Bayesian Inference of Stochastic Pursuit Models from Basketball Tracking Data

Harish S. Bhat, R. W. M. A. Madushani, and Shagun Rawat

1 Introduction

In 2010, the National Basketball Association (NBA) began to install a camera system to track the positions of the players and the ball as a function of time. For the ball and for each of the 10 players on the court, the system records an (x, y) position 25 times per second. Ultimately, this wealth of data should enable us to answer a number of questions regarding basketball strategy that would have seemed intractable just a few years ago. To bring this vision to reality, we must develop new algorithms that can efficiently use the data for inference of appropriate models.

In this work, we focus on so-called "fast break" situations where an offensive player races towards the basket in an attempt to score before the defensive team has time to set up their defense. In many such situations, it is relatively easy to identify from the data a runner and a chaser. This motivates the following question that is central to the present paper: using the NBA's spatial tracking data, how can we infer a stochastic model for the chaser's pursuit of the runner?

To answer this question, we first formulate a stochastic version of the classical pursuit model. Our model consists of a set of coupled, nonlinear stochastic differential equations with time-dependent coefficients. To perform Bayesian inference for this stochastic model, we develop a Markov Chain Monte Carlo (MCMC) algorithm. The MCMC algorithm is derived using a Metropolis scheme; our innovation is to evaluate the log likelihood efficiently using a novel, deterministic method called

Harish S. Bhat
University of California Merced, 5200 N. Lake Rd., Merced, CA, USA
e-mail: hbhat@ucmerced.edu

R. W. M. A. Madushani
University of California Merced, 5200 N. Lake Rd., Merced, CA, USA
e-mail: rmadushani@ucmerced.edu

Shagun Rawat
University of California Merced, 5200 N. Lake Rd., Merced, CA, USA
e-mail: srawat2@ucmerced.edu

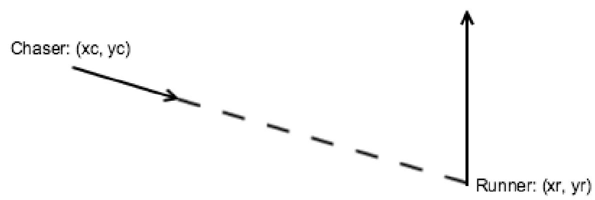

Fig. 1 Diagram illustrating motion of runner and chaser. At any instant of time, the chaser's velocity vector points toward the runner's current position.

density tracking by quadrature (DTQ). The DTQ method applies quadrature to the Chapman-Kolmogorov equation associated with a time-discretization of the original stochastic differential equation (SDE) [3]. For the case of scalar SDE, the DTQ method's density function converges to the true density of the SDE at a rate that is linear in the time step.

Note that the MCMC algorithm developed here can be applied for Bayesian inference of a class of two-dimensional SDE, not just the pursuit model considered here. Note that inference of SDE models is a challenging problem, due to the fact that a closed-form likelihood function is generally unavailable [8, 5, 4]. Most existing parametric inference methods for discretely observed SDE require inter-observation times to be small. As a way to facilitate approximation of the transition density for parametric inference for large inter-observation times, Bayesian methods are used to simulate missing values of the observations to form a high-frequency data set. In situations where the likelihood function is either analytically unavailable or computationally prohibitive to evaluate, Bayesian inference of SDE makes use of likelihood-free methods such as Approximate Bayesian Computation [6], variational methods [2, 9], and/or Gaussian processes [1, 7]. In ongoing and future work, we will conduct a careful comparison of our method against these other methods. For the purposes of the present paper, we are more interested in establishing the appropriateness of a stochastic pursuit model for basketball fast breaks.

2 Derivation of the Model and Inference Method

Let the *runner* be the player (on offense) who has the ball and is running toward the basket. Let the *chaser* be the player (on defense) who is trying to prevent the runner from scoring. Let the current spatial coordinates of the runner and chaser be, respectively, $(x^r(t), y^r(t))$ and $(x^c(t), y^c(t))$.

