# Particle Smoothing in Continuous Time: A Fast Approach Via Density Estimation

Lawrence Murray, *Member, IEEE,* and Amos Storkey

***Abstract*—We consider the particle smoothing problem for state-space models where the transition density is not available in closed form, in particular for continuous-time, nonlinear models expressed via stochastic differential equations (SDEs). Conventional forward-backward and two-filter smoothers for the particle filter require a closed-form transition density, with the linear-Gaussian Euler-Maruyama discretisation usually applied to the SDEs to achieve this. We develop a pair of variants using kernel density approximations to relieve the dependence, and in doing so enable use of faster and more accurate discretisation schemes such as Runge-Kutta. In addition, the new methods admit arbitrary proposal distributions, providing an avenue to deal with degeneracy issues. Experimental results on a Functional Magnetic Resonance Imaging (fMRI) deconvolution task demonstrate comparable accuracy and significantly improved runtime over conventional techniques.**

***Index Terms*—Particle filter, sequential Monte Carlo, smoothing, state-space models, density estimation, continuous time.**

## I. Introduction

In fields as diverse as finance, ecology, the physical sciences and biology, dynamical phenomena are naturally modelled using continuous-time stochastic processes. Such equations are a mainstay of stock market prediction, neural modelling, environmental monitoring and other applications. More recently, they have arisen in the hemodynamics underpinning Functional Magnetic Resonance Imaging (fMRI), the case study for this work. While continuous time may be a natural avenue of expression for such models, their use in Bayesian inference is underdeveloped, where a first step is often to discretise in time.

Continuous-time models most often take the form of ordinary (ODE) or stochastic (SDE) differential equations. The simulation of trajectories forward in time has been well studied, with various numerical integration schemes available [1], [2]. It therefore seems sensible to consider simulation-based approaches to inference with these models, such as the Kalman [3], unscented Kalman [4], [5] and particle [6] filters and associated smoothers. The particle filter is our focus here, being founded on the fewest assumptions and most generally applicable. We particularly consider the smoothing problem, where the challenges of continuous time appear most significant.

### A. State-space model formulation

For an ascending sequence of $T$ time points $t_1, \ldots, t_T$, we are provided with measurements $\mathbf{y}(t_1), \ldots, \mathbf{y}(t_T) \in \mathbb{R}^M$ indicative of the latent state $\mathbf{x}(t) \in \mathbb{R}^N$ of a dynamical system across time $t$. The state of the system transitions according to a stochastic Markov function $\mathbf{f}(\mathbf{x}, t)$, with observations of the state predicted via a stochastic function $\mathbf{g}(\mathbf{x})$.

For notational convenience we let $\mathbf{x}_0$ denote the initial state of the system at time $t = 0$, and define $\mathbf{x}_n := \mathbf{x}(t_n)$ and $\mathbf{y}_n := \mathbf{y}(t_n)$. In referring to sequences, $\mathbf{z}_{i:j}$ with $i < j$ refers to the set $\{\mathbf{z}_i, \ldots, \mathbf{z}_j\}$ for some symbol $\mathbf{z}$.

For all $n = 1, \ldots, T$, to solve the *filtering problem* we wish to estimate the *filter densities* $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n})$, and to solve the *smoothing problem* the *smooth densities* $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:T})$. The density $p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n)$ features prominently in our discussions, and we refer to it as the *transition density*.

### B. Stochastic differential equations

We are interested in the class of models where the transition density is not available in closed form, in particular where $\mathbf{f}(\cdot)$ is known only up to its time derivatives via the SDE:

$$d\mathbf{x} = \underbrace{\mathbf{a}(\mathbf{x}, t)dt}_{\text{drift}} + \underbrace{B(\mathbf{x}, t)d\mathbf{W}}_{\text{diffusion}}, \tag{1}$$

where $d\mathbf{W}$ is an increment of the multivariate *Wiener process*, a model of simple Brownian motion [7]. The equation consists of a deterministic *drift* component and stochastic *diffusion* component. Eliminating the diffusion component gives a deterministic ODE.

In some cases Itô integration or Fokker-Planck equations [7] can provide a closed form solution to the transition density. In the general case, however, we must rely on numerical integration of the differential equations, constituting a discretisation of time. The simplest method for doing so is the first-order Euler-Maruyama scheme [2]:

*Algorithm 1 (Euler-Maruyama scheme):* Let $t_n$ and $\mathbf{x}_n \approx \mathbf{x}(t_n)$ be given. Then, for time step $\Delta t$ and Wiener increment $\Delta\mathbf{W} \sim \mathcal{N}(\mathbf{0}, I\sqrt{\Delta t})$:

$$\begin{aligned} t_{n+1} &= t_n + \Delta t \\ \mathbf{x}_{n+1} &= \mathbf{x}_n + \mathbf{a}(\mathbf{x}_n, t_n)\Delta t + B(\mathbf{x}_n, t_n)\Delta\mathbf{W}. \end{aligned}$$

This is a linear discretisation with additive Gaussian noise, providing a closed-form transition density over a single step. The step size must be sufficiently small, however, for linearity to be a credible approximation of local dynamics. Higher-order nonlinear schemes, such as Runge-Kutta, admit larger step sizes while maintaining accuracy, and so tend to compute faster. Development of higher-order methods for SDEs is difficult, as each step squeezes the Wiener increment through a nonlinear function, such that Gaussianity is not maintained. It

L. Murray is with CSIRO Mathematics, Informatics & Statistics.
A. Storkey is with the School of Informatics, University of Edinburgh.

is not sufficient to simply apply a single step of a deterministic higher-order method for ODEs, such as Runge-Kutta, and then add a noise sample [8]. While specialist SDE integration schemes exist, a particularly practical choice is to observe that any scheme for ODEs can be adapted to SDEs with approximately half the order [9] by converting the Itô equation (1) into its equivalent Stratonovich form [2, p157]:

$$d\mathbf{x} = \left[\mathbf{a}(\mathbf{x},t) - \frac{1}{2}\sum_i \frac{\partial B(\mathbf{x},t)}{\partial x_i} B_{i*}(\mathbf{x},t)^T\right] dt + B(\mathbf{x},t) \circ d\mathbf{W},$$

where $B_{i*}(\mathbf{x},t)$ is the $i$th row of $B(\mathbf{x},t)$. The extra drift term arises as a result of calculating the derivative at the midpoint of the increment under Stratonovich, rather than at the beginning as under Itô. Under Stratonovich, the standard chain rule of calculus applies, unlike under Itô, so that existing solvers for ODEs may be applied. All of this leads to the following extension of the Runge-Kutta algorithm for ODEs to Itô SDEs [9]:

*Algorithm 2 (Stochastic Runge-Kutta scheme):* Let $t_n$ and $\mathbf{x}_n \approx \mathbf{x}(t_n)$ be given. Then, for time step $\Delta t$ and Wiener increment $\Delta\mathbf{W} \sim \mathcal{N}(\mathbf{0}, I\sqrt{\Delta t})$, let $\mathbf{h}(\mathbf{x}, t, \Delta t, \Delta\mathbf{W})$:

$$\mathbf{h}(\cdot) = \mathbf{a}(\mathbf{x},t) - \frac{1}{2}\sum_i \frac{\partial B(\mathbf{x},t)}{\partial x_i} B_{i*}(\mathbf{x},t)^T + B(\mathbf{x},t)\frac{\Delta\mathbf{W}}{\Delta t}$$

where $B_{i*}(\mathbf{x},t)$ is the $i$th row of $B(\mathbf{x},t)$. Then, for an $S$ stage numerical scheme:

$$\begin{aligned} t_{n+1} &= t_n + \Delta t \\ \mathbf{x}_{n+1} &= \mathbf{x}_n + \Delta t \sum_{i=1}^{s} b_i \mathbf{k}_i, \end{aligned}$$

where:

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{h}(\mathbf{x}_n, t_n, \Delta t, \Delta\mathbf{W}) \\ \mathbf{k}_{i=2,\ldots,S} &= \mathbf{h}(\mathbf{x}_n + \Delta t \sum_{j=1}^{i-1} a_{i,j}\mathbf{k}_j, t_n + c_i\Delta t, \Delta t, \Delta\mathbf{W}), \end{aligned}$$

The number of stages, $S$, and coefficients $a_{i,j}$, $b_i$ and $c_i$, are selected to satisfy the conditions for some desired order of accuracy [1]. Popular configurations are the six-stage ($S = 6$), fifth-order Dormand-Prince [10] and fourth-order Fehlberg [11] schemes.

### C. Particle smoothing

The particle filter [6], [12] recursively approximates the filter density at time $t_n$ with a weighted set of $P$ samples $\{(\mathbf{s}_n^{(i)}, \pi_n^{(i)})\}$, where $i = 1, \ldots, P$, so that $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n}) \approx \sum_{i=1}^P \pi_n^{(i)} \delta(\mathbf{x}_n - \mathbf{s}_n^{(i)}) / \sum_{i=1}^P \pi_n^{(i)}$. Smooth densities can be obtained from this via application of either the *two-filter* or *forward-backward* smoother.

