the closer it is to $p = 0.5$, the more chaotic the system will be, while a value of p close to zero means that the state trajectories are nearly deterministic, being governed tightly by the network function.

5.2 RNA-Seq Measurement Model

Next-generation sequencing (NGS) technologies are able to sequence millions of short DNA fragments in parallel; the length and number of reads vary with the specific technology [46]. The application of NGS technology to transcriptional profiling is called RNA-seq, which records how frequently each transcript is represented in a sequence sample [47]. RNA-seq is a probe-free approach that can capture any relevant transcript present in a sample, without the need of prior knowledge about the target sequence.

Let $\mathbf{Y}_k = (\mathbf{Y}_k(1), \ldots, \mathbf{Y}_k(d))$ be a vector containing the RNA-seq data at time k, for $k = 1, 2, \ldots$, such that $\mathbf{Y}_k(j)$ is the read count corresponding to transcript j in a single-lane platform, for $j = 1, \ldots, d$. We assume conditional independence of the transcript counts given the state,

$$P\left(\mathbf{Y}_k = \mathbf{y} \mid \mathbf{X}_k = \mathbf{x}\right) = \prod_{j=1}^{d} P(\mathbf{Y}_k(j) = \mathbf{y}(j) \mid \mathbf{X}_k(j) = \mathbf{x}(j)), \tag{51}$$

and adopt the negative binomial model for each count,

$$P(\mathbf{Y}_k(j) = \mathbf{y}(j) \mid \mathbf{X}_k(j) = \mathbf{x}(j)) = \frac{\Gamma(\mathbf{y}(j) + \phi_j)}{\mathbf{y}(j)!\,\Gamma(\phi_j)} \left(\frac{\lambda_j}{\lambda_j + \phi_j}\right)^{\mathbf{y}(j)} \left(\frac{\phi_j}{\lambda_j + \phi_j}\right)^{\phi_j}, \tag{52}$$

where Γ denotes the Gamma function, and $\phi_j, \lambda_j > 0$ are the real-valued inverse dispersion parameter and mean read count of transcript j, respectively, for $j = 1, \ldots, d$. The inverse dispersion parameter models observation noise; the smaller it is, the more variable the measurements are.

Now, recall that, according to the Boolean state model, there are two possible states for the abundance of transcript j: high, if $\mathbf{x}(j) = 1$, and low, if $\mathbf{x}(j) = 0$. Accordingly, we model the parameter λ_j in log-space as:

$$\log \lambda_j \;=\; \log s \;+\; \mu \;+\; \delta_j\, \mathbf{x}(j)\,, \tag{53}$$

where the parameter s is the *sequencing depth* (which is instrument-dependent), $\mu > 0$ is the baseline level of expression in the inactivated transcriptional state, and $\delta_j > 0$ expresses the effect on the observed RNA-seq read count as gene j goes from the inactivated to the activated state, for $j = 1, \ldots, d$.

Based on equations (51)–(53), given the particles $\tilde{\mathbf{x}}_{k,i}$, one can compute $P(\mathbf{Y}_k \mid \mathbf{X}_k = \tilde{\mathbf{x}}_{k,i})$ as :

$$P\left(\mathbf{Y}_k = \mathbf{y} \mid \mathbf{X}_k = \tilde{\mathbf{x}}_{k,i}\right) = \prod_{j=1}^{d} \left[\frac{\Gamma(\mathbf{y}(j) + \phi_j)}{\mathbf{y}(j)!\,\Gamma(\phi_j)} \left(\frac{s\, \exp(\mu + \delta_j\, \tilde{\mathbf{x}}_{k,i}(j))}{s\, \exp(\mu + \delta_j\, \tilde{\mathbf{x}}_{k,i}(j)) + \phi_j} \right)^{\mathbf{y}(j)} \times \left(\frac{\phi_j}{s\, \exp(\mu + \delta_j\, \tilde{\mathbf{x}}_{k,i}(j)) + \phi_j} \right)^{\phi_j} \right]. \tag{54}$$

The RNA-seq measurement model parameters are thus the sequencing depth s, the baseline expression level μ, the transcript-dependent differential expression levels δ_j, for $j = 1, \ldots, d$, and the transcript-dependent inverse dispersion parameters ϕ_j, for $j = 1, \ldots, d$. These are all continuous parameters.

