Fast Kalman filtering and forward-backward smoothing via a low-rank perturbative approach

Eftychios A. Pnevmatikakis *
Kamiar Rahnama Rad
Jonathan Huggins
Liam Paninski

October 15, 2012

Abstract

Kalman filtering-smoothing is a fundamental tool in statistical time series analysis. However, standard implementations of the Kalman filter-smoother require $O(d^3)$ time and $O(d^2)$ space per timestep, where d is the dimension of the state variable, and are therefore impractical in high-dimensional problems. In this paper we note that if a relatively small number of observations are available per time step, the Kalman equations may be approximated in terms of a low-rank perturbation of the prior state covariance matrix in the absence of any observations. In many cases this approximation may be computed and updated very efficiently (often in just $O(k^2 d)$ or $O(k^2 d + kd \log d)$ time and space per timestep, where k is the rank of the perturbation and in general $k \ll d$), using fast methods from numerical linear algebra. We justify our approach and give bounds on the rank of the perturbation as a function of the desired accuracy. For the case of smoothing we also quantify the error of our algorithm due to the low rank approximation and show that it can be made arbitrarily low at the expense of a moderate computational cost. We describe applications involving smoothing of spatiotemporal neuroscience data.

Keywords: covariance approximation, fast algorithm, low rank methods, numerical analysis, tracking

The document is accompanied by an appendix and Matlab code in the form of supplementary material.

*The authors are with the Department of Statistics and Center for Theoretical Neuroscience, Columbia University, New York, NY 10027. email: `eftychios@stat.columbia.edu`, `kamiar@stat.columbia.edu`, `jhuggins@mit.edu` and `liam@stat.columbia.edu`.

1 Introduction

Understanding the dynamics of large systems for which limited, noisy observations are available is a fundamental and recurring scientific problem. A key step in any such analysis involves data assimilation: we must incorporate incoming observations and update our beliefs about the dynamical state of the system accordingly. The Kalman filter may be considered the canonical method for data assimilation; this method provides a conceptually simple recursive framework for online Bayesian inference in the context of linear and Gaussian dynamics and observation processes. Furthermore, the Kalman filter serves as the underlying computational engine in a wide variety of more complicated non-Gaussian and nonlinear statistical models.

However, these methods face a major limitation: standard implementations of the Kalman filter require $O(d^3)$ time and $O(d^2)$ space per timestep, where d denotes the dimension of the system state variable, and are therefore impractical for applications involving very high-dimensional systems. The bottleneck is in the representation and computation of the forward covariance matrix $C_t = \mathrm{Cov}(x_t|Y_{1:t})$: this is the posterior covariance of the d-dimensional state vector x_t, given the sequence of observations $Y_{1:t}$ up to the current time t. Two natural ideas for reducing the computational burden of storing and computing this $d \times d$ matrix have been explored. First, if C_t is sparse (i.e., consists of mostly zeros), then we can clearly store and perform matrix-vector computations with C_t with $o(d^2)$ complexity. In many examples C_t has a nearly banded, or strongly tapered, structure (i.e., most of the large components of C_t are near the diagonal), and sparse approximate matrix updates can be exploited. This approach has been shown to be extremely effective in some cases (Furrer and Bengtsson, 2007; Khan and Moura, 2008; Bickel and Levina, 2008; Kaufman et al., 2008; El Karoui, 2008), but in many settings there is no a priori reason to expect C_t to have any useful sparse structure, and therefore this idea can not be applied generally.

Second, we could replace C_t with a low-rank approximation. For example, a major theme in the recent literature on numerical weather prediction (where the system of interest is the atmosphere discretized in a spatial grid, leading in many cases to a state dimension in the tens or hundreds of millions) has been the development of the theory of the "ensemble Kalman

filter" (Verlaan, 1998; Treebushny and Madsen, 2005; Chandrasekar et al., 2008; Evensen, 2009), which implements a Monte Carlo-based, low-rank approximation of the full Kalman filter. Low-rank approximations for C_t are typically justified on computational grounds but may also be justified statistically in the case that many high-signal-to-noise-ratio (high-SNR) observations are available: in this setting, we can argue that our posterior uncertainty C_t will be approximately restricted to a subspace of dimension significantly less than d, as discussed, e.g., by Solo (2004). Alternatively, we may impose a low-rank structure on the posterior covariance C_t directly by choosing our prior covariance matrix to be of low rank (Wikle and Cressie, 1999; Wood, 2006; Cressie and Johannesson, 2008; Banerjee et al., 2008; Cressie et al., 2010); however, our focus in this work is on approximating C_t given a prior covariance matrix which is of full rank.

The low-SNR setting, where a relatively small number of noisy observations are available per time step, has been explored less thoroughly. One exception is the neuronal dendritic application discussed by Paninski (2010), where we noted that C_t could be approximated very accurately in terms of a low-rank perturbation of C_0, the prior equilibrium covariance of the state variable x_t in the absence of any observations Y. (Note that this approximation is very different from the high-SNR case, where we approximate C_t as a low-rank perturbation of the zero matrix, not of C_0.) To efficiently update this low-SNR approximation to C_t, Paninski (2010) exploited the special structure of the dynamics in this application: dendritic voltage dynamics are governed by a cable equation on a tree (Koch, 1999), which may be solved using symmetric sparse matrix methods in $O(d)$ time (Hines, 1984). In turn, this implied that C_t could be updated in $O(k^2 d)$ time, where k is the rank of the perturbation of C_0 used to represent C_t. Since empirically a $k \ll d$ sufficed to accurately approximate C_t in this application, this approach resulted in a much faster implementation of the Kalman filter, with linear instead of cubic complexity in d.

In this paper we extend this basic idea in a number of ways. We first develop a methodology that provides upper bounds on the rank of the perturbation on C_0 required to represent C_t. Our analysis shows that the basic idea is applicable to both high and low-SNR cases, and that the rank of the perturbation is indeed small and thus the algorithm can lead to substantial computational gains. We also develop a similar fast algorithm for full forward-backward

smoothing by deriving an efficient low-rank block-Thomas (LRBT) recursive algorithm for the solution of block-tridiagonal systems. For this LRBT algorithm we also characterize the the tradeoff between the rank of the approximation (and thus the computational cost) and the induced approximation error. We show that the error can be made arbitrarily small, with a relatively moderate computational cost incurred by the corresponding increase in the rank of the perturbation. We also show that the LRBT algorithm efficiently calculates the steepest descent direction under an appropriate quadratic norm. As a result it can be used as an iterative steepest-descent algorithm, or as a preconditioner in standard iterative methods (e.g. conjugate gradients), to converge to the exact solution faster than exact forward-backward methods.

We describe a number of examples where special features of the system dynamics allow us to compute and update the low-rank approximation to C_t efficiently (often in just $O(k^2 d)$ or $O(k^2 d + kd \log d)$ time and $O(kd)$ space per timestep), using fast methods from numerical linear algebra. One particularly simple setting involves spatiotemporal smoothing applications; as a concrete example, we describe how to apply the proposed methods to efficiently smooth certain kinds of high-dimensional spatiotemporal neuroscience data. Finally, we briefly describe extensions of our methods to non-linear, non-Gaussian settings.

2 Basic Kalman filtering setup

We begin by briefly reviewing the Kalman filter and establishing notation. Again, let x_t denote our d-dimensional state variable, and y_t the observation at time t. We assume that x_t and y_t satisfy the following linear-Gaussian dynamics and observation equations:

$$x_{t+1} = Ax_t + u_t + \epsilon_t, \ \epsilon_t \sim \mathcal{N}(0, V) \tag{1}$$

$$y_t = B_t x_t + \eta_t, \ \eta_t \sim \mathcal{N}(\mu_t^\eta, W_t), \tag{2}$$

with initial conditions $x_0 \sim \mathcal{N}(\mu_0, V_0)$. Here A represents the system dynamics matrix; u_t is a deterministic input to the system at time t, and ϵ_t is an i.i.d. Gaussian vector with mean zero and covariance V. B_t denotes the observation gain matrix, W_t the observation noise

covariance, and μ_t^η an offset mean in the observation. Our methods are sufficiently general that the dimension of y_t can vary with time. From now on, without loss of generality we assume that $\mu_0 = 0$, and $u_t = 0$, $\mu_t^\eta = 0$ for all t. Nonlinear and non-Gaussian observations may also be incorporated in some cases, as we will discuss further below. Moreover, extensions to non-stationary models, where A and/or V in the dynamics equation vary with time, are also possible in some cases (Pnevmatikakis and Paninski, 2012), but will not be discussed here.

Now the focus of this paper is the efficient implementation of the Kalman filter recursion for computing the forward mean $\mu_t = \mathbb{E}(x_t|Y_{1:t})$, and covariance $C_t = \mathrm{Cov}(x_t|Y_{1:t})$, where $Y_{1:t}$ denotes the observed data $\{y_s\}$ up to time t. The Kalman recursions may be written as (Anderson and Moore, 1979):

$$C_t = \left(P_t^{-1} + B_t^T W_t^{-1} B_t\right)^{-1} \tag{3}$$

$$\mu_t = A\mu_{t-1} + P_t B_t^T (W_t + B_t P_t B_t^T)^{-1}(y_t - B_t A\mu_{t-1}) \tag{4}$$

with

$$P_t \triangleq \mathrm{Cov}(x_t|Y_{1:t-1}) = AC_{t-1}A^T + V. \tag{5}$$

Note that computing the inverses in the recursion for C_t requires $O(d^3)$ time in general, or $O(d^2)$ time via the Woodbury lemma (Golub and Van Loan, 1996) if the observation matrix B_t is of low rank (i.e., if $\mathrm{rank}(B_t) \ll d$). In either case, $O(d^2)$ space is required to store C_t.

A key quantity is the prior covariance $C_{0,t}$, i.e., the covariance of x_t in the absence of any observations. From the Kalman filter recursion, $C_{0,t}$ evolves as

$$C_{0,t} = AC_{0,t-1}A^T + V. \tag{6}$$

This is just the Kalman recursion for C_t above in the special case that $B = 0$ (i.e., no observations are available). Throughout the paper we make the assumption that A is stable, i.e., $\|A\| < 1$, where $\|\cdot\|$ denotes the spectral norm. In this case $C_{0,t}$ converges to the equilibrium prior covariance $C_0 = \lim_{t\to\infty} C_{0,t}$. To enforce stationarity of the prior, the Kalman recursion is often initialized with $V_0 \triangleq C_{0,0} = C_0$. In this case we have $C_{0,t} = C_0$ for

all t, since the equilibrium covariance C_0 satisfies the discrete Lyapunov equation

$$AC_0A^T + V = C_0. \tag{7}$$

This equation can be solved explicitly in many cases (Anderson and Moore, 1979), as we discuss briefly now. If A is normal (i.e., $AA^T = A^TA$), and commutes with the dynamics noise covariance V, then C_0 can be explicitly computed using the standard moving-average recursion (Brockwell and Davis, 1991) for the autoregressive model x_t:

$$C_0 = \sum_{i=0}^{\infty} A^i V (A^T)^i = V \sum_{i=0}^{\infty} (AA^T)^i = V(I - AA^T)^{-1}. \tag{8}$$

More generally, if V and A do not commute then we can employ the (linear) whitening change of variables $\tilde{x}_t = V^{-1/2}x_t$ (assuming V is of full rank). Defining the reparameterized covariance matrix C_0' via $C_0 = V^{1/2}C_0'V^{1/2}$, A_V through the similarity transformation $A_V = V^{-1/2}AV^{1/2}$, and assuming A_V is normal, we rewrite (7) as

$$A_V C_0' A_V^T + C_0' \Rightarrow C_0' = \left(I - A_V A_V^T\right)^{-1} \Rightarrow C_0 = V^{1/2}\left(I - A_V A_V^T\right)^{-1} V^{1/2}. \tag{9}$$

The case where V is of reduced rank, or the resulting A_V is non-normal, appears to be more difficult, as noted in more detail in the Discussion section below. From now on unless noted otherwise we make the assumption that A is normal and commutes with V.