Consider the following factorisation of the smooth density:

$$\begin{aligned} p(\mathbf{x}_n \,|\, \mathbf{y}_{1:T}) &\propto p(\mathbf{y}_{n:T} \,|\, \mathbf{x}_n) p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n-1}) \quad (2) \\ &\propto \frac{p(\mathbf{x}_n \,|\, \mathbf{y}_{n:T})}{p(\mathbf{x}_n)} p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n-1}). \quad (3) \end{aligned}$$

In this way the smooth density may be obtained via the combination of both forward and backward (in time) filters.

The priors $p(\mathbf{x}_n)$ will not typically be known in closed form except for linear-Gaussian models. In such cases each may be replaced by an arbitrary approximation $\gamma_n(\mathbf{x}_n)$ [13]. Suitable approximations include the equilibrium distribution for a stationary process, or an analytical form fit to the output of model simulations [13]. When these are incorporated into the backward filter only approximate filter densities are obtained, but their reappearance in Algorithm 4 corrects for this [14] to deliver exact smooth densities, up to effects on sampling efficacy [13]:

*Algorithm 3 (Backward particle filter):* Given the weighted sample set $\{(\tilde{\mathbf{s}}_{n+1}^{(i)}, \tilde{\pi}_{n+1}^{(i)})\}$, representing the backward filter density $p(\mathbf{x}_{n+1} \,|\, \mathbf{y}_{n+1:T})$, approximate priors $\gamma_n(\mathbf{x}_n)$ and $\gamma_{n+1}(\mathbf{x}_n)$, and a proposal distribution $q(\mathbf{x}_n)$, sample $\tilde{\mathbf{s}}_n^{(i)} \sim q(\mathbf{x}_n)$ and weight with:

$$\tilde{\pi}_n^{(i)} = \frac{p(\mathbf{y}_n \,|\, \tilde{\mathbf{s}}_n^{(i)}) \gamma_n(\tilde{\mathbf{s}}_n^{(i)})}{q(\tilde{\mathbf{s}}_n^{(i)})} \sum_{j=1}^P \frac{p(\tilde{\mathbf{s}}_{n+1}^{(j)} \,|\, \tilde{\mathbf{s}}_n^{(i)}) \tilde{\pi}_{n+1}^{(j)}}{\gamma_{n+1}(\tilde{\mathbf{s}}_{n+1}^{(j)})}.$$

The weighted sample set $\{(\tilde{\mathbf{s}}_n^{(i)}, \tilde{\pi}_n^{(i)})\}$ then approximates the backward filter density $p(\mathbf{x}_n \,|\, \mathbf{y}_{n:T})$.

All that remains is to fuse the results to obtain the smooth density.

*Algorithm 4 (Two-filter smoother):* Perform a filter forward in time, and a filter backward in time, to obtain weighted sample sets $\{(\mathbf{s}_n^{(i)}, \pi_n^{(i)})\}$ and $\{(\tilde{\mathbf{s}}_n^{(i)}, \tilde{\pi}_n^{(i)})\}$, respectively, for each time $t_n$. Then, for time $t_n$, let:

$$\tilde{\psi}_n^{(i)} = \frac{\tilde{\pi}_n^{(i)}}{\gamma_n(\mathbf{x}_n = \tilde{\mathbf{s}}_n^{(i)})} \sum_{j=1}^P p(\mathbf{x}_n = \tilde{\mathbf{s}}_n^{(i)} \,|\, \mathbf{x}_{n-1} = \mathbf{s}_{n-1}^{(j)}) \pi_{n-1}^{(j)}.$$

The weighted sample set $\{(\tilde{\mathbf{s}}_n^{(i)}, \tilde{\psi}_n^{(i)})\}$ then approximates the smooth density $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:T})$.

Runtime complexity of the method is $\mathcal{O}(TP^2)$ given the all-pairs transition density calculations in both Algorithms 3 and 4. In some cases this may be reduced to $\mathcal{O}(T \lg P \lg P)$ using $N$-body methods [15], which we discuss further in Sec. I-D. Memory requirements are $\mathcal{O}(TNP)$ for storage of filter results required by the smoother.

An alternative to the two-filter smoother is the forward-backward smoother:

*Algorithm 5 (Forward-backward smoother):* Perform a filter forward in time, at the conclusion of which $p(\mathbf{x}_T \,|\, \mathbf{y}_{1:T})$ is known and approximated by $\{(\mathbf{s}_T^{(i)}, \pi_T^{(i)})\}$. Initialise with $\psi_T^{(i)} = \pi_T^{(i)}$ and proceed recursively as follows:

$$\begin{aligned} \alpha_n^{(i,j)} &= p(\mathbf{x}_{n+1} = \mathbf{s}_{n+1}^{(i)} \,|\, \mathbf{x}_n = \mathbf{s}_n^{(j)}) \\ \gamma_n^{(i)} &= \sum_{j=1}^P \pi_n^{(j)} \alpha_n^{(i,j)} \\ \delta_n^{(j)} &= \sum_{i=1}^P \psi_{n+1}^{(i)} \frac{\alpha_n^{(i,j)}}{\gamma_n^{(i)}} \\ \psi_n^{(j)} &= \pi_n^{(j)} \delta_n^{(j)}. \end{aligned}$$

The weighted sample set $\{(\mathbf{s}_n^{(i)}, \psi_n^{(i)})\}$ then approximates the smooth density $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:T})$.

See [15]–[17] for further details. Runtime complexity is $\mathcal{O}(TP^2)$ given the all-pairs transition density calculations (the

$\alpha_n^{(i,j)}$ terms) at each step. Like the two-filter smoother, this may be reduced to $\mathcal{O}(T \lg P \lg P)$ in the best case for some circumstances [15]. Memory requirements are $\mathcal{O}(TNP)$ for storage of filter results required by the smoother. If $\alpha_n^{(i,j)}$ terms are precalculated ahead of both $\gamma_n^{(i)}$ and $\delta_n^{(i)}$, a constant factor gain to runtime is obtained at the burden of $\mathcal{O}(P^2)$ storage, although $\alpha_n$ may be sparse [18].

### D. Contribution

The crux of the problem that we address is this: in order to accurately and efficiently simulate from an SDE model, we wish to employ a high-order numerical integration scheme; in doing so, however, the transition density becomes unavailable in closed form. While we can use higher-order numerical integrators to draw some $\mathbf{s}_{n+1} \sim p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n = \mathbf{s}_n)$, we cannot evaluate $p(\mathbf{x}_{n+1} = \mathbf{s}_{n+1} \,|\, \mathbf{x}_n = \mathbf{s}_n)$. Existing particle smoothing methods assume that the transition density is available in closed form; the aim of this work is to develop methods which do not.

To elaborate, note the assumed closed form of the transition density $p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n)$ in all of the methods presented. This is available when using Euler-Maruyama over a single step, although doing so inextricably ties the step size of the filter or smoother to one sufficiently small to uphold the linear discretisation, which can be expensive. Use of a proposal that includes the transition density can cancel it and so facilitate use of higher-order discretisations. This is the case for the bootstrap [12] and regularised [19], [20] particle filters, and auxiliary particle filter [21] when the lookahead is a draw from the transition. It is not the case for more elaborate schemes such as the unscented particle filter [22]. We are not aware of the same idea having been used on the smoothing problem.

Both the two-filter and forward-backward smoothers obtain the smooth density by reweighting samples obtained during a filtering pass – a backward filter in the former case, and a forward filter in the latter. This may be problematic if these samples fail to support the smooth density adequately, and the smoother may then *degenerate* – heaping weight on a single point to represent the smooth density. This is a secondary issue that our methods may help to address, by admitting arbitrary proposal distributions to simulate new particles during the smoothing pass. Some recent work has considered the same [13], [14].

A number of existing works are worth contrasting. In [23], $m - 1$ latent points are introduced between each pair of observations, and then batch Gibbs sampling performed. This corresponds precisely to an Euler-Maruyama discretisation, with $m$ tuned to control error appropriately. Such a low-order discretisation may be unstable for stiff models, and we demonstrate in this work that it may be computationally prohibitive also. Our own methods admit higher-order discretisations. The random-weight particle filter [24] replaces the transition density with a rejection sampling that converges to it. This relies on the *exact algorithm* [25] for sampling Brownian bridges, which requires a particular form of SDE with gradient drift and additive diffusion. Our own methods apply to the broader class of equations embodied by (1), admitting arbitrary drift and correlated diffusion terms; this will be required for our experimental model in Section III. Finally, variational methods may also be applicable [26], [27]. These require the fit of a tractable density to the intractable posterior over trajectories, which typically means fitting a simplified linear-Gaussian model. Our own methods attempt to fit the original state space model.