Consider the diagram in Figure 1. Since the chaser is moving towards the runner, the velocity vector of the chaser points toward the runner's current position. Let $\phi = (x^r - x^c, y^r - y^c)$. Then the unit vector that points toward the runner from the chaser is $\phi/\|\phi\|$. The velocity of the chaser, $(\dot{x}^c, \dot{y}^c)$, can thus be given as

$$(\dot{x}^c, \dot{y}^c) = \gamma(t)\frac{\phi}{\|\phi\|}, \tag{1}$$

where $\gamma(t) = \|(\dot{x}^c, \dot{y}^c)\|$, the instantaneous speed of the chaser. Note that (1) is a coupled system of nonlinear ordinary differential equations known as the pursuit model—classically, one assumes that $\gamma(t)$ and $(x^r(t), y^r(t))$ are given, in which case one typically solves an initial-value problem for $(x^c(t), y^c(t))$. To generalize the classical model to the real data context considered here, we multiply both sides of (1) by $\mathrm{d}t$ and then add noise to each component:

$$\mathrm{d}(x^c, y^c) = \gamma(t)\frac{\phi}{\|\phi\|}\mathrm{d}t + (\nu_1 \mathrm{d}W_t^1, \nu_2 \mathrm{d}W_t^2) \tag{2}$$

Here $W_{1,t}$ and $W_{2,t}$ denote two independent Wiener processes with $W_{1,0} = W_{2,0} = 0$ almost surely. We refer to this model as the stochastic pursuit model.

Given time-discrete observations of (x^c, y^c) and (x^r, y^r), how do we infer $\gamma(t)$ together with ν_1 and ν_2? Our first step is to consider (2) as a particular example of a more general class of systems:

$$\mathrm{d}X_{1,t} = f_1(t, \mathbf{X}_t, \theta)\mathrm{d}t + g_1(t, \mathbf{X}_t, \theta)\mathrm{d}W_{1,t} \tag{3a}$$

$$\mathrm{d}X_{2,t} = f_2(t, \mathbf{X}_t, \theta)\mathrm{d}t + g_2(t, \mathbf{X}_t, \theta)\mathrm{d}W_{2,t}. \tag{3b}$$

Here $\mathbf{X}_t = (X_{1,t}, X_{2,t})$ is a two-dimensional stochastic process. For $j = 1,2$, we refer to f_j and g_j as, respectively, drift and diffusion functions. Both drift and diffusion functions may depend on a parameter vector $\theta \in \mathbb{R}^N$.

For the stochastic pursuit model (2), we take $\mathbf{X}_t = (x^c(t), y^c(t))$. We treat $\gamma(t)$ as piecewise constant. Each constant value of $\gamma(t)$ is one component of the parameter vector θ; the final two values of this vector are ν_1 and ν_2. Finally, if we treat $(x^r(t), y^r(t))$ as given, then we can identify the time-dependent drift functions f_1 and f_2 as the two components of $\gamma(t)\phi/\|\phi\|$.

Our goal is to infer θ from direct, discrete-time observations of $\mathbf{X}_t$. Suppose that at a sequence of times $0 = t_0 < t_1 < \cdots < t_M = T$, we have observations $\mathbf{x} := \{(x_{1,m}, x_{2,m})\}_{m=0}^M$. Here $\mathbf{x}_m = (x_{1,m}, x_{2,m})$ is a sample of $\mathbf{X}_{t_m}$. In this paper, we will assume equispaced temporal observations, i.e., $t_m = m\Delta t$ for fixed step size $\Delta t > 0$. We make this assumption purely for notational simplicity; the method we describe can be easily adapted for nonequispaced temporal observations. We refer to Δt as the time step of the data.

