Computing the Density Function for a Nonlinear Stochastic Delay System

H. S. Bhat [*] **R. W. M. A. Madushani** [**]

[*] *University of California, Merced, Merced, CA 95343 USA (e-mail: hbhat@ucmerced.edu)*
[**] *University of California, Merced, Merced, CA 95343 USA (e-mail: rmadushani@ucmerced.edu)*

Abstract: We develop a numerical method to compute the density of a specific nonlinear stochastic delay system, with no sampling. This system arises as a switch-type control model for human balance. Numerical tests against the Euler-Maruyama method show that our method is capable of computing accurate solutions. In particular, the method captures the covariance of the solution at the present and delayed times. This is accomplished through the time-evolution of a Gaussian approximation of the joint density at the present and delayed times. Issues of circularity prevent the numerical solution of the Fokker-Planck equation for stochastic delay systems. Our method bypasses these issues and offers one of the first deterministic algorithms to compute the density of a nonlinear stochastic delay system.

Keywords: stochastic differential equations, time delay, nonlinear systems, numerical analysis

1. INTRODUCTION

Research into the human nervous system's internal control mechanism for maintaining upright balance and posture has led to a number of mathematical models (Eurich and Milton, 1996; Cabrera, 2005; Milton et al., 2009; Boulet et al., 2009; Suzuki et al., 2012; Kowalczyk et al., 2014). Such models typically feature nonlinear feedback, time delay(s), and noise. In this work, we focus on a model that incorporates feedback through a switch-type control mechanism; the model we analyze is the $\tau = \tau'$ case of Eq. (1.2) in (Milton, 2011). The model consists of the following nonlinear stochastic delay differential equation (SDDE):

$$dX_t = f(X_{t-\tau})dt + dW_t \tag{1a}$$

$$f(y) = \alpha y + \begin{cases} C & y < -\beta \\ 0 & |y| \le \beta \\ -C & y > \beta, \end{cases} \tag{1b}$$

where α, β, C, τ are positive constants, and W_t is the Wiener process or standard Brownian motion. In (1a), X_t is the angle of displacement of the body from the line of gravity at time t. We assume that X_t is a given, deterministic function for $-\tau \le t \le 0$.

If the time delay τ in (1a) is zero, the resulting equation is a stochastic differential equation (SDE). With a smooth approximation to the drift function f, we can compute the p.d.f. of this SDE by solving the associated Fokker-Planck or Kolmogorov equation, a deterministic partial differential equation. Because the associated Fokker-Plank equation is circular when $\tau \neq 0$, (Longtin, 2009), SDDE are typically solved using Monte Carlo methods, which involve generating large numbers of sample paths.

In this paper we present a numerical method to solve for the p.d.f. of the nonlinear SDDE (1) with no sampling. The method represents the unknown p.d.f. using a discretization on a spatial grid; the p.d.f. that is computed is not constrained to come from a particular family of densities. In particular, though our method makes a Gaussian approximation for a long-range joint density, the marginal density that is computed is not necessarily Gaussian. While we do not establish rigorous convergence of the method here, we do provide evidence that the method is accurate enough for many purposes.

There is no existing method to compute the p.d.f. of a nonlinear SDDE with large delay, without simulating sample paths. The method described in this paper takes a first step in this direction.

2. DERIVATION OF NUMERICAL METHOD

Let $\ell \ge 1$ be an integer and set the time step $h = \tau/\ell$. Let $\{Z_n\}$ be an i.i.d. family of standard (zero mean, unit variance) Gaussian random variables. Then the Euler-Maruyama discretization of (1) is

$$x_{n+1} = x_n + f(x_{n-\ell})h + \sqrt{h}Z_{n+1}. \tag{2}$$

Because we assumed the initial segment $\{X_s : -\tau \le s \le 0\}$ is deterministic, the initial conditions $x_{-\ell}, \cdots, x_0$ are known constants. Our approach is to derive from (2) a deterministic evolution equation that yields an approximate density $p(x_n)$. We view (2) as a reasonable starting point since convergence as $h \to 0$ has been established (Buckwar et al., 2008; Gyöngy and Sabanis, 2013).

2.1 Evolution Equation

Let the joint density of $\mathbf{x}_n = (x_n, x_{n-1}, \ldots, x_{n-\ell})$ be denoted by $p(\mathbf{x}_n)$. Then we have

$$p(\mathbf{x}_{n+1}) = \int_{x_{n-\ell}} p(x_{n+1}|\mathbf{x}_n)p(\mathbf{x}_n)dx_{n-\ell}. \tag{3}$$

From (2), the conditional density $p(x_{n+1}|\mathbf{x}_n)$ is Gaussian with mean $x_n + f(x_{n-\ell})h$ and variance h. This observation shows that $p(x_{n+1}|\mathbf{x}_n) = p(x_{n+1}|x_n, x_{n-\ell})$. In principle, (3) is a self-contained system for evolving the joint density $p(\mathbf{x}_n)$. The joint density $p(\mathbf{x}_n)$ is a scalar function of $\ell + 1$ variables. It is impractical to store and manipulate spatial discretizations of such functions when $\ell \geq 4$, or equivalently, for time steps $h \leq \tau/4$. For this reason, we seek a simpler evolution equation. (As a Monte Carlo approach, one can directly simulate sample paths of (2); here we seek a method that does not involve sampling.)