6 Numerical Experiments

In this section, we carry out detailed numerical experiments to assess the performance of the developed particle-based methods. We base our experiments on the well-known *Mammalian Cell-Cycle network* [48]. The pathway diagram for this network is presented in Fig. 4. The state vector is $\mathbf{x} = (CycD, Rb, p27, E2F, CycE, CycA, Cdc20, Cdh1, UbcH10, CycB)$. The gene interaction parameters a_{ij} can be read off Fig. 4 easily. As an example, *Rb* is activated by *p27*, and is inactivated by *CycD*, *CycE*, *CycA*, *CycB*. These interactions can be expressed in terms of interaction parameters as: $a_{21} = -1$, $a_{22} = 0$, $a_{23} = +1$, $a_{24} = 0$, $a_{25} = -1$, $a_{26} = -1$, $a_{27} = 0$, $a_{28} = 0$, $a_{29} = 0$ and $a_{2\,10} = -1$.

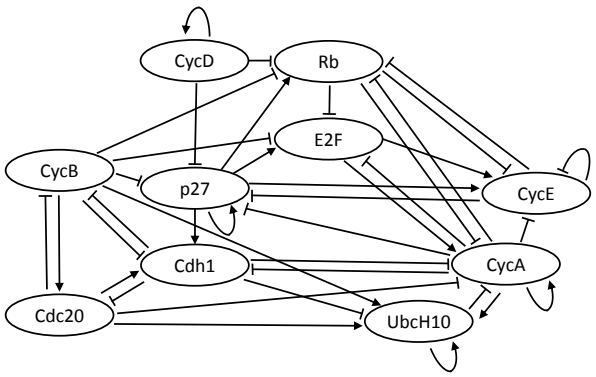

Fig. 4. Pathway diagram for the cell-cycle network.

In all numerical experiments to follow, we assume the same fixed set of "true" values for the system parameters, summarized in Table 2.

6.1 Experiment 1: State Estimation

In this section, the state estimation performance of the APF-BKF and APF-BKS is compared to that of the exact BKF and BKS, respectively. Given that the cell-cycle network comprises 10 genes, the size of the transition and update matrices required by both the BKF and BKS is $2^{10} \times 2^{10}$. As a result, the computational cost of the BKF and BKS is high. Table 3 shows the average rate of correct state estimation over 1000 independent runs for a time series with length 100.

As expected, the performance of both the AFP-BKF and APF-BKS is higher for large number of particles. However, the improvement is significantly larger by moving from 200 to 1000 particles in comparison to moving from 1000 to 5000 particles. One can also see the reduction in performance of all filters and smoothers as process noise or dispersion in measurements increases, which both make the estimation process more challenging. Also as expected, the BKS and APF-BKS outperform the BKF and APF-BKF, respectively, due to the availability of more data for estimation.

The average time to run the various algorithms for time series of length 100 and different number of particles is displayed in Fig. 5. Here, $p = 0.05$ and $\phi = 5$. Of course, the exact average running time might vary depending on the hardware and software setup used. However, we are more interested in the relative comparisons than the absolute values. For these results, we used the R programming language on a PC with an Intel Core i7-4790 CPU@3.60 GHz clock and 16 GB of RAM. The results show the large computational savings afforded by the APF-BKF and APF-BKS in comparison to the exact algorithms.

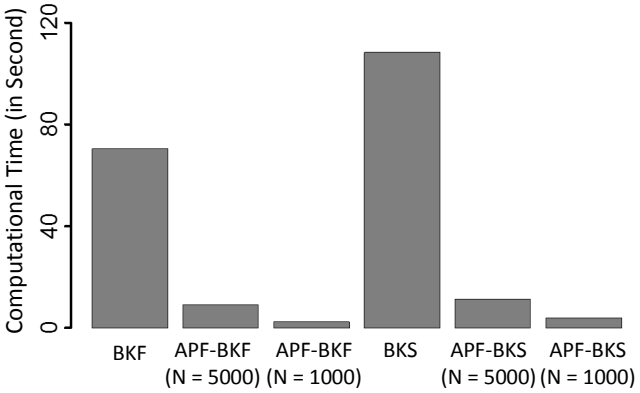

Fig. 5. Experiment 1: Average computation time (in seconds) for the exact and particle-based algorithms.

