Nonparametric Adjoint-Based Inference for Stochastic Differential Equations

Harish S. Bhat and R. W. M. A. Madushani
Applied Mathematics Unit
School of Natural Sciences
University of California, Merced
Merced, California 95343
hbhat@ucmerced.edu and rmadushani@ucmerced.edu

***Abstract*—We develop a nonparametric method to infer the drift and diffusion functions of a stochastic differential equation. With this method, we can build predictive models starting with repeated time series and/or high-dimensional longitudinal data. Typical use of the method includes forecasting the future density or distribution of the variable being measured. The key innovation in our method stems from efficient algorithms to evaluate a likelihood function and its gradient. These algorithms do not rely on sampling; instead, they use repeated quadrature and the adjoint method to enable the inference to scale well as the dimensionality of the parameter vector grows. In simulated data tests, when the number of sample paths is large, the method does an excellent job of inferring drift functions close to the ground truth. We show that even when the method does not infer the drift function correctly, it still yields models with good predictive power. Finally, we apply the method to real data on hourly measurements of ground level ozone, showing that it is capable of reasonable results.**

***Keywords*-nonparametric inference, stochastic differential equations, adjoint method**

I. Introduction

Consider a noisy, time-dependent system with a scalar observable. We define a time series to be a sequence of measurements, at regularly spaced times, of one sample path (or one realization) of the system. Suppose we have access to many sample paths; we observe all of the paths at regularly spaced times, thereby obtaining many time series. Starting with this type of data, we consider two questions:

1) How can we use all of the time series data to infer a fundamental equation of motion?
2) How well does such a model perform in predicting the future distribution of the observable variable?

We frame this problem as a non-parametric inference problem for the drift and diffusion functions of a general stochastic differential equation (SDE):

$$dX_t = f(X_t)dt + g(X_t)dW_t. \tag{1}$$

Here X_t is a scalar stochastic process and W_t is the Wiener process. In this SDE, the drift function f and the diffusion function g are considered to be unknown. We assume we observe (1) at times $t_j = j\Delta t$ for some fixed time-step $\Delta t > 0$, for $j = 0, \ldots, M$. At each time t_j, we collect ν samples of X_{t_j} and label these samples as $x_j \in \mathbb{R}^\nu$. We let $\mathbf{x} = x_0, x_1, \ldots, x_M$ denote all of the collected observations. Our goal is to use $\mathbf{x}$ to infer f and g from (1). We can then use (1) to predict the future distribution of X_t.

The type of data we have described arises in a diverse set of practical examples: time series measurements of large numbers of neurons [1], bid-ask prices in online auctions [2], or the position of a moving robotic arm [3]. The data we describe can also be viewed as a particular case of longitudinal high-dimensional data, of importance in a variety of biomedical contexts [4], [5].

The crux of our method is an algorithm for the efficient computation of a log likelihood function and its gradient. This likelihood function is the density $p(\mathbf{x}|\boldsymbol{\theta})$ under the model (1)—here $\boldsymbol{\theta}$, the parameter vector in our statistical model, is a vector of Hermite basis expansion coefficients that give finite-dimensional representations of the drift and diffusion functions that we seek to infer. We evaluate the likelihood deterministically, without recourse to sampling-based methods. The mathematical innovation in our work consists of the way in which we evolve densities forward in time. Rather than solve a partial differential equation (e.g., the Fokker-Planck equation), we iteratively apply quadrature to the Chapman-Kolmogorov equation associated with a time-discretization of (1). In this way, we evaluate the Markovian pieces $p(x_{j+1}|x_j, \boldsymbol{\theta})$ that make up the overall likelihood. We call our method density tracking by quadrature (DTQ).

While the DTQ method is by itself efficient, what is most relevant for the current nonparametric estimation problem is our ability to combine the DTQ method with the adjoint method to compute the gradient $\nabla_{\boldsymbol{\theta}} p(\mathbf{x}|\boldsymbol{\theta})$. The adjoint method enables us to compute this gradient with a computational cost (in time) that does not scale with the dimension of $\boldsymbol{\theta}$. This is important for nonparametric estimation, because we do not know the dimension of $\boldsymbol{\theta}$ a priori—indeed, it may be quite large. Of course, we can also apply the methods developed in this paper to parametric inference problems where the functional forms of f and g are known. Such problems are much more commonly studied in the literature—see [6]–[8].

Equipped with efficient algorithms to compute the likelihood and its gradient, we use quasi-Newton optimization

methods to compute $\boldsymbol{\theta}$, and by implication, infer the drift (f) and diffusion (g) functions from data.

In prior work, we have shown that the DTQ method computes a convergent approximation to the true density of the SDE (1). The convergence requires regularity of both f and g, together with restrictions on the rates at which the temporal and spatial grid spacing tends to zero [9]. In the context of parametric inference for the SDE (1), we have shown how to use the DTQ method to compute both the log likelihood and its gradient [10]. However, this earlier calculation of the gradient proceeds via a direct method, rather than the adjoint method considered in the present work. Finally, we have also used the DTQ method for Bayesian inference for both one- and two-dimensional SDE models [11], [12]. None of our past work considered the nonparametric inference of the drift and diffusion functions f and g.

Our work is related to but distinct from prior work carried out in both the machine learning and statistics communities. First, nonparametric inference of the drift function for the SDE model (1) has been approached using Gaussian process approximations [13], [14]; it is not immediately clear how well such an approach will carry over to the setting where we have multiple sample paths, each observed on a relatively coarse time scale. In [2], the authors seek to infer an empirical SDE from the type of data just described. However, the class of stochastic differential equations considered there is rather different than the drift-diffusion SDE (1) considered here. The work of [15] also fits the context of using data to infer general stochastic models; here the authors propose a model whose stochastic structure is completely unspecified. In contrast, by constraining the model to be (1), we form a connection to statistical physics, e.g., the Fokker-Planck and Langevin equations. In this framework, our question is: given enough realizations of a stochastic system, how well can we infer its potential function?

Moving outside the SDE framework, there are several approaches to use high-dimensional longitudinal data to build predictive models: a tree-based method [16], methods from functional data analysis [17], and time series methods [3], to name just a few. The purpose of the present work is to introduce and study our method; in future work, we plan to conduct a detailed comparison of our SDE-based method against these other methods. Note that all of our codes are available at the following URL: https://github.com/hbhat4000/sdeinference—see the "adjoint" directory.

We detail our mathematical methods in Section II, after which we provide brief notes on its R implementation in Section III. In Section IV, we report on several tests of our method using both artificial and real data sets. We show that the method does an excellent job of accurately inferring drift functions in the Hermite context, and we also show that the method can be used to generate predictive models. This is especially true if one is interested in forecasting the future distribution (e.g., the PDF or CDF) of a particular variable. Finally, we apply the method to develop an SDE model for ground level ozone pollution. The model performs reasonably well on a distributional forecasting problem.

II. Methodology

We assume that f and g are square-integrable, i.e., $f, g \in L^2(\mathbb{R})$. The Hermite functions $\{\psi_i(x)\}_{i=0}^{\infty}$ form an orthonormal basis of $L^2(\mathbb{R})$; additionally, we have that any $\phi \in L^2(\mathbb{R})$ can be represented as an expansion in Hermite functions,

$$\phi(x) = \sum_{i=0}^{\infty} c_i \psi_i(x), \tag{2}$$

where the coefficient c_j can be computed via $c_j = \int \phi(x)\psi_i(x)\,dx$. The j-th Hermite function is defined by

$$\psi_j(x) = (-1)^j (2^j j! \sqrt{\pi})^{-1/2} e^{x^2/2} \frac{d^j}{dx^j} e^{-x^2}. \tag{3}$$

For further definitions and properties of the Hermite functions, we refer the reader to [18].