Our first step is to integrate both sides of (3) with respect to $(x_{n-1}, \ldots, x_{n-\ell+1})$. This yields

$$p(x_{n+1}, x_n) = \int_{x_{n-\ell}} p(x_{n+1}|x_n, x_{n-\ell}) p(x_n, x_{n-\ell}) dx_{n-\ell}. \quad (4)$$

If $p(x_n, x_{n-\ell})$ is known, we can use (4) to compute $p(x_{n+1}, x_n)$. In order to close the loop, we must be able to compute $p(x_{n+1}, x_{n-\ell+1})$. This is explained next.

2.2 Long-Range Probability

We approximate $p(x_n, x_{n-\ell})$ by a two-dimensional Gaussian with mean vector $\boldsymbol{\mu}_{n,n-\ell}$ and covariance matrix $\Sigma_{n,n-\ell}$. While the true joint density is not Gaussian, a Gaussian is the simplest conceivable model that includes the possibility of covariance between the two random variables x_n and $x_{n-\ell}$. In contrast, if we were to approximate $p(x_n, x_{n-\ell}) \approx p(x_n)p(x_{n-\ell})$, although the joint density is allowed to have a non-Gaussian shape, we are essentially declaring x_n and $x_{n-\ell}$ to be independent and hence to have zero covariance. Through extensive tests of approximations of $p(x_n, x_{n-\ell})$, we find that modeling the covariance is essential; models that assume independence or even conditional independence produce errors that are orders of magnitude larger than the Gaussian approximation.

In order to compute $p(x_{n+1}, x_{n-\ell+1})$, we must determine the covariance matrix $\Sigma_{n+1,n-\ell+1}$. It will turn out to be necessary to compute the more general covariance matrix $\Sigma_{n+1,n+1-q}$. Hence we compute

$$\begin{aligned}
&E[x_{n+1}x_{n+1-q}] \\
&= \iint x_{n+1}x_{n+1-q}p(x_{n+1}, x_{n+1-q})dx_{n+1}dx_{n+1-q} \\
&= \iint \iint x_{n+1}x_{n+1-q}p(x_{n+1}|x_n, x_{n+1-q}, x_{n-\ell}) \\
&\qquad \times p(x_n, x_{n+1-q}, x_{n-\ell})dx_{n+1}dx_n dx_{n+1-q}dx_{n-\ell}.
\end{aligned}$$

Recall that $p(x_{n+1}|x_n, x_{n+1-q}, x_{n-\ell})$ is $p(x_{n+1}|x_n, x_{n-\ell})$, which is Gaussian with known mean and variance. Hence we carry out the integration over x_{n+1} first, and obtain

$$\begin{aligned}
&E[x_{n+1}x_{n+1-q}] \\
&= \iiint x_{n+1-q}(x_n + hf(x_{n-\ell})) \\
&\qquad \times p(x_n, x_{n+1-q}, x_{n-\ell})dx_n dx_{n+1-q}dx_{n-\ell} \\
&= \iint x_n x_{n+1-q}p(x_n, x_{n+1-q})dx_n dx_{n+1-q} \\
&+ h\iint x_{n+1-q}f(x_{n-\ell})p(x_{n+1-q}, x_{n-\ell})dx_{n+1-q}dx_{n-\ell} \\
&= E[x_n x_{n+1-q}] + hE[x_{n+1-q}f(x_{n-\ell})]. \qquad (5)
\end{aligned}$$

Overall, we have shown that

$$\begin{aligned}
\mathrm{Cov}[x_{n+1}, x_{n+1-q}] &= \mathrm{Cov}[x_n x_{n+1-q}] \\
&+ (E[x_n] - E[x_{n+1}])E[x_{n+1-q}] \\
&+ hE[x_{n+1-q}f(x_{n-\ell})]. \quad (6)
\end{aligned}$$

Substituting $q = \ell$, we see that we cannot compute $\Sigma_{n+1,n-\ell+1}$ without $\Sigma_{n,n-\ell+1}$. To ensure that we have $\Sigma_{n,n-\ell+1}$, we track all joint densities $p(x_n, x_{n+1-q})$ for all $q = 2, \ldots, \ell$. To make this tractable, we assume all of these joint densities are Gaussian with mean $\boldsymbol{\mu}_{n,n+1-q}$ and covariance matrix $\Sigma_{n,n+1-q}$.

We now show that this yields a self-consistent update scheme. Suppose that at time step n, we have used $p(x_n, x_{n-\ell})$ in (4) to compute $p(x_{n+1}, x_n)$. Using quadrature, we can compute the mean and variance of x_{n+1}, which we store. Further suppose we have all of the inputs required to evaluate the right-hand sides of (6) for $q = 2, \ldots, \ell$. This consists of $\{\Sigma_{n,n+1-q}\}_{q=2}^{\ell}$, $\{\Sigma_{n+1-q,n-\ell}\}_{q=2}^{\ell}$, together with the means of $\{x_j\}_{j=n-\ell}^{n+1}$.

