Solving Feynman-Kac Forward–Backward SDEs Using McKean-Markov Branched Sampling

Kelsey P. Hawkins, Ali Pakniyat, *Member, IEEE*, Evangelos Theodorou, and Panagiotis Tsiotras, *Fellow, IEEE*

***Abstract*—We propose a new method for the numerical solution of the forward–backward stochastic differential equations (FBSDE) appearing in the Feynman-Kac representation of the value function in stochastic optimal control problems. Using Girsanov's change of probability measures, it is demonstrated how a McKean-Markov branched sampling method can be utilized for the forward integration pass, as long as the controlled drift term is appropriately compensated in the backward integration pass. Subsequently, a numerical approximation of the value function is proposed by solving a series of function approximation problems backwards in time along the edges of a space-filling tree consisting of trajectory samples. Moreover, a local entropy-weighted least squares Monte Carlo (LSMC) method is developed to concentrate function approximation accuracy in regions most likely to be visited by optimally controlled trajectories. The proposed methodology is numerically demonstrated for linear and nonlinear stochastic optimal control problems with nonquadratic running costs, which reveal significant convergence improvements over previous FBSDE-based numerical solution methods.**

***Index Terms*—Least mean square methods, Monte Carlo methods, nonlinear control systems, optimal control, partial differential equations, stochastic processes, trajectory optimization, tree graphs.**

I. Introduction

THE Feynman-Kac representation theorem establishes the intrinsic relationship between the solution of a broad class of second-order parabolic and elliptic partial differential equations (PDEs) to the solution of forward–backward stochastic differential equations (FBSDEs) (see, e.g., [1, Ch. 7]). Investigations over these FBSDEs were brought to prominence in [2], [3], and [4], and they have been gaining traction as a framework to solve stochastic nonlinear control problems, including optimal control problems with quadratic cost [5], minimum-fuel (L_1-running cost) problems [6], differential games [7], [8], and reachability problems [5], [9]. FBSDE-based numerical methods have also received interest from the mathematical finance community [10], [11], [12]. This is due to the fact that the Hamilton-Jacobi-Bellman (HJB) second-order PDE appearing in stochastic optimal control (SOC) can be solved via FBSDE methods with general nonlinear dynamics and costs. Although initial results demonstrate promise in terms of flexibility and theoretical validity, numerical algorithms that leverage this theory have not yet matured. For even modest problems, state-of-the-art algorithms often have issues with slow and unstable convergence to the optimal policy. Producing more robust numerical methods is critical for the broader adoption of FBSDE methods for real-world tasks.

FBSDE numerical solution methods broadly consist of two steps, a forward pass, which generates Monte Carlo samples of the forward stochastic process, and a backward pass, which iteratively approximates the value function backwards in time. Typically, FBSDE methods perform this approximation using a least-squares Monte Carlo (LSMC) scheme, which implicitly solves the backward SDE using parametric function approximation [11]. The approximate value function fit in the backward pass is then used to improve sampling in an updated forward pass, leading to an iterative algorithm that, ideally, improves the approximation, till convergence. Although FBSDE methods share a distinct similarity to differential dynamic programming (DDP) techniques [13], [14], [15], as they also involve forward and backward passes, the latter are, in general, less flexible. For most DDP applications, a strictly positive definite running cost with respect to the control is required for convergence [16, Sec. 2.2.3]. Furthermore, in DDP the computation of first- and second-order derivatives of both the dynamics and the cost is necessary for the backward pass, making it challenging to apply this approach to problems where these derivatives cannot be computed analytically. In contrast, FBSDE techniques only require a good model of the value function and the evaluation of its gradient.

A key feature of FBSDE methods is their ability to generate a parametric model for the value function over the entire time horizon, which, in turn, can be used for the evaluation and assessment of the performance of closed-loop control policies. This feature differentiates both FBSDE and DDP methods from

Manuscript received 9 August 2023; accepted 28 December 2023. Date of publication 3 January 2024; date of current version 29 August 2024. This work was supported in part by NSF under Grant CMMI-1662523 and IIS-2008686 and in part by ONR under Grant N00014-18-1-2828. Recommended by Associate Editor A. A. Malikopoulos. *(Corresponding author: Panagiotis Tsiotras.)*

Kelsey P. Hawkins is with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: kphawkins@gmail.com).

Ali Pakniyat is with the Department of Mechanical Engineering, University of Alabama, Tuscaloosa, AL 35401 USA (e-mail: apakniyat@ua.edu).

Evangelos Theodorou and Panagiotis Tsiotras are with the School of Aerospace Engineering and the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: evangelos.theodorou@ae.gatech.edu; tsiotras@gatech.edu).

Digital Object Identifier 10.1109/TAC.2023.3349173

model predictive control (MPC) methods [17], which, in general, only produce the current-best optimal control signal, reevaluated at every time step [18].

FBSDE methods provide an alternative to grid-based methods for solving PDEs, typically utilizing finite-difference, finite-element, or level-set schemes, which are known to scale poorly in high-dimensional state spaces ($n \geq 4$). There is also ample research into the development of meshless methods for solving PDEs, such as radial basis function (RBF) collocation and RBF-finite difference (RBF-FD) formulations [19]. FBSDE methods share significant similarities with these methods, in the sense that the value function is approximated by solving the PDE over an unstructured set of collocation points. The primary drawback of RBF methods is that they do not offer an efficient method for choosing the collocation points, and since it is difficult to know a priori what the best points are, point selection might regress into a grid-based method. Specifically, sufficiently broad and dense sampling of a high-dimensional state space might require roughly the same number of collocation points as a grid-based method to be well-conditioned and to produce a quality estimate of the value function [20].

While Feynman-Kac-based FBSDE methods produce an unbiased estimator for the value function associated with HJB equations, a naïve application of the theory leads to estimators with high variance by producing sample trajectories away from the optimal one. Recent work has shown that Girsanov's change of probability measures can be employed to make changes to the sampling in the forward pass without adding intrinsic bias to the estimator [5], [6], and [8]. In other words, the drift appearing in the forward SDE as a consequence of the change in the probability measure can be employed to modify the sampling in the forward pass; this, in turn, requires appropriate accommodation for the change of measure and the associated conditional expectations in the backward pass.

In this work, we expand upon the above ideas and invoke Girsanov's theorem for Feynman-Kac FBSDEs in a broader setting than that of [5], [6], and [8], and show that the forward sampling measure can be modified at will; this enables us to incorporate methods from other domains, namely, rapidly-exploring random trees (RRTs) (see, e.g., [21] and the recent survey in [22]), in order to more efficiently explore the state space during the forward pass. RRTs are frequently applied to reachability-type motion planning problems, by biasing the samples toward regions of the state space that have low density. Using RRTs in the forward sampling allows us to spread samples evenly over the reachable state space, increasing the likelihood that near-optimal samples are well-represented in the forward pass sample distribution. By sampling more efficiently and relying less on incremental approximations of the value function to guide our search, we can achieve faster and more robust convergence than previous FBSDE methods. In the backward pass, we take advantage of the path-integrated running costs and the estimates of the value function to produce a heuristic that weighs paths according to a local-entropy measure-theoretic optimization. Although local-entropy path integral theory and RRTs have been used together in [23], the method of this article is more closely related to the path-integral approach to control [14]. Our method, similarly, performs forward passes to broadly sample the state space, but, in contrast to [23], it follows each forward pass with a backward pass to obtain an approximation of the value function, and, consequently, obtain a closed-loop policy over the full horizon.