To finite-dimensionalize the inference problem for (1), we expand the unknown functions f and g in the Hermite basis and then truncate the expansions:

$$f(x) \approx \sum_{i=0}^{N_f} \theta_i \psi_i(x) = \hat{f}(x;\boldsymbol{\theta}) \tag{4a}$$

$$g(x) \approx \sum_{i=0}^{N_g} \theta_{N_f+1+i} \psi_i(x) = \hat{g}(x;\boldsymbol{\theta}). \tag{4b}$$

These approximations of f and g induce an approximation of the original SDE (1) by the approximate SDE

$$dX_t = \hat{f}(X_t;\boldsymbol{\theta})dt + \hat{g}(X_t;\boldsymbol{\theta})dW_t. \tag{5}$$

Properties of the Hermite functions guarantee that both $\hat{f}$ and $\hat{g}$ and their derivatives are globally bounded. This is sufficient for the existence of a unique solution X_t of (5); moreover, we are guaranteed that for $t > 0$, the random variable X_t has a density function $p(x,t)$.

Let $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_{N_f}, \theta_{N_f+1}, \ldots, \theta_{N_f+N_g+1})$. Then the inference problem consists of using the data $\mathbf{x}$ to compute $\boldsymbol{\theta}$. We now describe a maximum likelihood method for doing this. In Section II-A, we start by explaining the inference method for the case where $\nu = 1$, i.e., we have collected only one observation at each point in time t_j. Later, in Section II-C, we discuss the generalization of the method for the case where we have many observations at each point in time ($\nu > 1$).

A. Inference for one time series

We compute the maximum likelihood estimator of $\boldsymbol{\theta}$ by minimizing the negative log likelihood of of the observed

time series given by

$$-\log \mathcal{L}(\boldsymbol{\theta}) = -\sum_{j=0}^{M-1} \log p_{X_{t_{j+1}}}(x_{j+1}|X_{t_j} = x_j;\boldsymbol{\theta}), \quad (6)$$

where $p_{X_{t_{j+1}}}(x_{j+1}|X_{t_j} = x_j;\boldsymbol{\theta})$ is the conditional density of $X_{t_{j+1}} = x_{j+1}$ given $X_{t_j} = x_j$. In our work, we approximate the transition density using a method called density tracking by quadrature (DTQ). The first step of the DTQ method is to discretize (5) in time using the Euler-Maruyama scheme. We select an internal time step h, a small fraction of Δt, and set $hF = \Delta t$ where $F \in \mathbb{Z}$ and $F \geq 2$. Then the Euler-Maruyama discretization gives

$$\widetilde{X}_{j+n/F} = \widetilde{X}_{j+(n-1)/F} + \hat{f}(\widetilde{X}_{j+(n-1)/F};\theta)h + \hat{g}(\widetilde{X}_{j+(n-1)/F};\theta)h^{1/2}Z_{j+n/F}, \quad (7)$$

for $n = 1,\ldots,F$. Here $\{Z_{j+n/F}\}$ is an i.i.d. family of Gaussian random variables with mean 0 and variance 1. The random variable $\widetilde{X}_j$ is intended to approximate X_{t_j} when the index j is an integer. When the index j is not an integer, $\widetilde{X}_j$ represents a random variable that *interpolates in time* between the random variables that have been sampled to give us our data. The idea now is to approximate $p_{X_{t_{j+1}}}(x_{j+1}|X_{t_j} = x_j;\boldsymbol{\theta})$ in (6) with $p_{\widetilde{X}_{j+1}}(x_{j+1}|\widetilde{X}_j = x_j;\boldsymbol{\theta})$. The Chapman-Kolmogorov equation for the Markov chain (7) is:

$$p_{\widetilde{X}_{j+n/F}}(y|\widetilde{X}_j = x_j;\boldsymbol{\theta}) = \int_z p_{\widetilde{X}_{j+n/F}}(y|\widetilde{X}_{j+(n-1)/F} = z;\boldsymbol{\theta}) \times p_{\widetilde{X}_{j+(n-1)/F}}(z|\widetilde{X}_j = x_j;\boldsymbol{\theta})\,dz. \quad (8)$$

Here y and z represent any value in the state spaces of the random variables $\widetilde{X}_{j+n/F}$ and $\widetilde{X}_{j+(n-1)/F}$, respectively. Now let $G^h_{\boldsymbol{\theta}}(y,z)$ denote the probability density funtion of a Gaussian random variable with mean $z + \hat{f}(z;\boldsymbol{\theta})h$ and variance $\hat{g}(z;\boldsymbol{\theta})^2 h$, evaluated at y. From (7) we observe that, for each $n \in \{1,\ldots,F\}$,

$$p_{\widetilde{X}_{j+n/F}}(y|\widetilde{X}_{j+(n-1)/F} = z;\boldsymbol{\theta}) = G^h_{\boldsymbol{\theta}}(y,z), \quad (9)$$

and (8) becomes

$$p_{\widetilde{X}_{j+n/F}}(y|\widetilde{X}_j = x_j;\boldsymbol{\theta}) = \int_z G^h_{\boldsymbol{\theta}}(y,z)\, p_{\widetilde{X}_{j+(n-1)/F}}(z|\widetilde{X}_j = x_j;\boldsymbol{\theta})\,dz. \quad (10)$$

When $n = 1$ on the right-hand side, we see that conditioning on $\widetilde{X}_j = x_j$ forces $p_{\widetilde{X}_j}(z|\widetilde{X}_j = x_j;\boldsymbol{\theta}) = \delta(z - x_j)$. This enables us to evaluate (10) at $n = 1$ to obtain

$$p_{\widetilde{X}_{j+1/F}}(y|\widetilde{X}_j = x_j;\boldsymbol{\theta}) = G^h_{\boldsymbol{\theta}}(y,x_j). \quad (11)$$

Starting with (11), we can now compute (10) iteratively to obtain the transition density $p_{\widetilde{X}_{j+n/F}}(y|\widetilde{X}_j = x_j;\boldsymbol{\theta})$. We compute (10) using quadrature. We first truncate the infinite domain of the integral to $(-z^A, z^A)$, and introduce the spatial grid spacing $k > 0$ such that $k = (z^A)/A$. Let us use superscripts to denote spatial grid locations, for example, $z^a = ak$ for all integers $a \in [-A, A]$. By applying the trapezoidal rule to the right-hand side of (10), we get

$$p_{\widetilde{X}_{j+n/F}}(y|\widetilde{X}_j = x_j;\boldsymbol{\theta}) \approx k \sum_{a_1=-A}^{A} G^h_{\boldsymbol{\theta}}(y,z^{a_1}) p_{\widetilde{X}_{j+(n-1)/F}}(z^{a_1}|\widetilde{X}_j = x_j;\boldsymbol{\theta}). \quad (12)$$

Let $kG^h_{\boldsymbol{\theta}}(y^{a_2}, z^{a_1})$ be the (a_2, a_1) element of a matrix $K_{(2A+1)\times(2A+1)}$. We also define the a_1-th element of the vector $\hat{p}_{j+n/F}$ by $\hat{p}^{a_1}_{j+n/F} = p_{\widetilde{X}_{j+n/F}}(y^{a_1}|\widetilde{X}_j = x_j;\boldsymbol{\theta})$. Now (12) reduces to matrix-vector multiplication:

$$\hat{p}_{j+n/F} = K\hat{p}_{j+(n-1)/F}. \quad (13)$$

Starting with the initial vector $\hat{p}_{j+1/F}$, given by the right-hand side of (11) discretized on the spatial grid, we apply (13) $F-1$ times to get $\hat{p}_{j+1} = K^{F-1}\hat{p}_{j+1/F}$, where $\hat{p}_{j+1}$ is the approximation of the transition density function $p_{\widetilde{X}_{j+1}}(y|\widetilde{X}_j = x_j;\boldsymbol{\theta})$ on the spatial grid. To find the value of the transition density at $y = x_j$ one could use interpolation on $\hat{p}_{j+1}$. Instead of using interpolation, in our method we first compute $\hat{p}_{j+(F-1)/F}$ using (13), i.e.,

$$\hat{p}_{j+(F-1)/F} = K^{F-2}\hat{p}_{j+1/F}. \quad (14)$$

Now define the vector Γ_{F-1} by $\Gamma^{a_{F-1}}_{F-1} = kG^h_{\theta_0}(x_{j+1}, y^{a_{F-1}})$, where a_{F_1} is any integer between $-A$ and A. Let ^{T} denote transpose. Putting everything together, we obtain, respectively, the transition density and the negative log likelihood:

$$p_{\widetilde{X}_{j+1}}(x_{j+1}|\widetilde{X}_j = x_j,\boldsymbol{\theta}) \approx [\Gamma_{F-1}]^T \hat{p}_{j+(F-1)/F}, \quad (15)$$

$$-\log \mathcal{L}(\boldsymbol{\theta}) \approx -\sum_{j=0}^{M-1} \log\left([\Gamma_{F-1}]^T K^{F-2}\hat{p}_{j+1/F}\right). \quad (16)$$