The posterior density of the parameter vector given the observations is $p(\theta \,|\, \mathbf{x}) \propto p(\mathbf{x} \,|\, \theta)p(\theta)$, where $p(\mathbf{x} \,|\, \theta)$ is the likelihood and $p(\theta)$ is the prior. We discretize the SDE (3) in time using the Euler-Maruyama scheme:

$$X_1^{n+1} = X_1^n + f_1(t_n, X_1^n, X_2^n, \theta)h + g_1(t_n, X_1^n, X_2^n, \theta)\sqrt{h}Z_1^{n+1} \tag{4a}$$

$$X_2^{n+1} = X_2^n + f_2(t_n, X_1^n, X_2^n, \theta)h + g_2(t_n, X_1^n, X_2^n, \theta)\sqrt{h}Z_2^{n+1}. \tag{4b}$$

Here $h > 0$ is a fixed time step, the time step of our numerical method. We shall choose h to be a fraction of Δt, i.e., $Fh = \Delta t$ for integer $F \geq 2$. The random variables X_i^n for $i = 1,2$ are approximations of $X_{i,nh}$. The Z_i^n are independent and identically distributed random variables, normally distributed with mean 0 and variance 1, i.e., $Z_i^n \sim \mathcal{N}(0,1)$.

Let $\widetilde{p}(\mathbf{x}\,|\,\theta)$ denote the likelihood under the discrete-time model (4), an approximation to the true likelihood $p(\mathbf{x}\,|\,\theta)$. Note that (4) describes a discrete-time Markov chain. By the Markov property, the likelihood $\widetilde{p}(\mathbf{x}\,|\,\theta)$ factors and we can write:

$$p(\mathbf{x}\,|\,\theta) \approx \widetilde{p}(\mathbf{x}\,|\,\theta) = \prod_{m=0}^{M-1} \widetilde{p}(\mathbf{x}_{m+1}\,|\,\mathbf{x}_m, \theta). \tag{5}$$

The term $\widetilde{p}(\mathbf{x}_{m+1}\,|\,\mathbf{x}_m, \theta)$ is the transition density for (4), from state $\mathbf{x}_m$ at time t_m to state $\mathbf{x}_{m+1}$ at time t_{m+1}. This suggests a numerical method for computing this density, which we explore in the next subsection.

2.1 Density Tracking by Quadrature (DTQ)

Equation (4) describes a Markov chain over a continuous state space. If we let $\widetilde{p}^n(x_1,x_2\,|\,\theta)$ denote the joint probability density function of X_1^n and X_2^n given θ, then the Chapman-Kolmogorov equation associated with (4) is

$$\widetilde{p}^{n+1}(x_1,x_2\,|\,\theta) = \int_{y_1,y_2 \in \mathbb{R}^2} K(x_1,x_2,y_1,y_2,t_n;\theta)\widetilde{p}^n(y_1,y_2\,|\,\theta)\,dy, \tag{6}$$

where

$$\begin{aligned} &K(x_1,x_2,y_1,y_2,t_n;\theta) = \widetilde{p}^{n+1|n}(x_1,x_2|y_1,y_2,\theta) \\ &= (2\pi\sigma_1^2)^{-1/2} \exp\left[-(x_1-\mu_1)^2/(2\sigma_1^2)\right] (2\pi\sigma_2^2)^{-1/2} \exp\left[-(x_2-\mu_2)^2/(2\sigma_2^2)\right]. \end{aligned}$$

Here $\mu_1 = y_1 + f_1(t_n,y_1,y_2;\theta)h, \mu_2 = y_2 + f_2(t_n,y_1,y_2;\theta)h, \sigma_1^2 = g_1^2(t_n,y_1,y_2;\theta)h$ and $\sigma_2^2 = g_2^2(t_n,y_1,y_2;\theta)h$. That is, $K(x_1,x_2,y_1,y_2,t_n;\theta)$ is the conditional density of X_1^{n+1} and X_2^{n+1} given $X_1^n = y_1$, $X_2^n = y_2$ and $\theta = \theta$, evaluated at the point (x_1,x_2). The fact that the conditional density is a product of normal distributions with means μ_1,μ_2 and variances σ_1^2,σ_2^2 can be shown using (4) together with the fact that X_1^{n+1} and X_2^{n+1} are conditionally independent given X_1^n and X_2^n. This conditional independence is a direct consequence of having two independent random variables Z_1^n and Z_2^n in (4).