The primary contributions of this article are as follows.

1) Provide the theoretical basis for the use of McKean-Markov branched sampling in the forward pass of FBSDE techniques.
2) Introduce an RRT-inspired algorithm for sampling the forward SDE.
3) Present a technique for concentrating value function approximation accuracy in regions containing optimal trajectories.
4) Propose an iterative numerical method for the purpose of approximating the optimal value function and its policy.

This article expands upon the authors' prior work in [24], by: first, providing missing proofs and adding the details for proving all stated theorems; second, providing a comprehensive discussion of the proposed algorithm; and third, by providing additional examples that further illustrate the theory and motivate the design choices of the proposed algorithm.

The rest of this article is organized as follows. Section II presents the SOC problem formulation, the on-policy value function representation, and the associated family of Hamilton-Jacobi equations. Next, Section III introduces a constructive series of alternative representations of the value function, first as the solution of "on-policy" FBSDEs which arise from the Feynman-Kac theorem, then as the solution of "off-policy" FBSDEs which arise from the application of Girsanov's theorem, and finally as the minimizer of a local-entropy weighted optimization problem over the off-policy FBSDE distribution. Section IV discusses branched forward SDE sampling and its novel interpretation as a discrete approximation of the continuous-time theory of Section III. In Section V, we propose the method of forward–backward rapidly exploring random trees for solving SDEs (FBRRT-SDE), a particular implementation of the representation introduced in Section IV. In Section VI, we apply FBRRT-SDE to three problems and demonstrate its ability to solve nonlinear SOC problems with nonquadratic running costs. Finally, Section VII concludes this article.

II. Hamilton-Jacobi Equation and On-Policy Value Function

In this section, we briefly introduce the SOC problem under consideration and its associated optimal value function, as well as the on-policy value function and its associated Hamilton-Jacobi PDE. Given an initial time $t \in [0, T]$ and a complete filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_s\}_{s\in[t,T]}, \mathsf{P})$ on which the n-dimensional standard Brownian (Wiener) process W_s is defined, consider a stochastic system with dynamics governed by

$$\mathrm{d}X_s = f(s, X_s, u_s)\,\mathrm{d}s + \sigma(s, X_s)\,\mathrm{d}W_s, \qquad X_t = x_t \qquad (1)$$

over the interval $[t, T]$, where X_s is an $\mathbb{R}^n$-valued, progressively measurable state process on the interval $s \in [t, T]$, $u_{[t,T]}$ is a progressively measurable input process on the same interval taking values in the compact set $U \subseteq \mathbb{R}^m$, and $f : [0, T] \times \mathbb{R}^n \times$

$U \to \mathbb{R}^n, \sigma : [0,T] \times \mathbb{R}^n \to \mathbb{R}^{n\times n}$ are the Markovian drift and diffusion functions, respectively.

For each $t \in [0,T]$, the cost over the time interval $[t,T]$ associated with a given control signal $u_{[t,T]}$ is

$$S_t[x_t, u_{[t,T]}] := \int_t^T \ell(s, X_s, u_s)\,\mathrm{d}s + g(X_T) \tag{2}$$

where $\ell : [0,T] \times \mathbb{R}^n \times U \to \mathbb{R}_+$ is the running cost, and $g : \mathbb{R}^n \to \mathbb{R}_+$ is the terminal cost. The SOC problem is to determine, given (f, σ, ℓ, g, T), the optimal value function $V^* : [0,T] \times \mathbb{R}^n \to \mathbb{R}_+$, defined as

$$V^*(t,x) = \inf_{u_{[t,T]} \in \mathcal{U}_{[t,T]}} \mathbf{E}\left[S_t[x, u_{[t,T]}] \right]) \tag{SOC}$$

where $\mathcal{U}_{[t,T]}$ is a set of admissible control processes satisfying the conditions described in the beginning of [1, Ch. 4, Sec. 3]. Among these conditions, $u_{[t,T]}$ must be progressively measurable, the SDE (1) must admit a unique solution X_s under the control $u_{[t,T]}$, and $f(\cdot, X(\cdot), u(\cdot))$ must be L^1 integrable.

Consider the HJB PDE

$$\partial_t V^* + \frac{1}{2}\operatorname{tr}[\sigma\sigma^\top \partial_{xx} V^*] + h^*(t,x,\partial_x V^*) = 0$$
$$V^*(T,x) = g(x) \tag{HJB}$$

where

$$h^*(t,x,p) := \min_{u \in U}\{\ell(t,x,u) + p^\top f(t,x,u)\} \tag{3}$$

and where $\partial_t V^*$ is the partial derivative with respect to t, $\partial_x V^*$ is the gradient with respect to state x, and $\partial_{xx} V^*$ is the Hessian with respect to state x.

Assumption 1: The following conditions hold for the problem data.

1) f, σ, ℓ, g are uniformly continuous, Lipschitz in x, for all $(t,u) \in [0,T] \times U$.
2) σ has sublinear growth in x and g has polynomial growth in x.
3) σ^{-1} exists and is bounded.

Under Assumption 1, there exists a unique weak solution X_s to (1) under each admissible control process u_s, there exists a unique viscosity solution V^* to (HJB), and the viscosity solution V^* has the representation (SOC) [1, Ch. 7, Th. 4.4].

In addition to the optimal value function V^*, we also seek to find an optimal feedback control policy π^*. According to [1, Ch. 5, Definition 6.1], we define an admissible feedback control policy $\pi : [0,T] \times \mathbb{R}^n \to U$ as a measurable function for which the SDE

$$\mathrm{d}X_s = f(s, X_s, \pi(s, X_s))\,\mathrm{d}s + \sigma(s, X_s)\,\mathrm{d}W_s, \quad X_t = x \tag{4}$$

has a weak solution. Due to the boundedness of σ^{-1}, there exists an optimal feedback control policy π^* with the verification property that $V^*(t,x) = \mathbf{E}[\,S_t[x, \pi^*]\,]$ [1, Ch. 5, Th. 6.6].

Remark II.1: Assuming the functions f, ℓ, σ are continuous and bounded along with their first partials in t and first and second partials in x, U is compact, and g is thrice continuously differential and bounded, Theorems 4.2 and 4.4 of [25, Ch. 4] yield that the PDE (HJB) admits a classical solution V^*. That is, V^* is continuously differentiable in t, and twice continuously differentiable in x, and the optimal feedback control policy satisfies

$$\pi^*(s,x) \in \arg\min_{u \in U}\{\ell(s,x,u) + (\partial_x V^*)^\top f(s,x,u)\}. \tag{5}$$

In this article, instead of a direct solution of (HJB), we work with a class of arbitrary control policies μ and their associated "on-policy" value functions V^μ, and we use iterative methods to approximate V^* and π^*. We allow more general policies of the form $\mu : [0,T] \times \mathbb{R}^n \times \mathbb{R}^n \to U$.

The Hamilton-Jacobi (HJ) PDE for the on-policy value function corresponding to the policy μ is given by

$$\partial_t V^\mu + \frac{1}{2}\operatorname{tr}[\sigma\sigma^\top \partial_{xx} V^\mu] + h^\mu(t,x,\sigma^\top \partial_x V^\mu) = 0$$
$$V^\mu(T,x) = g(x) \tag{HJ}$$

where

$$h^\mu(t,x;z) := \ell(t,x,\mu(t,x;z)) + z^\top \sigma^{-1}(t,x) f(t,x,\mu(t,x;z)). \tag{6}$$

Assumption 2: The function μ is selected such that h^μ in (6) is uniformly continuous in (t,x) and Lipschitz in z.