3 Fast Kalman filtering

Now the basic idea is that when $\text{rank}(B_t) \ll d$, C_t should be close to $C_{0,t}$: i.e., we should be able to represent the time-varying covariance C_t as a small perturbation about the prior covariance $C_{0,t}$, in some sense. Thus, more concretely, we will approximate C_t as

$$C_t \approx \tilde{C}_t \triangleq C_{0,t} - L_t \Sigma_t L_t^T, \tag{10}$$

where $L_t \Sigma_t L_t^T$ is a low-rank matrix we will update directly, and $C_{0,t} = \mathrm{Cov}(x_t)$. We will show that it is straightforward to compute and update the perturbations L_t and Σ_t efficiently whenever fast methods are available to solve linear equations involving A and $C_{0,t}$.

But first, why does the approximation in eq. (10) make sense? It is easy to see, using the Woodbury matrix lemma, that if we make b observations at time $t = 1$ then (10) will hold exactly, for $L_1 \Sigma_1 L_1^T$ of rank at most b. If we make no further observations, then C_t follows the simple update rule

$$C_t = AC_{t-1}A^T + V \Rightarrow C_2 = A(C_{0,1} - L_1 \Sigma_1 L_1^T)A^T + V = C_{0,2} - AL_1 \Sigma_1 L_1^T A^T;$$

the last equality follows from (6). Iterating, we see that

$$C_t = C_{0,t} - A^{t-s} L_s \Sigma_s L_s^T (A^{t-s})^T,$$

where s denotes the time of the last available observation. Since A is assumed to be stable, this implies that the perturbation to C_t around the equilibrium covariance $C_{0,t}$ caused by the observations up to time s will decay exponentially; for $t - s$ sufficiently large, we can discard some dimensions of the perturbation $A^{t-s} L_s \Sigma_s L_s^T (A^{t-s})^T$ without experiencing much error in C_t. In the case that additional observations become available with each timestep t, a similar phenomenon governs the behavior of C_t: long-ago observations are eventually "forgotten," due to the exponential decay caused by the double multiplication AC_tA^T. We may exploit this exponential decay by discarding some dimensions of $C_t - C_{0,t}$ as they become sufficiently small, and if the observations are sufficiently low-rank relative to the decay rate imposed by A, then the effective rank of $C_t - C_{0,t}$ will remain small.

3.1 The fast Kalman filtering algorithm

Now we can describe a method for efficiently updating L_t and Σ_t. We will use A and $C_{0,t}$ in what follows; it is easy to substitute the transformed matrices A_V and $C'_{0,t}$ (defined previously) if necessary. First, as above, for the approximate predictive covariance $\tilde{P}_t$ write

$$\begin{aligned}\tilde{P}_t^{-1} &\triangleq (A\tilde{C}_{t-1}A^T + V)^{-1} = (A(C_{0,t-1} - L_{t-1}\Sigma_{t-1}L_{t-1}^T)A^T + V)^{-1} \\ &= (C_{0,t} - AL_{t-1}\Sigma_{t-1}L_{t-1}^T A^T)^{-1} = C_{0,t}^{-1} + \Phi_t\Delta_t\Phi_t^T,\end{aligned} \tag{11}$$

where we applied (6) and the Woodbury lemma, and abbreviated $\Phi_t = C_{0,t}^{-1}AL_{t-1}$ and $\Delta_t = (\Sigma_{t-1}^{-1} - L_{t-1}^T A^T C_{0,t}^{-1} A L_{t-1})^{-1}$.

Now plug this into the covariance update and apply Woodbury[1] again:

$$\begin{aligned}\tilde{C}_t &= \left(C_{0,t}^{-1} + \Phi_t\Delta_t\Phi_t^T + B_t^T W_t^{-1} B_t\right)^{-1} = \left(C_{0,t}^{-1} + O_t Q_t O_t^T\right)^{-1} \\ &= C_{0,t} - C_{0,t}O_t(Q_t^{-1} + O_t^T C_{0,t} O_t)^{-1} O_t^T C_{0,t},\end{aligned} \tag{12}$$

where $$O_t = [\Phi_t \quad B_t^T], \quad Q_t = \text{blkdiag}\{\Delta_t, W_t^{-1}\}. \tag{13}$$

We obtain L_t and Σ_t by truncating the partial SVD of the right-hand side of (12):

$$[\hat{L}_t, \hat{\Sigma}_t^{1/2}] = \text{svd}(C_{0,t}O_t(Q_t^{-1} + O_t^T C_{0,t} O_t)^{-1/2}), \tag{14}$$

then choose L_t as the first k_t columns of $\hat{L}_t$ and Σ_t as the first k_t diagonal elements $\hat{\Sigma}_t$, where k_t is chosen to be large enough (for accuracy) and small enough (for computational tractability). A reasonable choice of k_t is as the least solution of the inequality:

$$\sum_{i \le k_t} [\hat{\Sigma}_t]_{ii} \ge \theta \sum_i [\hat{\Sigma}_t]_{ii}; \tag{15}$$

i.e., choose k_t to capture at least a large fraction θ of the term $\hat{L}_t\hat{\Sigma}_t^{1/2}$ (i.e., the square root of the term perturbing $C_{0,t}$ in (12)). Now for the update of the approximate Kalman mean $\tilde{\mu}_t$ we can use the exact formula (4) but replace P_t with the approximate predictive covariance $\tilde{P}_t$ (11). Note that we update the mean $\tilde{\mu}_t$ first, then truncate L_t and Σ_t.

[1]It is well-known that the Woodbury formula can be numerically unstable when the observation covariance W is small (i.e., the high-SNR case). It should be possible to derive a low-rank square-root filter (Treebushny and Madsen, 2005; Chandrasekar et al., 2008) to improve the numerical stability here, though we have not yet pursued this direction. Meanwhile, a crude but effective method to guarantee that C_t remains positive definite is to simply shrink Σ_t slightly if any negative eigenvalues are detected. This can be done easily in $O(d)$ time by restricting attention to the subspace spanned by L_t.

To review, we have introduced simple low-rank recursions for L_t, Σ_t, and μ_t in terms of $C_{0,t}$ and A. The key point is that $C_{0,t}$ or $C_{0,t}^{-1}$ need never be computed explicitly; instead, all we need is to multiply by A and multiply and divide by $C_{0,t}$ or $C_{0,t}^{-1}$, whichever is easiest (by "divide," we mean to solve equations of the form $C_{0,t}v = r$ for the unknown vector v and known vector r). The SVD step requires $O((k_{t-1}+b_t)^2 d)$ time, where k_{t-1} is the order of the perturbation (effective rank) at timestep $t-1$, and b_t is the number of measurements taken at timestep b_t. All the other steps involve $O(k_t)$ matrix-vector multiplications or divisions by $C_{0,t}$ or A. Thus, if $K(d)$ denotes the cost of such a single matrix-vector operation, the computational complexity of each low-rank update is approximately $O(k_t^2 d + k_t K(d))$. In many cases of interest (see below) $K(d) = o(d^2)$, and therefore the low-rank method is significantly faster than the standard Kalman recursion for large d. The algorithm is summarized below (Alg. 1).

Algorithm 1 Fast Kalman filtering algorithm

$L_1 = C_{0,1}B_1^T, \quad \Sigma_1 = (W_1 + B_1 C_{0,1} B_1^T)^{-1}$ (cost $O(b_1^3 + b_1 K(d))$)
$\tilde{C}_1 = C_{0,1} - L_1 \Sigma_1 L_1^T$
$\tilde{\mu}_1 = L_1 \Sigma_1^{-1} y_1$
for $t = 2$ to T **do**
  $C_{0,t} = A C_{0,t-1} A^T + V$
  $\Phi_t = C_{0,t}^{-1} A L_{t-1}, \quad \Delta_t = (\Sigma_{t-1}^{-1} - L_{t-1}^T A^T C_{0,t}^{-1} A L_{t-1})^{-1}$ (cost $O(k_{t-1}^3 + k_{t-1} K(d))$)
  $O_t = [\Phi_t \;\; B_t], \quad Q_t = \mathrm{blkdiag}\{\Delta_t, W_t^{-1}\}$
  $[\hat{L}_t, \hat{\Sigma}_t^{1/2}] = \mathrm{svd}(C_{0,t} O_t (Q_t^{-1} + O_t^T C_{0,t} O_t)^{-1/2})$ (cost $O((b_t + k_{t-1})^2 d)$)
  Truncate $\hat{L}_t$ and $\hat{\Sigma}_t$ to L_t and Σ_t. (effective rank $k_t \leq b_t + k_{t-1} \ll d$)
  $\tilde{C}_t = C_{0,t} - L_t \Sigma_t L_t^T$
  $\tilde{P}_t = C_{0,t} - A L_{t-1} \Sigma_{t-1} L_{t-1}^T A^T$ (cost $O(k_{t-1} K(d))$)
  $\tilde{\mu}_t = A\tilde{\mu}_{t-1} + \tilde{P}_t B_t^T (W_t + B_t \tilde{P}_t B_t^T)^{-1} (y_t - B_t A \tilde{\mu}_{t-1})$ (cost $O(b_t^3 + b_t K(d))$)

We close this section by noting that the posterior marginal variance difference $[\tilde{C}_t - C_{0,t}]_{ii}$ can be computed in $O(k_t d)$ time, since computing the diagonal of $\tilde{C}_t - C_{0,t}$ just requires us to sum the squared elements of $\Sigma_t^{1/2} L_t$. This quantity is useful in a number of contexts (Huggins and Paninski, 2012). In addition, the method can be sped up significantly in the special case that B and W are time-invariant (or vary in a periodic manner): in this case, $\tilde{C}_t$ will converge to a limit as an approximate solution of the corresponding Riccati equation, (or $\tilde{C}_t$ will also be periodic) and we can stop recomputing L_t and Σ_t on every time step.