B. An adjoint-based computation of the gradient

To compute the maximum likelihood estimator of $\boldsymbol{\theta}$ using a numerical optimization technique, we need to compute the gradient of $-\log \mathcal{L}(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. In this section we discuss an adjoint-based approach to compute this gradient of the negative log likelihood. Let us first consider (14), which can be written as a system of block matrices to compute $\bar{\boldsymbol{p}}_{\boldsymbol{j+1}} = (\hat{p}^T_{j+1/F}, \cdots, \hat{p}^T_{j+(F-1)/F})^T$ as follows:

$$\bar{\boldsymbol{K}}\bar{\boldsymbol{p}}_{\boldsymbol{j+1}} = \boldsymbol{v}_{\boldsymbol{j+1}}, \quad (17)$$

where $\boldsymbol{v}_{\boldsymbol{j+1}} = (\hat{p}^T_{j+1/F}, \cdots, \boldsymbol{0})^T$, and

$$\bar{\boldsymbol{K}} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ -K & \boldsymbol{I} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & -K & \boldsymbol{I} & \cdots & \boldsymbol{O} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & -K & \boldsymbol{I} \end{pmatrix}.$$

Here $\mathbf{0}$ denotes a zero row vector of the same length as $\hat{p}_{j+1/F}$. Also, $\boldsymbol{O}$ denotes a zero matrix and $\boldsymbol{I}$ denotes an identity matrix, both with the same dimension as that of the matrix K. Therefore, the block matrix $\bar{\boldsymbol{K}}$ has dimension $(2A+1)(F-1) \times (2A+1)(F-1)$. Now define $\boldsymbol{w_{j+1}} = (\mathbf{0}, \mathbf{0}, \cdots, \mathbf{0}, \Gamma_{F-1}^T)^T$, a vector of the same length as $\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}$. Then we can write the negative log likelihood in (16) as

$$-\mathcal{L}(\theta) \approx -\sum_{j=0}^{M-1} \log\left(\boldsymbol{w_{j+1}}^T \bar{\boldsymbol{p}}_{\boldsymbol{j+1}}\right).$$

Let us define the Lagrangian as follows:

$$\mathbf{L}(\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}, \boldsymbol{\theta}, \bar{\boldsymbol{u}}_{\boldsymbol{j+1}}) = \sum_{j=0}^{M-1} \Big[-\log\left(\boldsymbol{w_{j+1}}^T \bar{\boldsymbol{p}}_{\boldsymbol{j+1}}\right) + \bar{\boldsymbol{u}}_{\boldsymbol{j+1}}^T \left(\bar{\boldsymbol{K}} \bar{\boldsymbol{p}}_{\boldsymbol{j+1}} - \boldsymbol{v_{j+1}}\right) \Big].$$

Here, $\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}$ is the state solution from (17) and $\bar{\boldsymbol{u}}_{\boldsymbol{j+1}}$ is a vector of Lagrange multipliers. Now, by taking the variations of the Lagrangian with respect to $\bar{\boldsymbol{u}}_{\boldsymbol{j+1}}$ and $\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}$, we obtain the state equation,

$$\mathbf{L}_{\bar{\boldsymbol{u}}_{j+1}}(\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}, \boldsymbol{\theta}, \bar{\boldsymbol{u}}_{\boldsymbol{j+1}}) = \bar{\boldsymbol{K}} \bar{\boldsymbol{p}}_{\boldsymbol{j+1}} - \boldsymbol{v_{j+1}} = 0, \tag{18}$$

and the adjoint equation,

$$\mathbf{L}_{\bar{\boldsymbol{p}}_{j+1}}(\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}, \boldsymbol{\theta}, \bar{\boldsymbol{u}}_{\boldsymbol{j+1}}) = -\frac{\boldsymbol{w_{j+1}}}{\boldsymbol{w_{j+1}}^T \bar{\boldsymbol{p}}_{\boldsymbol{j+1}}} + \bar{\boldsymbol{K}}^T \bar{\boldsymbol{u}}_{\boldsymbol{j+1}} = 0. \tag{19}$$

In the above equations, we have used the subscript on $\mathbf{L}$ to denote the variables with respect to which we have differentiated. Recall that $\boldsymbol{\theta}$ is the vector of coefficients in the drift and the diffusion of (5), i.e., $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_{N_f+N_g+1})$. We compute the gradient of the log likelihood by taking the derivative of the Lagrangian with respect to each θ_i:

$$\mathbf{L}_{\theta_i}(\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}, \boldsymbol{\theta}, \bar{\boldsymbol{u}}_{\boldsymbol{j+1}}) = -\sum_{j=0}^{M-1} \frac{\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}^T \frac{\partial \boldsymbol{w_{j+1}}}{\partial \theta_i}}{\boldsymbol{w_{j+1}}^T \bar{\boldsymbol{p}}_{\boldsymbol{j+1}}} + \sum_{j=0}^{M-1} Q^T \bar{\boldsymbol{u}}_{\boldsymbol{j+1}} = -\frac{\partial \mathcal{L}(\theta)}{\partial \theta_i}, \tag{20}$$

where

$$Q = \frac{\partial \bar{\boldsymbol{K}}}{\partial \theta_i} \bar{\boldsymbol{p}}_{\boldsymbol{j+1}} - \frac{\partial}{\partial \theta_i} \boldsymbol{v_{j+1}}. \tag{21}$$

We rewrite (20) as

$$-\frac{\partial \mathcal{L}(\theta)}{\partial \theta_i} = -\sum_{j=0}^{M-1} \frac{\hat{p}_{j+(F-1)/F}^T \frac{\partial \Gamma_{F-1}}{\partial \theta_i}}{\Gamma_{F-1}^T \hat{p}_{j+(F-1)/F}} - \sum_{j=0}^{M-1} \left[\frac{\partial}{\partial \theta_i} \hat{p}_{j+1/F}\right]^T u_{j+1/F} - \sum_{j=0}^{M-1} \sum_{\ell=1}^{F-2} \left[\frac{\partial K}{\partial \theta_i} \hat{p}_{j+\ell/F}\right]^T u_{j+(\ell+1)/F}. \tag{22}$$

Let $\bar{\boldsymbol{u}}_{\boldsymbol{j+1}} = (u_{j+1/F}^T, u_{j+2/F}^T, \cdots, u_{j+(F-1)/F}^T)^T$. Then, the adjoint equation (19) can be written as the following temporally evolving system of equations, which can be solved backward in time for $n = 1, \cdots, F-2$:

$$u_{j+(F-(n+1))/F} - K^T u_{j+(F-n)/F} = \mathbf{0}, \tag{23a}$$

$$u_{j+(F-1)/F} = \frac{1}{\Gamma_{F-1}^T \hat{p}_{j+(F-1)/F}} \Gamma_{F-1}, \tag{23b}$$

The adjoint method to compute the gradient can now be summarized. For each fixed j, the procedure is as follows.

1 Given the unknown parameter vector $\boldsymbol{\theta}$, solve the state/forward problem equation (13) to find $\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}$. This is the same as solving the time evolution system in (14).
2 Given $\boldsymbol{\theta}$ and $\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}$, solve the adjoint equation (19) to find $\bar{\boldsymbol{u}}_{\boldsymbol{j+1}}$. This is same as solving the time evolution system in (23a-23b).
3 With $\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}$, $\bar{\boldsymbol{u}}_{\boldsymbol{j+1}}$, use (22) to compute the gradient.