With all of this information, we use (6) together with the stored variances of x_{n+1} and x_{n+1-q} to generate and store the new covariance matrix $\Sigma_{n+1,n+1-q}$ for each $q = 2, \ldots, \ell$. The new mean vector $\boldsymbol{\mu}_{n+1,n+1-q}$ is directly determined by the stored means of x_{n+1} and x_{n+1-q}. This is because the vector of means of the marginal densities of $p(x_{n+1}, x_{n+1-q})$ equals the mean vector of the joint density. The $q = \ell$ case of the new mean vector and new covariance matrix gives us enough information to compute $p(x_{n+1}, x_{n+1-\ell})$, enabling the use of (4) at the next time step. Hence we compute $p(x_{n+2}, x_{n+1})$, which in turn yields the expected value of x_{n+2}. This is the only new mean value required to use (6) at the next time step.

To assemble the remaining information required to use (6) at the next time step, we need $\{\Sigma_{n+1,n+2-q}\}_{q=2}^{\ell}\}$ and $\{\Sigma_{n+2-q,n+1-\ell}\}_{q=2}^{\ell}$. In $\{\Sigma_{n+1,n+2-q}\}_{q=2}^{\ell}$, the $q = 2$ term can be computed by applying quadrature to $p(x_{n+1}, x_n)$. The remaining terms in $\{\Sigma_{n+1,n+2-q}\}_{q=2}^{\ell}$ are a subset of the new covariance matrices we generated above. The collection $\{\Sigma_{n+2-q,n+1-\ell}\}_{q=2}^{\ell}$ consists entirely of covariance matrices computed and stored at previous time steps.

2.3 Computation of the Nonlinear Term in (6)

Let us explain in detail how we compute $E[x_{n+1-q}f(x_{n-\ell})]$ for the particular nonlinearity f given by (1b). We proceed as follows, abbreviating $p(x_{n+1-q}, x_{n-\ell})$ as p:

$$\begin{aligned}
&E[x_{n+1-q}f(x_{n-\ell})] \\
&= \iint x_{n+1-q}f(x_{n-\ell})\underbrace{p(x_{n+1-q}, x_{n-\ell})}_{p}dx_{n+1-q}dx_{n-\ell} \\
&= \int\int_{x_{n-\ell}=-\infty}^{-\beta} x_{n+1-q}(\alpha x_{n-\ell} + C)p dx_{n+1-q}dx_{n-\ell} \\
&+ \int\int_{x_{n-\ell}=-\beta}^{\beta} x_{n+1-q}\alpha x_{n-\ell}p dx_{n+1-q}dx_{n-\ell} \\
&+ \int\int_{x_{n-\ell}=\beta}^{\infty} x_{n+1-q}(\alpha x_{n-\ell} - C)p dx_{n+1-q}dx_{n-\ell}.
\end{aligned}$$

Combining terms with α and C, respectively, we obtain

$$
\begin{aligned}
E[x_{n+1-q}f(x_{n-\ell})] &= \alpha E[x_{n+1-q}x_{n-\ell}] \\
&+ C\int_{x_{n-\ell}=-\infty}^{-\beta}\left(\int x_{n+1-q}p\,dx_{n+1-q}\right)dx_{n-\ell} \\
&- C\int_{x_{n-\ell}=\beta}^{\infty}\left(\int x_{n+1-q}p\,dx_{n+1-q}\right)dx_{n-\ell}.
\end{aligned}
\tag{7}
$$

Because $p = p(x_{n+1-q}, x_{n-\ell})$ has been approximated by a two-dimensional Gaussian, the term in parentheses can be evaluated:

$$
\begin{aligned}
&\int x_{n+1-q}p\,dx_{n+1-q} \\
&= p(x_{n-\ell})\int x_{n+1-q}p(x_{n+1-q}|x_{n-\ell})dx_{n+1-q} \\
&= p(x_{n-\ell})\left(\mu_{n+1-q} + \frac{\mathrm{Cov}[x_{n+1-q}x_{n-\ell}]}{\mathrm{Var}[x_{n-\ell}]}(x_{n-\ell}-\mu_{n-\ell})\right).
\end{aligned}
$$

Using this in (7), the computation of $E[x_{n+1-q}f(x_{n-\ell})]$ is reduced to two one-dimensional quadratures in the $x_{n-\ell}$ variable. The quadratures are of the form

$$
\int p(x_{n-\ell})(c_1 x_{n-\ell} + c_2)dx_{n-\ell}, \tag{8}
$$

for constants c_1 and c_2. When we perform this quadrature, we use the marginal density $p(x_{n-\ell})$ computed via (4), not the marginal of the two-dimensional Gaussian.