3.2 Examples for which the proposed fast methods are applicable

There are many examples where the required manipulations with A, V and C_0 are relatively easy. The following list is certainly non-exhaustive. First, if A or its inverse is banded (or tree-banded, in the sense that $A_{ij} \neq 0$ only if i and j are neighbors on a tree) then so is C_0^{-1}, and multiplying and dividing by C_0 costs just $O(d)$ time and space per timestep (Rue and Held, 2005; Davis, 2006).

Second, in many cases A is defined in terms of a partial differential operator. (The example discussed in Paninski (2010) falls in this category; the voltage evolution on the dendritic tree is governed by a cable equation.) A in these cases is typically sparse and has a specialized local structure; multiplication by A and C_0^{-1} requires just $O(d)$ time and space. In many of these cases multigrid methods or other specialized PDE solvers can be used to divide by C_0^{-1} in $O(d)$ time and space (Briggs et al., 2000). As one specific example, multigrid methods are well-established in electroencephalographic (EEG) and magnetoencephalographic (MEG) analysis (Wolters, 2007; Lew et al., 2009), and therefore could potentially be utilized to significantly speed up the Kalman-based analyses described in Long et al. (2006); Galka et al. (2008); Freestone et al. (2011).

Third, A will have a Toeplitz (or block-Toeplitz) structure in many physical settings, e.g. whenever the state variable x_t has a spatial structure and the dynamics are spatially-invariant in some sense. Multiplication by A and C_0^{-1} via the fast Fourier transform (FFT) requires just $O(d \log d)$ time and space in these cases (Press et al., 1992). Similarly, division by C_0^{-1} can be performed via preconditioned conjugate gradient descent, which in many cases again requires $O(d \log d)$ time and space (Chan and Ng, 1996). Of course, if A is circulant then FFT methods may be employed directly to multiply and divide by C_0 with cost $O(d \log d)$.

Finally, in all of these cases, block or Kronecker structure in A may be exploited easily, since the transpose and product involved in the construction of C_0 will preserve this structure.

3.3 Analysis of the effective rank

As discussed above, the complexity of each iteration is $O(k_t^2 d + k_t K(d))$, where k_t is the effective rank of the perturbation to $C_{0,t}$ at time t. In this section we formalize the notion

of the effective rank and present some simple bounds that provide some insight into the efficiency of our algorithm. A more detailed treatment can be found in appendix B.

Definition 3.1. *Let U be a matrix and θ a constant with $0 \leq \theta \leq 1$. The effective rank of U at threshold θ, $z_\theta(U)$, is defined as the minimum integer k, such that there exists a matrix X with* $\operatorname{rank}(X) = k$ *and*

$$\|X - U\|_F^2 \leq (1-\theta)\|U\|_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

Based on the above definition, the number of singular values k_t (15) in the fast Kalman recursion can be expressed as $k_t = z_\theta(G_t^{1/2})$, with G_t defined as

$$G_t = C_{0,t} O_t (Q_t^{-1} + O_t^T C_{0,t} O_t)^{-1} O_t^T C_{0,t}. \tag{16}$$

To estimate the complexity of the algorithm, we need to characterize $z_\theta(G_t^{1/2})$. However, this is challenging since G_t is obtained from a series of successive low-rank approximations. From (12), G_t corresponds to the perturbing term of the approximate covariance $\tilde{C}_t$. We instead analyze the effective rank of the perturbing term of the exact covariance C_t as given in the following proposition (proved in appendix B).

Proposition 3.2. *The covariance matrices C_t can be written recursively as*

$$C_t = C_{0,t} - C_{0,t} U_t^T Z_t^{-1} U_t C_{0,t} \tag{17}$$

where

$$Z_t = F_t^{-1} + U_t C_{0,t} U_t^T, \tag{18}$$

and the matrices U_t and F_t are defined recursively as

$$\begin{aligned} U_t &= \begin{bmatrix} B_t \\ U_{t-1} C_{0,t-1} A^T C_{0,t}^{-1} \end{bmatrix}, \quad U_1 = B_1 \\ F_t^{-1} &= \begin{bmatrix} W & 0 \\ 0 & F_{t-1}^{-1} + U_{t-1}(C_{0,t-1} + C_{0,t-1} A^T C_{0,t}^{-1} A C_{0,t-1}) U_{t-1}^T \end{bmatrix}, \quad F_1 = W^{-1}. \end{aligned} \tag{19}$$

Since $\tilde{C}_t$ is an approximation of C_t we expect that for at least high threshold θ we have

$$z_\theta(G_t^{1/2}) \approx z_\theta(Z_t^{-1/2} U_t C_{0,t}). \tag{20}$$

Here we analyze the effective rank of the matrices $U_t C_{0,t}$. In the appendix we analyze the effective rank of $Z_t^{-1/2} U_t C_{0,t}$ and also provide a heuristic method for estimating the actual effective rank of $G_t^{-1/2}$. Our analysis and simulations show that $z_\theta(G_t^{1/2}) \leq z_\theta(Z_t^{-1/2} U_t C_{0,t})$, with equality when $\theta \uparrow 1$; this is unsurprising, since $Z_t^{-1/2} U_t C_{0,t}$ corresponds to the full perturbation in C_t away from $C_{0,t}$, while $G_t^{1/2}$ is an approximation of this perturbation.

For large t, the prior covariance $C_{0,t}$ converges to the equilibrium covariance C_0. Therefore, under the assumption that A is normal and commutes with V, we can make the approximation $C_{0,t} \approx C_0 = V(I - AA^T)^{-1}$ and the recursion of (19) can be rewritten as

$$U_t \approx [B_t^T \quad AU_{t-1}^T]^T, \quad F_t^{-1} \approx \text{blkdiag}\{W, F_{t-1}^{-1} + U_{t-1} V U_{t-1}^T\}. \tag{21}$$

If b_t is the number of measurements taken at time t, then the matrices U_t, F_t have dimensions $\left[\sum_{l=1}^t b_t, d\right]$ and $\left[\sum_{l=1}^t b_t, \sum_{l=1}^t b_t\right]$ respectively. However we see that at each timestep t, all the blocks of U_t that correspond to times $1, \ldots, t-1$ are multiplied with A^T. Therefore at time t, the effect of the measurements from time $t - s$ will be limited and thus past measurements are eventually "forgotten," as discussed above.

To characterize the effective rank in a specific tractable setting, suppose that each B_t is a $b \times d$ i.i.d. random matrix where each entry has zero mean and variance $1/d$. Let $[U_t]_{1:l}$ be the matrix that consists of the first l blocks of U_t, and define k_U as the minimum number of blocks required to capture a θ fraction of the expected energy,

$$k_U = \arg\min_{l \in \mathbb{N}} \{l : \mathbb{E}\|[U_t]_{1:l} C_{0,t}\|_F^2 \geq \theta \mathbb{E}\|U_t C_{0,t}\|_F^2\}. \tag{22}$$

Using (21) the $(m+1)$-th block of $U_t C_{0,t}$ is approximately $B_{t-m}(A^T)^m C_0$. Using the identity

$\|X\|_F^2 = \mathrm{Tr}(X^T X)$, we have that the expected energy of the $(m+1)$-th block is equal to

$$\begin{aligned}\mathbb{E}\|[[U_t]_{m+1} C_{0,t}\|_F^2 \approx \mathbb{E}\|B_{t-m}(A^T)^m C_0\|_F^2 &= \mathbb{E}\left(\mathrm{Tr}[B_{t-m}(A^T)^m C_0^2 A^m B_{t-m}^T]\right) \\ &= \frac{b}{d}\mathrm{Tr}[(A^T)^m C_0^2 A^m] = \frac{b}{d}\sum_{i=1}^{d} c_i^2 \alpha_i^{2m},\end{aligned} \tag{23}$$

where $\alpha_1 \geq \ldots \geq \alpha_d$ are the singular values of A and $c_1, \ldots, c_d$ are the corresponding singular values of C_0. Plugging into (22) and summing over the blocks, assuming $t \to \infty$, we get

$$k_U = \arg\min_{l \in \mathbb{N}} \left\{ \sum_{i=1}^{d} c_i^2 \frac{1-\alpha_i^{2l}}{1-\alpha_i^2} \geq \theta \sum_{i=1}^{d} c_i^2 \frac{1}{1-\alpha_i^2} \right\} \overset{(*)}{\leq} \arg\min_{l \in \mathbb{N}} \left\{ 1 - \alpha_1^{2l} \geq \theta \right\} = \left\lceil \frac{\log(1-\theta)}{2\log(\|A\|)} \right\rceil, \tag{24}$$

where $(*)$ follows since $1 - \alpha_1^{2l} \geq \theta \Rightarrow 1 - \alpha_i^{2l} \geq \theta$ for all the other singular values α_i and $\lceil x \rceil$ denotes the least integer greater or equal than x. The bound of (24) becomes tight if $c_1 \gg c_2, \ldots, c_d$ or if all the singular values of A are approximately equal, i.e., A becomes proportional to the identity matrix. Note that the bound of (24) covers only the expected case and is probabilistic. It is possible to derive concentration inequalities on the probability that the bound does not hold, but for our purposes it suffices to state that the bound is expected to hold with high probability. Therefore with high probability the first bk_U rows of $U_t C_{0,t}$ capture a θ fraction of its energy and

$$z_\theta(U_t C_{0,t}) \leq b k_U. \tag{25}$$

In other words, we expect that the algorithm will lead to high computational gains if $d \gg bk_U$. Note that the derived bound grows only mildly with θ and is also independent of d. Therefore for large d we see that the total cost of the fast Kalman filtering algorithm becomes at most $O((k_U^2 d + k_U K(d))T)$. In appendix B we argue that a tighter bound for $z_\theta(G^{1/2})$ can be derived by taking into account the recursive nature of the thresholding procedure. More specifically, we argue that

$$z_\theta(G_t^{1/2}) \leq b \arg\min_{l \in \mathbb{N}} \{\mathbb{E}\|[U_t]_{1:l} C_{0,t}\|_F^2 \geq \theta \mathbb{E}\|[U_t]_{1:l+1} C_{0,t}\|_F^2\} \leq b \left\lceil \frac{\log(1-\theta) - \log(1-\|A\|^2\theta)}{2\log(\|A\|)} \right\rceil,$$

which provides a significantly tighter bound. Moreover, we examine the effective rank of $Z_t^{-1/2}U_tC_{0,t}$ and show that $z_\theta(Z_t^{-1/2}U_tC_{0,t}) \leq z_\theta(U_tC_{0,t})$ with equality holding in the limiting case where the noise power becomes infinite, i.e., in the low-SNR regime. Finally, we derive another heuristic bound on $z_\theta(G_t^{1/2})$, based on $z_\theta(Z_t^{-1/2}U_tC_{0,t})$ and present a simulation example that supports the several bounds.