We follow this procedure for each Markovian piece of the likelihood $p(x_{j+1}|x_j, \boldsymbol{\theta})$. For each piece, we need F steps in time to solve for the state $\bar{\boldsymbol{p}}$ and F steps in time to solve for the adjoint $\bar{\boldsymbol{u}}$. In total, across the entire time series, we need $2MF$ steps to compute the log likelihood and its gradient. In particular, note that the number of steps in time does not depend on the dimension of $\boldsymbol{\theta}$. This is in sharp contrast to a direct (i.e., non-adjoint) method computation of the gradient, in which one would have to perform a forward evolution in time (analogous to (14)) for each component of $\boldsymbol{\theta}$. Such a method would require $MF(1 + |\boldsymbol{\theta}|)$ steps to compute the log likelihood and its gradient, where $|\boldsymbol{\theta}| = N_f + N_g + 2$ is the dimension of $\boldsymbol{\theta}$.

C. Inference for many time series

In this section, we reinterpret $\mathbf{x} = x_0, x_1, \ldots, x_M$ as a sequence of vector-valued observations. For each $s = 1, 2, \ldots, \nu$, the sequence $x_0^s, x_1^s, \ldots, x_M^s$ is a scalar time series. With these changes, the log likelihood becomes

$$\log \mathcal{L}(\boldsymbol{\theta}) \approx \sum_{j=0}^{M-1} \sum_{a_F=1}^{\nu} \log p_{\widetilde{X}_{j+1}}(x_{j+1}^{a_F} | \widetilde{X}_j = \{x_j^s\}_{s=1}^{\nu}; \boldsymbol{\theta}). \tag{24}$$

The derivation of the above negative log likelihood is identical to that given in Section II-A. The only real change is that when $\nu > 1$, we use the samples $\{x_j^s\}_{s=1}^{\nu}$ of the random variable $\widetilde{X}_j$ to estimate the density of X_j as follows:

$$p_{\widetilde{X}_j}(z) \approx \frac{1}{\nu} \sum_{q=1}^{\nu} \delta(z - x_j^q). \tag{25}$$

This approximation is a density estimate that corresponds to the spatial derivative of the empirical cumulative distribution function of the samples. By logic analogous to (11) in Section II-A, we can then compute the initial density function

$p_{\widetilde{X}_{j+1}}(y|\widetilde{X}_j = \{x_j^s\}_{s=1}^{\nu}; \boldsymbol{\theta})$ by

$$p_{\widetilde{X}_{j+1/F}}(y|\widetilde{X}_j = x_j, \theta) = \int_z G_\theta^h(y,z) p_{\widetilde{X}_j}(z)\, dy \approx \frac{1}{\nu}\sum_{q=1}^{\nu} G_\theta^h(y, x_j^q). \tag{26}$$

We calculate the negative log likelihood by repeatedly applying (8), starting with the initial density given by (26) for $n = 2$. We make the approximation (25) so that we can evolve the density along each sample path with the same initial condition. Otherwise, we would have to repeat the calculation (16) ν times.

Now let us redefine $\hat{p}_{j+1/F}$ such that its a_1-th element is (26) evaluated at the spatial grid point y^{a_1}. Then we can use (14) to compute $\hat{p}_{j+(F-1)/F}$. We also redefine Γ_{F-1} to be a matrix of dimension $\nu \times (2A+1)$ whose (a_F, a_{F-1}) entry is $\Gamma_{F-1}^{a_F, a_{F-1}} = kG_\theta^h(x_{j+1}^{a_F}, y^{a_{F-1}})$. Note that the superscript in $x_{j+1}^{a_F}$ is used to denote the a_F-th sample observation taken at time t_{j+1}, whereas the superscript in $y^{a_{F-1}}$ denotes the spatial grid location. With these changes, (16) becomes

$$-\log \mathcal{L}(\boldsymbol{\theta}) \approx -\sum_{j=0}^{M-1}\sum_{a_F=1}^{\nu} \log\left(\Gamma_{F-1}\hat{p}_{j+(F-1)/F}\right)_{a_F}. \tag{27}$$

D. Gradient computation

To compute the gradient using the adjoint method, we write the negative log likelihood in (27) as

$$-\mathcal{L}(\theta) \approx -\sum_{j=0}^{M-1}\sum_{a_F=1}^{\nu} \log\left(\mathbf{e}_{\mathbf{a_F}}^{\mathbf{T}} \boldsymbol{W_{j+1}} \bar{\boldsymbol{p}}_{\boldsymbol{j+1}}\right), \tag{28}$$

where $\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}$ comes from (17), and $\mathbf{e}_{\mathbf{a_F}}$ is the a_F-th canonical basis unit vector. Consider a row vector $\gamma_{\mathbf{aF}}$, the a_F-th row of the matrix Γ_{F-1}. Then, each aF-th row of the matrix $\boldsymbol{W_{j+1}}$ consists of the vector $(\mathbf{0}, \mathbf{0}, \cdots, \mathbf{0}, \gamma_{\mathbf{aF}})$, where $\mathbf{0}$ denotes a zero row vector of length $2A+1$. Note that $\boldsymbol{W_{j+1}}$ is of dimension $\nu \times (2A+1)(F-1)$.

Now the Lagrangian is given by

$$L(\theta) = \sum_{j=0}^{M-1}\Bigg[-\sum_{a_F=1}^{\nu}\log\left(\mathbf{e}_{\mathbf{a_F}}^{\mathbf{T}} \boldsymbol{W_{j+1}} \bar{\boldsymbol{p}}_{\boldsymbol{j+1}}\right) + \bar{\boldsymbol{u}}_{\boldsymbol{j+1}}{}^T\left(\bar{\boldsymbol{K}}\bar{\boldsymbol{p}}_{\boldsymbol{j+1}} - \boldsymbol{v_{j+1}}\right)\Bigg].$$

As a result, (18) remains the same. However, (19) becomes

$$\mathbf{L}_{\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}}(\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}, \boldsymbol{\theta}, \bar{\boldsymbol{u}}_{\boldsymbol{j+1}}) = -\sum_{a_F=1}^{\nu}\frac{\boldsymbol{W_{j+1}^T}\mathbf{e}_{\mathbf{a_F}}}{\mathbf{e}_{\mathbf{a_F}}^{\mathbf{T}}\boldsymbol{W_{j+1}}\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}} + \bar{\boldsymbol{K}}^T\bar{\boldsymbol{u}}_{\boldsymbol{j+1}} = 0. \tag{29}$$

Let us denote the quantity $\mathbf{e}_{\mathbf{a_F}}^{\mathbf{T}}\boldsymbol{W_{j+1}}\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}$ by a constant c_{a_F} that depends on a_F. Then, $\sum_{a_F=1}^{\nu}\frac{\boldsymbol{W_{j+1}^T}\mathbf{e}_{\mathbf{a_F}}}{\mathbf{e}_{\mathbf{a_F}}^{\mathbf{T}}\boldsymbol{W_{j+1}}\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}}$ is a vector of length $(2A+1)(F-1)$ given by $(\mathbf{0}, \mathbf{0}, \cdots, \mathbf{y}^T)^T$, where $\mathbf{y} = (\sum_{a_F=1}^{\nu}\frac{\gamma_{a_F}^1}{c_{a_F}}, \cdots, \sum_{a_F=1}^{\nu}\frac{\gamma_{a_F}^{(2A+1)}}{c_{a_F}})^T$. Here $\gamma_{a_F}^{\ell}$ denotes the ℓ-th element of the vector $\gamma_{\mathbf{aF}}$.