The two kernel methods presented in this work are of computational complexity $\mathcal{O}(T \lg P \lg P)$ in the best case and $\mathcal{O}(TP^2)$ in the worst. These derive from performing evaluations on a kernel density [28] over $P$ support points for $P$ query points, by constructing a partition tree (such as a $kd$ tree) over both sets of points and performing a dual-tree computation – techniques inherited from $N$-body simulation [29]. The best case corresponds to only one support point contributing significantly to the density approximation at each query point, while the worst case occurs when all support points provide significant contributions for all query points. Similar ideas have been explored in [15] to address the $\mathcal{O}(TP^2)$ bottleneck of the conventional forward-backward and two-filter smoothers. In that work, the transition density must be of a closed-form similarity kernel $\mathcal{K}(d(\mathbf{x}_n, \mathbf{x}_{n-1}))$, where $\mathcal{K}(\cdot)$ denotes the closed-form kernel over the distance $d(\cdot)$. This may be difficult to apply for anything but additive noise structures. By not requiring the transition density, the ideas presented here are not limited in this way, albeit via the introduction of additional approximations over those in [15]. A less complex $\mathcal{O}(TP)$ smoother is given in [14]. This again requires a closed-form transition density, something not assumed here in order support a broader class of models, specifically those defined with SDEs.

## II. Methods

We propose two novel methods for particle smoothing with SDE models, using density approximations to facilitate the cancellation of the transition density. Further details are available in [30].

### A. Kernel forward-backward smoother

The kernel forward-backward smoother follows a similar derivation to that of the forward-backward smoother, with the introduction of density approximation to permit arbitrary proposal distributions for importance sampling. Factorise the smooth density as follows:

$$\begin{aligned} & p(\mathbf{x}_n \,|\, \mathbf{y}_{1:T}) \\ \propto\ & p(\mathbf{y}_{n+1:T} \,|\, \mathbf{x}_n) p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n}) \\ \propto\ & p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n}) \int p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n) p(\mathbf{y}_{n+1:T} \,|\, \mathbf{x}_{n+1}) d\mathbf{x}_{n+1} \\ \propto\ & p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n}) \int p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n) \frac{p(\mathbf{x}_{n+1} \,|\, \mathbf{y}_{1:T})}{p(\mathbf{x}_{n+1} \,|\, \mathbf{y}_{1:n})} d\mathbf{x}_{n+1}. \end{aligned}$$

To eliminate the integral and simplify the derivation, consider the joint:

$$p(\mathbf{x}_{n:n+1} \,|\, \mathbf{y}_{1:T}) = \frac{p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n}) p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n) p(\mathbf{x}_{n+1} \,|\, \mathbf{y}_{1:T})}{p(\mathbf{x}_{n+1} \,|\, \mathbf{y}_{1:n})}. \tag{4}$$

Now consider importance sampling from this with a proposal distribution $q'(\mathbf{x}_{n:n+1})$ of the form:

$$q'(\mathbf{x}_{n:n+1}) = p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n) q(\mathbf{x}_n),$$

so as to cancel the intractable transition density in (4). Drawing $(\mathbf{s}_n^{(i)}, \mathbf{r}_{n+1}^{(i)}) \sim q'(\mathbf{x}_{n:n+1})$, the weight calculation is reduced to:

$$\psi_n^{(i)} = \frac{p(\mathbf{x}_n = \mathbf{s}_n^{(i)} \,|\, \mathbf{y}_{1:n}) p(\mathbf{x}_{n+1} = \mathbf{r}_{n+1}^{(i)} \,|\, \mathbf{y}_{1:T})}{p(\mathbf{x}_{n+1} = \mathbf{r}_{n+1}^{(i)} \,|\, \mathbf{y}_{1:n}) q(\mathbf{x}_n = \mathbf{s}_n^{(i)})},$$

and the weighted sample set $\{(\mathbf{s}_n^{(i)}, \psi_n^{(i)})\}$ represents the smooth density at time $t_n$.

The filter densities $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n})$ and $p(\mathbf{x}_{n+1} \,|\, \mathbf{y}_{1:n})$ may be obtained through a preceding filter, and the smooth density $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:T})$ is known recursively. Depending on the selection of $q(\mathbf{x}_n)$, some of these will need to be approximated. Several techniques could be used for this, such as Gaussian mixtures or variational fits. We choose to use kernel density estimates [28], which we denote using $p_{\mathcal{K}}(\cdot)$. The algorithm is summarised as:

*Algorithm 6 (Kernel forward-backward smoother):* Perform a filter forward in time, at the conclusion of which $p(\mathbf{x}_T \,|\, \mathbf{y}_{1:T})$ is known and approximated by $\{(\mathbf{s}_T^{(i)}, \psi_T^{(i)} = \pi_T^{(i)})\}$. Then, for time $t_n$, draw $\mathbf{s}_n^{(i)}$ from some importance distribution $q(\mathbf{x}_n)$, draw $\mathbf{r}_{n+1}^{(i)} \sim p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n = \mathbf{s}_n^{(i)})$ via numerical integration, and let:

$$\psi_n^{(i)} = \frac{p_{\mathcal{K}}(\mathbf{x}_n = \mathbf{s}_n^{(i)} \,|\, \mathbf{y}_{1:n}) p_{\mathcal{K}}(\mathbf{x}_{n+1} = \mathbf{r}_{n+1}^{(i)} \,|\, \mathbf{y}_{1:T})}{p_{\mathcal{K}}(\mathbf{x}_{n+1} = \mathbf{r}_{n+1}^{(i)} \,|\, \mathbf{y}_{1:n}) q(\mathbf{x}_n = \mathbf{s}_n^{(i)})},$$

The weighted sample set $\{(\mathbf{s}_n^{(i)}, \psi_n^{(i)})\}$ then approximates the smooth density $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:T})$.

The interest now is to select an appropriate proposal distribution $q(\mathbf{x}_n)$. One option is to set $q(\mathbf{x}_n) = p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n})$. Recall that $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n})$ is approximated by the weighted sample set $\{(\mathbf{s}_n^{(i)}, \pi_n^{(i)})\}$. By preserving these samples and propagating each through the SDEs of the system to obtain $\{\mathbf{r}_{n+1}^{(i)}\}$, the smoothed weight reduces to:

$$\psi_n^{(i)} = \frac{p_{\mathcal{K}}(\mathbf{x}_{n+1} = \mathbf{r}_{n+1}^{(i)} \,|\, \mathbf{y}_{1:T})}{p_{\mathcal{K}}(\mathbf{x}_{n+1} = \mathbf{r}_{n+1}^{(i)} \,|\, \mathbf{y}_{1:n})} \pi_n^{(i)}. \tag{5}$$

Clearly, this will suffer from the same degeneracy issue of the forward-backward smoother if the filter density fails to adequately support the smooth density. Nevertheless, it is suitable in many situations and worth considering for its potential efficiency, given that one kernel density evaluation is eliminated. Other selections of proposal are likely to be model-specific, e.g. we use a tailored choice in Section III.

The algorithm requires two kernel density evaluations at each time step, and these dominate the runtime complexity. If these are performed using a dual-tree algorithm [29], runtime complexity is $\mathcal{O}(T \lg P \lg P)$ in the best case, and $\mathcal{O}(TP^2)$ in the worst. Various optimisations are available, particularly if resampling can be limited during the forward pass [30].

### B. Kernel two-filter smoother

Like its counterpart above, the kernel two-filter smoother may be compared to the two-filter smoother, having a similar derivation, and exploiting kernel densities for applicability to continuous-time models. In contrast to the standard two-filter smoother, however, the approach does not involve an explicit calculation of the backward filter density $p(\mathbf{x}_n \,|\, \mathbf{y}_{n:T})$, nor the prior $p(\mathbf{x}_n)$ or its substitute $\gamma_n(\mathbf{x}_n)$.