Under Assumptions 1–2, (HJ) admits a unique viscosity solution [1, Ch. 7, Th. 4.1]. When the control policy is chosen as the optimal policy $\mu = \pi^*$, by comparing (6) and (3), we confirm that (HJ) coincides with (HJB), and therefore the on-policy value function corresponding to the optimal policy V^{π^*} is indeed an optimal value function.

III. Feynman-Kac-Girsanov FBSDE Representation

A. Off-Policy Drifted FBSDEs

In language originating from reinforcement learning, an "on-policy" method learns a value function V^μ from trajectory samples generated by following the same policy μ, whereas "off-policy" methods learn from trajectory samples generated by following a different policy [26]. Off-policy methods are generally more desirable because by disentangling the sampling distribution from the target value function being learned, we allow for broader exploration and thus more rapid convergence to the optimal value function. Following this language, in stochastic control, an "on-policy" method would sample from the FSDE (4) with drift $f^\mu(s, X_s) := f(s, X_s, \mu(s, X_s; Z_s))$ to learn V^μ, but an "off-policy" method would sample from a different FSDE to learn V^μ. In this section, we present an off-policy stochastic control method for representing V^μ. The on-policy version of the theory then arises naturally from a particular specialization.

The off-policy method utilizes the connection between the solution of a pair of FBSDEs and the on-policy value function V^μ solving (HJ). We first introduce a class of off-policy FBSDEs, and then provide a theorem establishing their connection to V^μ. We call this class of FBSDEs *off-policy* because the drift term of the FSDE is a random process that can be chosen at will.

As before, let $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\in[0,T]}, \mathsf{P})$ be a complete, filtered probability space, and let W_s^{P} be a Brownian process in the measure P. Denote $f^\mu_s := f^\mu(s, X_s)$, and similarly for ℓ^μ_s, σ_s, and let $\mathbf{E}_\mathsf{P}$ refer to the expectation taken in the probability measure P. Further, let K_s be an arbitrary $\mathcal{F}_s$-progressively measurable process such that $\sigma_s^{-1}K_s$ satisfies Novikov's criterion ($\mathbf{E}_\mathsf{P}\left[\exp\left(\frac{1}{2}\int_0^T \|\sigma_s^{-1}K_s\|^2\,\mathrm{d}s\right)\right] < \infty$) [27, Lemma 9]. We call the pair of FBSDEs

$$\mathrm{d}X_s = K_s\,\mathrm{d}s + \sigma_s\,\mathrm{d}W^\mathsf{P}_s, \qquad X_t = x \tag{7}$$

$$\mathrm{d}Y_s = -(\ell^\mu_s + Z_s^\top D_s)\,\mathrm{d}s + Z_s^\top \mathrm{d}W^\mathsf{P}_s, \quad Y_T = g(X_T) \tag{8}$$

where

$$D_s := \sigma_s^{-1}(f^\mu_s - K_s) \tag{9}$$

the off-policy *drifted FBSDEs* for the target policy μ and drift process K_s. A solution to (7)–(8) is the triple of $\mathcal{F}_s$-adapted processes (X_s, Y_s, Z_s) for which X_s satisfies the FSDE (7) and Y_s, Z_s satisfy the BSDE (8).

Theorem 3.1: Let Assumptions 1–2 hold, and let V^μ be the unique viscosity solution of (HJ). Assume K_s is chosen such that (7) admits a unique square-integrable solution X_s (i.e., it satisfies the properties of [1, Ch. 1, Th. 6.16]), $\sigma_s^{-1}K_s$ satisfies Novikov's criterion, and D_s is bounded. Then, there exists a solution (Y_s, Z_s) to the BSDE (8), and it holds that

$$Y_s = V^\mu(s, X_s), \qquad s \in [t, T] \tag{10}$$

$$Z_s = \sigma_s^\top G(s, X_s), \qquad \text{a.e. } s \in [t, T] \tag{11}$$

for some function $G : [0, T] \times \mathbb{R}^n \to \mathbb{R}^n$, P-a.s., and, in particular

$$Y_t = \mathbf{E}_\mathsf{P}[\widehat{Y}_{t,\tau}] = V^\mu(t, x), \qquad \mathsf{P}\text{-a.s.} \tag{12}$$

where

$$\widehat{Y}_{t,\tau} := Y_\tau + \int_t^\tau (\ell^\mu_s + Z_s^\top D_s)\,\mathrm{d}s. \tag{13}$$

Whenever $\partial_x V^\mu$ exists, then $G(x, X_s) = \partial_x V^\mu(s, X_s)$, and hence

$$Z_s = \sigma_s^\top \partial_x V^\mu(s, X_s), \qquad \text{a.e. } s \in [t, T]. \tag{14}$$

□

Proof: Since (7) has a square-integrable solution, by Assumption 1 on g, the BSDE terminal condition $Y_T = g(X_T)$ is square-integrable. By [1, Ch. 7, Th. 3.2] and because $\ell^\mu_s + z^\top D_s$ is Lipschitz in y, z, then there exists a solution (Y_s, Z_s) to (8) which is unique and square-integrable. Girsanov's theorem (see, e.g. [27, Th. 10]) indicates that if we construct the measure Q from P via the Radon-Nikodym derivative

$$\mathrm{d}\mathsf{Q} = \Theta^{\mathsf{Q}|\mathsf{P}}_{t,T}\mathrm{d}\mathsf{P} \tag{15}$$

where

$$\Theta^{\mathsf{Q}|\mathsf{P}}_{t,t'} := \exp\left(-\frac{1}{2}\int_t^{t'} \|\sigma_s^{-1}K_s\|^2\,\mathrm{d}s - \int_t^{t'} (\sigma_s^{-1}K_s)^\top \mathrm{d}W^\mathsf{P}_s\right) \tag{16}$$

then the process

$$W^\mathsf{Q}_{t'} := W^\mathsf{P}_{t'} + \int_t^{t'} \sigma_s^{-1}K_s\,\mathrm{d}s \tag{17}$$

is Brownian in the newly constructed measure Q. A further consequence of Girsanov's theorem (see e.g., [28, Ch. 5, Th. 10.1]) is illustrated, through an abuse of notation, by substituting the relationship $\mathrm{d}W^\mathsf{P}_s = \mathrm{d}W^\mathsf{Q}_s - \sigma_s^{-1}K_s\,\mathrm{d}s$ into (7) and (8). Performing this substitution yields that the solution (X_s, Y_s, Z_s) to (7) and (8) with the P-Brownian process W^P_s also solves the zero-drift FBSDE

$$\mathrm{d}X_s = \sigma_s\,\mathrm{d}W^\mathsf{Q}_s, \qquad X_t = x \tag{18}$$

$$\mathrm{d}Y_s = -(\ell^\mu_s + Z_s^\top \sigma_s^{-1} f^\mu_s)\,\mathrm{d}s + Z_s^\top\,\mathrm{d}W^\mathsf{Q}_s, \quad Y_T = g(X_T) \tag{19}$$

with Q-Brownian W^Q_s.