Now we can write (29) as the system of equations

$$u_{j+(F-(n+1))/F} - K^T u_{j+(F-n)/F} = \mathbf{0}, \tag{30a}$$

$$u_{j+(F-1)/F} = \mathbf{y}, \tag{30b}$$

which can be solved backward. Equation (20), required to compute the gradient, becomes

$$\mathbf{L}_{\theta_i}(\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}, \boldsymbol{\theta}, \bar{\boldsymbol{u}}_{\boldsymbol{j+1}}) = -\sum_{j=0}^{M-1}\sum_{a_F=1}^{\nu}\frac{\boldsymbol{e_{a_F}}^T\frac{\partial \boldsymbol{W_{j+1}}}{\partial\theta_i}\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}}{\boldsymbol{e_{a_F}}^T\boldsymbol{W_{j+1}}\bar{\boldsymbol{p}}_{\boldsymbol{j+1}}} + \sum_{j=0}^{M-1} Q^T\bar{\boldsymbol{u}}_{\boldsymbol{j+1}} = -\frac{\partial\mathcal{L}(\theta)}{\partial\theta_i}, \tag{31}$$

while (21), required to compute Q^T, remains the same. As a consequence of (31), only the first term in the right hand-side of (22) changes while other two terms remain the same. After all of these changes, the algorithm to compute the gradient of (28) remains the same as explained in Section (II-B). There are only two changes to make: (1) in the second step, we compute the adjoint solution by solving the system in (30a-30b) backwards; (2) we compute the gradient using (22) after replacing the first term by $-\sum_{a_F=1}^{\nu}\frac{\frac{\partial\gamma_{\mathbf{aF}}}{\partial\theta_i}\hat{p}_{j+(F-1)/F}}{\gamma_{\mathbf{aF}}\hat{p}_{j+(F-1)/F}^T}$.

E. Dirichlet penalty term

Thus far, we have assumed that the dimensionality of $\boldsymbol{\theta}$, which by (4) is $N_f + N_g + 2$, has been fixed ahead of time by the user. In any practical data science context, the danger is that by choosing N_f and N_g sufficiently large, one can infer functions $\hat{f}(x)$ and $\hat{g}(x)$ that do not yield predictive models on test data, even though the models may fit the training data arbitrarily well. In many data science settings, we use computational methods such as cross-validation to choose optimal values of parameters such as N_f and N_g. Because our methodology already requires substantial computational effort, we seek an alternative method to choose parameters so as to avoid overfitting. Beyond computational considerations, we also seek a method that naturally constrains $\boldsymbol{\theta}$, guiding the optimizer away from large magnitude coefficients without the use of, e.g., ad hoc box constraints.

We describe this method using $\hat{f}$, but all of our considerations apply to $\hat{g}$ as well.

We begin with the Dirichlet energy of $\hat{f}$:

$$E = \int_{x=-\infty}^{\infty} |\hat{f}'(x)|^2\, dx. \tag{32}$$

Minimizers of E consist of constant functions $\hat{f}$. For a constant $\gamma \geq 0$, we propose to add γE to the negative log likelihood. Minimizing the resulting penalized log likelihood

will yield estimates of $\hat{f}$ that are less oscillatory than would obtained by minimizing the negative log likelihood alone. This is analogous to the penalty term used in natural smoothing splines; the only difference is that we penalize the L^2 norm of $\hat{f}'$ rather than $\hat{f}''$.

Due to properties of the Hermite basis, it is possible to substitute (4a) in (32) and evaluate E in closed form. The main property that we employ is

$$\psi_j'(x) = \sqrt{\frac{j}{2}}\psi_{j-1}(x) - \sqrt{\frac{j+1}{2}}\psi_{j+1}(x).$$

With this, we obtain

$$E(\boldsymbol{\theta}) = \frac{1}{2}\theta_1^2 + \frac{N_f}{2}\theta_{N_f}^2 + \sum_{j=1}^{N_f-1}\left(\sqrt{\frac{j+1}{2}}\theta_{j+1} - \sqrt{\frac{j}{2}}\theta_{j-1}\right)^2 \tag{33}$$

The overall penalized objective function is then

$$J(\boldsymbol{\theta}) = -\log\mathcal{L}(\boldsymbol{\theta}) + \gamma E(\boldsymbol{\theta}) \tag{34}$$

with the negative log likelihood defined by (24) and constant $\gamma \geq 0$.

III. Implementation

Using R, we have implemented the algorithms described in Section II. Specifically, we have coded functions that take as input a trial value of $\boldsymbol{\theta}$ and the data matrix $\mathbf{x}$, and that return as output the penalized objective function (34) and its gradient with respect to $\boldsymbol{\theta}$. We pass these functions to a gradient-based optimizer. In this work, we make use of two R packages for optimization:

1) **nloptr** [19]: this package is an R interface to NLopt, a general-purpose optimization package [20]. From this package, we use the low-storage BFGS method labeled as "NLOPT_LD_LBFGS." Internally, this method uses a line search method to find an optimal step size once the step direction has been chosen.
2) **trustOptim** [21]: this package implements trust region methods for nonlinear optimization. Essentially, the method finds an optimal step direction once a step size (the trust region radius) has been computed. From this package, we use the SR1 (Symmetric Rank 1) method.

Both of the optimization methods are quasi-Newton methods that seek to numerically approximate the Hessian using previously computed gradients. Such methods are well-suited for our problem: the adjoint method described in Section II enables us to compute the gradient of the log likelihood with computational effort that does not depend on the dimensionality of $\boldsymbol{\theta}$.

Note that though the methods described in Section II are valid for nonparametric inference of both f and g, for the present paper, we have implemented the nonparametric inference only for the drift function f. In the present implementation, we treat the diffusion function g as parametric; in the results that follow, we treat g as an unknown constant. Preliminary tests indicate that though simultaneous nonparametric inference of both f and g is possible, as one might expect, such inference requires more sample paths than are required for nonparametric of f only. The present code is written entirely in R; the adjoint-based computation of the gradient features a highly intuitive but relatively slow "for" loop over each time series or sample path (from 1 to ν). In future work, we will reimplement this code in either C++ or Scala (in conjunction with Apache Spark), eliminate the sample path bottleneck, and study simultaneous nonparametric inference of the drift and diffusion functions.

IV. Results

A. Hermite inference test

In our first set of tests, we use artificial data generated with known drift and diffusion functions f and g. In particular, for each $i \in \{0, 1, 2, 3, 4\}$, we consider (1) with

$$f(x) = \psi_i(x) \tag{35}$$

and a fixed diffusion coefficient $g = 1$. For each resulting SDE, we generate ν sample paths using the Euler-Maruyama scheme with initial condition $X_0 = 0$. When we step the solution forward in time, we use the fixed time step $h_{\text{EM}} = 10^{-4}$, i.e.,

$$\widetilde{X}_{n+1} = \widetilde{X}_n + f(\widetilde{X}_n)h_{\text{EM}} + g(\widetilde{X}_n)h_{\text{EM}}^{1/2}Z_{n+1} \tag{36}$$

where Z_{n+1} is an independent sample of a standard Gaussian (mean 0 and variance 1) random variable. Here $\widetilde{X}_n$ is intended to approximate $X_{nh_{\text{EM}}}$. We step the solution forward in time until $T = 4$, i.e., until $n = T/h_{\text{EM}}$. However, we retain the solution only at times $t = 0, 1, 2, 3, 4$, i.e., when $n = t/h_{\text{EM}}$. In this way, each sample path we generate consists of the solution of (1) sampled at five points in time, one of which is the initial condition.

For each choice of f given by (35), we generate artificial data sets with $\nu = 100$, 1000, and 10000 sample paths.

For this problem, the parameter vector $\boldsymbol{\theta}$ consists of coefficients $(\theta_0, \ldots, \theta_{N_f})$ in (4a) together with one coefficient θ_{N_f+1} that we use to parameterize the unknown (yet constant) diffusion function g. In the following set of tests, we set $N_f = 4$. We give the optimizer an initial guess consisting of $\boldsymbol{\theta}_0 = (1, 1, \ldots, 1)$; for each of three values of the penalty parameter $\gamma \in \{0, 50, 500\}$, we run the optimizer to find an inferred parameter vector $\widehat{\boldsymbol{\theta}}$. When we run the optimizer, we must choose internal parameters for the DTQ method. We choose an internal DTQ time step of $h = 0.02$, spatial grid spacing $k = h^{0.6}$, and grid truncation $A = 2\lceil\pi/k^{1.5}\rceil$. These parameters are chosen so that the DTQ method preserves the total probability of the density function as it evolves.