2.4 Initialization

As we remarked after (2), the initial conditions $\{x_j\}_{j=-\ell}^{j=0}$ are known constants which we denote as $\{\mu_j\}_{j=-\ell}^{j=0}$. Using these initial conditions, direct substitution of $n = 0, 1, \ldots, \ell$ into (2) yields the solution

$$
x_m = x_0 + h\left[f(x_{m-1-\ell}) + \cdots + f(x_{-\ell})\right] + h^{1/2}\sum_{j=1}^{m} Z_j, \tag{9}
$$

valid for $m = 1, 2, \ldots, \ell+1$. The means and variances of x_m are thus determined exactly for $m = 1, 2, \ldots, \ell+1$. Now suppose $\ell+1 \geq n \geq m$. Then we can use (9) together with properties of the i.i.d. standard Gaussian random variables $\{Z_j\}_{j=1}^{\ell+1}$ to derive $\mathrm{Cov}[x_n, x_m] = hm$. This gives us enough information to initialize the covariance update algorithm described in Sections 2.2 and 2.3.

The final piece of initial data we need to use (4) at time $n = \ell+1$ is $p(x_{\ell+1}, x_1) = p(x_{\ell+1}|x_1)p(x_1)$. The conditional density $p(x_{\ell+1}|x_1)$ can be computed from the difference of (9) at $m = \ell+1$ and $m = 1$. This shows that $p(x_{\ell+1}|x_1)$ is Gaussian with mean $h\left[f(x_0) + \cdots + f(x_{-\ell+1})\right]$ and variance ℓh.

2.5 Quadrature and Grids

At each time step, we must perform quadrature to compute (4) together with the updates described in Sections 2.2 and 2.3. To perform this quadrature, we truncate all integrals from $(-\infty, \infty)$ to $[-A, A]$ for fixed $A > 0$. We then introduce the grid $\{\zeta_i\}_{i=1}^m$ with $\zeta_1 = -A$ and $\zeta_m = A$. This grid need not be uniform or equispaced. Using this grid, we can store a finite-dimensional representation of $p(x_{n+1}, x_n)$ as an $m \times m$ matrix, where the (i,j)-th element is $p_{i,j}^{n+1} := p(x_{n+1} = \zeta_i, x_n = \zeta_j)$. The marginal density $p(x_n)$ is stored as an $m \times 1$ vector where the i-th element is $p_i^n := p(x_n = \zeta_i)$. To compute integrals such as $E[x_n]$, we use the trapezoidal rule, i.e.,

$$
\begin{aligned}
E[x_n] &= \int_{-\infty}^{\infty} x_n p(x_n)dx_n = \int_{-A}^{A} x_n p(x_n)dx_n + e_1 \\
&= \frac{1}{2}\sum_{i=1}^{m-1}(\zeta_{i+1}-\zeta_i)\left[\zeta_{i+1}p_{i+1}^n + \zeta_i p_i^n\right] + e_1 + e_2.
\end{aligned}
$$

Here e_1 is the error due to truncating the integral to a finite domain, and e_2 is the error due to the trapezoidal rule. Using the trapezoidal rule in a similar way, we can write formulas for all quadrature calculations in our code.

Note that we do not artificially enforce normalization, e.g., by dividing the p.d.f. by its total integral at each time step. We find in all tests that for sufficiently large grids, our algorithm preserves normalization automatically.

2.6 Zero Initial Data

When the initial data $\{\mu_j\}_{j=-\ell}^{j=0}$ vanishes identically, we can substantially simplify the algorithm using symmetry arguments. First note that (9) implies that $E[x_m] = 0$ for $1 \leq m \leq \ell+1$, which further implies that $\boldsymbol{\mu}_{x_{\ell+1},x_1} = \mathbf{0}$. Because $p(x_{\ell+1} = a_{\ell+1}, x_1 = a_1)$ is Gaussian, it must then be symmetric with respect to the reflection $(a_{\ell+1}, a_1) \mapsto (-a_{\ell+1}, -a_1)$. Fix $b > 0$ and consider

$$
\begin{aligned}
&P(x_{n+1} \leq -b) = (2\pi h)^{-1/2} \\
&\times \int_{a_{n+1}=-\infty}^{-b}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp\left[-\frac{(a_{n+1}-(a_n + hf(a_{n-\ell})))^2}{2h}\right] \\
&\qquad \times p(x_n = a_n, x_{n-\ell} = a_{n-\ell})da_{n+1}da_n da_{n-\ell}. \quad (10)
\end{aligned}
$$

We now change variables via $a_j' = -a_j$ for $j \in \{n+1, n, n-\ell\}$. Using the fact that f defined by (1b) is always odd, we obtain

$$
\begin{aligned}
&P(x_{n+1} \leq -b) = (2\pi h)^{-1/2} \\
&\times \int_{a_{n+1}'=b}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp\left[-\frac{(a_{n+1}'-(a_n' + hf(a_{n-\ell}')))^2}{2h}\right] \\
&\qquad \times p(x_n = -a_n', x_{n-\ell} = -a_{n-\ell}')da_{n+1}'da_n' da_{n-\ell}'. \quad (11)
\end{aligned}
$$