6.2 Experiment 2: Incomplete Network Topology

In this experiment, we assume that the interaction between genes *Rb* and *E2F*, or equivalently the gene inter-

Table 1
Derivatives of $Q(\theta, \theta^{(n)})$ with respect to different parameters needed in Numerical Experiment.

Parameter	derivation
p	$\frac{1}{N}\sum_{s=1}^{T}\sum_{i=1}^{N}\left(\frac{\|\tilde{\mathbf{x}}_{s,i} \oplus \mathbf{f}(\tilde{\mathbf{x}}_{s-1,i})\|_1 - d\,p}{p\,(1-p)}\right)$
s	$\frac{1}{Ns}\sum_{s=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{d}\left(\frac{\phi_j\,\left(\mathbf{Y}_s(j) - s\,\exp(\mu+\delta_j\,\tilde{\mathbf{x}}_{s,i}(j))\right)}{\phi_j + s\,\exp(\mu+\delta_j\,\tilde{\mathbf{x}}_{s,i}(j))}\right)$
μ	$\frac{1}{N}\sum_{s=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{d}\left(\frac{\phi_j\,\left(\mathbf{Y}_s(j) - s\,\exp(\mu+\delta_j\,\tilde{\mathbf{x}}_{s,i}(j))\right)}{\phi_j + s\,\exp(\mu+\delta_j\,\tilde{\mathbf{x}}_{s,i}(j))}\right)$
δ_j	$\frac{1}{N}\sum_{s=1}^{T}\sum_{i=1}^{N}\left(\frac{\tilde{\mathbf{x}}_{s,i}(j)\,\phi_j\,\left(\mathbf{Y}_s(j) - s\,\exp(\mu+\delta_j\,\tilde{\mathbf{x}}_{s,i}(j))\right)}{\phi_j + s\,\exp(\mu+\delta_j\,\tilde{\mathbf{x}}_{s,i}(j))}\right)$
ϕ_j	$\frac{1}{N}\sum_{s=1}^{T}\sum_{i=1}^{N}\left(\frac{\Gamma'(\mathbf{Y}_s(j)+\phi_j)}{\Gamma(\mathbf{Y}_s(j))+\phi_j)} - \frac{\Gamma'(\phi_j)}{\Gamma(\phi_j)} + \frac{s\,\exp(\mu+\delta_j\,\tilde{\mathbf{x}}_{s,i}(j)) - \mathbf{Y}_s(j)\phi_j}{\phi_j(s\,\exp(\mu+\delta_j\,\tilde{\mathbf{x}}_{s,i}(j))+\phi_j)} + \log\frac{\phi_j}{s\,\exp(\mu+\delta_j\,\tilde{\mathbf{x}}_{s,i}(j))+\phi_j}\right)$

Table 2
Parameter values for numerical experiments using the Cell-Cycle gene regulatory network.

Parameter	**Value**
Length of time series T	50, 100
Number of genes d	10
Initial distribution $P(\mathbf{X}_0 = \mathbf{x}^i), i = 1 : 2^{10}$	$1/2^{10}$
Number of particles N	200, 1000, 5000
Bias $b_i, i = 1, \ldots, 10$	-1/2
Transition noise intensity p	0.01, 0.05
Sequencing depth s	1.02 (1K-50K reads)
Baseline expression μ	0.1
Differential expression δ_i, $i = 1, \ldots, 10$	2
Inverse dispersion ϕ_i, $i = 1, \ldots, 10$	1, 5
APF-CPMLA-BKS stopping criterion ε	10^{-4}

action parameter a_{42}, is unknown, and all other parameters are known. Since this is a discrete parameter estimation problem, the APF-DPMLA-BKF is run, which in this case consists of three APF-BKFs running in parallel — one for each possible kind of interaction (activation, inhibition, no interaction).