The crux of the DTQ method is to apply quadrature to (6) to evolve an initial density forward in time. Consider a spatial grid with fixed spacing $k > 0$ and grid points $x_1^i = ik$, $x_2^j = jk$, $y_1^{i'} = i'k$, and $y_2^{j'} = j'k$. Then we apply the trapezoidal rule in both the y_1 and y_2 variables to obtain:

$$\hat{p}^{n+1}(x_1^i, x_2^j; \theta) = k^2 \sum_{i'=-\infty}^{\infty} \sum_{j'=-\infty}^{\infty} K(x_1^i, x_2^j, y_i^{i'}, y_2^{j'}, t_n; \theta) \hat{p}^n(y_1^{i'}, y_2^{j'}; \theta) \tag{7}$$

It is unnecessary to sum over all of $\mathbb{Z}^2$. We know that a two-dimensional Gaussian decays to zero far from its mean. Since the mean (μ_1, μ_2) is approximately (y_1, y_2), we sum only from $y_1 = x_1 - \zeta k$ to $y_1 = x_1 + \zeta k$ and similarly for y_2:

$$\hat{p}^{n+1}(x_1^i, x_2^j; \theta) = k^2 \sum_{i'=i-\zeta}^{i+\zeta} \sum_{j'=j-\zeta}^{j+\zeta} K(x_1^i, x_2^j, y_i^{i'}, y_2^{j'}, t_n; \theta) \hat{p}^n(y_1^{i'}, y_2^{j'}; \theta) \tag{8}$$

We choose ζ manually to ensure the accuracy of the computation. We now have our method to evaluate $\widetilde{p}(\mathbf{x}_{m+1} \,|\, \mathbf{x}_m, \theta)$. Let us take $n = 0$ in (8) to correspond to the time t_m. We start with the deterministic initial condition $\mathbf{X}^0 = \mathbf{x}_m$, corresponding to the density $\widetilde{p}^0(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_m)$. Inserting this point mass into (6), we obtain a Gaussian density for $\widetilde{p}^1(\mathbf{x})$. For each $i, j \in [-y_M/k, y_M/k]$, we set

$$\hat{p}^1(x_1^i, x_2^j; \theta) = \widetilde{p}^1(x_1^i, x_2^j; \theta).$$

Now that we have $\hat{p}^1$, we use (8) repeatedly to compute $\hat{p}^2$, $\hat{p}^3$, and so on until we reach $\hat{p}^F$. The object $\hat{p}^F$ is then a spatially discrete approximation of the transition density from time t_m to time $t_m + Fh = t_{m+1}$. For this last density, instead of evaluating it on the spatial grid used by the trapezoidal rule, we evaluate the density at the data $\mathbf{x}_{m+1}$. This avoids interpolation. In this way, we compute a numerical approximation of $\widetilde{p}(\mathbf{x}_{m+1} \,|\, \mathbf{x}_m, \theta)$, as required for the likelihood function.

2.2 Metropolis Algorithm

Once we have an efficient method to compute the likelihood, we can use the method in a Metropolis algorithm to sample from the posterior. In the Metropolis algorithm, we construct an auxiliary Markov chain $\{\hat{\theta}_N\}_{N \geq 0}$ which is designed to have an invariant distribution given by the posterior $p(\theta \,|\, \mathbf{x})$. This Markov chain is constructed as $\hat{\theta}_{N+1} = \hat{\theta}_N + Z_{N+1}$, where Z_{N+1} is a random vector with dimension equal to that of the parameter vector θ. In this paper, we choose all components of Z_{N+1} to be independent normal random variables with known means and variances.