When $n = \ell+1$, we appeal to the reflection symmetry of $p(x_{\ell+1}, x_1)$ to conclude that

$$
P(x_{n+1} \leq -b) = P(x_{n+1} \geq b). \tag{12}
$$

We also have, at $n = \ell+1$,

$$
\begin{aligned}
E[x_{n+1}] &= \iint E[x_{n+1}|x_n, x_{n-\ell}]p(x_n, x_{n-\ell})dx_n dx_{n-\ell} \\
&= \iint \left[x_n + f(x_{n-\ell})h\right]p(x_n, x_{n-\ell})dx_n dx_{n-\ell} \\
&= E[x_n] + h\alpha E[x_{n-\ell}] \\
&\quad + ChP(x_{n-\ell} \leq -\beta) - ChP(x_{n-\ell} \geq \beta) = 0.
\end{aligned}
$$

To obtain the last equality, we have used (12) at $n = 0$. This result forces $\boldsymbol{\mu}_{\ell+2,2} = \mathbf{0}$, leading to the reflection symmetry of $p(x_{\ell+2}, x_2)$, which by (10) and (11) implies (12) at $n = \ell+2$. We have essentially proven, by induction, that $E[x_{n+1}] = 0$ for all n.

Therefore, when the initial data vanishes, we do not need to track any mean values or mean vectors in the code. In this case, $\text{Cov}[x_n, x_m] = E[x_n x_m]$ and $\text{Var}[x_n] = E[x_n^2]$. Finally, the two one-dimensional integrals (8) described in Section 2.3 are equal, a direct consequence of (12). We have developed a separate code, incorporating these simplifications, to handle zero initial data.

3. NUMERICAL RESULTS

We implemented our algorithm in MATLAB, and for this we considered two sets of parameter values for α, β, C, and τ in (1). For each of these parameter sets, we compute the p.d.f.'s for two cases of initial values: (i) zero initial conditions, in which case we use the algorithm described in Section 2.6, and (ii) non-zero initial conditions. For a particular choice of parameters and initial conditions, we compute the p.d.f. at a final time T for a fixed value ℓ. Because $h = \tau/\ell$, this is equivalent to choosing a fixed step size h. The two sets of parameters are:

	α	β	C	τ	ℓ	h	T
Param. I	0.1	1	1	1	20	0.05	3
Param. II	1	0.5	0.5	0.2	20	0.01	1

The first set of parameters has been chosen to match a set of parameters described in the literature (Milton et al., 2009). For the Param. I test problem, when we use nonzero initial conditions, we set $x_j = 0$ for $-\ell \le j < 0$ and $x_0 = 0.3$. For the Param. II test problem, when we use nonzero initial conditions, we set $x_j = 0.2$ for $-\ell \le j \le 0$.

On a uniform grid, the grid spacing is $\Delta\zeta = \zeta_{i+1} - \zeta_i$. To test the convergence of our method, we repeatedly compute the marginal p.d.f. $f_{X(T)}(x)$, i.e., $p(x_N)$ where $N = T/h$, each time decreasing the grid spacing $\Delta\zeta$ by increasing the number of grid points. For comparison, we use the Euler-Maruyama method to generate 10^8 sample paths of (1). Whenever we make a comparison between our method and Euler-Maruyama, the same step size h is used for both methods. Because Euler-Maruyama is known to converge as $h \to 0$, our goal is to show that for each fixed $h > 0$, our method converges (as $\Delta\zeta \to 0$) to the same solution to which Euler-Maruyama converges as the number of sample paths goes to infinity.

We compute the error in two metrics: the Kolmogorov-Smirnov (K-S) metric and the Wasserstein metric.

Let $F(x)$ be the cumulative distribution function (c.d.f.) of $X(T)$ computed using the method described in this paper. We compute $F(x)$ numerically by applying the trapezoidal rule to the p.d.f. $f_{X(T)}(x)$. Let $F_{\text{EM}}(x)$ be the c.d.f. of $X(T)$ computed via the Euler-Maruyama method; in this case, $F_{\text{EM}}(x)$ is the empirical c.d.f. of the 10^8 samples, interpolated onto the same grid used to calculate $F(x)$. Then the K-S error is defined by

$$\|F - F_{\text{EM}}\|_{\text{K-S}} = \sup_{x \in \mathbb{R}} |F(x) - F_{\text{EM}}(x)|,$$

and the Wasserstein error is defined by

$$\|F - F_{\text{EM}}\|_{\text{Wasserstein}} = \int_{x=-\infty}^{\infty} |F(x) - F_{\text{EM}}(x)| \, dx.$$

For both error computations, we truncate the domain to $[-A, A]$. When we calculate the Wasserstein error, we apply quadrature as described in Section 2.5.

In addition, we compute the relative error in the first and second moment of $X(T)$. When we use our method, the moments are computed by applying quadrature to $\int x^j f_{X(T)}(x)\, dx$. For comparison, we use the first and second moments of the 10^8 Euler-Maruyama samples.

The results of the tests are plotted in Fig. 1 using logarithmically scaled horizontal and vertical axes. The Param I. problem yields better results in the K-S and Wasserstein metrics, while the Param II. problem yields better results in terms of the relative error in the first two moments. For the Param. I problem, the K-S and Wasserstein errors decrease monotonically as the grid size increases up to 801. For the Param. I zero (respectively, non-zero) initial condition (IC) problem, the least-squares regression line fit to the log-scaled K-S errors yields a slope of -0.840 (respectively, -0.724).