4 Full forward-backward smoothing

So far we have focused on the forward problem of computing estimates of x_t given the data available up to time t. To incorporate all of the available information $Y_{1:T}$ (not just $Y_{1:t}$), we need to perform a backward recursion. Two methods are available: we can use the Kalman backward smoother (Shumway and Stoffer, 2006), which provides both $\mathbb{E}(x_t|Y_{1:T})$ and $\mathrm{Cov}(x_t|Y_{1:T})$, or a version of the Thomas recursion for solving block-tridiagonal systems.

Both recursions can be adapted to our low-rank setting. In the Kalman backward smoother we can approximate $\mathrm{Cov}(x_t|Y_{1:T}) \approx C_0 - L_t^s\Sigma_t^s(L_t^s)^T$, for an appropriately chosen low-rank matrix $L_t^s\Sigma_t^s(L_t^s)^T$, which can be updated efficiently using methods similar to those we have described here for the forward low-rank approximation $C_0 - L_t\Sigma_tL_t^T$; see Huggins and Paninski (2012) for full details. Here we focus on deriving an efficient low-rank block-Thomas (LRBT) approach, and examining its convergence characteristics.

4.1 The low-rank block-Thomas algorithm

First we recall that the output of Kalman filter-smoother, $s_t = \mathbb{E}(x_t|Y_{1:T})$, may be written as the solution to a block-tridiagonal linear system (Fahrmeir and Kaufmann, 1991; Paninski et al., 2010), i.e.

$$H\boldsymbol{s} = -\nabla\big|_{\mathbf{x}=0}, \tag{26}$$

where $\nabla\big|_{\mathbf{x}=0}$, H denote the gradient evaluated at zero and the Hessian of the negative log-posterior $f = -\log p(X|Y_{1:T})$ with respect to X, because f is simply a quadratic function

in this linear-Gaussian setting. We have

$$f \propto \frac{1}{2}\sum_{t=1}^{T}(y_t - B_t x_t)^T W_t^{-1}(y_t - B_t x_t) + \frac{1}{2}\sum_{t=1}^{T-1}(x_{t+1} - Ax_t)^T V^{-1}(x_{t+1} - Ax_t) + \frac{1}{2}x_1^T V_0^{-1} x_1$$

$$\nabla_t \triangleq \frac{\partial f}{\partial x_t} = -B_t^T W_t^{-1}(y_t - B_t x_t) - A^T V^{-1}(x_{t+1} - Ax_t) + V^{-1}(x_t - Ax_{t-1}) \tag{27}$$

$$H = \begin{bmatrix} D_1 + B_1^T W_1^{-1} B_1 & -E_1 & 0 & \dots & 0 \\ -E_1^T & D_2 + B_2^T W_2^{-1} B_2 & -E_2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & -E_{T-1}^T & D_T + B_T^T W_T^{-1} B_T \end{bmatrix},$$

with $$D_t = \begin{cases} V_0^{-1} + A^T V^{-1} A, & t = 1 \\ V^{-1} + A^T V^{-1} A, & 1 < t < T \\ V^{-1}, & t = T \end{cases} \quad \text{and} \quad E_t = A^T V^{-1},\ 1 \le t \le T.$$

The solution of (26), which corresponds to the full forward-backward smoothing can be given by the classic block-Thomas (BT) algorithm (Isaacson and Keller, 1994), which we repeat here for completeness (Alg. 2).

Algorithm 2 Classic Block-Thomas Algorithm (computes $\mathbf{s} = -H^{-1}\nabla$)

$M_1 = D_1 + B_1^T W_1^{-1} B_1, \quad \Gamma_1 = M_1^{-1} E_1$ (cost $O(d^3)$)
$\boldsymbol{q}_1 = -M_1^{-1}\nabla_1$ (cost $O(d^2)$)
for $i = 2$ to T **do**
  $M_t = D_t + B_t^T W_t^{-1} B_t - E_{t-1} M_{t-1}^{-1} E_{t-1}^T, \quad \Gamma_t = M_t^{-1} E_t$ (cost $O(d^3)$)
  $\boldsymbol{q}_t = -M_t^{-1}(\nabla_t - E_{t-1}^T \boldsymbol{q}_{t-1})$ (cost $O(d^2)$)
$\mathbf{s}_T = \boldsymbol{q}_T$
for $t = T - 1$ to 1 **do**
  $\mathbf{s}_t = \boldsymbol{q}_t + \Gamma_t \mathbf{s}_{t+1}$ (cost $O(d^2)$)

The expensive part in the BT algorithm is the multiplication and division with the matrices M_t, which correspond to a modified version of the inverse covariance matrices C_t^{-1}. In the case where $B_s = 0$ for all $s \le t$, we have that $M_t = \tilde{D}_t$ where the matrices $\tilde{D}_t$ correspond to a modified version of the inverse equilibrium covariance $C_{0,t}^{-1}$ (in fact for $t = T$

we have that $\tilde{D}_T = C_{0,T}^{-1}$ and $M_T = C_T^{-1}$) and are defined recursively as

$$\tilde{D}_t = D_t - E_{t-1}\tilde{D}_{t-1}^{-1}E_{t-1}^T, \text{ with } \tilde{D}_1 = D_1. \tag{28}$$

Using a similar argument as in the fast Kalman filtering case (see (10)), to derive a similar fast algorithm in the case where $B_s \neq 0$, we want to approximate the matrices M_t^{-1} as

$$M_t^{-1} \approx \tilde{M}_t^{-1} = \tilde{D}_t^{-1} - L_t\Sigma_t L_t^T, \tag{29}$$

where $L_t\Sigma_t L_t^T$ is a suitable low rank matrix. To gain some insight into this approximation, suppose that (29) holds at time $t-1$. Then following the BT recursion we can define the matrices $\hat{M}_t$ as

$$\begin{aligned}
\hat{M}_t &= D_t + B_t^T W_t^{-1} B_t - E_{t-1}\tilde{M}_{t-1}^{-1}E_{t-1}^T \\
&\overset{(28)}{=} \tilde{D}_t + B_t^T W_t^{-1} B_t + E_{t-1}L_{t-1}\Sigma_{t-1}L_{t-1}^T E_{t-1}^T = \tilde{D}_t + O_t Q_t O_t^T \Rightarrow \\
\hat{M}_t^{-1} &\overset{(w)}{=} \tilde{D}_t^{-1} - \tilde{D}_t^{-1}O_t(Q_t^{-1} + O_t^T\tilde{D}_t^{-1}O_t)^{-1}O_t^T\tilde{D}_t^{-1},
\end{aligned} \tag{30}$$

with $$O_t = [B_t^T \quad E_{t-1}L_{t-1}], \quad Q_t = \text{blkdiag}\{W_t^{-1}, \Sigma_{t-1}\}. \tag{31}$$

L_t, Σ_t can be derived by taking the partial SVD of the term $\tilde{D}_t^{-1}O_t(Q_t^{-1} + O_t^T\tilde{D}_t^{-1}O_t)^{-1/2}$ and keeping only the singular values/vectors that express a θ fraction of the energy[2]. This results in the approximation of (29). Note again the resemblance of (30) and (31) with (12) and (13) respectively, for the fast KF case. The resulting LRBT algorithm is summarized below (Alg. 3). As in the fast Kalman filtering case, the use of this fast low-rank approach will lead to substantial gains if the cost $K(d)$ of matrix-vector multiplication or division with the matrices $\tilde{D}_t^{-1}$ and E_t, satisfies $K(d) = o(d^2)$ for d large. (Again, we assume that fast methods are available for updating $\tilde{D}_t$; for example, in the case that A is normal and commutes with V, if we choose the stationary initial condition $C_{0,t} = C_0$, then updating $\tilde{D}_t$ turns out to be trivial, as can be demonstrated with a simple direct computation.)

[2]Note that in a slight abuse of notation, we will recycle the names of some matrices (e.g., O_t and Q_t) that play a similar role in the LRBT approach as in the fast Kalman method described in the previous sections.

Algorithm 3 Low-Rank Block-Thomas Algorithm

$\tilde{D}_1 = D_1, L_1 = D_1^{-1} B_1^T$ (cost $O(b_1 d)$, $k_1 = b_1$)
$\Sigma_1 = (W_1 + B_1 D_1^{-1} B_1^T)^{-1}$ (cost $O(b_1^3)$)
$\tilde{\boldsymbol{q}}_1 = (-D_1^{-1} + L_1 \Sigma_1 L_1^T)\nabla_1 \quad (= -\tilde{M}_1^{-1}\nabla_1)$ (cost $O(b_1 K(d))$)
for $t = 2$ to T **do**
  $\tilde{D}_t = D_t - E_{t-1}\tilde{D}_{t-1}^{-1} E_{t-1}^T$
  $O_t = [B_t^T \quad E_{t-1} L_{t-1}], \quad Q_t = \text{blkdiag}\{W_t^{-1}, \Sigma_{t-1}\}$
  $[\hat{L}_t, \hat{\Sigma}_t^{1/2}] = \text{svd}(\tilde{D}_t^{-1} O_t (Q_t^{-1} + O_t^T \tilde{D}_t^{-1} O_t)^{-1/2})$ (cost $O((b_t + k_{t-1})^2 K(d))$)
  Truncate $\hat{L}_t$ and $\hat{\Sigma}_t$ to L_t and Σ_t. (effective rank $k_t \leq b_t + k_{t-1} \ll d$)
  $\tilde{\boldsymbol{q}}_t = -(\tilde{D}_t^{-1} - L_t \Sigma_t L_t^T)(\nabla_t - E_{t-1}^T \tilde{\boldsymbol{q}}_{t-1}) \quad (= -\tilde{M}_t^{-1}(\nabla_t - E_{t-1}^T \tilde{\boldsymbol{q}}_{t-1}))$ (cost $O(k_t K(d))$)
$\tilde{\mathbf{s}}_T = \tilde{\boldsymbol{q}}_T$
for $i = T - 1$ to 1 **do**
  $\tilde{\mathbf{s}}_t = \tilde{\boldsymbol{q}}_t + (\tilde{D}_t^{-1} E_t^T - L_t \Sigma_t L_t^T E_t^T)\tilde{\mathbf{s}}_{t+1} \quad (= \tilde{\boldsymbol{q}}_t + \tilde{\Gamma}_t \tilde{\mathbf{s}}_{t+1})$ (cost $O(k_t K(d))$)

The BT algorithm provides the smoothed mean $\mathbb{E}(x_t | Y_{1:T})$ by solving (26). The Hessian H can also be used to obtain the smoothed covariance $C_t^s = \text{Cov}(x_t | Y_{1:T})$ since C_t^s is equal to the t-th diagonal block of H^{-1}. We can obtain the diagonal blocks of H^{-1} in $O(d^3 T)$ time and $O(d^2 T)$ space using the algorithm of Rybicki and Hummer (1991) for the fast solution for the diagonal elements of the inverse of a tridiagonal matrix. In appendix D we present this algorithm and show how we can modify it in a similar fashion to the LRBT algorithm, to obtain estimates of C_t^s in just $O(K(d)T)$ time and $O(dT)$ space.