Table 4 displays the average accuracy rate in the estimation of the interaction type between *Rb* and *E2F* over 100 different runs. We can observe that performance increases with longer time series and larger number of particles, as expected. The performance is better for larger transition noise. The reason is that large transition noise gets the system out of its attractors more often and, as a result, helps the estimation process.

Table 3
Experiment 1: Average rates of correct state estimation over 1000 independent runs for a time series with length 100.

p	ϕ	N	BKF	APF-BKF	BKS	APF-BKS
0.01	5	200	93.9	85.4	96.6	88.1
		1000		92.1		95.0
		5000		93.2		95.7
	1	200	83.8	74.6	90.7	80.4
		1000		80.6		88.3
		5000		82.1		89.8
0.05	5	200	82.9	75.0	93.4	82.3
		1000		80.3		91.3
		5000		81.9		92.6
	1	200	58.5	50.1	70.8	62.3
		1000		55.1		68.2
		5000		56.9		69.9

The evolution of estimated state and discrete-parameter for a single sample run of the experiment, for transition noise $p = 0.05$, $\phi = 5$, and $N = 5000$ is displayed in Fig. 6. We can observe that the discrete parameter a_{42}

Table 4
Experiment 2 results. Average accuracy rates for estimation of the gene interaction parameter a_{42}.

n	p	N	High Disp. $\phi = 1$	Low Disp. $\phi = 5$
30	0.01	200	0.45	0.62
		1000	0.58	0.71
		5000	0.59	0.73
	0.05	200	0.52	0.71
		1000	0.64	0.74
		5000	0.67	0.75
60	0.01	200	0.82	0.86
		1000	0.87	0.93
		5000	0.89	0.93
	0.05	200	0.86	0.91
		1000	0.89	0.95
		5000	0.92	0.96

is estimated correctly after less than 20 time step. In addition, we can see that the state estimator of each gene eventually converges to the true state value.

6.3 Experiment 3: Unknown Noise and Expression Parameters

In the final experiment, the Boolean network topology (gene interaction parameters a_{ij} and biases b_i) is assumed to be completely known, whereas the transition noise parameter p, the baseline expression μ, and differential expression δ_i, $i = 1, \ldots, 10$, are unknown. The inverse dispersion parameters are assumed to be $\phi_i = 5$, $i = 1, \ldots, 10$.

In order to assess continuous-parameter estimation accuracy, we define the relative distance between estimated and true parameter values as

$$\text{Relative Distance}(\hat{\theta}) = \frac{|\hat{\theta} - \theta^*|}{R(\theta)}, \tag{55}$$

where θ^* is the true parameter value, and $R(\theta)$ is the range of parameter θ assumed in the M-step of the APF-CPMLA-BKS algorithm. Here, the range is $R(p) = [0, 0.5]$ for the transition noise p, $R(\mu) = [0, 2]$ for the baseline expression μ, and $R(\delta_i) = [0.1, 10]$ for the differential expression δ_i, $i = 1, \ldots, 10$.

A new version of the "augmented Lagrange method" [49] is used for optimization in the M-Step of the particle-based EM algorithm. The gradient vector at each step is computed based on Table 1. The procedure terminates when the maximum of the absolute values of the changes in the parameter estimates in two consecutive iterations gets smaller than 10^{-4}.

The average relative distance between estimated and true parameter values over 100 independent runs for different inverse dispersion parameters and time series lengths are plotted in Fig. 7. As expected, the performance of APF-CPMLA-BKS improves steadily as time goes on. Performance improves by increasing the number of particles; however, the curves get close to each other as the length of the time series increases. All curves show a decreasing trend, which indicates that the parameter estimates become arbitrarily close to the true values for sufficiently long time.

7 Conclusion

In this paper, we introduced approximate particle-based algorithms for state and simultaneous state and parameter estimation for large partially-observed Boolean dynamical systems. For approximate state estimation, filtering and smoothing methods based on auxiliary particle filtering (APF) were developed to approximate the optimal BKF and BKS. These algorithms are called APF-BKF and APF-BKS, and are original contributions of this work.