Under Assumptions 1–2, it follows from [1, Ch. 7, Th. 4.5] that there exists a unique solution $(\widetilde{X}_s, \widetilde{Y}_s, \widetilde{Z}_s)$ to the FBSDE

$$\mathrm{d}\widetilde{X}_s = \sigma_s\,\mathrm{d}W^\mathsf{Q}_s, \qquad \widetilde{X}_t = x \tag{20}$$

$$\mathrm{d}\widetilde{Y}_s = -h^\mu(s, \widetilde{X}_s, \widetilde{Z}_s)\,\mathrm{d}s + \widetilde{Z}_s^\top\,\mathrm{d}W^\mathsf{Q}_s, \quad \widetilde{Y}_T = g(\widetilde{X}_T). \tag{21}$$

and $\widetilde{Y}_s = V^\mu(s, \widetilde{X}_s)$ holds Q-a.s., and, in particular, $\widetilde{Y}_t = V^\mu(t, x)$. That $\widetilde{Z}_s = \sigma_s^\top G(s, \widetilde{X}_s)$ holds Q-a.s. for the viscosity solution V^μ follows from [4, Th. 4.1]. Similarly, that $\widetilde{Z}_s = \sigma_s^\top \partial_x V^\mu(s, \widetilde{X}_s)$ holds for a classical solution follows from [4, Prop. 4.3]. It is easy to see that

$$h^\mu(s, \widetilde{X}_s, \widetilde{Z}_s) = \ell^\mu_s + \widetilde{Z}_s^\top \sigma_s^{-1} f^\mu_s \tag{22}$$

and from the uniqueness of solutions [1, Ch. 1, Definition 6.6; Ch. 7, Definition 2.1], it follows that (X_s, Y_s, Z_s) also solves (20)–(21), and thus $Y_s = V^\mu(s, X_s)$, $Z_s = \sigma_s^\top \partial_x V^\mu(s, X_s) \equiv \sigma_s^\top G(x, X_s)$ hold Q-a.s.

A further consequence of Girsanov's theorem is that P and Q are equivalent measures [27], that is, $\mathsf{P}(N) = 0$ iff $\mathsf{Q}(N) = 0$ for $N \in \mathcal{F}$. Thus the previous statements said to hold Q-a.s. also hold P-a.s.

To show (12), note that from (13) and the definition of the Itô integral [1, p. 33, eq. (5.23)], we have

$$\widehat{Y}_{t,\tau} = Y_t - \int_t^\tau Z_s^\top \mathrm{d}W^\mathsf{Q}_s.$$

Taking the conditional expectation of both sides, and by noting that by a basic property of the Itô integral [1, p. 34, eq. (5.26)]

$$\mathbf{E}_\mathsf{P}\left[\int_t^\tau Z_s^\top \mathrm{d}W^\mathsf{P}_s\right] = 0$$

we have $\mathbf{E}_\mathsf{P}[\widehat{Y}_{t,\tau}] = \mathbf{E}_\mathsf{P}[Y_t] = \mathbf{E}_\mathsf{P}[V^\mu(t, x)] = V^\mu(t, x)$. ■

Note that while $Z = \sigma^\top \partial_x V^\mu$ cannot be guaranteed for all viscosity solutions of (HJ) (see the discussion in [4, pp. 46–47]), it does hold for smooth convergent approximations of V^μ (see, for example, the proof of Theorem 4.5 in [1, Ch. 7]). Thus, for the numerical approximations generated in this work, we will assume that this equality holds.

Example 1: Consider the optimal control of the scalar linear system

$$\mathrm{d}X_s = (X_s + u_s)\,\mathrm{d}s + \frac{1}{5}\,\mathrm{d}W_s$$

whose value function is defined as the solution of

$$V^*(t,x) = \inf_{u[t,T]} \mathbf{E}\left[\frac{1}{2}(X_T)^2 + \int_t^T \frac{1}{2}u_s^2\,\mathrm{d}s\right].$$

It can be verified that the value function is analytically expressed as

$$V^*(t,x) = \alpha_t x^2 + \beta_t$$

where

$$\alpha(t) := (e^{-2(1-t)}+1)^{-1}$$

$$\beta(t) := \frac{1}{50}\left(\log\left(\frac{1}{2}+\frac{1}{2}e^{2(1-t)}\right)+2(1-t)\right)$$

with corresponding optimal policy $\pi^*(t,x) = -2\alpha_t x$. For this example, let the target policy be the optimal policy $\mu = \pi^*$, which can be computed analytically. For any adapted process K_s, the associated drifted FBSDE is

$$\mathrm{d}X_s = K_s\,\mathrm{d}s + \frac{1}{5}\,\mathrm{d}W_s^{\mathsf{P}}$$

$$\mathrm{d}Y_s = -\left(\frac{1}{2}\mu_s^2 + 5(X_s+\mu_s) - 5Z_sK_s\right)\mathrm{d}s + Z_s\,\mathrm{d}W_s^{\mathsf{P}}$$

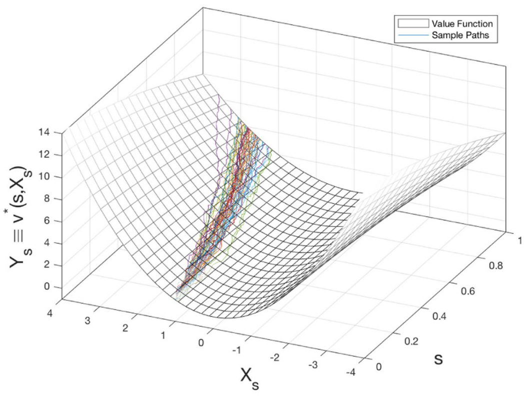
with boundary conditions

$$X_0 = x_0, \quad Y_T = \frac{1}{2}(X_T)^2$$

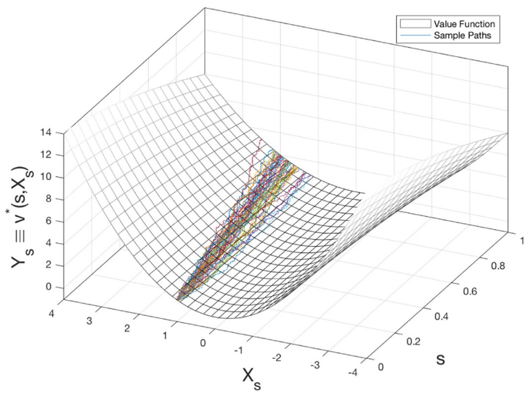
where we choose $x_0 = 1$. We also have $Z_t = (2/5)\alpha_t X_t$. The correspondence between the value function and the solutions to the above FBSDE for three different choices of K_s, are illustrated in Fig. 1 . The drift term in (c) generates a distribution for X_s, which matches the system guided by the optimal control policy.

Regardless of the drift value chosen, the process $[X_s \quad Y_s]^\top$ lies on the surface characterized by the function V^*. Since, in general, function approximation has higher accuracy when interpolating in a region of dense samples compared to extrapolating in a region with no samples, the case in Fig. 1(c) is more desirable from an optimal control perspective, compared to Fig. 1(a) and (b), since the samples have a correspondence with the optimal trajectories.