At a grid size of 401, the K-S error for the Param. II nonzero IC problem is close to 0.01. To visualize what this means in terms of the computed density functions, we present Fig. 2. Here we plot the p.d.f.'s for the same problem at the final time $t = T$ computed using Euler-Maruyama (solid black) and our method (red dash). We see that the difference is quite small. If we were to present similar plots for the other problems solved at 401 grid points, the p.d.f.'s become indistinguishable.

As expected, keeping all problem parameters the same but using zero instead of non-zero initial conditions causes the error to decrease. This is because the zero initial condition problem is solved using a modified algorithm that exploits symmetry as described in Section 2.6.

The bottom half of Fig. 1 shows interesting behavior in the relative error in the first two moments. If one is primarily interested in the mean (first moment) of the solution, then our method appears to provide a convergent algorithm. For the Param. I and II non-zero IC problems, the error curves display nearly the same rate of convergence for grid sizes greater than 201: the least-squares lines have slopes -0.88 and -0.89. The relative error in the second moment decreases monotonically for all problems up to a grid size of 401. The Param. I problem's second moment error increases as we increase the grid size to 801 and 1601. We discuss possible reasons for this below.

4. DISCUSSION

There are two primary sources of error in our algorithm: quadrature and the long-range Gaussian assumption. By long-range Gaussian error, we mean the error incurred by approximating $p(x_n, x_{n-\ell})$ by a two-dimensional Gaussian, as discussed in Section 2.2. By quadrature error, we mean the error generated by our approximation of the integral in (4), as well as other integrals that we must do in order to compute $p(x_n, x_{n-\ell})$ in (4). Note that even if we somehow knew an exact formula for $p(x_n, x_{n-\ell})$, we would still incur quadrature error when we compute the right-hand side of (4).

In order to determine which source of error is dominant for our nonlinear problem, we ran a series of tests on the linear problem obtained by setting $C = 0$ in (1b). In this case, (i) the Gaussian approximation is exact, and (ii) we have an exact solution (Bhat, 2014) against which to compare. For