Moreover, we considered the case where some of the parameters may not be known. In the discrete parameter case, an adaptive filter scheme is developed based on APF-BKF algorithms running in parallel. For continuous parameter problems, a particle-based EM algorithm for POBDS was presented.

The methodology was applied to a model of Boolean gene regulatory networks observed through RNA sequencing data. The numerical experiments with a cell-cycle Boolean network demonstrated the ability of the proposed methodologies to efficiently estimate the state and also the parameters of the large Boolean regulatory network observed through noisy measurements.

Acknowledgment

The authors acknowledge the support of the National Science Foundation, through NSF award CCF-1320884.

References

[1] U. Braga-Neto, "Optimal state estimation for Boolean dynamical systems," 2011. Proceedings of 45th Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA.

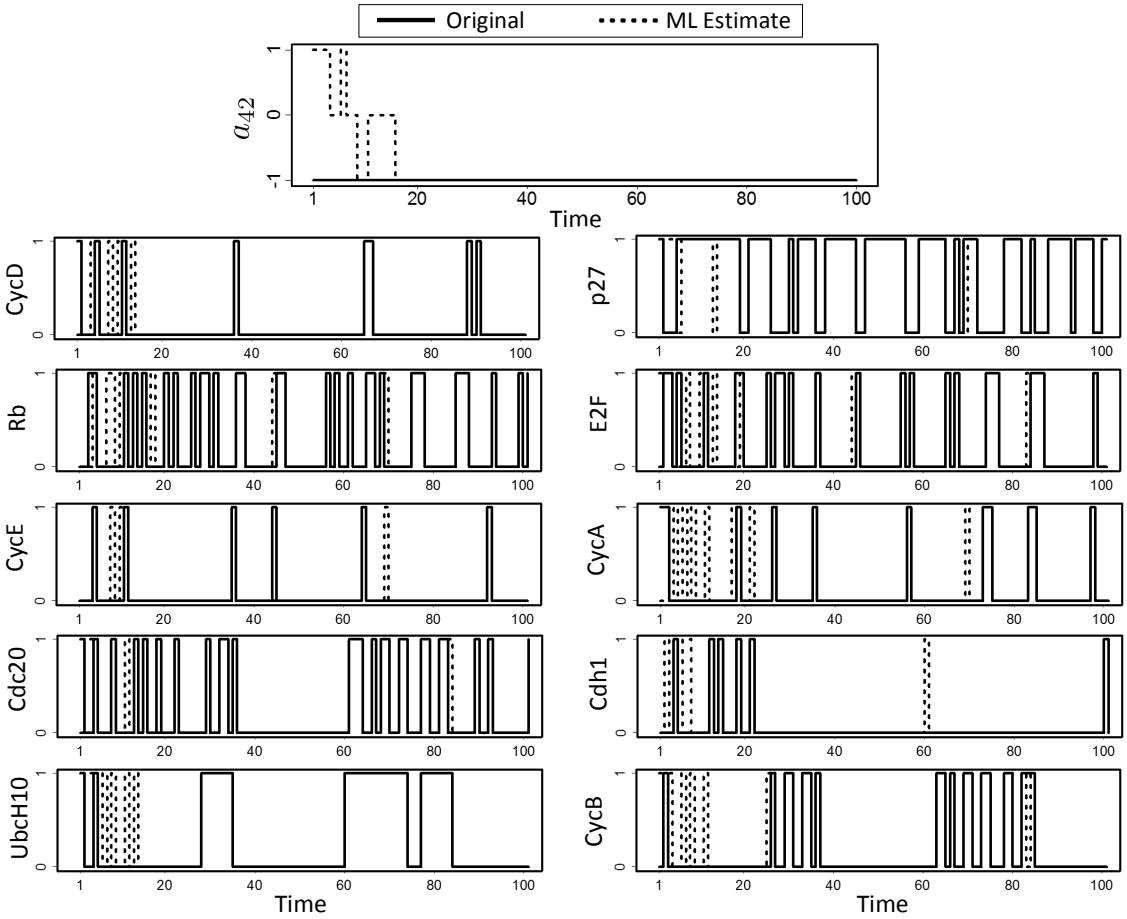

Fig. 6. Experiment 2: Estimated interaction type from *Rb* to *E2F* and true gene trajectories for single sample run of the experiment.