Factorise the smooth density as follows:

$$p(\mathbf{x}_n \,|\, \mathbf{y}_{1:T}) \quad \propto \quad p(\mathbf{y}_{n:T} \,|\, \mathbf{x}_n) p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n-1}),$$

expanding the likelihood term:

$$\begin{aligned} & p(\mathbf{y}_{n:T} \,|\, \mathbf{x}_n) \\ = \; & p(\mathbf{y}_n \,|\, \mathbf{x}_n) p(\mathbf{y}_{n+1:T} \,|\, \mathbf{x}_n) \\ = \; & p(\mathbf{y}_n \,|\, \mathbf{x}_n) \int p(\mathbf{y}_{n+1:T} \,|\, \mathbf{x}_{n+1}) p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n) d\mathbf{x}_{n+1}. \end{aligned}$$

Now observe:

$$\begin{aligned} & p(\mathbf{x}_n \,|\, \mathbf{y}_{n:T}) \\ \propto \; & p(\mathbf{x}_n) p(\mathbf{y}_{n:T} \,|\, \mathbf{x}_n) \\ \propto \; & p(\mathbf{x}_n) p(\mathbf{y}_n \,|\, \mathbf{x}_n) \int p(\mathbf{y}_{n+1:T} \,|\, \mathbf{x}_{n+1}) p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n) d\mathbf{x}_{n+1}, \end{aligned}$$

and consider the joint:

$$p(\mathbf{x}_{n:n+1} \,|\, \mathbf{y}_{n:T}) \tag{6}$$

$$\propto \quad p(\mathbf{x}_n) p(\mathbf{y}_n \,|\, \mathbf{x}_n) p(\mathbf{y}_{n+1:T} \,|\, \mathbf{x}_{n+1}) p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n). \tag{7}$$

Now consider importance sampling from this using a proposal distribution $q'(\mathbf{x}_{n:n+1})$ of the form:

$$q'(\mathbf{x}_{n:n+1}) = p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n) q(\mathbf{x}_n),$$

so as to cancel the intractable transition density in (7). Drawing $(\mathbf{s}_n^{(i)}, \mathbf{r}_{n+1}^{(i)}) \sim q'(\mathbf{x}_{n:n+1})$, the weight calculation for the backward filter is:

$$\tilde{\pi}_n^{(i)} = \frac{p(\mathbf{x}_n) p(\mathbf{y}_n \,|\, \mathbf{x}_n = \mathbf{s}_n^{(i)}) p_{\mathcal{K}}(\mathbf{y}_{n+1:T} \,|\, \mathbf{x}_{n+1} = \mathbf{r}_{n+1}^{(i)})}{q(\mathbf{x}_n = \mathbf{s}_n^{(i)})},$$

so that the weighted sample set $\{(\mathbf{s}_n^{(i)}, \tilde{\pi}_n^{(i)})\}$ would represent the backward filter density, if not for the expected unavailability of the prior $p(\mathbf{x}_n)$. Consider the weight calculation for the backward likelihood:

$$\begin{aligned} \beta_n^{(i)} \quad &= \quad \frac{\tilde{\pi}_n^{(i)}}{p(\mathbf{x}_n)} \\ &= \quad \frac{p(\mathbf{y}_n \,|\, \mathbf{x}_n = \mathbf{s}_n^{(i)}) p_{\mathcal{K}}(\mathbf{y}_{n+1:T} \,|\, \mathbf{x}_{n+1} = \mathbf{r}_{n+1}^{(i)})}{q(\mathbf{x}_n = \mathbf{s}_n^{(i)})}, \end{aligned}$$

noting that the prior $p(\mathbf{x}_n)$ has now been cancelled. The weighted sample set $\{(\mathbf{s}_n^{(i)}, \beta_n^{(i)})\}$ now represents the backward likelihood $p(\mathbf{y}_{n:T} \,|\, \mathbf{x}_n)$. This alone is a sufficient basis for the recursion of the method; the backward filter density is not required.

Note that $p_{\mathcal{K}}(\mathbf{y}_{n+1:T} \,|\, \mathbf{x}_{n+1})$ gives a kernel *likelihood* estimate. In terms of implementation, this is, for all intents and purposes, identical to a kernel *density* estimate, although need not integrate to one over $\mathbf{x}_n$, and indeed may be infinite.

Recalling (3), the smooth density weights may be calculated by:

$$\psi_n^{(i)} = \beta_n^{(i)} p_{\mathcal{K}}(\mathbf{x}_n = \mathbf{s}_n^{(i)} \,|\, \mathbf{y}_{1:n-1}),$$

and the weighted sample set $\{(\mathbf{s}_n^{(i)}, \psi_n^{(i)})\}$ represents the smooth density at time $t_n$. The algorithm is summarised below:

*Algorithm 7 (Kernel two-filter smoother):* Perform a filter forward in time, at the conclusion of which $p(\mathbf{x}_T \,|\, \mathbf{y}_{1:T})$ is known and approximated by $\{(\mathbf{s}_T^{(i)}, \psi_T^{(i)} = \pi_T^{(i)})\}$. Let:

$$\beta_T^{(i)} = \frac{p(\mathbf{y}_T \,|\, \mathbf{x}_T = \mathbf{s}_T^{(i)})}{p_{\mathcal{K}}(\mathbf{x}_T = \mathbf{s}_T^{(i)} \,|\, \mathbf{y}_{1:T})} \psi_T^{(i)},$$

so that $p(\mathbf{y}_T \,|\, \mathbf{x}_T)$ is approximated by a kernel likelihood over $\{(\mathbf{s}_T^{(i)}, \beta_T^{(i)})\}$.

Then, for time $t_n$, draw $\mathbf{s}_n^{(i)}$ from some importance distribution $q(\mathbf{x}_n)$, draw $\mathbf{r}_{n+1}^{(i)} \sim p(\mathbf{x}_{n+1} \,|\, \mathbf{x}_n = \mathbf{s}_n^{(i)})$, and let:

$$\beta_n^{(i)} = \frac{p(\mathbf{y}_n \,|\, \mathbf{x}_n = \mathbf{s}_n^{(i)}) p_{\mathcal{K}}(\mathbf{y}_{n+1:T} \,|\, \mathbf{x}_{n+1} = \mathbf{r}_{n+1}^{(i)})}{q(\mathbf{x}_n = \mathbf{s}_n^{(i)})}.$$

The weighted sample set $\{(\mathbf{s}_n^{(i)}, \beta_n^{(i)})\}$ then approximates the backward likelihood $p(\mathbf{y}_{n:T} \,|\, \mathbf{x}_n)$. Now, let:

$$\psi_n^{(i)} = \beta_n^{(i)} p_{\mathcal{K}}(\mathbf{x}_n = \mathbf{s}_n^{(i)} \,|\, \mathbf{y}_{1:n-1}).$$

The weighted sample set $\{(\mathbf{s}_n^{(i)}, \psi_n^{(i)})\}$ then approximates the smooth density $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:T})$.

Similar options for the proposal may be used as for the kernel forward-backward smoother, although in this case note that the filter density provides no additional cancellation. No other obvious cancelling proposals are available, partly because unlike the forward-backward smoother, the smooth density is calculated as an aside – it is the likelihood calculation that is essential for the recursion. Even the uncorrected filter density $p(\mathbf{x}_n \,|\, \mathbf{y}_{1:n-1})$, providing cancellation for smoothed weights, needs to be evaluated to recover likelihood weights for the next step of the recursion.

The algorithm requires two kernel density (likelihood) evaluations. This is fewer than for the kernel forward-backward smoother, although fewer optimisations are available also [30]. If these are performed using the dual-tree algorithm [29], runtime complexity is $\mathcal{O}(T \lg P \lg P)$ in the best case, and $\mathcal{O}(TP^2)$ in the worst.

## III. Experiments

We apply the methods to a test case in the domain of Functional Magnetic Resonance Imaging (fMRI) [31], [32], a medical imaging modality exploiting the Blood Oxygen Level Dependent (BOLD) contrast [33] to form images of functional activity in the brain. Whole brain volumes of order $10^5$ voxels are acquired at a rate of one every few seconds while a subject performs some task – simple finger tapping or visual stimulation being common examples.

The basic intuition behind fMRI is that neural activity has metabolic demands, such that an increase in neural activity in part of the brain causes an increase in the Cerebral Metabolic Rate of Oxygen ($CMRO_2$) in the surrounding capillary bed. The vascular system responds with a delayed surge of fresh arterial blood, increasing Cerebral Blood Flow (CBF) through the affected area, and consequently Cerebral Blood Volume (CBV). The response overcompensates for demand, such that the concentration of oxy- compared to deoxy-hemoglobin in the area increases. The BOLD signal, being a contrast between the two, varies accordingly.

Relative to the rapid fluctuations typical of neural activity, the hemodynamic response is slow and delayed, taking on the order of seconds. To study neural correlations or interactions across the brain, it is necessary to strip away the observed hemodynamic response to reveal the underlying neural activity [34]. We concentrate on this task, deconvolving the hemodynamic response in individual voxels in a time series of brain volumes acquired by fMRI.

### A. Model

We build a stochastic differential model of the input, neural, hemodynamic and BOLD activity in a single region of interest in the brain during an fMRI experiment. The BOLD signal is observed, while all other variables are latent state variables. Fig. 1 depicts sample trajectories from this model to help illustrate its typical behaviour.

Input, $u$, consists of the stimulus associated with the experimental task, commonly a simple box-car function denoting the intervals of rest and experimental stimulus.

Neural activity, $z$, responds to $u$ according to a stochastic extension of a one-dimensional *dynamic causal model* [35]:

$$dz = [(a + bu)z + cu]\, dt + \sigma_z dW,$$

where $a$, $b$, $c$ and $\sigma_z$ are parameters, noting that the latter scales the diffusion term on $z$.

The hemodynamic response to this neural activity is based around the *balloon model* [36], [37], which models a venous compartment in the brain as a balloon using Windkessel dynamics [38]. It begins with an inflow of arterial blood, $f$, which responds to $z$ via an abstract "vasodilatory" damped oscillation signal, $s$ [39]:

$$\begin{aligned} df &= s\,dt + f\sigma_f dW \\ ds &= \left(\epsilon z - \frac{s}{\tau_s} - \frac{f-1}{\tau_f}\right) dt, \end{aligned}$$

where $\epsilon$, $\tau_s$, $\tau_f$ and $\sigma_f$ are parameters.