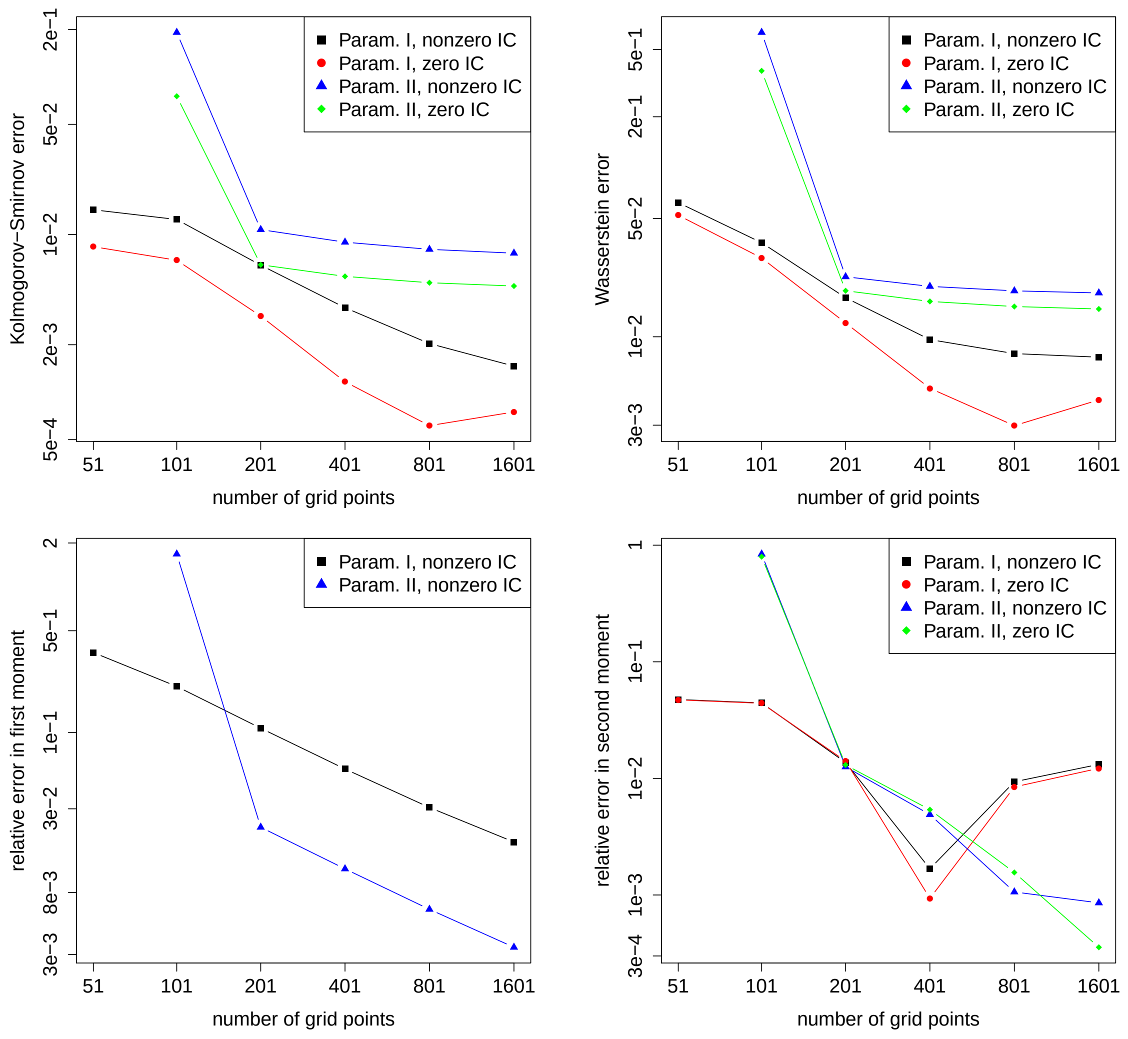

Fig. 1. We plot the errors obtained by comparing, using four different metrics, our method against the Euler-Maruyama method. All axes on all plots are logarithmically scaled. Each plot shows errors on up to four different problems, where each problem is defined by a set of parameters and initial conditions (IC). For details regarding the choices of parameters (Param. I and II) and initial conditions, see Section 3. The results show that, in the Kolmogorov-Smirnov and Wasserstein metrics, errors decrease monotonically as the grid size increases up to 801 points. The error in the first moment appears to converge at a problem-independent rate. The error in the second moment is discussed in the text—see Section 3.

the linear problem, convergence as $\Delta\zeta \to 0$ is problematic if we use the trapezoidal rule. Based on this finding, we wrote and tested a version of the code that uses Gauss-Hermite quadrature. In this case, we modify (4) by writing

$$p(x_{n+1}, x_n) = \int\limits_{x_{n-\ell}} p(x_{n+1}|x_n, x_{n-\ell}) p(x_{n-\ell}|x_n) p(x_n) dx_{n-\ell}. \tag{13}$$

Because $p(x_n, x_{n-\ell})$ is Gaussian, $p(x_{n-\ell}|x_n)$ is a one-dimensional Gaussian, which can be transformed to κe^{-z^2} through an appropriate change of variables. Here κ stands for a normalization constant. Schematically, then, (13) is of the form

$$p(x_{n+1}, x_n) = \kappa p(x_n) \int \varphi(z) e^{-z^2} \, dz. \tag{14}$$

Gauss-Hermite quadrature produces highly accurate answers to this integral. Using Gauss-Hermite quadrature throughout the code, we find convergence of $f_{X(T)}(x)$ to the exact solution as $\Delta\zeta \to 0$, for the linear problem.

Returning to the nonlinear problem, we reran the tests described above using Gauss-Hermite quadrature. The abscissae in Gauss-Hermite quadrature consist of zeros of the Hermite polynomials; we could not reliably find arbitrary large numbers of these zeros, and hence use a grid of at most 401 points. We find that the typical error of our method on either nonlinear problem increases when we use Gauss-Hermite rather than trapezoidal quadrature.

To visualize directly the error that is made using the current Gaussian approximation, we present Fig. 3. Here we plot the joint density $p(x_n, x_{n-\ell})$ at the final time $t = T$ for the Param. I problem with nonzero initial conditions, as defined in Section 3. On the left, the plotted density is computed by applying kernel density estimation to 10^7 samples calculated using Euler-Maruyama. On the right, the plotted density is the two-dimensional Gaussian approximation from Section 2.2. The approximation is in qualitative agreement with the kernel density estimate, with regards to the overall shape, center, and decay rate. Quantitatively, we can compare the mean vectors

$$\boldsymbol{\mu}_{\mathrm{EM}} = \begin{bmatrix} 0.1248 \\ 0.2285 \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} 0.1321 \\ 0.2284 \end{bmatrix},$$

and the covariance matrices

$$\Sigma_{\mathrm{EM}} = \begin{bmatrix} 1.971 & 1.250 \\ 1.250 & 1.719 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1.967 & 1.268 \\ 1.268 & 1.723 \end{bmatrix}.$$

Overall, our algorithm does a reasonable job of capturing the covariance between x_n and $x_{n-\ell}$, which was our goal.

However, it is clear from Fig. 3 that the empirical joint density $p(x_n, x_{n-\ell})$ is not exactly Gaussian. In our method, at each time step, we compute an integral involving a Gaussian approximation to this joint density. When the spatial grid is coarse enough (i.e., at most 401 points for the trapezoidal results in Fig. 1), the error committed by spatial quadrature exceeds the error of the Gaussian approximation. In this regime, when we go from say $N = 51$ points to $N = 401$ points, we observe monotonic convergence in all metrics. As we increase the grid resolution, the quadrature error drops off and we expose the error in the Gaussian approximation. This explains the non-monotonic convergence curves in Fig. 1, and it also explains the Gauss-Hermite results described above.