The Metropolis algorithm is as follows:

- Choose value q_0 for $\hat{\theta}_0$.
- Once the values $q_0, \cdots, q_N$ of $\hat{\theta}_0, \cdots, \hat{\theta}_N$ have been found:
  - Generate a proposal from the auxiliary Markov chain: $q_{N+1}^* = q_N + Z_{N+1}$.
  - Calculate the ratio $\rho = \frac{p(q_{N+1}^* \,|\, \mathbf{x})}{p(q_N \,|\, \mathbf{x})}$, where $p(q_{N+1}^* \,|\, \mathbf{x}) \approx \widetilde{p}(\mathbf{x} \,|\, q_{N+1}^*) p(q_{N+1}^*) = p(q_{N+1}^*) \prod_{m=0}^{M-1} \widetilde{p}(\mathbf{x}_{m+1} \,|\, \mathbf{x}_m, q_{N+1}^*)$. Now each term $\widetilde{p}(\mathbf{x}_{m+1} \,|\, \mathbf{x}_m, q_{N+1}^*)$ can be computed using the DTQ method discussed in Section 2.1.

– Sample $u_N \sim \mathscr{U}(0,1)$. If $\rho > u_N$ set $\hat{\theta}_{N+1} = q^*_{N+1}$; in this case, the proposal is accepted. Else set $\hat{\theta}_{N+1} = q_N$ and the proposal is rejected.

Once we have obtained all the samples $q_0, q_1, \cdots, q_N$ from the Metropolis algorithm, we discard a sufficient number of initial samples to ensure the Markov chain has converged to its invariant distribution.

3 Numerical Tests

We implement the Metropolis algorithm in R. Inside the Metropolis algorithm, we evaluate the likelihood function using the DTQ method, which is implemented in C++ as an R package. Note that all code and data used in this work is available online (https://github.com/hbhat4000/sdeinference)—see the "Rdtq2d" and "pursuit2d" directories. To test the method, we first consider the SDE

$$dX_{1,t} = -\frac{X_{2,t}}{L}dt + \frac{s_1^2}{L}dW_{1,t}, \qquad dX_{2,t} = \frac{X_{1,t}}{C}dt + \frac{s_2^2}{C}dW_{2,t}. \tag{9}$$

This system describes a noisy electrical oscillator with one inductor (with inductance L) and one capacitor (with capacitance C). The dependent variables $X_{1,t}$ and $X_{2,t}$ represent, respectively, the current and voltage of the circuit at time t.

Our goal here is to test the performance of the algorithm using simulated data. To generate this data, we start with known values of the parameters: $L = C = (2\pi)^{-1}$ and $s_1 = s_2 = .4/\sqrt{2\pi}$. Using a fixed initial condition $(X_{1,0}, X_{2,0})$, we then use the Euler-Maruyama method to step (9) forward in time until a final time $T > 0$. When we carry out this time-stepping, we use a step size of 0.001 and then retain only those samples at times $t_m = m\Delta t$, from $m = 0$ to $m = M$, where $M\Delta t = T$. The simulated data is taken over two periods of the oscillator ($T = 2$) with a full resolution of $\Delta t = 0.01$. By, for example, taking every other row of this data set, we can obtain data with a resolution of $\Delta t = 0.02$.

Using the samples $\{\mathbf{x}_m\}_{m=0}^M$ thus constructed, we run the Metropolis algorithm. Because capacitance and inductance are physically constrained to be positive, we set $1/L = \theta_1^2$. For the tests presented here, we infer only θ_1, keeping other parameters fixed at their known values. For θ_1, we use a diffuse Gaussian prior with mean 0 and standard deviation 100. For the proposal distribution Z_{N+1} in the auxiliary Markov chain, we choose i.i.d. Gaussians with mean 0 and standard deviation 0.35.