The inflow of blood affects the CBV, normalised to volume at rest, $v$, and deoxyhemoglobin (dHb) concentration, normalised to that at rest, $q$. Oxygen is extracted from the inflow, and partially deoxygenated blood expelled, and so $v$ and $q$ vary according to [36]:

$$\begin{aligned} dq &= \frac{1}{\tau_0}\left(f\frac{1-(1-E_0)^{\frac{1}{f}}}{E_0} - v^{\frac{1}{\alpha}-1}q\right) dt + q\sigma_q dW \\ dv &= \frac{1}{\tau_0}(f - v^{\frac{1}{\alpha}})dt + v\sigma_v dW, \end{aligned}$$

with parameters $\tau_0$, $E_0$, $\alpha$, $\sigma_q$ and $\sigma_v$.

Finally, we introduce $w$ as a baseline BOLD signal during rest periods, allowed to slowly move via a random walk with

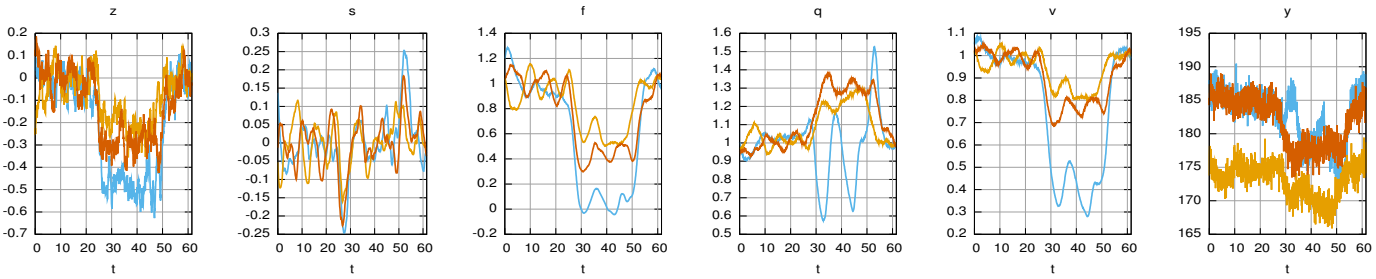


Fig. 1. Three sample trajectories simulated from the fMRI model described in Sec. III-A with random parameter configurations drawn from the prior. Input, $u$, consists of a 0-1 box-car function switching every 24.6s, as in the experimental data sets. Note in particular the several seconds delay of the BOLD signal ($y$) over the state variable of most interest, neural activity ($z$), motivating the deconvolution task.

diffusion parameter $\sigma_w$ to absorb large-scale trends in the data:

$$dw \quad = \quad \sigma_w dW.$$

For the observation model, the relative BOLD signal change at any time is given by [36]:

$$\Delta\hat{y} = V_0 \left[ k_1(1-q) + k_2 \left(1 - \frac{q}{v}\right) + k_3(1-v) \right],$$

converted to an absolute measurement $\hat{y}$ for comparison with actual measurements $y_{1:T}$ via the baseline signal $w$ and an independent noise source $\xi \sim \mathcal{N}(0,1)$:

$$\hat{y} = w(1 + \Delta\hat{y}) + \sigma_y \xi,$$

with parameters $V_0$, $k_1$, $k_2$ and $k_3$.

Hemodynamic parameters are fixed to experimentally derived values from the fMRI literature [35], [36]. Diffusion parameters are fixed with magnitudes that account for epistemic uncertainty in the otherwise deterministic model itself. See Table I.

A fixed value of -1 is used for $a$ to ensure identifiability. Remaining parameters, $b$ and $c$, are estimated online, with an orthogonal Gaussian prior given in Table I. Priors favour a slower decay to equilibrium, important for identifiability under infrequent observations.

An orthogonal Gaussian prior is given over the initial values of all state variables, as in Table I. Recall that $f$, $q$ and $v$ are normalised to their rest level, so that a prior mean of 1 is appropriate. Standard deviations are nominal. While $s$ and $z$ may be negative, their magnitude is expected to be comparable to $f$, $q$ and $v$; their priors reflect this. A sample from the prior is obtained by drawing from the Gaussian and then simulating the system for 12 s with a fixed input of zero to reach equilibrium.

While the introduction of stochasticity, in the form of the Wiener process, is novel for this model, we have taken care to maintain its intuition, carefully using it to emphasise the uncertainty in the coupling between neural activity and induced blood flow, while also accounting for the expected noise in a biological system such as this.

### B. Data

Real experimental data is used in the form of the *SessFX* data set, collected during a simple finger tapping exercise. Using a Siemens Vision at 2 T with a $TR$ of 4.1 s, a healthy 23-year-old right-handed male was scanned on 33 separate sessions over a period of two months. In each session, 80 whole volumes were taken, with the first two discarded to account for T1 saturation effects. The experimental paradigm consists of alternating $6TR$ blocks of rest and tapping of the right index finger at 1.5Hz, where tapping frequency is provided by a constant audio cue, present during both rest and tapping phases. Input, $u$, therefore consists of a box-car function that starts with a value of zero for $6TR$, followed by one for $6TR$, and repeating.

All scans across all sessions were then realigned using SPM5 [40] and a two-level random effects analysis performed, from which four voxels were selected as exhibiting interesting activity. For each voxel, the mean across all sessions is taken to act as the data set.

A comparable artificial data set, denoted *Sim*, is constructed by fixing all parameters ($b$ and $c$ drawn from prior) and simulating the model for the same length of time as the *SessFX* set, taking an observation every 4.1 s. The model is simulated using an RK9(8) Dormand-Prince integrator [10], different to, and of higher-order than, those used during inference. The importance of this artificial set is that it provides a ground-truth trace of underlying neural activity with which to compare the deconvolution obtained by various methods.

### C. Results

We first apply the bootstrap, auxiliary and regularised particle filters to deconvolution of the Sim set, using both Euler-Maruyama and order 4(5) Fehlberg Runge-Kutta [11] numerical integrators, abbreviated EM1 and RK4(5). EM1 uses a fixed time step of .05 to ensure tractability of the transition density – this value selected by prior simulation with an adaptive time step. RK4(5) adapts its time step to absolute and relative error bounds of $10^{-3}$ and $10^{-2}$ respectively, starting with a suggested size of .067, chosen as the mean step size when simulating under a range of conditions (see [30]).

For the regularised particle filter, we use a Gaussian kernel with bandwidth of $kh_{\text{opt}}$, where:

$$h_{\text{opt}}(N,P) = \left[ \frac{4}{(N+2)P} \right]^{\frac{1}{N+4}} \tag{8}$$

is the optimal bandwidth for kernel density approximation of an $N$-dimensional Gaussian using $P$ Gaussian kernels [28], and $k$ is some constant fraction, for which we try several

| Parameter ($\theta$) | Value |
|---|---|
| $\epsilon$ | .8 |
| $E_0$ | .4 |
| $\tau_0$ | 1.02 |
| $\tau_f^{-1}$ | .41 |
| $\tau_s^{-1}$ | .65 |

| Parameter ($\theta$) | Value |
|---|---|
| $\alpha$ | .32 |
| $V_0$ | .018 |
| $k_1$ | .28 |
| $k_2$ | 2 |
| $k_3$ | .4 |

| Parameter ($\theta$) | Value |
|---|---|
| $\sigma_z$ | .1 |
| $\sigma_f$, $\sigma_q$, $\sigma_v$ | .01 |
| $\sigma_w$ | .05 |
| $\sigma_y$ | 1.25 |
| $a$ | $-1$ |

| Variable ($x$) | $\mu_x$ | $\sigma_x$ |
|---|---|---|
| $b$ | 0 | .1 |
| $c$ | 0 | .5 |
| $w$ | * | 5 |
| $z$, $s$ | 0 | .1 |
| $f$, $q$, $v$ | 1 | .1 |

TABLE I
FIXED VALUES OF HEMODYNAMIC (BASED ON [35], [36]), NEURAL AND DIFFUSION PARAMETERS. GAUSSIAN PRIOR OVER NEURAL PARAMETERS TO BE ESTIMATED AND INITIAL VALUES OF STATE VARIABLES. THE MEAN FOR $w$ IS GIVEN BY THE MEAN OF ALL OBSERVATIONS IN REST BLOCKS.

values. While more rigorous means of bandwidth selection exist, such as cross-validation, these are expensive to perform at every step of a filter or smoother, so we prefer this simpler approach. For the bootstrap and auxiliary particle filters, samples of parameters will deplete as a side effect of resampling as the filter proceeds, as they have no dynamic of their own. Conversely, the kernel density resampling of the regularised particle filter introduces stochastic innovations to parameter samples at each step [19], [20], [41], such that alternative values are explored. This is equivalent to introducing an artificial dynamic to these otherwise static parameters.