Clearly, in order for our method to converge, we must use more accurate approximations of the long-range joint density $p(x_n, x_{n-\ell})$. One approach to doing this is motivated by the theory of kernel density estimation: specifically, if we represent the joint density by a sum of two-dimensional Gaussians, each centered at a different point, then as the number of Gaussians increases, we can more accurately represent any joint density $p(x_n, x_{n-\ell})$. We could then use our method to handle problems in which the true joint density is multimodal.

An Euler-Maruyama run with 10^8 samples takes considerably longer to run than our method, again using the same time step h for both methods. Taking this into consideration, the accuracy of our method may be sufficient, depending on the application.

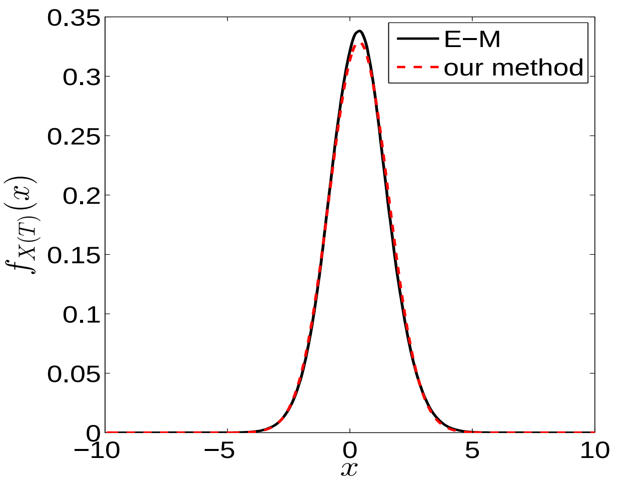

Fig. 2. Comparison of p.d.f.'s for $X(T)$ computed using Euler-Maruyama (10^7 samples) and our method for the Param. II problem with nonzero initial conditions on a grid of size 401. The p.d.f.'s are in close agreement, with only a slight discrepancy near the center.

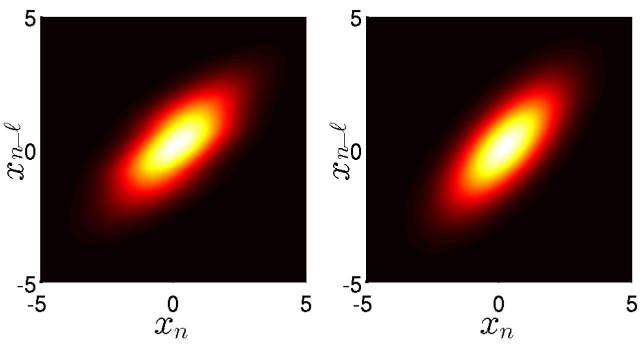

Fig. 3. Joint densities $p(x_n, x_{n-\ell})$ at the final time $t = T$ computed using Euler-Maruyama (left, with 10^7 samples) and our method (right, with 401 grid points). The joint densities are in qualitative agreement.

REFERENCES

Bhat, H.S. (2014). Algorithms for linear stochastic delay differential equations. In V.B. Melas, S. Mignani, P. Monari, and L. Salmaso (eds.), *Topics in Statistical Simulation*, 57–65. Springer.

Boulet, J., Balasubramaniam, R., Daffertshofer, A., and Longtin, A. (2009). Stochastic two-delay differential model of delayed visual feedback effects on postural dynamics. *Phil. Trans. Roy. Soc. A*, 368(1911), 423–438.

Buckwar, E., Kuske, R., Mohammed, S.E., and Shardlow, T. (2008). Weak convergence of the Euler scheme for stochastic differential delay equation. *LMS J. Mathematics and Computing*, 11, 60–99.

Cabrera, J.L. (2005). Controlling instability with delayed antagonistic stochastic dynamics. *Physica A*, 356(1), 25–30.

Eurich, C.W. and Milton, J.G. (1996). Noise-induced transitions in human postural sway. *Phys. Rev. E*, 54, 6681–6684.

Gyöngy, I. and Sabanis, S. (2013). A note on Euler approximations for stochastic differential equations with delay. *Appl. Math. Opt.*, 68, 391–412.

Kowalczyk, P., Nema, S., Glendinning, P., Loram, I., and Brown, M. (2014). Auto-regressive moving average analysis of linear and discontinuous models of human balance during quiet standing. *Chaos*, 24, 022101.

Longtin, A. (2009). Stochastic delay-differential equations. In F.M. Atay (ed.), *Complex Time-Delay Systems*, 177–195. Springer, Berlin, Heidelberg.

Milton, J.G. (2011). The delayed and noisy nervous system: implications for neural control. *J. Neural Engineering*, 8(6), 065005.

Milton, J.G., Townsend, J.L., King, M.A., and Ohira, T. (2009). Balancing with positive feedback: the case for discontinuous control. *Phil. Trans. Roy. Soc. A*, 367(1891), 1181–1193.

Suzuki, Y., Nomura, T., Casadio, M., and Morasso, P. (2012). Intermittent control with ankle, hip, and mixed strategies during quiet standing: A theoretical proposal based on a double inverted pendulum model. *J. Theoretical Biology*, 310, 55–79.