[2] M. Imani and U. Braga-Neto, "Optimal gene regulatory network inference using the Boolean Kalman filter and multiple model adaptive estimation," in *Proceedings of the 49th Annual Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA*, pp. 423–427, 2015.

[3] M. Imani and U. Braga-Neto, "Optimal state estimation for Boolean dynamical systems using a Boolean Kalman smoother," in *Proceedings of the 3rd IEEE Global Conference on Signal and Information Processing (GlobalSIP'2015), Orlando, FL*, pp. 972–976, 2015.

[4] M. Imani and U. Braga-Neto, "Point-based value iteration for partially-observed Boolean dynamical systems with finite observation space," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pp. 4208–4213, IEEE, 2016.

[5] S. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, pp. 437–467, 1969.

[6] A. Roli, M. Manfroni, C. Pinciroli, and M. Birattari, "On the design of Boolean network robots," in *Applications of Evolutionary Computation*, pp. 43–52, Springer, 2011.

[7] D. Messerschmitt, "Synchronization in digital system design," *IEEE Journal on Selected Areasin Communications*, vol. 8, no. 8, pp. 1404–1419, 1990.

[8] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, vol. 64. Academic Press, 1970.

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

[10] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *American Control Conference, Proceedings of the 1995*, vol. 3, pp. 1628–1632, IEEE, 1995.

[11] K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE transactions on automatic control*, vol. 45, no. 5, pp. 910–927, 2000.

[12] M. NøRgaard, N. K. Poulsen, and O. Ravn, "New developments in state estimation for nonlinear systems," *Automatica*, vol. 36, no. 11, pp. 1627–1638, 2000.

[13] A. Doucet, N. De Freitas, and N. Gordon, "Sequential monte carlo methods in practice," 2001.

[14] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, N. Chopin, *et al.*, "On particle methods for parameter estimation in state-space models," *Statistical science*, vol. 30, no. 3, pp. 328–351, 2015.

[15] U. Braga-Neto, "Particle filtering approach to state estimation in Boolean dynamical systems," 2013. Proceedings of the IEEE Global Conference on Signal and Image Processing (GlobalSIP'13), Austin, TX.