We apply the kernel forward-backward and kernel two-filter smoothers over the results of the regularised particle filter, using the same bandwidth as the filter. While any of the filters may be used for this purpose, the regularised particle filter provides a consistent interpretation of the time marginals as kernel densities, and so has some intuitive appeal.

We use two proposals for these kernel smoothers. The first is the filter density, uncorrected in the case of the kernel two-filter smoother. The second is constructed by drawing $P$ samples from the last two blocks (recall each block is $6TR = 24.6s$) of the filter results, replacing their $w$ components with a sample drawn from the filtered marginal of $w$ at the current time, and building a kernel density over the result. We refer to the former as the *Filter* proposal, and the latter as the *Custom* proposal. The motivation behind the latter is that neural and hemodynamic activity converges to an input-switched stationary regime. We expect the filter to poorly fit this regime while it converges at low $t$, but nearing $T$, certainly in the last two blocks, we expect the estimate to be reasonably good, making it a suitable proposal for smoothing at all times. The BOLD baseline $w$, on the other hand, wanders randomly, and the marginal filter results are our best guess at it.

For comparison we apply the conventional forward-backward smoother over the results of the EM1 filter runs, with one caveat: as $s$ is deterministic, for each particle at time $t_{n+1}$, only one particle at time $t_n$ has positive probability of reaching it. For this reason we add nominal additive Gaussian noise with variance $10^{-4}$ to $s$ at each step for the smoothing pass only. Our implementation is $\mathcal{O}(TP^2)$, using sparse matrices for improved performance [18]. While methods exist using dual-tree evaluations as for our kernel smoothers [15], the correlated noise in this model confounds their application.

Results across many runs of each filter and smoother, as applied to the Sim data set, are summarised in Table II. As a measure of accuracy in deconvolving the neural signal we use

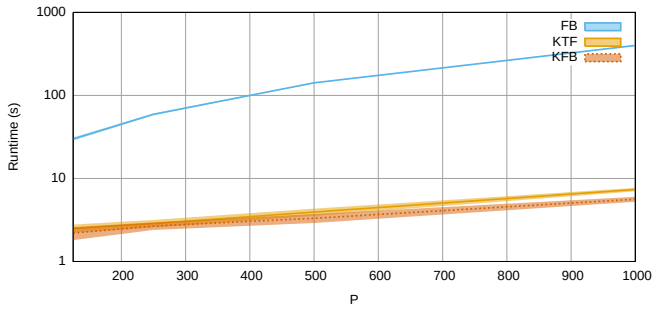


Fig. 3. Runtime scaling across $P$. Solid lines represent means and shaded area one standard deviation either side, across 40 runs. Both kernel smoothers use bandwidth of $0.1h_{\text{opt}}$ and filter as proposal.

weighted Root Mean Squared Error (RMSE) against known ground truth, providing a combined error for both state and parameters, as well as separate errors for each. A weighted RMSE may be distorted when the variance in weights is large, if a sample of significant weight happens to fall close to the ground truth. To qualify this, the table also includes Effective Sample Size (ESS) [42] across the normalised weights at all time points. Resampling introduces dependencies between samples, while ESS assumes independent samples – as a simple measure of weight variance to qualify RMSE figures this does not seem critical, however.

Results for the four regions of the SessFX set are given for the regularised particle filter, kernel forward-backward smoother and kernel two-filter smoother in Fig. 2, using the custom proposal at a bandwidth of $0.1h_{\text{opt}}$ in all cases.

Fig. 3 gives some indication of how runtime scales across $P$. All experiments are carried out on a quad-core Intel Xeon CPU at 3 GHz with 4 Gb memory; the job distributed over all four cores using the dysii Dynamic Systems Library (www.indii.org/software/dysii).

## IV. DISCUSSION

For the fMRI deconvolution test case, the kernel smoothers, using higher-order integrators, appear to be at least as accurate as, and substantially faster than the conventional forward-backward smoother, using the low-order Euler-Maruyama integrator (Table II). The visual comparison in Fig. 2 demonstrates the practical utility of the methods on real data, where they clearly correct the filtered neural signals for the SessFX data set in early blocks. On the Sim data set, all filters achieve comparable RMSE and ESS, but those utilising the higher-

| Method | Integrator | Resampler | Proposal | Band. | RMSE $\mathbf{x}, \boldsymbol{\theta}$ | | RMSE $\mathbf{x}$ | | RMSE $\boldsymbol{\theta}$ | | ESS | | Runtime | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PF | EM1 | Boot | | | 0.923 | (0.05) | 0.863 | (0.05) | 0.325 | (0.03) | 14471 | (1085) | 18.42 | (0.5) |
| PF | EM1 | Aux | | | 0.943 | (0.06) | 0.883 | (0.05) | 0.330 | (0.03) | 15478 | (1995) | 19.20 | (0.5) |
| PF | RK4(5) | Boot | | | 0.920 | (0.05) | 0.855 | (0.05) | 0.339 | (0.03) | 14338 | (1210) | 4.03 | (0.4) |
| PF | RK4(5) | Aux | | | 0.909 | (0.06) | 0.847 | (0.05) | 0.329 | (0.03) | 15626 | (1441) | 4.26 | (0.5) |
| PF | RK4(5) | Reg | | 0.1 | 0.903 | (0.05) | 0.840 | (0.05) | 0.331 | (0.03) | 14037 | (1502) | 3.98 | (0.4) |
| PF | RK4(5) | Reg | | 0.2 | 0.913 | (0.06) | 0.850 | (0.05) | 0.333 | (0.04) | 13889 | (1516) | 3.95 | (0.4) |
| PF | RK4(5) | Reg | | 0.3 | 0.925 | (0.05) | 0.863 | (0.05) | 0.332 | (0.03) | 13875 | (1424) | 4.11 | (0.3) |
| FB | EM1 | Boot | | | 0.598 | (0.05) | 0.521 | (0.05) | 0.290 | (0.05) | 2943 | (800) | 253.88 | (12.4) |
| FB | EM1 | Aux | | | 0.605 | (0.06) | 0.524 | (0.05) | 0.299 | (0.05) | 2985 | (790) | 253.72 | (15.5) |
| KFB | RK4(5) | Reg | Custom | 0.1 | 0.559 | (0.08) | 0.524 | (0.08) | 0.181 | (0.07) | 727 | (218) | 6.26 | (0.4) |
| KFB | RK4(5) | Reg | Custom | 0.2 | 0.622 | (0.09) | 0.587 | (0.09) | 0.196 | (0.06) | 2220 | (907) | 6.03 | (0.5) |
| KFB | RK4(5) | Reg | Custom | 0.3 | 0.690 | (0.08) | 0.659 | (0.07) | 0.201 | (0.06) | 4048 | (1238) | 6.20 | (0.4) |
| KFB | RK4(5) | Reg | Filter | 0.1 | 0.507 | (0.04) | 0.477 | (0.04) | 0.164 | (0.05) | 3826 | (1389) | 3.29 | (0.3) |
| KFB | RK4(5) | Reg | Filter | 0.2 | 0.596 | (0.09) | 0.555 | (0.07) | 0.211 | (0.07) | 5741 | (2344) | 3.28 | (0.4) |
| KFB | RK4(5) | Reg | Filter | 0.3 | 0.756 | (0.05) | 0.690 | (0.05) | 0.306 | (0.05) | 6690 | (1630) | 3.25 | (0.4) |
| KTF | RK4(5) | Reg | Custom | 0.1 | 0.542 | (0.07) | 0.510 | (0.06) | 0.166 | (0.09) | 528 | (165) | 6.21 | (0.4) |
| KTF | RK4(5) | Reg | Custom | 0.2 | 0.578 | (0.07) | 0.546 | (0.07) | 0.179 | (0.06) | 1653 | (396) | 5.95 | (0.5) |
| KTF | RK4(5) | Reg | Custom | 0.3 | 0.626 | (0.07) | 0.598 | (0.07) | 0.177 | (0.06) | 2403 | (444) | 6.22 | (0.4) |
| KTF | RK4(5) | Reg | Filter | 0.1 | 0.672 | (0.09) | 0.607 | (0.09) | 0.280 | (0.06) | 1206 | (415) | 3.93 | (0.3) |
| KTF | RK4(5) | Reg | Filter | 0.2 | 0.705 | (0.07) | 0.639 | (0.06) | 0.293 | (0.05) | 3702 | (1036) | 3.96 | (0.3) |
| KTF | RK4(5) | Reg | Filter | 0.3 | 0.738 | (0.05) | 0.671 | (0.05) | 0.305 | (0.05) | 5955 | (1327) | 3.82 | (0.4) |

TABLE II

RESULTS FOR EXPERIMENTS ON THE SIM DATA SET FOR THE PARTICLE FILTER (PF), AND FORWARD-BACKWARD (FB), KERNEL FORWARD-BACKWARD (KFB) AND KERNEL TWO-FILTER (KTF) SMOOTHERS, USING BOOTSTRAP (BOOT), AUXILIARY (AUX) AND REGULARISED (REG) STRATEGIES FOR THE FORWARD FILTER, AND EULER-MARUYAMA (EM1) AND FEHLBERG (RK4(5)) NUMERICAL INTEGRATORS. FOR KERNEL-BASED METHODS, $k$ GIVES THE CONSTANT FACTOR IN THE BANDWIDTH USED (SEE EQN. 8). ROOT MEAN SQUARE ERROR (RMSE) AGAINST GROUND TRUTH, EFFECTIVE SAMPLE SIZE (ESS) AND RUNTIME (IN WALLCLOCK SECONDS) ARE GIVEN AS MEANS (AND STANDARD DEVIATIONS) ACROSS 40 RUNS WITH DIFFERENT RANDOM NUMBER SEEDS USING $P = 500$ PARTICLES. RUNTIMES FOR SMOOTHERS DO NOT INCLUDE THAT OF THE PRECEDING FILTER.