When we run the Metropolis algorithm, we discard the first 100 samples and retain the next 1000 samples. For each value of Δt and the DTQ time step h, we compute both the mean of the samples of θ_1^2 and the mode of the kernel density estimate of θ_1^2. We compare these values against the true value of the parameter $1/L = 2\pi$ and record the relative errors as, respectively, e_1 and e_2:

Δt	h	e_1 (relative error of mean)	e_2 (relative error of mode)
0.04	0.04	6.1%	7.6%
0.04	0.02	0.54%	6.8%
0.04	0.01	5.1%	1.1%
0.02	0.02	12%	14%
0.02	0.01	4.9%	2.3%

When $h = \Delta t$, only one step of the method described in Section 2.1 is required to go from time t_m to t_{m+1}. This step does not use any quadrature at all—one merely evaluates (6) using a point mass for the density at time t_m. The resulting likelihood function is a product of Gaussians. On the other hand, when h is strictly less than Δt, we must use quadrature (i.e., the actual DTQ method) to step forward in time from t_m to t_{m+1}. Clearly, using the DTQ method to compute the likelihood yields more accurate posteriors than using a purely Gaussian likelihood.

To visualize the results, we present Figure 2. The true value of $1/L = 2\pi$ is indicated by the dashed black line. The posterior mode for $h = 0.01$ is indicated by the solid black line. The curves are kernel density estimates computed using the posterior samples described above.

Fig. 2 We plot kernel density estimates of posterior densities $p(\theta_1^2|\mathbf{x})$. We use simulated data with $\Delta t = 0.04$, generated as described above. Each posterior density corresponds to a finer DTQ step h. As we take a finer DTQ step (i.e., as h decreases), the posterior mode approaches the true value indicated by the solid vertical line at $1/L = 2\pi$.

Next, we test the method using the pursuit SDE (2). We set the runner's trajectory equal to a sinusoidal curve $y = \sin(\pi x)$ from $x = -1$ to $x = 1$. We assume the runner covers this trajectory over the time period $0 \le t \le 8$. The chaser's trajectory is simulated using the Euler-Maruyama method to step (2) forward in time from a fixed initial condition $\mathbf{X}_0 = (x_0^c, y_0^c)$. During the generation of the data, we use a step size of 10^{-4}. By downsampling this single time series, we generate time series with spacing $\Delta t = 0.4, 0.2, 0.1$.

For the test presented here, the values of the parameters are $\nu_1 = 0.15$, $\nu_2 = 0.1$, and $\gamma(t) = \begin{cases} \gamma_1 = 0.4 \text{ if } 0 \le t < 4 \\ \gamma_2 = 1.0 \text{ if } 4 \le t \le 8. \end{cases}$

Because we want all speeds and diffusion constants to be positive, we take $\gamma_i = e^{\theta_i}$ and $\nu_i = e^{\theta_{i+2}}$ for $i = 1, 2$. The priors for θ_1 and θ_2 are normal with variance one and mean equal to the log of the mean speed of the chaser computed over the chaser's entire trajectory. The priors for θ_3 and θ_4 are normal with mean $\log(0.4)$ and variance 1. We use mean zero Gaussian proposals for all components of θ. We choose the variances of these proposals so that the acceptance rate for all runs is near 30%.