Because we generated the artificial data ourselves, we know that the ground truth parameter vector consists of $\theta_i = 1$, $\theta_5 = 1$, and all other entries of $\boldsymbol{\theta}$ equal zero. In

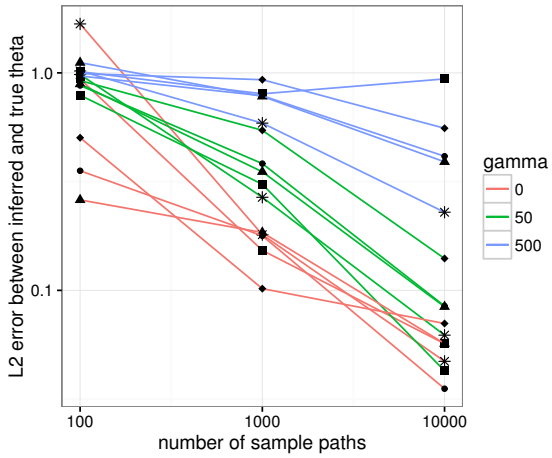

Figure 1. We plot the two-norm error between the true and inferred parameter vectors as a function of the number of sample paths ν. Note that both axes are log-scaled. For each value of ν, there are 15 data points corresponding to 3 possible values of γ, the penalty parameter, and 5 possible drift functions $f(x)$. In particular, the asterisk, square, triangle, diamond, and circles correspond to, respectively $i = 0, 1, 2, 3, 4$ in (35). Overall, we see that the error decreases as the number of sample paths increases; for the $\gamma = 0$ case, a least-squares line has slope -0.5285, consistent with an error rate of $\nu^{-1/2}$.

Figure 1, we plot the two-norm error between the inferred and ground truth parameter vectors, i.e., $\|\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_2$, as a function of ν, the number of samples. Note that both axes on this plot have been logarithmically scaled. The overall trend is clear: as we feed our algorithm a larger number of sample paths, we obtain estimates that are closer to the ground truth values. We also see that, for this simple test, the penalty term is not necessary; the best performance is achieved by setting $\gamma = 0$. If we isolate the $\gamma = 0$ data points from the plot and fit a least-squares regression line to the log-transformed data, the line has slope -0.5285, consistent with an error decay rate of $\nu^{-1/2}$. When the number of sample paths is limited, i.e., $\nu = 100$, it is possible to achieve better performance by setting $\gamma = 50$. Even in this case, the improved inference is observed for some, but not all, choices of $\psi_i(x)$ in (35).

In Figure 2, we focus our attention on the results obtained when $\gamma = 0$ and $\nu = 10000$, i.e., the best results from Figure 1. For each $i = 0, 1, 2, 3, 4$, we plot the true drift function $\psi_i(x)$ (in green) together with the inferred drift function $\hat{f}(x)$ (in red). Using the R function optimHess, we recover the numerical value of the Hessian matrix at the optimal solution $\widehat{\boldsymbol{\theta}}$. This numerical Hessian is the observed Fisher information matrix $I(\boldsymbol{\theta})$. Let $\boldsymbol{\theta}^{\text{SE}}$ denote the square root of the diagonal of $I(\boldsymbol{\theta})^{-1}$; this vector is an estimate of the standard error of $\widehat{\boldsymbol{\theta}}$. Using these standard errors, we compute all $2^5 = 32$ possible curves $\hat{f}(x)$ that result from taking $\boldsymbol{\theta}^* = (\theta_0 \pm \theta_0^{\text{SE}}, \ldots, \theta_4 \pm \theta_4^{\text{SE}})$. We plot the upper and lower envelopes of these curves as dotted lines in Figure 2.

Overall, Figure 2 shows excellent agreement between the true and inferred drift functions. When there is disagreement, the true drift function almost always lies between the dotted lines, i.e., between our upper/lower band estimates for $\hat{f}$.

For this same case ($\gamma = 0$ and $\nu = 10000$), we also mention the inferred values of the constant diffusion coefficient $\hat{g}$. Recall that the true value is $g = 1$. For each i, the inferred values are 0.993,1.012,0.994,0.986, and 0.995. The standard errors for these estimates—obtained in the same way as above—are 0.00762, 0.00636, 0.00873, 0.00936, and 0.00883.

B. Prediction test

In the next test, we focus our attention on the predictive power of the inferred model. For a particular data set $\mathbf{x}$, we set a window size $w \geq 2$. Let $a \vee b$ denote the maximum of the two integers a and b. For each $j \geq 1$, we then use the windowed data $\{x_{(j-w+1)\vee 1}, \ldots, x_j\}$ to train our model, i.e., to run our optimization method and infer a parameter vector $\widehat{\boldsymbol{\theta}}$. We then use this inferred parameter vector in (5). Using the true data vector x_j as a vector of initial conditions, we step (5) forward in time until we reach the time corresponding to x_{j+1}. Let us label the samples thus obtained as $\widehat{x}_{j+1}$; we treat these samples as predictions. We then compute the Kolmogorov-Smirnov distance $E_j = \sup_z |F_{\widehat{x}_{j+1}}(z) - F_{x_{j+1}}(z)|$. Here F_y stands for the empirical CDF of the sample vector y.

To see what the test error is like when we *know* the drift function perfectly, we also use $\hat{f}(x) = \sin x$ in (5) and compute the error just as above for each j. In what follows, we refer to this as the "control" parameter.

Note that in this test, we do not use the penalty parameter, i.e., we set $\gamma = 0$.

As we go through the data set, incrementing j by 1 and computing the test set error repeatedly, we try to avoid recomputing the parameter vector from scratch. Specifically, the parameter vector $\widehat{\boldsymbol{\theta}}$ that is obtained using training data that includes x_j is used as an initial guess when we optimize with training data that includes x_{j+1}. For this test, we use the `nloptr` optimization package mentioned above. If the optimizer fails to find a minimizer, we restart the optimizer with an initial guess perturbed by uniformly distributed $U(-0.1, 0.1)$ noise and an initial diffusion coefficient of 0.7. For the very first initial guess, we take $\boldsymbol{\theta} = (0, \ldots, 0, 0.7)$. As in the first test, we set $N_f = 4$.

For the internal DTQ parameters, we use the same parameters as in Section IV-A, except for the grid truncation level A. We compare two values for A: $A_1 = \lceil \pi/k^{1.5} \rceil$ and $A_2 = 2A_1$.

As for the artificial data generation, we again follow the procedure in Section IV-A. Compared to what we described above, the only differences are that we now take $T = 50$ and $\nu = 100$; despite the small Euler-Maruyama time step h_{EM}, we still save the solution only at integer times t.