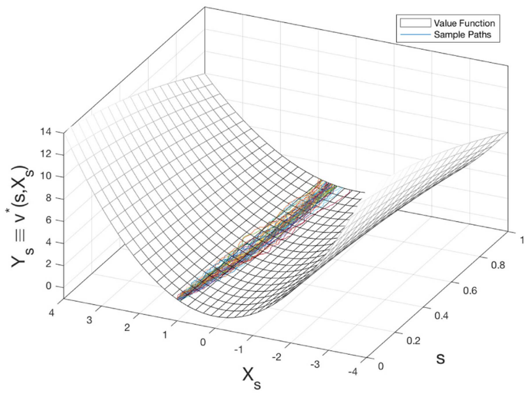
However, as illustrated in Fig. 2, there are other selections for the sampling policy outperforming the optimal control policy, yielding better function approximation. In this example, the function approximation at $t = 0.5$, is illustrated on the right-hand side of Fig. 2. In particular, Fig. 2(b) illustrates that such a randomized optimal policy explores a larger region compared to a pure implementation of the optimal drift in Fig. 2(a), thus resulting in a more accurate approximation of the value function. In other words, a broader exploration of the state space contributes to better function approximations in the presence of numerical error that builds from recursive function approximation during the backward pass. □



(a)



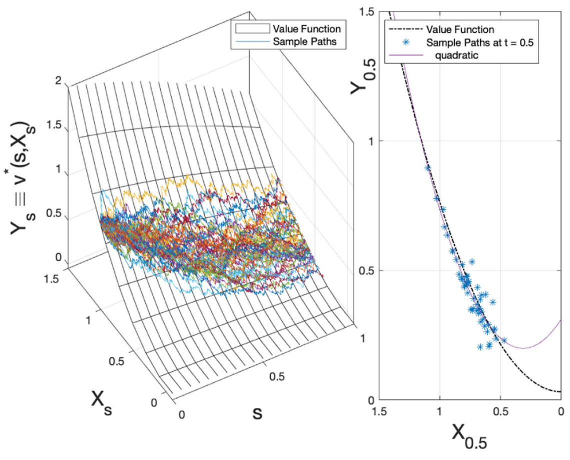
(b)



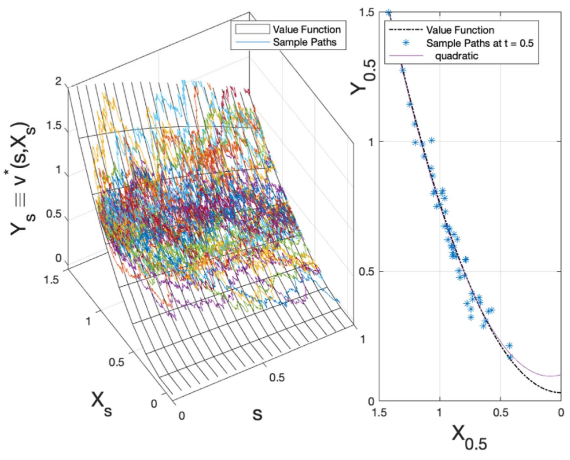
(c)

Fig. 1. Surface of the optimal value function for Example 1. Fifty sample paths of drifted forward–backward SDEs (7) and (8) are compared with different drifts. The initial condition is chosen as $x_0 = 1$. (a) $K_s = X_s$ ($= f_s^\pi$ with $\pi \equiv 0$). (b) $K_s = \frac{3}{5}X_s$ ($= f_s^\pi$ with $\pi = -\frac{2}{5}X_s$). (c) $K_s = f_s^{\pi^*}$ (Optimal policy guided dynamics).

Example 1 and Fig. 1 illustrate the link between the drifted FBSDE and the value function. We can interpret this result in the following sense. We can pick an arbitrary process K_s to be the drift term, which generates a distribution for the forward process X_s in the corresponding measure P. The BSDE yields an expression for Y_t using the same process W_s^{P} used in the FSDE. The term $Z_s^\top D_s$ acts as a correction in the BSDE to compensate for changing the drift of the FSDE. We can then use the relationship (12) to solve for the value function V^μ, whose conditional expectation can be evaluated in P.



(a)



(b)

Fig. 2. Implementation of the drift corresponding to (a) optimal input, and (b) optimal input with 10% randomization, for the same system as in Example 1. (a) $K_s = f_s^{\pi^*}$, the same case as in Fig. 1(c). (b) $K_s = \begin{cases} f(X_s,\pi^*(X_s)), & \text{w.p. } 0.8, \\ f(X_s,-30), & \text{w.p. } 0.1, \\ f(X_s,30), & \text{w.p. } 0.1. \end{cases}$.

It should be noted that K_s need not be a deterministic function of the random variable X_s, as is the case with f_s^{μ}. For instance, it can be selected as $K_s(\omega) = \kappa(s, X_s(\omega), \omega)$ for some appropriate function κ, producing a nontrivial joint distribution for the random variables (X_t, K_t).

A remarkable feature of the off-policy FBSDE formulation is that the forward pass is decoupled from the backward pass, that is, the evolution of the forward SDE does not explicitly depend on Y_s or Z_s (whereas in the stochastic maximum principle formulations (see, e.g., [1, Ch. 3]), the decoupling is irremovable). This feature forms the basis of FBSDE numerical investigations of SOC [5], [29]. The significant difference of Theorem 3.1 in comparison to those results is that the focus here is shifted from the solution of (HJB) toward the broader class of functions satisfying the (HJ). This provides a stronger case for policy iteration methodologies, because the theory does not require, or expect, μ to be an optimal policy, as is in [5] and [29]. Although not evaluated in this work, μ can be chosen according to design specifications other than estimating the optimal policy, such as, for instance, to ensure that the current policy is close to the previously estimated policy.

B. Local Entropy Weighing

As discussed in Section III-A, the disentanglement of the forward sampling from the backward function approximation provides the opportunity to employ broad sampling schemes to cover the state space with potential paths. However, fitting a value function broadly to a wide support distribution might degrade the quality of the function approximation since high accuracy of function approximation is more crucial in those parts of the state space that are in proximity to the optimal trajectories. Once forward sampling has been performed and some parts of the value function have been approximated, we can apply a heuristic in which sample paths closer to optimal trajectories are weighted more so as to concentrate value function approximation accuracy in those regions.

To this end, we propose using a bounded random variable ρ_t to produce a new measure R_t, the weighted counterpart to P_t, defined as

$$\mathrm{d}\mathsf{Q}_t = \Theta_{0,t}^{\mathsf{Q}|\mathsf{P}} \mathrm{d}\mathsf{P}_t. \tag{23}$$

In order to avoid underdetermination of the regression by concentrating to a single or a few samples, we select R_t as

$$\mathsf{R}_t \in \arg\min_{\mathsf{R}_t} \{\mathbf{E}_{\mathsf{R}_t}[\rho_t] + \lambda \mathcal{H}(\mathsf{R}_t \| \mathsf{P}_t)\} \tag{24}$$

with $\lambda > 0$, a tuning variable, and

$$\mathcal{H}(\mathsf{R}_t\|\mathsf{P}_t) = \mathbf{E}_{\mathsf{R}_t}\left[\log\left(\frac{\mathrm{d}\mathsf{R}_t}{\mathrm{d}\mathsf{P}_t}\right)\right] \tag{25}$$

is the relative entropy of R_t that takes its minimum value when $\mathsf{R}_t = \mathsf{P}_t$, the distribution in which all sampled paths have equal weight.

The minimizer of (24), which balances between the value of ρ_t and the relative entropy of its induced measure, has a solution R_t^*, given by [30, p. 2]

$$\mathrm{d}\mathsf{R}_t^* = \Theta_t^{\mathsf{R}|\mathsf{P}} \mathrm{d}\mathsf{P}_t, \qquad \Theta_t^{\mathsf{R}|\mathsf{P}} := \frac{\exp\left(-\frac{1}{\lambda}\rho_t\right)}{\mathbf{E}_{\mathsf{P}_t}\left[\exp\left(-\frac{1}{\lambda}\rho_t\right)\right]}. \tag{26}$$

Henceforth, for simplicity, we let R_t refer to this minimizer R_t^*. During numerical approximation we can interpret the weights as a *softmin* operation over paths according to this heuristic, a method often used in the deep learning literature [31].