4.2 Analysis of the LRBT algorithm

The forward-backward procedure allows us to analyze the error of our LRBT algorithm. In appendix C we prove that although the algorithm involves an approximation at every step the error does not accumulate, and thus remains of the order $O(1 - \theta)$.

Theorem 4.1. *The solution* $\tilde{\mathbf{s}}$ *of the LRBT algorithm can be written as*

$$\tilde{\mathbf{s}} = -\tilde{H}^{-1} \nabla\big|_{\mathbf{x}=0} \tag{32}$$

with

$$\tilde{H} = \begin{bmatrix} \tilde{M}_1 & -E_1 & \dots & 0 \\ -E_1^T & \tilde{M}_2 + E_1 \tilde{M}_1^{-1} E_1^T & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & -E_{T-1}^T & \tilde{M}_T + E_{T-1} \tilde{M}_{T-1}^{-1} E_{T-1} \end{bmatrix}. \tag{33}$$

Moreover, $\tilde{H}$ is positive definite and, under the assumption $\|A\| < 1$, it approximates the true Hessian H, defined in (27)*, as*

$$\|\tilde{H} - H\| = O(1 - \theta). \tag{34}$$

Theorem 4.1 has several useful implications. First, it establishes that the LRBT smoother approximation error is also of order $O(1 - \theta)$, since

$$\|\tilde{s} - s\| = \|(\tilde{H}^{-1} - H^{-1})\nabla\big|_{\mathbf{x}=0}\| \leq \|(\tilde{H}^{-1} - H^{-1})\|\|\nabla\big|_{\mathbf{x}=0}\|. \tag{35}$$

Moreover, since $\tilde{H}$ is positive definite, it follows that the LRBT performs the steepest descent step for the quadratic norm $\|x\|_{\tilde{H}} = (x^T \tilde{H} x)^{1/2}$. Therefore when applied as a search direction in an iterative algorithm, it converges to the solution of (26). The convergence rate is linear, but can be made arbitrarily fast since it is controlled by the threshold θ. In fact if f^* is the minimum value of the negative log-likelihood function f (27), then we can show that $f(\mathbf{s}_n) - f^* \propto \gamma_\theta^n$, with $\gamma_\theta = O(1 - \theta)$. A short discussion can be found in appendix C. $\tilde{H}$ can also be used as an effective preconditioner for other iterative methods, e.g. conjugate gradients that in general lead to faster convergence than plain steepest descent. Note that the condition $\|A\| < 1$ is critical for the $O(1 - \theta)$ approximation error, since it guarantees that the matrix M_t that we approximate stays finite. This issue is discussed in somewhat more detail in appendix C (remark C.3). Finally, we note that the effective rank of the matrices involved in the LRBT algorithm has exactly the same scaling properties as in the fast KF case. The interested reader is referred to appendix B for more details.

5 Application to high-dimensional smoothing

Now for the main statistical examples we have in mind. In many statistical settings, the dynamics matrix A and noise covariance V are not directly defined; the analyst has some flexibility in choosing these matrices according to criteria including physical realism and computational tractability. Perhaps the simplest approach is to use a separable prior, de-

fined most easily as $A = aI$, $0 < a < 1$. Now $C_0 = (1 - a^2)^{-1}V$; thus it is clear that when it is easy to multiply and divide by V, we may apply the fast methods discussed above with no modifications. Note that in this case the prior covariance of the vector X is separable: $\mathrm{Cov}(X) = C_0 \otimes C_{AR}$, where $\otimes$ denotes the Kronecker product and C_{AR} denotes the covariance of the standardized one-dimensional autoregressive AR(1) process, $x_{t+1} = ax_t + \sqrt{1 - a^2}\epsilon_t, \epsilon_t \sim \mathcal{N}(0, 1)$. However the posterior covariance $\mathrm{Cov}(X|Y)$ is not separable in general, which complicates exact inference.

It is straightforward to construct more interesting nonseparable examples. For example, in many cases we may choose a basis so that V and A are diagonal and the transformation back to the "standard" basis is fast. Examples include the discrete Fourier basis, common spline bases and wavelet bases. Now the interpretation is that each basis element is endowed with an AR(1) prior: the (i, i)-th element of A defines the temporal autocorrelation width of the i-th process, while the elements of the diagonal matrix $(I - A^2)^{-1}V$ set the processes' prior variance (and therefore $(I - A^2)^{-1}V$ expressed in the "standard" basis sets the prior covariance C_0). The difficulty in applying the standard Kalman recursion in this setting is that if B is not also diagonal in this representation, then direct implementations of the Kalman filter require $O(d^3)$ time per timestep, since C_t does not remain diagonal in general. Nonetheless, the fast low-rank smoother may be applied in a straightforward manner in this setting: computing $\mathbb{E}(x_t|Y)$ and $\mathrm{Cov}(x_t|Y)$ requires $O(k_t^2 d)$ time, to which we add the time necessary to transform back into the standard basis.

A further speedup is possible in this diagonal case, if the observation matrices B_t are sparse; i.e., if each observation y_t only provides information about a few elements of the state vector x_t. This setting arises frequently in environmental applications, for example, where just a few sampling stations are often available to take spatially-localized samples of large spatiotemporal processes of interest (Stroud et al., 2001). Another example, from neuroscience, will be discussed in the following section. If I_t denotes the set of indices for which B_s is nonzero for $s \leq t$, then it is easy to show that the forward covariance C_t matrix need only be evaluated on the $|I_t| \times |I_t|$ submatrix indexed by I_t; if i or j are not in I_t, then $[C_t]_{ij} = [C_0]_{ij}$. Thus, we need only update the low-rank matrix L_t at the indices I_t, reducing the computational complexity of each update from $O(k_t^2 d)$ to $O(k_t^2 |I_t|)$. Clearly,

with each new update at time t, we will add some elements to I_t, but we can also discard some elements as we go because our low-rank updates will effectively "forget" information as time progresses, as discussed above. (In particular, the indices for which the recent observations provide no information will eventually be dropped.) Thus in practice $|I_t|$ often remains much smaller than d, leading to a significant speedup.

5.1 Two neuroscience examples

To make these ideas more concrete, we now examine two examples from neuroscience. For our first example we consider neurons in the rodent hippocampal brain region; many of these neurons respond selectively depending on the animal's current location. This spatial dependency can be summarized in terms of a "place field" $f(\vec{x})$, where $f(\vec{x})$ is the expected response of the neuron (quantified by the number of action potentials emitted by the neuron in a fixed time interval), given that the animal is located at position $\vec{x}$. It is known that these place fields can in some cases change with time; in this case we might replace $f(\vec{x})$ with $f(\vec{x}, t)$. These time-varying place fields $f(\vec{x}, t)$ are often represented as a sum of some fixed spatial basis functions (Brown et al., 2001; Frank et al., 2002; Czanner et al., 2008), weighted by some appropriate weights which are to be inferred:

$$f(\vec{x}, t) = \sum_i q_{it} f_i(\vec{x}). \tag{36}$$

For example, the basis $\{f_i(\vec{x})\}$ could consist of spline functions defined on the spatial variable $\vec{x}$. Now we place a prior on how the weights q_{it} evolve with time. In the simplest case, q_{it} could evolve according to independent AR(1) processes; as emphasized above, this means that the dynamics matrix A is diagonal. Now the observation model in this setting may be taken to be $y_t = f(\vec{x}_t, t) + \eta_t$, with η_t denoting an i.i.d. Gaussian noise source, or we can use a slightly more accurate Poisson model, $y_t \sim Poiss\{\exp(f(\vec{x}_t, t))\}$, where in either case $\vec{x}_t$ represents the (known) location of the animal as a function of time t, and $Poiss\{\lambda_t\}$ denotes a Poisson process with rate λ_t. The observation matrix B_t is just a d-dimensional vector, $B_{it} = f_i(\vec{x}_t)$, if we use d basis vectors to represent the place field f. Computing B_t requires

at most $O(d)$ time; if the basis functions f_i have compact support, then B_t will be sparse (i.e., computable in $O(1)$ time), and we can employ the speedup based on the sparse index vector I_t described above. More detailed models are possible, of course (Czanner et al., 2008; Rahnama Rad and Paninski, 2010), but this basic formulation is sufficient to illustrate the key points here.

A second example comes from sensory systems neuroscience. The activity of a neuron in a sensory brain region depends on the stimulus which is presented to the animal. The activity of a visual neuron, for example, is typically discussed using the notion of a "receptive field," which summarizes the expected response of the neuron as a function of the visual stimulus presented to the eye (Dayan and Abbott, 2001). We can use a similar model structure to capture these stimulus-dependent responses; for example, we might model $y_t = s_t^T f^t + \eta_t$ in the Gaussian case, or $y_t \sim Poiss\{\exp(s_t^T f^t)\}$ in the Poisson case, where s_t is the sensory stimulus presented to the neuron at time t, $s_t^T f^t = \sum_x s(\vec{x}, t) f(\vec{x}, t)$ denotes the linear projection of the stimulus s_t onto the receptive field f^t at time t, and $f(\vec{x}, t)$ is proportional to the expectation of y_t given that a light of intensity $s(\vec{x}, t)$ was projected onto the retina at location $\vec{x}$. As indicated by the notation f^t, these receptive fields can in many cases themselves vary with time, and to capture this temporal dependence it is common to use a weighted sum of basis functions model, as in equation (36). This implies that the observation matrix B_t can be written as $B_t = s_t^T F$, where the i-th column of the basis matrix F is given by f_i. If the basis functions f_i are Fourier or wavelet functions, then the matrix-vector multiplications $s_t^T F$ can be performed in $O(d \log d)$ time per timestep; if f_i are compactly supported, F will be sparse, and computing $s_t^T F$ requires just $O(d)$ time.

Now in each of these settings the fast Kalman filter is easy to compute. In the case of Gaussian observation noise η_t we proceed exactly as described above, once the observation matrices B_t are defined; in the Poisson case we can employ well-known extensions of the Kalman filter described, for example, in (Fahrmeir and Kaufmann, 1991; Fahrmeir and Tutz, 1994; Brown et al., 1998; Paninski et al., 2010); see appendix A for details. In either case, the filtering requires $O(k_t^2 |I_t|)$ time for timestep t. When the filtering is complete (i.e., $\mathbb{E}(q_t|Y)$ has been computed for each desired t), we typically want to transform from the q_t space to represent $E(f|Y)$; again, if the basis functions f_i correspond to wavelet or

Fourier functions, this costs $O(d \log d)$ time per timestep, or $O(d)$ time if the f_i functions are compactly supported.