We perform the above test for window sizes $w = 2$, $w = 10$, and $w = 50$. Because $T = 50$, $w = 50$ corresponds to

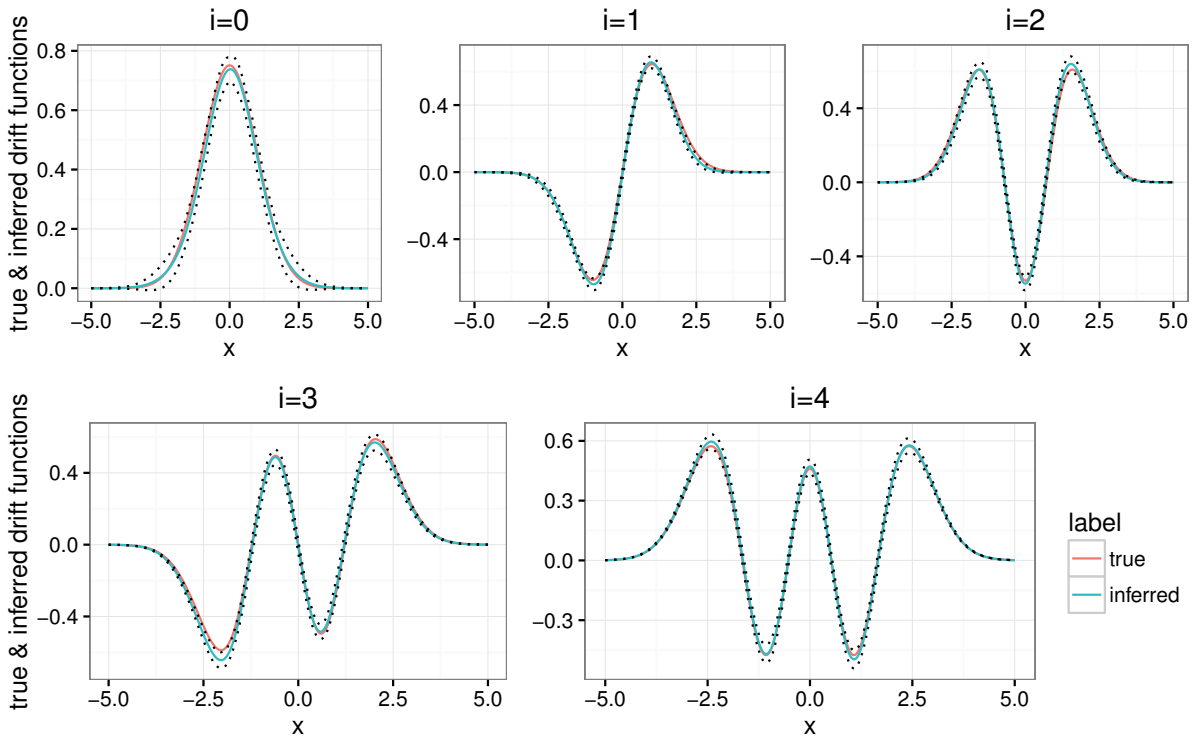

Figure 2. For each value of i, we plot the true drift function $\psi_i(x)$ (in green), the inferred drift function $\hat{f}(x)$ (in red), and upper/lower standard error bands about $\hat{f}(x)$ (in dotted lines). Overall, we find excellent agreement between the true and inferred drift functions. These results correspond to the $\gamma = 0$, $\nu = 10000$ results from Figure 1. Further details are given in the main text.

using all available past data to train the model.

We plot the results in Figure 3. The plotted curves lead to four main conclusions. First, there are several test set error curves that stay consistently below 0.15, indicating good agreement between the predicted and true CDFs. This is true in spite of the fact that the true drift function $f(x) = \sin x$ explicitly violates an assumption made in Section II; in particular, $f(x)$ is not square-integrable, i.e., $f \notin L^2$.

Second, in the DTQ method, we must pay attention to the size of the spatial domain. If we set $A = A_1$, it turns out that the spatial domain is too small to preserve the total area under the density curve as it evolves forward in time. As we will see, this leads to unreasonably large estimates of the coefficient vector $\boldsymbol{\theta}$. Once we set $A = A_2$, the method performs far better.

Third, the window size has an insignificant effect on the test set error. The three test set error curves corresponding to $A = A_2$ are quite close to one another, even when w is as small ($w = 2$) or as large ($w = 50$) as possible.

The fourth conclusion is perhaps the most interesting one: the seemingly ideal case in which $\hat{f}(x) = f(x) = \sin x$ leads to rather poor test set error, as we can see from the control curve. We hypothesize that because the number of sample paths is fairly small ($\nu = 100$), the artificial data we have generated may at times be more consistent with a drift function *other* than $f(x)$. In Figure 4, we have plotted five drift functions: the true drift $f(x)$ together with four inferred drifts corresponding to the $j = 50$ points in Figure 3. The $A = A_1$ case leads to unreasonably large coefficients, which we see immediately in the plot. The $A = A_2$ cases lead to perfectly reasonable drift functions, only one of which bears much resemblance to $f(x) = \sin x$. Overall, we are encouraged by the fact that our method can pick out predictive models from the data even when the inference is far from the ground truth.

C. Real data test

Our final test involves real data obtained from the California Air Resources Board [22]. Specifically, we obtained data consisting of hourly measurements of ground level ozone (in parts per million) conducted daily at numerous sites across California in the years 2010-2011. We first narrowed our scope to 25 sites in the San Joaquin Valley. For these sites, we treated each day as a separate time series with 24 points. When fewer than 24 were available, we used linear interpolation to impute missing points; however, this procedure was only applied if the time series had at least 16 points to begin with.

In this way, we formed a data set consisting of $\nu = 16675$ sample paths, each with 24 points. To allow us to use a large spatial grid spacing in the DTQ method, we multiplied each

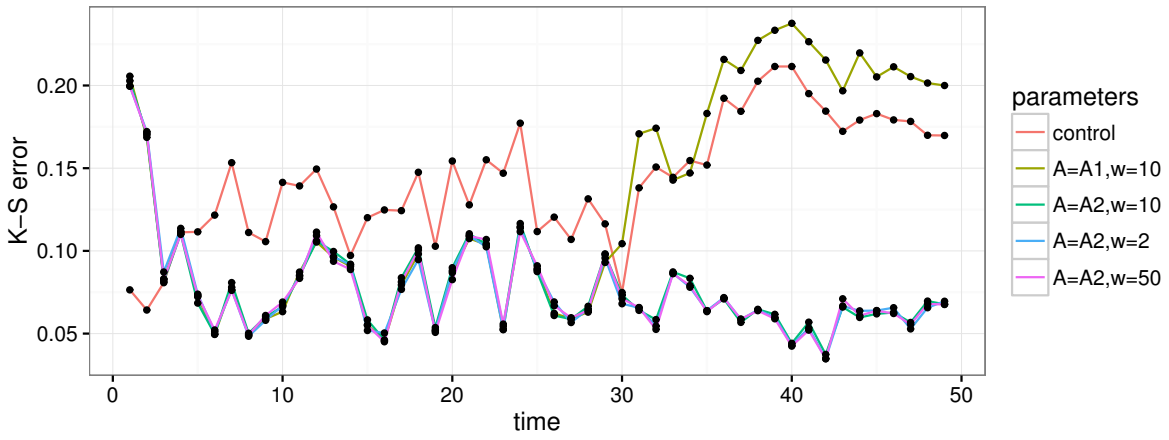

Figure 3. Following the procedure in Section IV-B, we evaluate the predictive power of models inferred using our method. Specifically, for each point in time j, we plot the Kolmogorov-Smirnov (K-S) error between the empirical distributions of x_{j+1} (real data) and $\widehat{x}_{j+1}$ (predictions). We form the predictions by inferring the drift and diffusion functions, and then using them in a forward simulation of the SDE (5). The parameters A and w are, respectively the spatial grid truncation in the DTQ method and a window size, i.e., how many points along each sample path are actually used to fit the model. The $A = A_2$ results show that as long as the DTQ method's spatial grid truncation is chosen well, the inferred models yield good predictive results. These results also show insensitivity to the window size w. The control curve corresponds to predictions generated from $f(x) = \sin x$, i.e., the drift function used to create the data set.

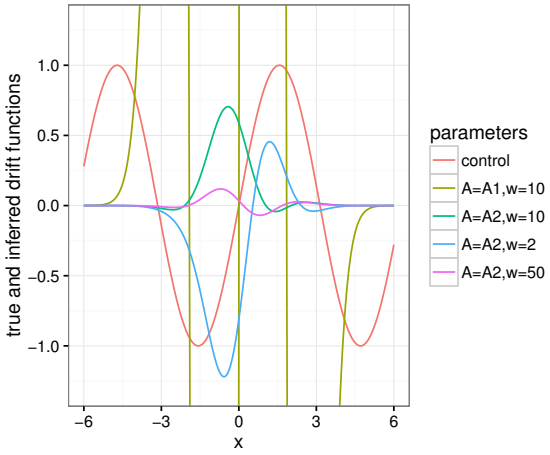

Figure 4. We plot the true drift function $f(x) = \sin x$ (control) together with four inferred drift functions $\hat{f}(x)$. These drift functions correspond to the $j = 50$ runs from Figure 3. Combining what we see here with Figure 3, we conclude that even though the $A = A_2$ drifts are not close to $\sin x$, they still yield good test set results.

ozone level by a factor of 10. We then applied our inference method to this data set.