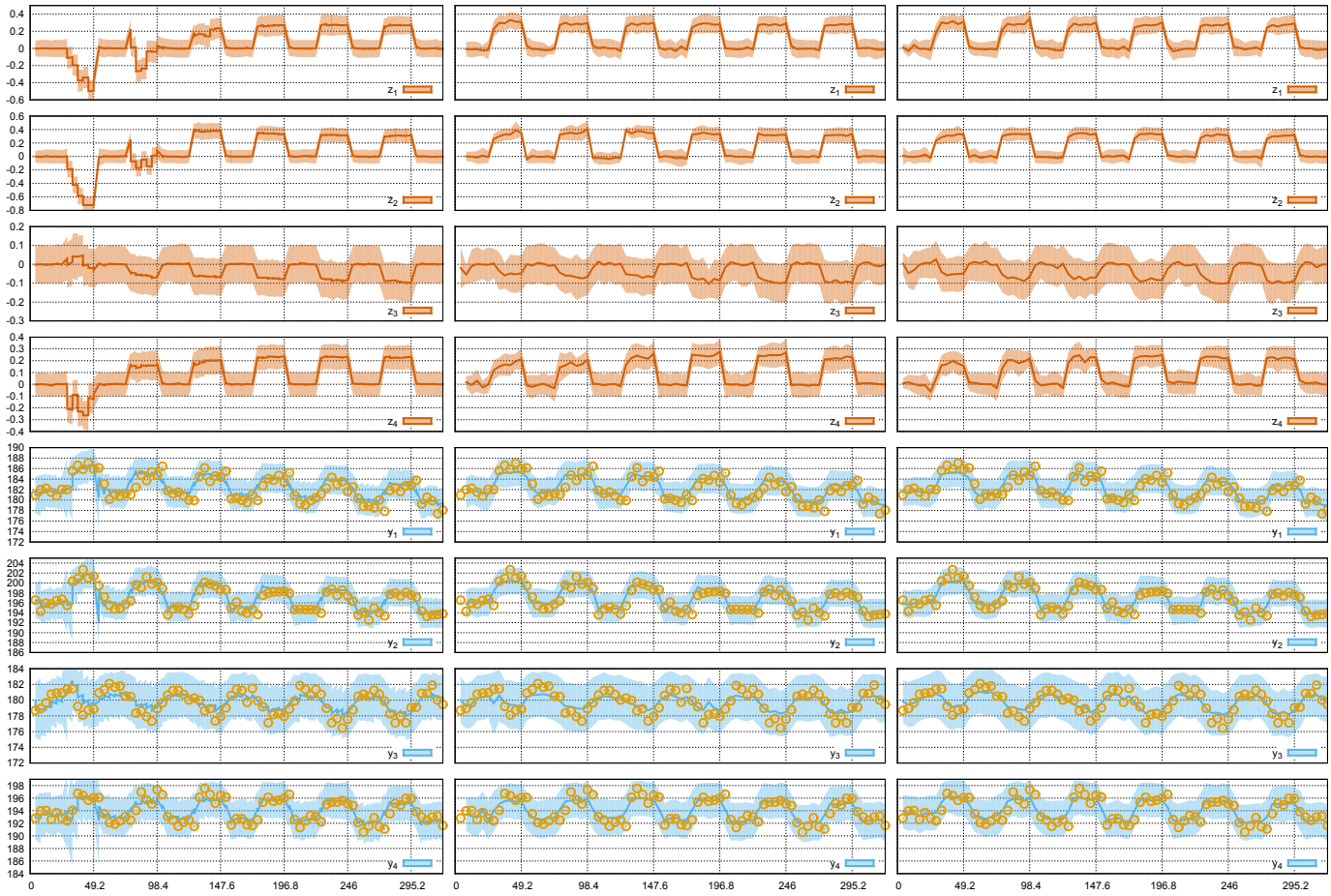


Fig. 2. Regularised particle filter (first column), kernel forward-backward smoother (second column) and kernel two-filter smoother (third column) results for the SessFX data set: deconvolved neural activity (first four rows) and predicted BOLD signal (final four rows, actual observations as circles). Solid lines represent means and shaded area two standard deviations either side, with vertical grid lines delimiting the periods of the box-car input. The custom proposal with a bandwidth of $0.1h_{\text{opt}}$ has been used.

order RK4(5) integrator do so approximately five times faster than those using the low-order EM1.

All smoothers improve on the accuracy of their base particle filter, as should be expected. Runtime of the kernel smoothers represents a major improvement over that of the forward-backward smoother. The kernel forward-backward smoother with filter proposal performs fastest overall – to be expected given the additional optimisations available in this case [30]. Some caution should be taken in interpreting runtime results given that implementations have not been thoroughly optimised or tuned for the underlying hardware. However, the faster performance of the kernel smoothers is consistent with expectations given algorithmic complexity, recalling that they are $\mathcal{O}(T \lg P \lg P)$ in the best case to $\mathcal{O}(TP^2)$ for the forward-backward smoother. Fig. 3 seems to confirm this.

Accuracy of the kernel smoothers compared to the forward-backward smoother is largely comparable. One result stands out: that of the kernel forward-backward smoother at $k = .1$ with the filter proposal. This achieves significantly better RMSE, while achieving a high ESS that musters confidence. Both kernel smoothers achieve noticeably better RMSE with the custom proposal at $k = .1$, but a low ESS indicates that this fit may rely on only a few heavily weighted particles close to the ground truth trajectory, without adequately supporting the whole distribution. The custom proposal achieves less error on parameter estimates – to be expected given it is based on parameter estimates from the last time step of the filter.

The regularised particle filters and kernel smoothers introduce error in the form of kernel density approximations in addition to the sampling error inherent in all methods. Table II gives some indication of the magnitude of this, noting that RMSE results given are means and standard deviations over multiple runs. Comparing, say, the bootstrap and auxiliary against the regularised particle filters, with the RK4(5) integrator, a small increase in RMSE is apparent as bandwidth of the regularised method increases. This is dwarfed by the total RMSE, however, suggesting that the additional error is only a small addition to sampling error.

Kernel smoothers using the custom proposal rest on significantly less ESS than those utilising a filter proposal. This is not overly surprising: given that the custom proposal is bimodal, only one mode will match the smooth density well at each time point, so we might expect ESS to roughly halve. The custom proposal improves RMSE for the kernel two-filter smoother, but only at the highest bandwidth attempted for the kernel forward-backward smoother. Runtime increases in both cases.

Kernel density approximations require selection of an appropriate bandwidth parameter. RMSE noticeably improves for the smoothers as bandwidth decreases in Table II, but ESS also decreases sharply, so that one might question the usefulness of some of the low-bandwidth results. This is to be expected when one considers how the breadth of support for the various densities approximated increases with bandwidth. Larger bandwidths produce broader distributions that are likely to increase ESS and provide greater robustness to degeneracy in extreme cases. The combination of a custom proposal with inflated bandwidth may be a generally good strategy to mitigate degeneracy risks, albeit at some loss of precision.

The parameter estimation regimen employed here treats parameters as slowly moving state variables. If static terms are sought, estimates might be improved by adding shrinkage to the kernel density approximations of the regularised particle filter, compensating for the information loss caused by the artificial dynamic [20]. How this then might work in a smoothing context is not clear, however. At the end of the forward pass we would have the posterior over parameters $p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T})$, and would wish to apply this at all time steps in place of the filter densities $p(\boldsymbol{\theta} \mid \mathbf{y}_{1:n})$. Presumably this would involve a draw from $p(\mathbf{x}_n \mid \mathbf{y}_{1:T}, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathbf{y}_{1:T})$, but no existing samples of this would be available. Arbitrary pairing of parameters $\boldsymbol{\theta}'$ and state $\mathbf{x}'$ from marginals would not work, as forward propagation would then yield transient effects whenever $\mathbf{x}'$ is not a member of the equilibrium dynamic induced by $\boldsymbol{\theta}'$. We have avoided these complexities in the less sophisticated treatment of parameters here. In practice, running multiple smoothers, each conditioned on a posterior parameter sample, may be as viable an option as any other to build a joint posterior over both parameters and state.