Figures 1 and 2 illustrate the output of the fast filter-smoother applied to simulated place field data. The spatial variable $\vec{x}$ is chosen to be one-dimensional here, for clarity. We chose the true place field $f(\vec{x}, t)$ to be a Gaussian bump (as a function of $\vec{x}$) whose mean varied sinusoidally in time but whose height and width were held constant (see the upper left panel of Fig. 1). The basis matrix F consisted of 50 equally-spaced bump functions with compact support (specifically, spatial Gaussians truncated at $\sigma \approx 4$, with each bump located one standard deviation σ apart from the next.) The dynamics coefficient a (in the diagonal dynamics matrix $A = aI$) was about 0.97, which corresponds to a temporal correlation time of $\tau = 30$ timesteps; the simulation shown in Fig. 1 lasted for $T = 1000$ timesteps. To explore the behavior of the filter in two regimes, we let $\vec{x}_t$ begin by sampling a wide range of locations (see Fig. 1 for $t < 200$ or so), but then settling down to a small spatial subset for larger values of t. We used the Gaussian noise model for y_t in this simulation with standard deviation 0.1.

We find that, as expected, the filter does a good job of tracking $f(\vec{x}, t)$ for locations $\vec{x}$ near the observation points $\vec{x}_t$, where the observations y_t carry a good deal of information, but far from $\vec{x}_t$ the filter defaults to its prior mean value, significantly underestimating $f(\vec{x}, t)$. The posterior uncertainty $V(f(\vec{x}, t)|Y) = \mathrm{diag}\{F \mathrm{Cov}(q_t|Y) F^T\}$ remains near the prior uncertainty $\mathrm{diag}\{F C_0 F^T\}$ in locations far from $\vec{x}_t$, as expected. Fig. 2 illustrates that the low-rank approximation works well in this setting, despite the fact that (at least for t sufficiently large) only a few singular values are retained in our low-rank approximation (c.f. Fig. 1, lower left panel). We set $\theta = 0.99$ in (15) for this simulation.

We have also applied the filter to real neuronal data, recorded from single neurons in the mouse hippocampal region by Dr. Pablo Jercog. In these experiments the mouse was exploring a two-dimensional cage, and so we estimated the firing rate surface $f(\vec{x}, t)$ as a function of time t and a two-dimensional spatial variable $\vec{x}$. The results are most easily viewed in movie form; see `http://www.stat.columbia.edu/~liam/research/abstracts/fast-Kalman-abs.html` for details.

Figure 3 illustrates an application of the fast filter-smoother to the second context de-

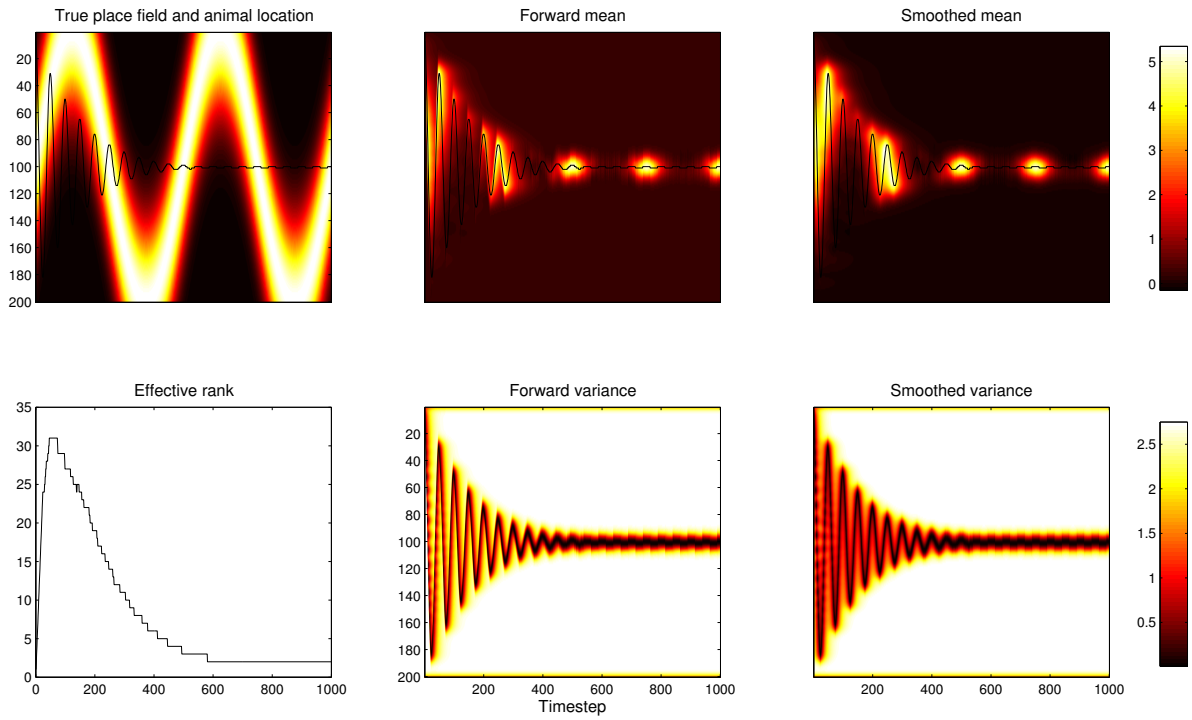

Figure 1: Output of the filter-smoother applied to simulated one-dimensional place field data. The superimposed black trace in all but the lower left panel indicates the simulated path $\vec{x}_t$ of the animal; $\vec{x}_t$ begins by sampling a wide range of locations for $t < 200$, but settles down to a small spatial subset for larger values of t. Upper left: true simulated place field $f(\vec{x}, t)$ is shown in color; $f(\vec{x}, t)$ has a Gaussian shape as a function of $\vec{x}$, and the center of this Gaussian varies sinusoidally as a function of time t. Top middle and right panels: estimated place fields, forward ($E(f(\vec{x}, t)|Y_{1:t})$) and forward-backward ($E(f(\vec{x}, t)|Y_{1:T})$), respectively. Here (in a slight abuse of notation) we use $E(f^t|Y)$ to denote the projected mean $FE(q_t|Y)$, where F is the basis matrix corresponding to the basis coefficients q. Note that the estimated place fields are accurate near the observed positions $\vec{x}_t$, but revert to the prior mean when no information is available. Bottom middle and right panels: marginal variance of the estimated place fields, forward ($V(f(\vec{x}, t)|Y_{1:t})$) and forward-backward ($V(f(\vec{x}, t)|Y_{1:T})$), respectively. Again, note that the filter output is most confident near $\vec{x}_t$. Lower left panel: effective rank of $C_0 - C_t^s$ as a function of t in the forward-backward smoother; the effective rank is largest when $\vec{x}_t$ samples many locations in a short time period.

scribed above. We simulated neuronal responses of the form $y_t = s_t^T f^t + \eta_t$, where the sensory stimulus s_t was taken to be a spatiotemporal Gaussian white noise process normalized to unit energy and the response noise η_t was also modeled as Gaussian and white, for simplicity with variance 0.1. As discussed above, we represented f^t as a time-varying weighted sum of fixed

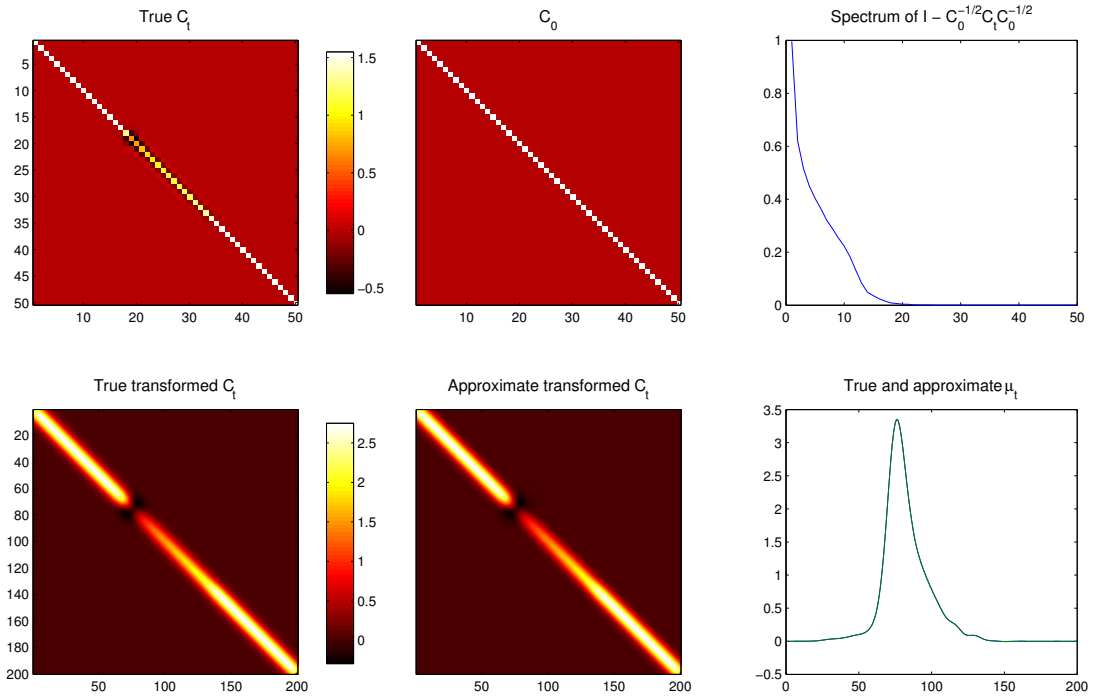

Figure 2: Justification of our low rank approximation in the place field example (section 5.1). Upper row: C_t is fairly close to C_0. Left: true C_t. Middle: C_0. Both C_0 and C_t are plotted on the same colorscale, to facilitate direct comparison. Right: eigenvalue spectrum of $I - C_0^{-1/2} C_t C_0^{-1/2}$; an approximation of rank about 20 seems to suffice here. Lower row: Comparison of the true vs. approximate projected covariance $F C_t F^T$ and mean $F \mu_t$ at $t = 200$. Left panel: true forward projected covariance $F C_t F^T$. Middle panel: approximate forward covariance $F(C_0 - L_t \Sigma_t L_t) F^T$. The maximal pointwise error between these two matrices less than 1%. Right panel: true (exact) and approximate means. The traces for the exact and approximate means are barely distinguishable.

basis functions f_i. In this case the basis F consisted of real-valued Fourier functions (sines and cosines), and multiplication by this basis matrix was implemented via the fast Fourier transform. As in the previous example, we chose the dynamics matrix A to be proportional to the identity; the effective autocorrelation time was $\tau = 50$ time steps here. The dynamics noise covariance V was diagonal (and therefore so was the prior covariance C_0), with the diagonal elements chosen so that the prior variance of the ω-th frequency basis coefficient falls off proportionally to ω^{-2}; this led to an effective smoothing prior. The dimensionality of this basis was chosen equal to $d = 2^9$. Figure 3 provides a one-dimensional example, where the full spatiotemporal output of the filter-smoother can be visualized directly. We have also applied