The specific DTQ parameters used are: $h = 0.02$, $k = h$, and $A = \lceil 0.5/k^{1.5} \rceil$. As before, we take $N_f = 6$.

As one might expect, with a real data set, the objective function required penalization in order to yield reasonable results. We chose a penalty parameter of $\gamma = 50$ based on several trials.

We fit the model using the first 6 hours worth of data, i.e., a window size of $w = 6$. The drift function together with upper/lower standard error bands is displayed in Figure 5.

For this test, the inferred diffusion coefficient is $\hat{g} = 0.141$. Note that when running using the `nloptr` optimization package for this test, we have constrained the diffusion function via $\hat{g} \geq \sqrt{k}$. In the DTQ method, as the diffusion function approaches zero at a particular point, it becomes impossible to resolve the Gaussian kernel $G^h_{\boldsymbol{\theta}}$ on a spatial grid with fixed spacing $k > 0$. In the particular inference problem we describe here, the optimizer does tend to send the diffusion constant to its lower bound—we believe this indicates the need for a spatially-dependent (rather than constant, as assumed here) diffusion function $\hat{g}$.

Following the same procedure outlined in Section IV-B, we generated predictions for the next 6 hours. The Kolmogorov-Smirnov errors between the true and predicted CDFs are, respectively, 0.148, 0.161, 0.164, 0.179, 0.190, and 0.220. The errors in the true and predicted means are, respectively, 0.0549, 0.0657, 0.0772, 0.0867, 0.1069, and 0.1333. As expected, the errors increase as we get further away from the last data point used to build the model. Please note that to convert the errors in the mean to parts per million, we must divide by 10; we believe the resulting errors indicate reasonable agreement between predictions and reality.

V. Conclusion

In this paper, we have developed a new nonparametric method to infer the drift and diffusion functions for a stochastic differential equation. We have demonstrated through three tests that the method has the capability to succeed. There are three main future directions that we plan to pursue. First, we have only explored the Hermite function basis in this work and we believe other bases may yield better results for particular data sets. Second, we have not yet tested the method on inference problems where we seek

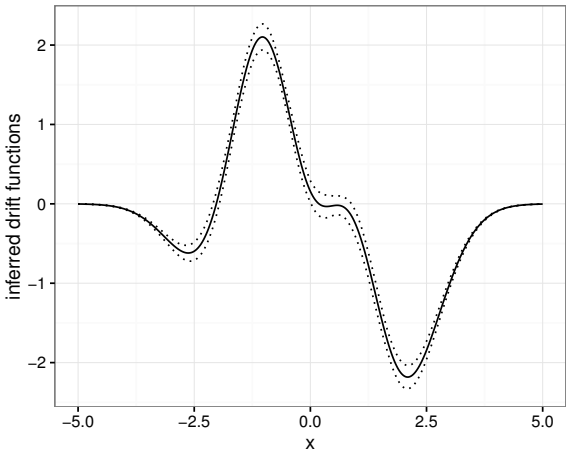

Figure 5. The inferred drift $\hat{f}(x)$ for hourly ozone levels in the San Joaquin Valley, together with upper/lower standard error bands.

a non-constant diffusion function $g(x)$. Third, we plan to seek out improved methods to find optimal values of the penalization parameter γ. All of these tasks will be enabled by porting computationally intensive parts of our algorithm to a compiled language, a task we have already begun.

ACKNOWLDGMENT

H.S. Bhat and R.W.M.A. Madushani gratefully acknowledge research support from, respectively, UC Merced's Committee on Research and a UC Merced Applied Mathematics USAP Fellowship.

REFERENCES

[1] I. H. Stevenson and K. P. Kording, "How advances in neural recording affect data analysis," *Nature Neuroscience*, vol. 14, no. 2, pp. 139–142, 201.

[2] H.-G. Müller and F. Yao, "Empirical dynamics for longitudinal data," *The Annals of Statistics*, vol. 38, no. 6, pp. 3458–3486, 2010.

[3] W.-K. Wong and R. B. Miller, "Repeated time series analysis of ARIMA-noise models," *Journal of Business and Economic Statistics*, vol. 8, no. 2, pp. 243–250, 1990.

[4] S. Chen, E. Grant, T. T. Wu, and F. D. Bowman, "Statistical learning methods for longitudinal high-dimensional data," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, no. 1, pp. 10–18, 2014.

[5] R. Küffner, N. Zach, R. Norel, J. Hawe, D. Schoenfeld, L. Wang, G. Li, L. Fang, L. Mackey, O. Hardiman, M. Cudkowicz, A. Sherman, G. Ertaylan, M. Grosse-Wentrup, T. Hothorn, J. van Ligtenberg, J. H. Macke, T. Meyer, B. Schoelkopf, L. Tran, R. Vaughan, G. Stolovitzky, and M. L. Leitner, "Crowdsourced analysis of clinical trial data to predict amyotrophic lateral sclerosis progression," *Nature Biotechnology*, vol. 33, no. 1, pp. 51–U292, 2015.

[6] H. Sørensen, "Parametric inference for diffusion processes observed at discrete points in time: a survey," *International Statistical Review*, vol. 72, no. 3, pp. 337–354, 2004.

[7] S. M. Iacus, *Simulation and Inference for Stochastic Differential Equations: With R Examples*, ser. Springer Series in Statistics. New York: Springer, 2009.

[8] C. Fuchs, *Inference for Diffusion Processes: With Applications in Life Sciences*. Berlin: Springer, 2013.

[9] H. S. Bhat and R. W. M. A. Madushani, "Density tracking by quadrature for stochastic differential equations," 2016, preprint.

[10] H. S. Bhat, R. W. M. A. Madushani, and S. Rawat, "Parameter inference for stochastic differential equations with density tracking by quadrature," 2016, submitted and under review.

[11] ——, "Scalable SDE filtering and inference with Apache Spark," *Journal of Machine Learning Research W&CP*, 2016, in press.

[12] ——, "Bayesian inference of stochastic pursuit models from basketball tracking data," 2016, submitted and under review.

[13] A. Ruttor, P. Batz, and M. Opper, "Approximate Gaussian process inference for the drift function in stochastic differential equations," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 2040–2048.

[14] 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.

[15] D. Y. Lin and Z. Ying, "Semiparametric and nonparametric regression analysis of longitudinal data," *Journal of the American Statistical Association*, vol. 96, no. 453, pp. 103–13, 2001.

[16] R. J. Sela and J. S. Simonoff, "RE-EM trees: a data mining approach for longitudinal and clustered data," *Machine Learning*, vol. 86, pp. 169–207, 2012.

[17] J.-L. Wang, "Nonparametric regression analysis of longitudinal data," in *Encyclopedia of Biostatistics*. New York, NY: Wiley, 2005.

[18] N. Wiener, *The Fourier Integral and Certain of Its Applications*, ser. Cambridge Mathematical Library. Cambridge, UK: Cambridge University Press, 1988.

[19] J. Ypma, "nloptr: R interface to nlopt," 2014, http://cran.r-project.org/web/packages/nloptr/.

[20] S. G. Johnson, "The nlopt nonlinear-optimization package," 2016, http://ab-initio.mit.edu/nlopt.

[21] M. Braun, "trustoptim: Trust region optimization for nonlinear functions with sparse hessians," 2015, http://cran.r-project.org/web/packages/trustOptim/.

[22] C. A. R. Board, "Database: California Air Quality Data," 2016, http://www.arb.ca.gov/aqd/aqdcd/aqdcddld.htm.