Using the samples $\{\mathbf{x}_m\}_{m=0}^M$ thus constructed, we run the Metropolis algorithm with $h = \Delta t/i$ with $i = 1, 2, 3, 4$. For each choice of parameters Δt and h, we compute 10100 samples and discard the first 100. To compute the runner's trajectory at intermediate points, we use linear interpolation between times t_m and t_{m+1}. We tabulate the results below; each value of γ_1 represents the mean of e^{θ_1} over all Metropolis samples of θ_1:

parameters	γ_1	γ_2	ν_1	ν_2
$\Delta t = 0.1; h = 0.1/1$	0.301	0.748	0.124	0.0886
$\Delta t = 0.1; h = 0.1/2$	0.311	0.956	0.124	0.0858
$\Delta t = 0.1; h = 0.1/3$	0.307	1.011	0.117	0.0805
$\Delta t = 0.1; h = 0.1/4$	0.308	1.025	0.120	0.0829
$\Delta t = 0.2; h = 0.2/1$	0.306	0.650	0.142	0.1146
$\Delta t = 0.2; h = 0.2/2$	0.310	0.877	0.137	0.1197
$\Delta t = 0.2; h = 0.2/3$	0.309	1.015	0.112	0.0844
$\Delta t = 0.2; h = 0.2/4$	0.304	1.019	0.111	0.0852
$\Delta t = 0.4; h = 0.4/1$	0.292	0.514	0.188	0.2010
$\Delta t = 0.4; h = 0.4/2$	0.312	0.960	0.177	0.1774
$\Delta t = 0.4; h = 0.4/3$	0.307	0.987	0.124	0.1447
$\Delta t = 0.4; h = 0.4/4$	0.303	1.014	0.145	0.1130

Overall, the results show that our algorithm produces mean posterior estimates that are reasonably close to the ground truth values. When the spacing of the data Δt is large, we see greater benefit from using the DTQ method. For instance, when $\Delta t = 0.4$, the mean estimates of γ_2 improve dramatically from 0.514 to 1.014 as we decrease h, i.e., as we take more internal DTQ steps. Similar trends can be seen for ν_1 and ν_2.

4 NBA Tracking Data

We now turn to real tracking data taken from the game played between the Golden State Warriors and the Sacramento Kings on October 29, 2014. Reviewing this game, we found a fast break where Stephen Curry (of the Warriors) was the runner and Ramon Sessions (of the Kings) was the chaser. The entire fast break lasts 4.12 seconds. The spatial tracking data is recorded at intervals of 0.04 seconds, for a total of 104 observations. The tracking data uses the position on a court of dimension 94×50. We have rescaled the data to lie in a square with center $(0,0)$ and side length equal to one.

To parameterize the chaser's speed $\gamma(t)$, we have used a piecewise constant approximation with 8 equispaced pieces. Combined with the diffusion constants ν_1 and ν_2, this yields a 10-dimensional parameter vector θ. As in the previous simulated data test, we set the true parameters γ_i and ν_i to be the exponentials of the corresponding elements of the θ vector.

For the Metropolis sampler, the priors and proposals are higher-dimensional versions of those described in the simulated data test above. The main difference is that we now generate only 1000 post-burnin samples.

Using the Metropolis samples, we compute a kernel density estimate of each parameter. We then treat the mode of each computed density as the MAP (maximum a posteriori) estimate of the corresponding parameter. We then use the MAP estimates of the parameters in the pursuit SDE (2). We generate 100 sample paths of this SDE using the Euler-Maruyama method with time step 10^{-4}. As shown in Figure 3, the mean of these sample paths (plotted in black) agrees very well with the chaser's trajectory (plotted in red). This gives evidence that our stochastic pursuit system is an appropriate model for NBA fast breaks involving one runner and one chaser.

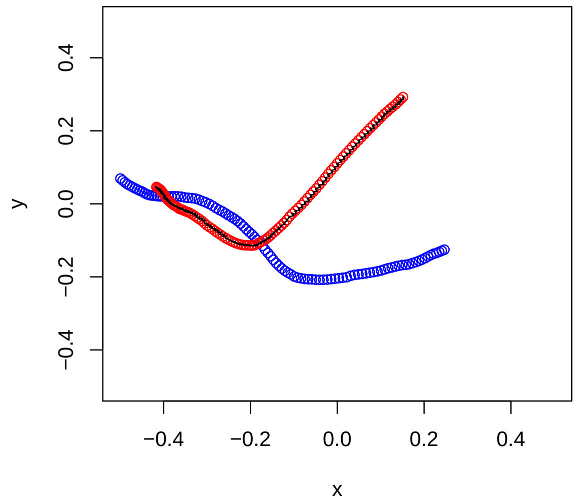

Fig. 3 The agreement between the black curve (mean of simulated stochastic pursuit trajectories using MAP estimated parameters) and the red curve (chaser's trajectory) shows that the stochastic pursuit model is appropriate. The runner's trajectory is given in blue.