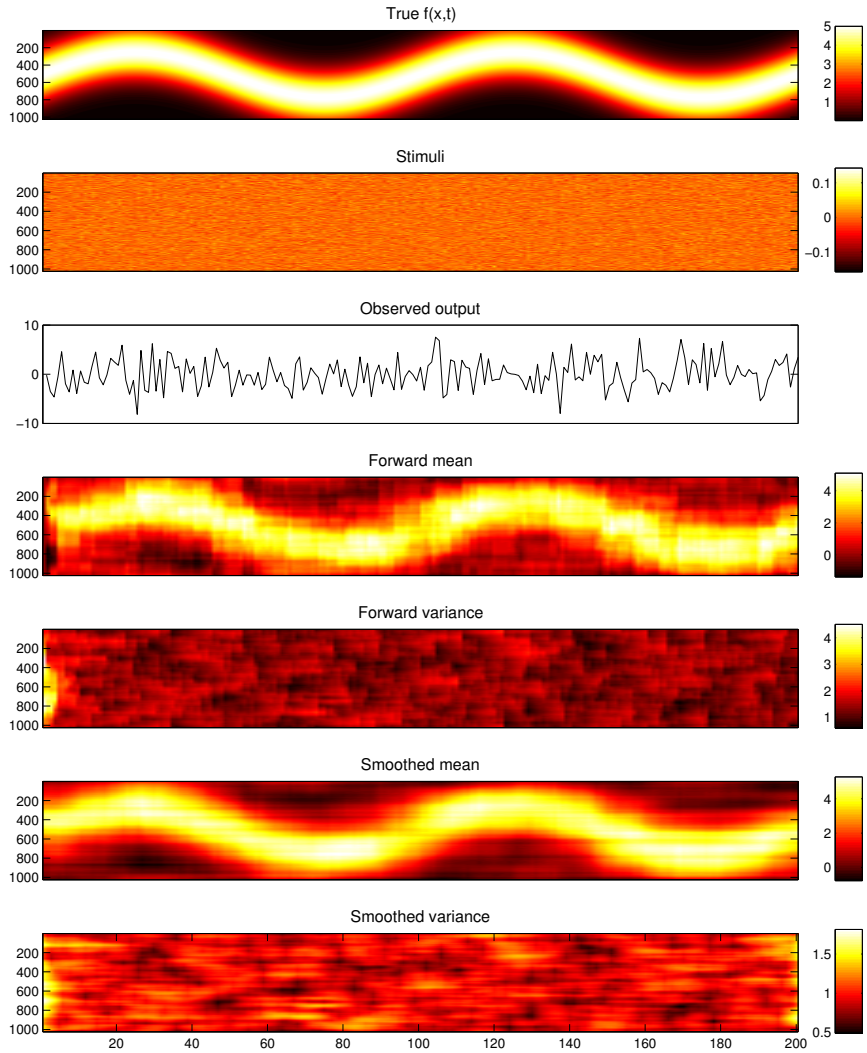

Figure 3: Tracking a time-varying one-dimensional receptive field (section 5.1). Top panel: the true receptive field f^t was chosen to be a spatial Gaussian bump whose center varied sinusoidally as a function of time t. Second panel: the stimulus s_t was chosen to be spatiotemporal white Gaussian noise. Third panel: simulated output observed according to the Gaussian model $y_t = s_t^T f^t + \eta_t$ with $\eta_t \sim \mathcal{N}(0, 0.1)$. Lower four panels: the forward filter mean $\mathbb{E}(f^t|Y_{1:t})$ and marginal variance $\mathrm{Var}(f^t(\vec{x})|Y_{1:t})$ and the full forward-backward smoother mean and marginal variance $\mathbb{E}(f^t|Y_{1:T})$ and marginal variance $\mathrm{Var}(f^t(\vec{x})|Y_{1:T})$. The dimension of the state variable f^t here was 2^{10}; inference required seconds on a standard laptop. Time units are arbitrary here; the assumed prior autocorrelation time was $\tau = 50$ timesteps, leading to $A = \alpha I$ with $\alpha \approx 0.98$, while the total length of the experiment was $T = 200$ timesteps.

the filter to higher-dimensional examples; a two-dimensional example movie is available at `http://www.stat.columbia.edu/~liam/research/abstracts/fast-Kalman-abs.html`.

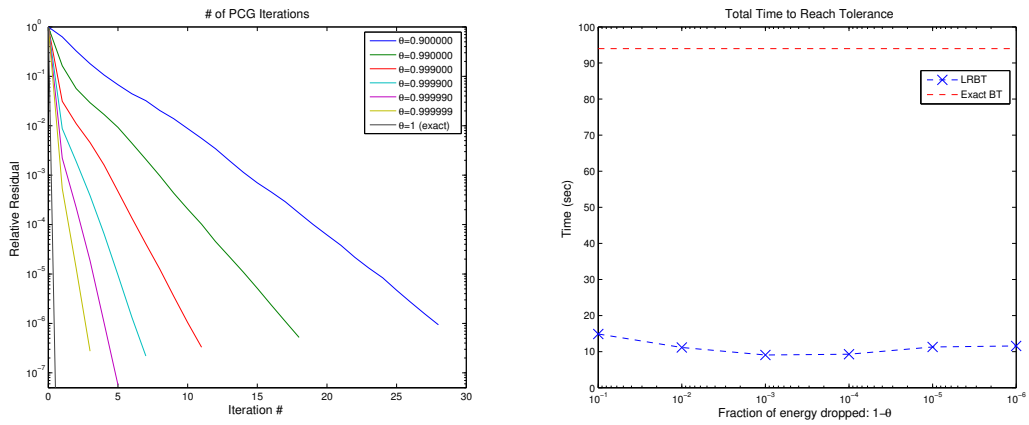

Figure 4: Solving a quadratic problem using preconditioned conjugate gradients with a LRBT preconditioner. Left: Relative residual error $\|Hs_k + \nabla_0\|/\|\nabla_0\|$, where H denotes the Hessian and s_k the search direction at the k-th iteration, and ∇_0 the gradient at 0, as a function of the number of iterations for various choices of the threshold θ. As θ approaches 1 the PCG method requires fewer iterations to converge to the exact solution within a small tolerance (10^{-6}). Right: Total computational cost to reach desired tolerance as a function of the threshold. The total cost for convergence is always smaller than the cost required by the exact BT algorithm. The computational gain (ratio of the required time at best θ vs $\theta = 1$) scales with d (data not shown).

As discussed in section 4, both the LRBT approach and the fast Kalman filter-smoother can be used to approximate the Newton direction for maximizing the posterior. Apart from approximating the exact solution, the LRBT algorithm can also be used as a preconditioner for solving for the exact direction $\mathbf{s} = -H^{-1}\nabla$. We investigated this is in Fig. 4, where we examine the convergence rates of the preconditioned conjugate gradient method (PCG) using the LRBT method as a preconditioner, for the example presented in Fig. 3. As expected, the number of iterations required for convergence drops as the threshold θ approaches 1 (see Fig. 4 left). Trivially, when $\theta = 1$, the exact BT algorithm is performed and we achieve convergence within one iteration. The cost per iteration increases slowly with the threshold (since the effective rank scales only as $O(|\log(1-\theta)|)$), so the overall cost to reach a desired tolerance is always significantly smaller than the cost required for the exact BT algorithm. The total cost for the PCG method reaches a global minimum for an intermediate value of

θ. This value depends on the behavior of the effective rank (and therefore on the dynamics A and the size of the observation matrices B_t), as well as the desired relative tolerance level (set to 10^{-6} here).

6 Discussion

We have presented methods for efficiently computing the Kalman filter and the block-Thomas (BT) smoother recursions in the few-observation setting. For the Kalman filter, the basic idea is that, when fast methods are available for multiplying and dividing by the prior equilibrium state covariance C_0, then the posterior state covariance C_t can be well-approximated by forming a low-rank perturbation of the prior C_0. A similar argument holds for the BT smoother. These low-rank perturbations, in turn, can be updated in an efficient recursive manner. We provided a theoretical analysis that characterizes the tradeoff between the computational cost of the algorithm (via the effective rank), and the accuracy of the low rank approximation. We also showed that our methods can be applied in an iterative fashion to reach any level of accuracy, at a reduced cost compared to standard exact methods.

There are a number of clear opportunities for application of this basic idea. Some exciting examples involve optimal control and online experimental design in high-dimensional settings; for instance, optimal online experimental design requires us to choose the observation matrix B_t adaptively, in real time, to minimize some objective function that expresses the posterior uncertainty in some sense (Fedorov, 1972; Lewi et al., 2009; Seeger and Nickisch, 2011). Our fast methods can be adapted to compute many of these objective functions, including those based on the posterior state entropy, or weighted sums of the marginal posterior state variance. See Huggins and Paninski (2012) for an application of these ideas to the neuronal dendritic setting.

The fast low-rank methods can also greatly facilitate the selection of hyperparameters in the smoothing setting: typically the data analyst will need to set the scale over which the data are smoothed, both temporally and spatially, and we would often like to do this in a data-dependent manner. There are a number of standard approaches for choosing hyperparameters (Hastie et al., 2001), including cross-validation, generalized cross-validation,

expectation-maximization, and maximum marginal likelihood or empirical Bayes methods. In all of these cases, it is clearly beneficial to be able to compute the estimate more rapidly for a variety of hyperparameter settings. In addition, the output of the filter-smoother is often a necessary ingredient in hyperparameter selection. For example, the standard expectation-maximization method of Shumway and Stoffer (2006) can be easily adapted to the low-rank setting: we have already discussed the computation of the sufficient statistics $\mathbb{E}(x_t|Y)$ and $\mathrm{Cov}(x_t|Y)$, and the remaining needed sufficient statistics $\mathbb{E}(x_t x_{t+1}^T|Y)$ follow easily. Similarly, a straightforward application of the low-rank determinant lemma allows us to efficiently compute the marginal log-likelihood $\log p(Y)$, via a simple adaptation of the standard forward recursion for the log-likelihood in the Kalman filter model (Rabiner, 1989).

We have seen that the prior covariance is especially easy to compute in the case that the dynamics matrix A is normal: here C_0 may be computed analytically, assuming the dynamics noise covariance V can be transformed via a convenient whitening transformation. A key direction for future work will be to extend these methods to the case that A is a non-normal matrix, a situation that arises quite frequently in practice. For example, weather prediction applications involve dynamics with strong drift (not just diffusion) terms, making A non-symmetric and perhaps non-normal in many cases. Standard direct methods for solving the Lyapunov equation given a non-normal dynamics matrix A (e.g., the Bartels-Stewart algorithm (Antoulas, 2005)) require an orthogonalization step that takes $O(d^3)$ time in general. There is a large applied mathematics literature on the approximate solution of Lyapunov equations with sparse dynamics (see e.g. Sabino (2007) for a nice review), but the focus of this literature is on the case that the noise covariance matrix is of low rank, which may be less relevant in some statistical applications. Further research is needed into how to adapt modern methods for solving the Lyapunov equation to the fast Kalman filter setting.