Theorem 3.2: Assume ρ_τ is selected such that W_s^{P} is Brownian on the interval $[t, \tau]$ with respect to the induced measure R_τ. It then holds that

$$Y_t = \mathbf{E}_{\mathsf{R}_\tau}[\widehat{Y}_{t,\tau}|X_t] = V^{\mu}(t, X_t), \qquad \mathsf{R}_\tau\text{-a.s.} \tag{27}$$

where $\widehat{Y}_{t,\tau}$ is defined in (13). Furthermore, the minimizer ϕ^* of the optimization problem

$$\begin{aligned} &\inf_{\phi\in L^2} \mathbf{E}_{\mathsf{R}_\tau}[(\widehat{Y}_{t,\tau} - \phi(X_t))^2] \\ &= \inf_{\phi\in L^2} \mathbf{E}_{\mathsf{P}_\tau}[\Theta_\tau^{\mathsf{R}|\mathsf{P}}(\widehat{Y}_{t,\tau} - \phi(X_t))^2] \end{aligned} \tag{28}$$

over X_t-measurable square integrable variables $\phi(X_t)$ coincides with the value function $\phi^*(X_t) = V^\mu(t, X_t)$. □

Proof: First, note that X_s, Y_s, Z_s, W_s^{P}, and $\widehat{Y}_{t,\tau}$ are $\mathcal{F}_\tau$-measurable for $s, t \in [0, \tau]$. Thus, (10) and

$$\widehat{Y}_{t,\tau} = Y_t - \int_t^\tau Z_s^\top \mathrm{d}W_s^{\mathsf{P}} \tag{29}$$

hold P_τ-a.s. We now show that they hold R_τ-a.s. as well. Since $\mathbf{E}_{\mathsf{P}_\tau}[\Theta_\tau^{\mathsf{R}|\mathsf{P}}] = 1$ and P_τ is a probability measure, then R_τ is also a probability measure. Furthermore, since ρ_τ is bounded, $\Theta_\tau > 0$ P_τ-a.s. It follows that R_τ and P_τ are equivalent measures because they reciprocally have strictly positive densities [32, Ch. 10, Remark 10.4]. The proof of Theorem 3.1 shows that since these measures are equivalent, (10) and (29) hold P_τ-a.s. if and only if they also hold R_τ-a.s. Since W_s^{P} is Brownian in R_τ over the integral, the second term in the right hand side of (29) will drop out when taking the conditional expectation $\mathbf{E}_{\mathsf{R}_\tau}[\widehat{Y}_{t,\tau} | X_t]$, yielding (27).

Equation (28) is a result of the L^2-projective properties of conditional expectation [33, Ch. 10.3, Property 11] applied to (27), followed by a change of measure applied to the expected value integration with (26). ■

In the following section, we approximate the minimization of the right-hand side of (28) over parameterized value function models to obtain an estimate of the value function.

To summarize, in this section, we introduced three measures: 1) Q, the measure associated with the target policy μ for the value function V^μ; 2) P, the sampling measure used in the forward pass to explore the state space; and 3) R_τ, the local-entropy weighted measure used in the backward pass to control function approximation accuracy.

IV. Branching Path LSMC

In the previous section, we provided the theoretical results justifying our approach for the case of continuous-time stochastic processes. In this section, we discuss how each of these measures is represented numerically. In particular, we propose a novel discrete-time, finite-dimensional numerical scheme to produce the FSDE distribution, along with a procedure to solve for the value function in a backward pass using the BSDE. The FSDE distribution is represented as a branching-path tree and the BSDE is used to produce estimators, stepping backwards along each of the branching paths, to estimate the value function parameters using LSMC regression. In Section V, we propose particular choices for the drift process and the heuristic weight function used in the proposed FBRRT-SDE numerical method.

Henceforth, we assume a discrete-time partition of the interval $[0, T] (t_0 = 0, \ldots, t_i = (\Delta t)i, \ldots, t_N = T)$ for some partition length Δt. For brevity, we abbreviate X_{t_i} as X_i and similarly for most other variables.

Fig. 3 motivates our approach, illustrating how the method is able to rapidly find the optimal distribution. An on-policy method assumes knowledge of an initial suboptimal control policy, sampled as in Fig. 3(b) using the approach in [6] and the suboptimal value function is solved in that distribution. The on-policy method requires iterative improvement of the policy to

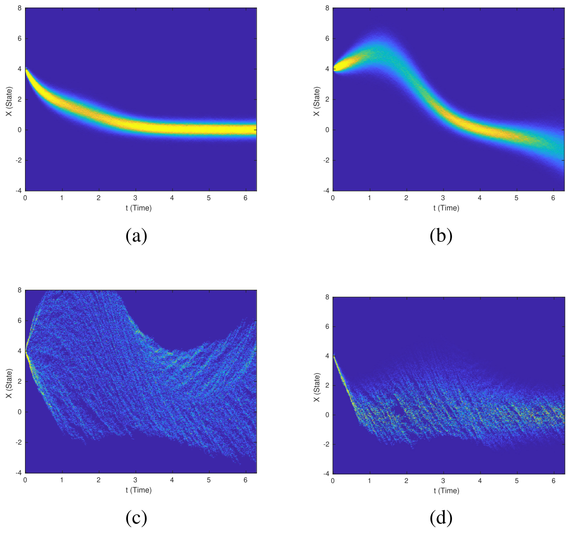


Fig. 3. Heatmap of different measure distributions for a 1-D SOC problem, illustrating how RRT-sampling and local-entropy weighing can accelerate discovery of the optimal distribution. (a) Optimal distribution. (b) Parallel-sampled suboptimal. (c) RRT-sampled (P). (d) RRT-sampled, weighted (R).

produce a distribution that overlaps with the optimal distribution. However, if we begin with a sampling measure that broadly explores the state space as in Fig. 3(c), we can produce an informed heuristic that weighs this distribution as in Fig. 3(d), so that the function approximation is concentrated in a near-optimal distribution. Thus, often, we need only one iteration to get a good approximation of the optimal value function and policy.

In Section IV-A, we summarize the construction of a stochastically sampled tree as a generalized data structure to approximate the FSDE distribution over the partition $t_0, t_1, \ldots, t_N$. Then, in Section IV-B, we demonstrate how this data structure can be interpreted as a series of McKean-Markov path measures $\{\overrightarrow{\mathsf{P}}_i\}_{i=0}^N$ to approximate the forward sampling distributions. Finally, in Section IV-C, we discuss how these measures can be used in the backward pass to approximate the BSDE solution by estimating the optimal value function.

A. Forward SDE Branched Sampling

We begin by discussing the construction of a tree data structure $\mathcal{G}$ representing the FSDE (7). In this section, we only describe how edges of $\mathcal{G}$ are added and what data are stored. Later, in Section V-A, we propose a specific methodology for selecting nodes for expansion and choosing the drift value. The tree is initialized with a root node at the initial state x_0 and is constructed asynchronously as long as new nodes and directed edges are added using the following procedure.