To visualize the insight provided by the model, we plot in Figure 4 the MAP estimated $\gamma(t)$ function over the time period of the fast break, $0 \leq t \leq 4.12$. The speed $\gamma(t)$ is the piecewise constant function plotted in black, while the mean speed computed directly from the data is given by a red horizontal line. The inferred speed shows that the chaser slows down dramatically approximately 1.5 seconds into the fast break. If one reviews the video footage of the play, this corresponds to the runner confusing the chaser and evading him.

Given our stochastic pursuit model's success in fitting the real data, in future work, we seek to apply the same methodology to a much larger sample of fast breaks. In this way, we can quantify a runner's ability to evade a chaser and/or a chaser's ability to stay near a runner who is actively trying to score.

References

1. Archambeau, C., Cornford, D., Opper, M., Shawe-Taylor, J.: Gaussian process approximations of stochastic differential equations. Journal of Machine Learning Research: Workshop and Conference Proceedings **1**, 1–16 (2007)
2. Archambeau, C., Opper, M., Shen, Y., Cornford, D., Shawe-Taylor, J.: Variational inference for diffusion processes. In: Advances in Neural Information Processing Systems 20, pp. 17–24 (2007)
3. Bhat, H.S., Madushani, R.W.M.A.: Density tracking by quadrature for stochastic differential equations (2016). Preprint

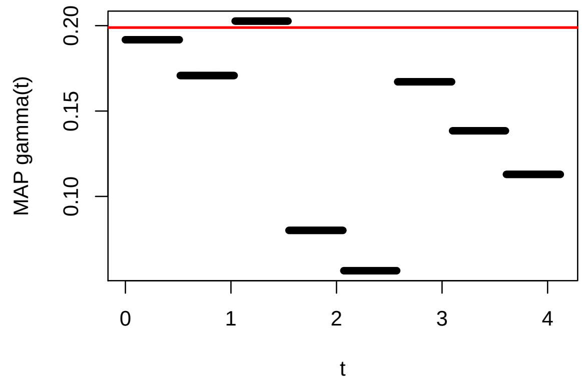

Fig. 4 For the fast break tracking data described in the text, we plot the MAP estimate of the chaser's speed $\gamma(t)$ in black. Note that the inferred speed differs greatly from the mean speed across the entire trajectory, plotted as a horizontal red line.

4. Fuchs, C.: Inference for Diffusion Processes: With Applications in Life Sciences. Springer, Berlin (2013)
5. Iacus, S.M.: Simulation and Inference for Stochastic Differential Equations: With R Examples. Springer Series in Statistics. Springer, New York (2009)
6. Picchini, U.: Inference for SDE models via approximate Bayesian computation. Journal of Computational and Graphical Statistics **23**(4), 1080–1100 (2014)
7. Ruttor, A., Batz, P., Opper, M.: Approximate Gaussian process inference for the drift function in stochastic differential equations. In: Advances in Neural Information Processing Systems 26, pp. 2040–2048 (2013)
8. Sørensen, H.: Parametric inference for diffusion processes observed at discrete points in time: a survey. International Statistical Review **72**(3), 337–354 (2004)
9. Vrettas, M.D., Opper, M., Cornford, D.: Variational mean-field algorithm for efficient inference in large systems of stochastic differential equations. Physical Review E **91**(012148) (2015)