[16] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American statistical*

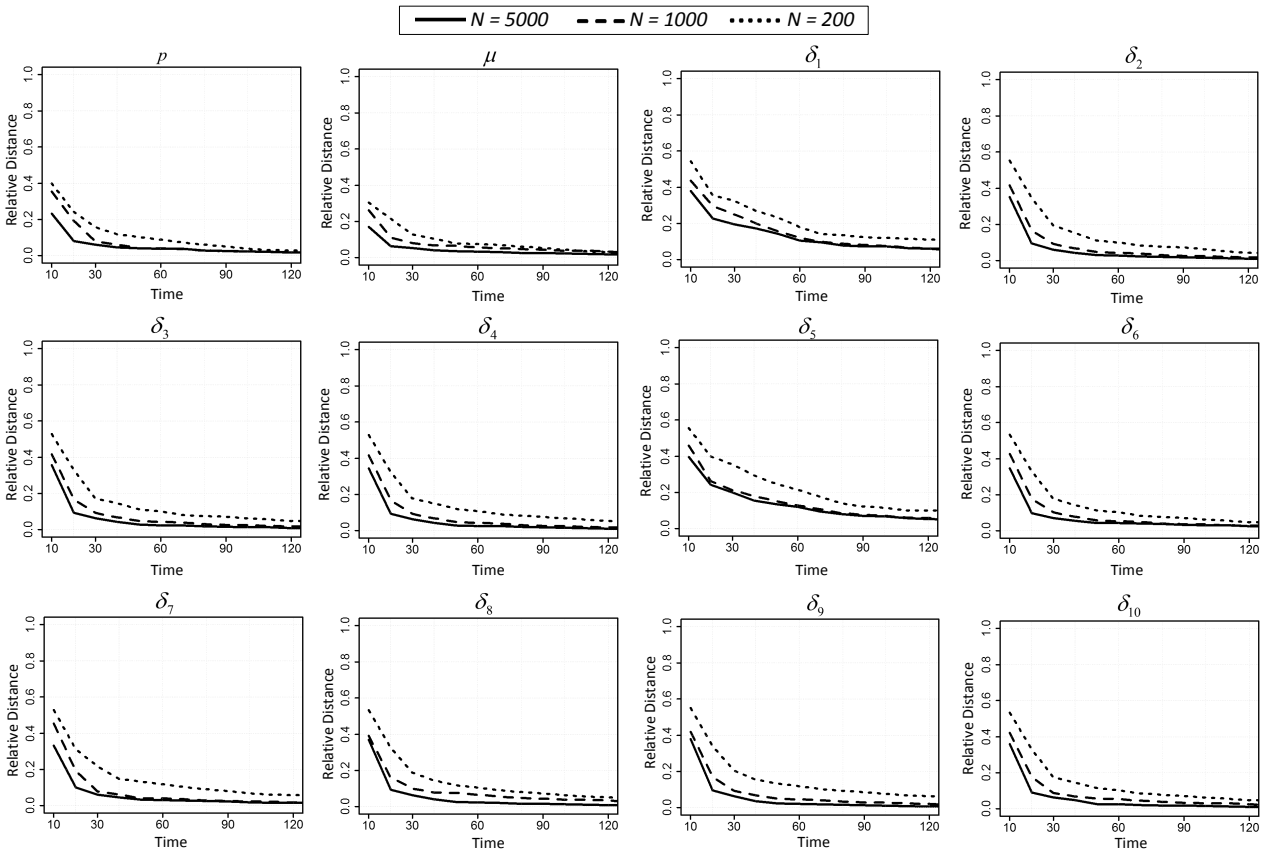

Fig. 7. Experiment 3: Average relative distance between estimated and true parameter values as a function of time series length.

association, vol. 94, no. 446, pp. 590–599, 1999.

[17] M. Imani and U. Braga-Neto, "Maximum-likelihood adaptive filter for partially observed Boolean dynamical systems," *IEEE Transactions on Signal Processing*, vol. 65, no. 2, pp. 359–371, 2017.

[18] T. Chen, J. Morris, and E. Martin, "Particle filters for state and parameter estimation in batch processes," *Journal of Process Control*, vol. 15, no. 6, pp. 665–673, 2005.

[19] J. Liu and M. West, "Combined parameter and state estimation in simulation-based filtering," in *Sequential Monte Carlo methods in practice*, pp. 197–223, Springer, 2001.

[20] M. Hürzeler and H. R. Künsch, "Approximating and maximising the likelihood for a general state-space model," in *Sequential Monte Carlo methods in practice*, pp. 159–175, Springer, 2001.

[21] S. Malik and M. K. Pitt, "Particle filters for continuous likelihood evaluation and maximisation," *Journal of Econometrics*, vol. 165, no. 2, pp. 190–209, 2011.

[22] D. N. DeJong, R. Liesenfeld, G. V. Moura, J.-F. Richard, and H. Dharmarajan, "Efficient likelihood evaluation of state-space representations," *The Review of Economic Studies*, vol. 80, no. 2, pp. 538–567, 2013.

[23] A. D. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm (with Discussion)," *Journal of the Royal Statistical Society, Series B*, vol. 39, pp. 1–38, 1977.

[24] Z. Ghahramani, "Parameter estimation for linear dynamical systems," tech. rep., Dept. of Computer Science, University of Toronto, 1996.

[25] T. B. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," *Automatica*, vol. 47, no. 1, pp. 39–49, 2011.

[26] A. Wills, T. B. Schön, L. Ljung, and B. Ninness, "Identification of hammerstein–wiener models," *Automatica*, vol. 49, no. 1, pp. 70–81, 2013.

[27] R. Gopaluni, "A particle filter approach to identification of nonlinear processes under missing observations," *The Canadian Journal of Chemical Engineering*, vol. 86, no. 6, pp. 1081–1092, 2008.

[28] V. J. Väänänen *et al.*, "Gaussian filtering and smoothing based parameter estimation in nonlinear models for sequential data," 2012.