Let $x_i^{\text{parent}} \in \mathbb{R}^n$ be a state node in the tree at time i selected for expansion, as the parent of a new edge. The drift $k_i \sim \kappa(x_i^{\text{parent}}, \{x_i^k\}_k)$ (representing the random variable K_i) is sampled from some random function κ, which can depend on both the state and the distribution of the nodes at that time. Independently, the noise is sampled $w_i \sim \mathcal{N}(0, \Delta t I_n)$. The child state node is computed using an Euler-Maruyama SDE

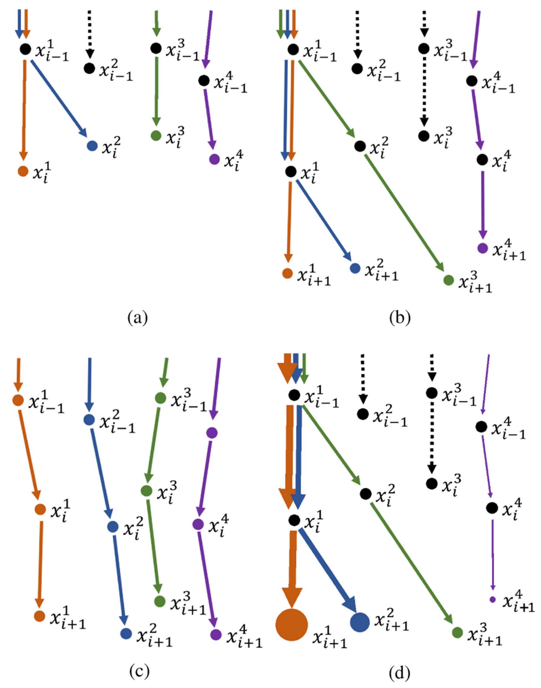


Fig. 4. (a) and (b) Illustration of how the branch sampled measures are represented based on the underlying data structure. The colored paths show the collection of paths representing the respective measure. Dotted lines represent edges in the data structure which are not included in the path measure for that time step. (c) and (d) Comparison of the unweighted parallel-sampling method from previous approaches to the proposed weighted and branch-sampled method. (a) Branch-sampled ($\overrightarrow{\mathsf{P}}_i$). (b) Branch-sampled ($\overrightarrow{\mathsf{P}}_{i+1}$). (c) Parallel-sampled. (d) Weighted-branched ($\overleftarrow{\mathsf{R}}_{i+1}$).

step approximation of the FSDE (7)

$$x_{i+1}^{\text{child}} = x_i^{\text{parent}} + k_i \Delta t + \sigma(t_i, x_i^{\text{parent}}) w_i. \tag{30}$$

The edge $(x_i^{\text{parent}}, d_i^{\text{data}}, x_{i+1}^{\text{child}})$ is added to the tree, where $d_i^{\text{data}} = (k_i, w_i, \ldots)$ is the data attached to the edge. A new parent can then be selected for expansion, including selecting the same parent again. Fig. 4(a) and (b) illustrates the branching tree data structure.

B. McKean-Markov Measure Representation

We approximate the continuous-time sampling distributions with discrete-time McKean-Markov branch sampled paths, as in [34]. The tree data structure $\mathcal{G}$ represents a series of path measures $\{\overrightarrow{\mathsf{P}}_i\}_{i=0}^N$, each approximating the distribution $\overrightarrow{\mathsf{P}}_i \approx \mathsf{P}_{t_i} \circ \xi_i^{-1}$, where ξ_i is the discrete-time random path defined as $\xi_i := (X_0, \mathcal{D}_0, X_1, \ldots, \mathcal{D}_{i-1}, X_i)$, and ξ_i^{-1} is the inverse map from events on the path space to events on the sample space Ω [33, Ch. 3]. Here, we use $\mathcal{D}_i$ to refer to the set of random variables associated with the edges of the tree, including K_i and W_i^{P}. The empirical measure approximations are defined as

$$\overrightarrow{\mathsf{P}}_i := \frac{1}{M} \sum_{j=1}^{M} \delta_{\xi_i^j} \tag{31}$$

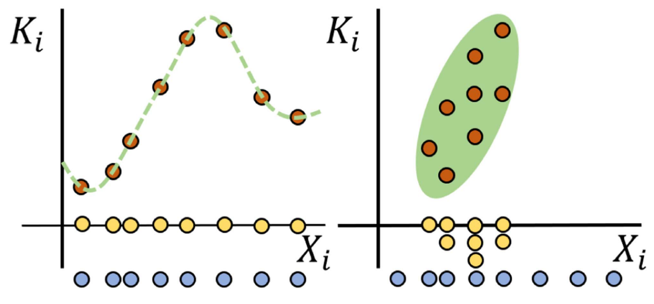


Fig. 5. Comparison of the approximations of the joint distribution $\mathsf{P}_{(X_i,K_i)}$. Left: A parallel-sampling method is used to approximate a joint distribution wherein K_i is a deterministic function of X_i. Right: A branch-sampling method is used to approximate a nontrivial joint distribution. The blue dots are the approximation of the distribution P_{X_i}, the green curves are the ideal continuous distribution $\mathsf{P}_{(X_i,K_i)}$, the red dots are its sampled approximation and the yellow dots represent how many children that node in the tree will have.

where δ is the Dirac-delta measure acting on sample paths

$$\xi_i^j := (x_{0,i}^j, d_{0,i}^j, x_{1,i}^j, d_{1,i}^j, \ldots, d_{i-1,i}^j, x_{i,i}^j). \tag{32}$$

The notation $x_{m,i}^j$ indicates that this element is the sample of a random variable X_m that is the ancestor of sample $x_{i,i}^j$ in the path ξ_i^j, and similarly for the edge variables $\mathcal{D}_m$. Each node in the tree x_i^j (alternatively called a particle) is associated with a unique path ξ_i^j whose final term is $x_{i,i}^j = x_i^j$. Fig. 4(a)–(b) illustrates how each colored node at a particular time step is associated with its matching colored path, and that all of these paths collectively constitute the path measure.

It is worth noting that in this construction there is no requirement for $\overrightarrow{\mathsf{P}}_i$ and $\overrightarrow{\mathsf{P}}_{i+1}$ to agree over the interval $[0, t_i]$. This property is illustrated by the fact that, for example, the path ending at x_i^3 in Fig. 4(a) is represented in $\overrightarrow{\mathsf{P}}_i$ but not represented in $\overrightarrow{\mathsf{P}}_{i+1}$ in Fig. 4(b). To see why such constructions are permissible in the proposed numerical scheme, notice that in the backward step (e.g., in Theorems 3.1 and 3.2) the measure P_i is only employed to compute the instantaneous conditional expectation; thus, there is no sample path matching required when taking $\tau = t_{i+1}$ and $t = t_i$ to obtain $V^\mu(t_i, x)$, and when taking $\tau = t_i$ and $t = t_{i-1}$ to obtain $V^\mu(t_{i-1}, x)$.

It is observed in Fig. 4(b) that some edges are multiply represented in the distribution. If the drift term K_i were restricted to be a deterministic function of X_i (as is the case in [5], [6], and [8]), such a construction would represent an unfaithful characterization of the path distribution because samples of the Brownian process are independent, and thus should be sampled as in Fig. 4(c). However, since K_i itself is permitted to have a distribution, the overlapping of paths is justified as the drift having been selected so as to concentrate the paths in a certain part of the state space. Fig. 5 illustrates why parallel sampling is naturally suited for representing deterministic functions and why branch sampling is necessary for representing nontrivial joint distributions (X_i, K_i).