While kernel densities are easy to apply, mileage may vary, particularly for high-dimensional models. Other techniques may be worth exploring, such as variational approximations. The novel particle smoothers presented in this work are equally applicable in such cases by replacing the $p_{\mathcal{K}}(\cdot)$ density approximations with an alternative form.

Despite density approximation, the kernel smoothers achieve comparable accuracy to the forward-backward smoother with its exact evaluation of the transition density (up to the additional noise on $s$ introduced in Section III-C). The use of density approximations facilitates higher-order numerical integrators; we might speculate that the former's compromise in the probabilistic model is compensated for by the latter's more accurate simulation of the model dynamics.

## V. Conclusion

This work has presented two novel methods for particle smoothing in state-space models: the kernel forward-backward and kernel two-filter smoothers. These are applicable to a broader class of models than conventional techniques, in particular where the formulation does not provide a closed-form transition density, such as for continuous-time ODE and SDE models. The methods are substantially faster than conventional techniques, while accuracy is not compromised.

## Acknowledgements

The authors would like to thank David McGonigle for advice on preparation of the SessFX data set.

## References

[1] E. Hairer, S. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd ed. Springer-Verlag, 1993.

[2] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*. Springer, 1992.

[3] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 80, pp. 35–45, 1960.

[4] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management*, 1997.
[5] E. A. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of IEEE Symposium on Adaptive Systems for Signal Processing Communications and Control*, 2000, pp. 153–158.
[6] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo Methods in Practice*. Springer, 2001.
[7] C. W. Gardiner, *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*, 3rd ed. Springer, 2004.
[8] A. Greiner, W. Strittmatter, and J. Honerkamp, "Numerical integration of stochastic differential equations," *Journal of Statistical Physics*, vol. 51, pp. 95–107, 1988.
[9] J. Wilkie, "Numerical methods for stochastic differential equations," *Physical Review E*, vol. 70, 2004.
[10] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, pp. 19–26, 1980.
[11] E. Fehlberg, "Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems," National Aeronautics and Space Administration, Tech. Rep. R-315, 1969.
[12] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings-F*, vol. 140, pp. 107–113, 1993.
[13] M. Briers, A. Doucet, and S. Maskell, "Smoothing algorithms for state-space models," *Annals of the Institute of Statistical Mathematics*, vol. 62, pp. 61–89, 2010.
[14] P. Fearnhead, D. Wyncoll, and J. Tawn, "A sequential smoothing algorithm with linear computational cost," *Biometrika*, vol. 97, pp. 447–464, 2010.
[15] M. Klaas, M. Briers, N. de Freitas, A. Doucet, S. Maskell, and D. Lung, "Fast particle smoothing: If I had a million particles," *Proceedings of the 23rd International Conference on Machine Learning*, 2006.
[16] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, pp. 1–25, 1996.
[17] M. Isard and A. Blake, "A smoothing filter for Condensation," *Proceedings of the 5th European Conference on Computer Vision*, vol. 1, pp. 767–781, 1998.
[18] L. Murray and A. Storkey, "Continuous time particle filtering for fMRI," *Advances in Neural Information Processing Systems*, vol. 20, 2008, to appear.
[19] C. Musso, N. Oudjane, and F. L. Gland, *Sequential Monte Carlo Methods in Practice*. Springer, 2001, ch. Improving Regularised Particle Filters.
[20] J. Liu and M. West, *Sequential Monte Carlo Methods in Practice*. Springer, 2001, ch. Combined Parameter and State Estimation in Simulation-Based Filtering, pp. 197–223.
[21] M. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, pp. 590–599, 1999.
[22] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan, "The unscented particle filter," *Advances in Neural Information Processing Systems*, vol. 13, 2000.
[23] A. Golightly and D. J. Wilkinson, "Bayesian sequential inference for nonlinear multivariate diffusions," *Statistics and Computing*, vol. 16, pp. 323–338, 2006.
[24] P. Fearnhead, O. Papaspiliopoulos, and G. O. Roberts, "Particle filters for partially observed diffusions," *Journal of the Royal Statistical Society Series B*, vol. 70, pp. 755–777, 2008.
[25] A. Beskos, O. Papaspiliopoulos, G. Roberts, and P. Fearnhead, "Exact and efficient likelihood-based inference for discretely observed diffusion processes (with discussion)," *Journal of the Royal Statistical Society Series B*, vol. 68, pp. 333–382, 2006.
[26] C. Archambeau, D. Cornford, M. Opper, and J. Shawe-Taylor, "Gaussian process approximations of stochastic differential equations," *Journal of Machine Learning Research Workshop and Conference Proceedings*, vol. 1, pp. 1–16, 2007.
[27] C. Archambeau, M. Opper, Y. Shen, D. Cornford, and J. Shawe-Taylor, "Variational inference for diffusion processes," in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008.
[28] B. Silverman, *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.
[29] A. G. Gray and A. W. Moore, "'N-body' problems in statistical learning," *Advances in Neural Information Processing Systems*, vol. 13, 2001.
[30] L. Murray, "Bayesian learning of continuous time dynamical systems," Ph.D. dissertation, School of Informatics, University of Edinburgh, 2009. [Online]. Available: http://www.indii.org/research/
[31] R. B. Buxton, *Introduction to Functional Magnetic Resonance Imaging: Principles & Techniques*. Cambridge University Press, 2002.
[32] R. S. Frackowiak, K. J. Friston, C. D. Frith, R. J. Dolan, C. J. Price, S. Zeki, J. Ashburner, and W. Penny, *Human Brain Function*, 2nd ed. Elsevier Academic Press, 2004.
[33] S. Ogawa, T. Lee, A. Kay, and D. Tank, "Brain magnetic resonance imaging with contrast dependent on blood oxygenation," *Proceedings of the National Academy of Sciences*, vol. 87, pp. 9868–9872, 1990.
[34] D. R. Gitelman, W. D. Penny, J. Ashburner, and K. J. Friston, "Modeling regional and psychophysiologic interactions in fMRI: the importance of hemodynamic deconvolution," *NeuroImage*, vol. 19, pp. 200–207, 2003.
[35] K. Friston, L. Harrison, and W. Penny, "Dynamic causal modelling," *NeuroImage*, vol. 19, pp. 1273–1302, 2003.
[36] R. B. Buxton, E. C. Wong, and L. R. Frank, "Dynamics of blood flow and oxygenation changes during brain activation: The balloon model," *Magnetic Resonance in Medicine*, vol. 39, pp. 855–864, 1998.
[37] D. Glaser, K. Friston, A. Mechelli, R. Turner, and C. Price, *Human Brain Function*. Elsevier, 2004, ch. 41, pp. 823–842.
[38] J. B. Mandeville, J. J. A. Marot, C. Ayata, G. Zaharchuk, M. A. Moskowitz, B. R. Rosen, and R. M. Weisskoff, "Evidence of a cerebrovascular postarteriole Windkessel with delayed compliance," *Journal of Cerebral Blood Flow & Metabolism*, vol. 19, pp. 679–689, 1999.
[39] K. J. Friston, A. Mechelli, R. Turner, and C. J. Price, "Nonlinear responses in fMRI: The balloon model, Volterra kernels, and other hemodynamics," *NeuroImage*, vol. 12, pp. 466–477, 2000.
[40] Wellcome Department of Imaging Neuroscience, "Statistical parametric mapping," Online at www.fil.ion.ucl.ac.uk/spm/, 2006. [Online]. Available: www.fil.ion.ucl.ac.uk/spm/
[41] G. Kitagawa, "A self-organising state-space model," *Journal of the American Statistical Association*, vol. 93, pp. 1203–1215, 1998.
[42] J. S. Liu and R. Chen, "Blind deconvolution via sequential imputations," *Journal of the American Statistical Association*, vol. 90, pp. 567–576, 1995.

**Lawrence Murray** is a research scientist with CSIRO Mathematics, Informatics and Statistics in Perth, Western Australia. He received the bachelor's degree in software engineering from the Australian National University in 2004, before the Ph.D. degree in informatics at the University of Edinburgh in 2009. His research interests include Bayesian inference in state-space models, machine learning, high performance computing and general-purpose computation on graphical processing units (GPGPU). He is a member of the IEEE.

**Amos Storkey** is a lecturer in the School of Informatics, University of Edinburgh. He received an MA and MMath in Mathematics from Trinity College, Cambridge and an MSc in Transport from Imperial College/UCL University of London. His Ph.D. was in Neural Networks and Gaussian processes, also from Imperial College. Prior to his lectureship he was on a Research Fellowship in Edinburgh, funded by Microsoft Research. His research focus is in the development of probabilistic techniques and representations for domains with significant prior structure, especially those related to image analysis and sequential data.