[29] P. S. Maybeck and P. D. Hanlon, "Performance enhancement of a multiple model adaptive estimator," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 31, no. 4, pp. 1240–1254, 1995.

[30] S. J. Godsill, A. Doucet, and M. West, "Monte carlo smoothing for nonlinear time series," *Journal of the american statistical association*, 2012.

[31] I. Schmulevich, E. Dougherty, and W. Zhang, "From Boolean to probabilistic Boolean networks as models of genetic regulatory networks," *Proceedings of the IEEE*, vol. 90, pp. 1778–1792, 2002.

[32] S. Marguerat and J. Bahler, "RNA-seq: from technology to

biology," *Cellular and molecular life science*, vol. 67, no. 4, pp. 569–579, 2010.

[33] A. Faure, A. Naldi, C. Chaouiya, and D. Thieffry, "Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle," *"Bionformatics"*, vol. 22, no. 14, pp. e124–e131, 2006.

[34] M. Imani and U. Braga-Neto, "Multiple model adaptive controller for partially-observed Boolean dynamical systems," in *2017 Ammerical Control Conference*, IEEE, 2017.

[35] L. D. McClenny, M. Imani, and U. M. Braga-Neto, "Boolean Kalman filter with correlated observation noise," in *The 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, 2017.

[36] M. Imani and U. Braga-Neto, "Control of gene regulatory networks with noisy measurements and uncertain inputs," *arXiv preprint arXiv:1702.07652*, 2017.

[37] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[38] N. Whiteley and A. Johansen, "Recent developments in auxiliary particle filtering," in *Bayesian time series models* (D. Barber, A. Cemgil, and S. Chiappa, eds.), pp. 52–79, New York, NY: Cambridge University Press, 2011.

[39] M. K. Pitt, "Smooth particle filters for likelihood evaluation and maximisation," tech. rep., University of Warwick, Department of Economics, 2002.

[40] G. Kitagawa, "Monte carlo filter and smoother for non-gaussian nonlinear state space models," *Journal of computational and graphical statistics*, vol. 5, no. 1, pp. 1–25, 1996.

[41] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and computing*, vol. 10, no. 3, pp. 197–208, 2000.

[42] S. Barembruch, A. Garivier, and E. Moulines, "On approximate maximum-likelihood methods for blind identification: how to cope with the curse of dimensionality," *Signal Processing, IEEE Transactions on*, vol. 57, no. 11, pp. 4247–4259, 2009.

[43] M. Hürzeler and H. R. Künsch, "Monte carlo approximations for general state-space models," *Journal of Computational and graphical Statistics*, vol. 7, no. 2, pp. 175–193, 1998.

[44] Y. Chen, E. R. Dougherty, and M. L. Bittner, "Ratio-based decisions and the quantitative analysis of cdna microarray images," *Journal of Biomedical optics*, vol. 2, no. 4, pp. 364–374, 1997.

[45] J. Hua, C. Sima, M. Cypert, G. C. Gooden, S. Shack, L. Alla, E. A. Smith, J. M. Trent, E. R. Dougherty, and M. L. Bittner, "Dynamical analysis of drug efficacy and mechanism of action using gfp reporters," *Journal of Biological Systems*, vol. 20, no. 04, pp. 403–422, 2012.

[46] N. Ghaffari, M. R. Yousefi, C. D. Johnson, I. Ivanov, and E. R. Dougherty, "Modeling the next generation sequencing sample processing pipeline for the purposes of classification," *BMC bioinformatics*, vol. 14, no. 1, p. 307, 2013.

[47] M. Imani and U. Braga-Neto, "State-feedback control of partially-observed Boolean dynamical systems using RNA-seq time series data," in *American Control Conference (ACC), 2016*, pp. 227–232, IEEE, 2016.

[48] A. Fauré, A. Naldi, C. Chaouiya, and D. Thieffry, "Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle," *Bioinformatics*, vol. 22, no. 14, pp. e124–e131, 2006.

[49] E. G. Birgin and J. M. Martínez, "Improving ultimate convergence of an augmented lagrangian method," *Optimization Methods and Software*, vol. 23, no. 2, pp. 177–195, 2008.