Another important direction for future research involves generalizations beyond the simple Kalman setting explored here. The smoothers we have discussed are all based on a simple AR(1) framework. It is natural to ask if similar methods can be employed to efficiently handle the general autoregressive-moving average (ARMA) case, or other temporal smoothing methodologies (e.g., penalized spline methods (Green and Silverman, 1994; DiMatteo et al., 2001; Wood, 2006)), since all of these techniques rely heavily on solving linear equations for

which the corresponding matrices are block banded in the temporal domain.

Finally, for the methods discussed here, we assumed that the underlying dynamics model (A, V) does not change with time, in order to compute the equilibrium state covariance C_0. However, as noted in the presentation of our methods C_0 can be interpreted as the limit of a time varying prior covariance $C_{0,t}$. In this case, our methods can be applied when $C_{0,t}$ can be updated efficiently and used for fast matrix-vector operations. This is possible in a number of setups (Pnevmatikakis and Paninski, 2012), and opens up some interesting applications involving the incorporation of non-Gaussian priors (Park and Casella, 2008) and efficient sampling of the full posterior $p(X|Y)$ using the perturbation technique of Papandreou and Yuille (2010). We are currently pursuing these directions further.

Acknowledgments

LP is supported by a McKnight Scholar award, an NSF CAREER award, an NSF grant IIS-0904353, an NEI grant EY018003, and a DARPA contract N66001-11-1-4205. JH is supported by the Columbia College Rabi Scholars Program. The authors thank P. Jercog for kindly sharing his hippocampal data with us.

Supplementary Material

Appendix.pdf: Appendix including the omitted proofs and further technical results. Section A deals with extensions to non-Gaussian measurements. Section B provides further analysis and tighter bounds for the effective rank. Section C provides the proof for Theorem 4.1 and discusses the convergence properties of the LRBT algorithm when applied in an iterative fashion. Section D discusses how the LRBT can be extended to provide estimates of the smoothed covariance in $O(K(d)T)$ time and $O(dT)$ space.

Matlab code: Matlab code including routines for all the algorithms described in the paper (fast Kalman filter, LRBT, steepest descent LRBT, conjugate gradients with LRBT preconditioner). The wrapper code calls all the algorithms as well the exact Kalman

filter and BT algorithms for the receptive field example presented in section 5.1.

References

Anderson, B. and J. Moore (1979). *Optimal Filtering*. Prentice Hall.

Antoulas, A. (2005). *Approximation of Large-scale Dynamical Systems*. Cambridge University Press.

Banerjee, S., A. E. Gelfand, A. O. Finley, and H. Sang (2008). Gaussian predictive process models for large spatial data sets. *Journal Of The Royal Statistical Society Series B 70*, 825–848.

Bickel, J. and E. Levina (2008). Regularized estimation of large covariance matrices. *Annals of Statistics 36*, 199–227.

Briggs, W. L., V. E. Henson, and S. F. McCormick (2000). *A Multigrid Tutorial* (2nd ed.). SIAM.

Brockwell, P. and R. Davis (1991). *Time Series: Theory and Methods*. Springer.

Brown, E., L. Frank, D. Tang, M. Quirk, and M. Wilson (1998). A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *Journal of Neuroscience 18*, 7411–7425.

Brown, E., D. Nguyen, L. Frank, M. Wilson, and V. Solo (2001). An analysis of neural receptive field plasticity by point process adaptive filtering. *PNAS 98*, 12261–12266.

Chan, R. H. and M. K. Ng (1996). Conjugate gradient methods for toeplitz systems. *SIAM Review 38*, 427–482.

Chandrasekar, J., I. Kim, D. Bernstein, and A. Ridley (2008). Cholesky-based reduced-rank square-root Kalman filtering. In *American Control Conference*, pp. 3987–3992. IEEE.

Cressie, N. and G. Johannesson (2008). Fixed rank kriging for very large spatial data sets. *Journal Of The Royal Statistical Society Series B 70*, 209–226.

Cressie, N., T. Shi, and E. L. Kang (2010). Fixed rank filtering for spatio-temporal data. *Journal of Computational and Graphical Statistics 19*, 724–745.

Czanner, G., U. Eden, S. Wirth, M. Yanike, W. Suzuki, and E. Brown (2008). Analysis of between-trial and within-trial neural spiking dynamics. *Journal of Neurophysiology 99*, 2672–2693.

Davis, T. (2006). *Direct Methods for Sparse Linear Systems*. SIAM.

Dayan, P. and L. Abbott (2001). *Theoretical Neuroscience*. MIT Press.

DiMatteo, I., C. Genovese, and R. Kass (2001). Bayesian curve fitting with free-knot splines. *Biometrika 88*, 1055–1073.

El Karoui, N. (2008). Operator norm consistent estimation of large-dimensional sparse covariance matrices. *Annals of Statistics 36*, 2717–2756.

Evensen, G. (2009). *Data assimilation: the Ensemble Kalman Filter*. Springer.

Fahrmeir, L. and H. Kaufmann (1991). On Kalman filtering, posterior mode estimation and fisher scoring in dynamic exponential family regression. *Metrika 38*, 37–60.

Fahrmeir, L. and G. Tutz (1994). *Multivariate Statistical Modelling Based on Generalized Linear Models*. Springer.

Fedorov, V. (1972). *Theory of Optimal Experiments*. New York: Academic Press.

Frank, L., U. Eden, V. Solo, M. Wilson, and E. Brown (2002). Contrasting patterns of receptive field plasticity in the hippocampus and the entorhinal cortex: An adaptive filtering approach. *J. Neurosci. 22*(9), 3817–3830.

Freestone, D., P. Aram, M. Dewar, K. Scerri, D. Grayden, and V. Kadirkamanathan (2011). A data-driven framework for neural field modeling. *NeuroImage 56*(3), 1043–1058.

Furrer, R. and T. Bengtsson (2007). Estimation of high-dimensional prior and posterior covariance matrices in Kalman filter variants. *J. Multivar. Anal. 98*, 227–255.

Galka, A., T. Ozaki, H. Muhle, U. Stephani, and M. Siniatchkin (2008). A data-driven model of the generation of human EEG based on a spatially distributed stochastic wave equation. *Cognitive Neurodynamics 2*(2), 101–13.

Golub, G. H. and C. F. Van Loan (1996). *Matrix Computations*. The Johns Hopkins University Press.

Green, P. and B. Silverman (1994). *Nonparametric Regression and Generalized Linear Models*. CRC Press.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *The Elements of Statistical Learning*. Springer.

Hines, M. (1984). Efficient computation of branched nerve equations. *International Journal of Bio-Medical Computing 15*(1), 69 – 76.

Huggins, J. and L. Paninski (2012). Optimal experimental design for sampling voltage on dendritic trees. *Journal of Computational Neuroscience 32*, 347–366.

Isaacson, E. and H. Keller (1994). *Analysis of numerical methods*. Dover Publications.

Kaufman, C. G., M. J. Schervish, and D. W. Nychka (2008). Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association 103*(484), 1545–1555.

Khan, U. A. and J. M. F. Moura (2008). Distributing the Kalman filter for large-scale systems. *IEEE Transactions on Signal Processing 56*, 4919–4935.

Koch, C. (1999). *Biophysics of Computation*. Oxford University Press.

Lew, S., C. H. Wolters, T. Dierkes, C. Röer, and R. S. MacLeod (2009). Accuracy and run-time comparison for different potential approaches and iterative solvers in finite element method based EEG source analysis. *Appl. Numer. Math. 59*, 1970–1988.

Lewi, J., R. Butera, and L. Paninski (2009). Sequential optimal design of neurophysiology experiments. *Neural Computation 21*, 619–687.

Long, C. J., R. L. Purdon, S. Temereanca, N. U. Desai, M. Hämäläinen, and E. N. Brown (2006). Large scale Kalman filtering solutions to the electrophysiological source localization problem–a MEG case study. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 4532–4535.

Paninski, L. (2010). Fast Kalman filtering on quasilinear dendritic trees. *Journal of Computational Neuroscience 28*, 211–28.

Paninski, L., Y. Ahmadian, D. Ferreira, S. Koyama, K. Rahnama, M. Vidne, J. Vogelstein, and W. Wu (2010). A new look at state-space models for neural data. *Journal of Computational Neuroscience 29*, 107–126.

Papandreou, G. and A. Yuille (2010). Gaussian sampling by local perturbations. In *Proc. NIPS*.

Park, T. and G. Casella (2008). The Bayesian Lasso. *Journal of the American Statistical Association 103*, 681–686.

Pnevmatikakis, E. and L. Paninski (2012). Fast interior-point inference in high-dimensional, sparse, penalized state-space models. In *Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS). J Mach Learn Res*, Volume 22, pp. 895–904.

Press, W., S. Teukolsky, W. Vetterling, and B. Flannery (1992). *Numerical recipes in C*. Cambridge University Press.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 257–286.

Rahnama Rad, K. and L. Paninski (2010). Efficient estimation of two-dimensional firing rate surfaces via Gaussian process methods. *Network 21*, 142–168.

Rue, H. and L. Held (2005). *Gaussian Markov Random Fields: Theory and Applications*. CRC Press.

Rybicki, G. and D. Hummer (1991). An accelerated lambda iteration method for multilevel radiative transfer. I-Non-overlapping lines with background continuum. *Astronomy and Astrophysics 245*, 171–181.

Sabino, J. (2007). *Solution of large-scale Lyapunov equations via the block modified Smith method*. Ph. D. thesis, Rice University.

Seeger, M. and H. Nickisch (2011). Large scale Bayesian inference and experimental design for sparse linear models. *SIAM Journal of Imaging Sciences 4*(1), 166–199.

Shumway, R. and D. Stoffer (2006). *Time Series Analysis and Its Applications*. Springer.

Solo, V. (2004). State estimation from high-dimensional data. *ICASSP 2*, 685–688.

Stroud, J. R., P. Muller, and B. Sanso (2001). Dynamic models for spatiotemporal data. *Journal of the Royal Statistical Society. Series B (Statistical Methodology) 63*, pp. 673–689.

Treebushny, D. and H. Madsen (2005). On the construction of a reduced rank square-root Kalman filter for efficient uncertainty propagation. *Future Gener. Comput. Syst. 21*, 1047–1055.

Verlaan, M. (1998). *Efficient Kalman filtering algorithms for hydrodynamic models*. Ph. D. thesis, TU Delft.

Wikle, C. and N. Cressie (1999). A dimension-reduced approach to space-time Kalman filtering. *Biometrika 86*(4), 815–829.

Wolters, C. (2007). The finite element method in EEG/MEG source analysis. *SIAM News 40*(2), 1–2.

Wood, S. N. (2006). Low-rank scale-invariant tensor product smooths for generalized additive mixed models. *Biometrics 62*, 1025–36.