For any arbitrary function G_i evaluated on path ξ_i, we have the almost sure convergence

$$\mathbf{E}_{\overrightarrow{\mathsf{P}}_i}[G_i(\xi_i)] = \sum_{j=1}^{M} \frac{1}{M} G_i(\xi_i^j) \xrightarrow{\text{a.s.}} \mathbf{E}_{\widetilde{\mathsf{P}}_i}[G_i(\xi_i)] \tag{33}$$

as the number of particles $M \to \infty$, where $\widetilde{\mathsf{P}}_i$ is the ideal discrete-time approximation of the distribution $\mathsf{P}_i \circ \xi_i^{-1}$ under the Euler-Maruyama scheme [34, Sec. 4.1.2]. It follows from the change of variables theorem [35, Ch. 3, Th. 3.6.1] that this expectation is exact up to the error due to time discretization

$$\mathbf{E}_{\widetilde{\mathsf{P}}_i}[G_i(\xi_i)] \approx \int G_i(\xi_i)\, \mathsf{P}_i \circ \xi_i^{-1}(\mathrm{d}\xi_i)$$
$$= \int G_i(\xi_i(\omega))\, \mathsf{P}_i(\mathrm{d}\omega) =: \mathbf{E}_{\mathsf{P}_i}[G_i(\xi_i)].$$

According to [36, Ch. 10, Th. 10.2.2], when a linear growth condition in x is imposed on f_s^μ, σ_s, and ℓ_s^μ along with a few other conditions, the Euler-Maruyama scheme's error varies as $\mathcal{O}((\Delta t)^{\frac{1}{2}})$. When σ_s is constant with respect to x, the error bound improves to $\mathcal{O}(\Delta t)$ [36, Ch. 10, Th. 10.3.5]. Thus, our approximation converges with large numbers of particles and a decreasing time interval.

C. Path Integral Least Squares Monte Carlo

To approximate the measure R_{i+1} in Theorem 3.2 we use a path integral-weighted measure

$$\overleftarrow{\mathsf{R}}_{i+1} := \frac{1}{\eta} \sum_{j=1}^{M} \Theta_{i+1}^j \delta_{\xi_{i+1}^j} \tag{34}$$

where the weights for each heuristic value ρ_{i+1}^j are

$$\Theta_{i+1}^j = \exp\left(-\frac{1}{\lambda}\rho_{i+1}^j\right) \tag{35}$$

and η is a normalizing constant. The heuristic value is calculated as $\rho_{i+1}^j = \rho_{i+1}(\xi_{i+1}^j)$, taking care to exclude $w_{i,i+1}^j$ so that its distribution remains Brownian.

In each step of the backward pass, we use $\overleftarrow{\mathsf{R}}_{i+1}$ and the value function approximation $V(x;\alpha_{i+1}) \approx V^\mu(t_{i+1},x)$, parameterized by $\alpha_{i+1} \in \mathcal{A}$, where $\mathcal{A}$ is the parameter space, to estimate the value function at the previous time step $V(x;\alpha_i) \approx V^\mu(t_i,x)$, by producing $\alpha_i \in \mathcal{A}$. We assume that the parameterization $V(x;\alpha)$ results in a function that is C^2 for all $\alpha \in \mathcal{A}$, and approximate the optimization in (28) as

$$\alpha_i^* = \arg\min_{\alpha_i \in \mathcal{A}} \mathbf{E}_{\overleftarrow{\mathsf{R}}_{i+1}}[(\widehat{Y}_{i,i+1} - V(X_i;\alpha_i))^2]$$
$$= \arg\min_{\alpha_i \in \mathcal{A}} \mathbf{E}_{\overrightarrow{\mathsf{P}}_{i+1}}[\Theta_{i+1}^j(\widehat{Y}_{i,i+1} - V(X_i;\alpha_i))^2]$$
$$= \arg\min_{\alpha_i \in \mathcal{A}} \frac{1}{\eta} \sum_{k=1}^{M} \Theta_{i+1}^j(\widehat{y}_{i,i+1}^j - V(x_i^j;\alpha_i))^2 \tag{36}$$

where (13) is approximated as

$$\widehat{y}_{i,i+1}^j \approx V(x_{i+1}^j;\alpha_{i+1}) + (\ell_i^{\mu j} + z_{i+1}^{j\top} d_i^j)\Delta t. \tag{37}$$

Algorithm 1: Forward–Backward RRT-SDE.

```
1:  procedure FBRRT-SDE($x_0$)
2:    $\widetilde{\mathcal{G}}$.init($\xi_0$)
3:    for $k = 1, \ldots, N_{\text{iter}}$ do
4:      $\mathcal{G} \leftarrow$ ForwardPass ($\widetilde{\mathcal{G}}, (\alpha_i)_i$)
                ▷ Generate tree which represents $\{\overrightarrow{\mathsf{P}}_i\}_i$
5:      $(\alpha_i)_i \leftarrow$ BackwardPass ($\mathcal{G}$)
                ▷ Approximate value functions $\{V(\cdot;\alpha_i)\}_i$
6:      $\overline{J}_k \leftarrow$ PolicyCost ($x_0, (\alpha_i)_i$)
                ▷ Evaluate computed policy $\{\mu_i(\cdot;\alpha_{i+1})\}_i$
7:      $\widetilde{\mathcal{G}} \leftarrow$ ERODE ($\mathcal{G}, (\alpha_i)_i$)
                ▷ Prune tree to remove suboptimal paths
8:    end for
9:    return $(\alpha_i)_i$
10: end procedure
```

The novelty of this method over classic LSMC [11], developed for parallel-sampled paths comes from: 1) the observation that introducing the drift process K_i with a nontrivial joint distribution $\mathsf{P}_{(X_i,K_i)}$ validates the choice of branch-sampled path distributions; 2) we can weigh regression points using a heuristic that acts on the entire path, not just the immediate states; and 3) weighing as in (35) has a particular interpretation as the selection of a measure with desirable properties for robustness using (24).

V. Forward–Backward RRT-SDE

In this section, we present a novel algorithm (FBRRT-SDE) that uses rapidly exploring random trees to construct the graph $\mathcal{G}$ of samples for solving the corresponding system of FBSDEs. The FBRRT-SDE algorithm is a particular numerical application of the generalized theory presented in Section IV. The ultimate goal of the FBRRT-SDE algorithm is to produce the set of parameters $\{\alpha_i\}_{i=1}^N$ which approximate the optimal value function as $V^*(t_i,x) \approx V(x;\alpha_i)$. This is achieved by generating a forward pass to produce a graph representation $\mathcal{G}$ of the path measures $\{\overrightarrow{\mathsf{P}}_i\}_{i=1}^N$. Given that the optimal policy has the form (5), we define the target policy

$$\mu_i(x;\alpha_{i+1})$$
$$= \arg\min_{u \in U}\{\ell(t_i,x,u) + f(t_i,x,u)^\top \partial_x V(x;\alpha_{i+1})\} \tag{38}$$

so that it coincides with the optimal control policy when the value function approximation is exact. The backward pass uses $\mathcal{G}$, μ_i, and ρ_{i+1} to produce α_i, backwards in time. At each iteration k of the algorithm, the policy cost $\overline{J}_k := \mathbf{E}[S_0[x_0,\mu]]$ associated with a set of parameterized policies is evaluated by sampling a parallel-sampled set of trajectories and computing the mean cost. At the end of each iteration, nodes with a high heuristic value ρ_{i+1} are pruned from the tree $\mathcal{G}$, and new nodes are added in the forward pass in the next iteration. This outer loop of the FBRRT-SDE algorithm is summarized in Algorithm 1.

A. Kinodynamic RRT Forward Sampling

We desire sampling methods that seek to explore the whole state space, thus increasing the likelihood of sampling in the proximity of optimal trajectories. For this reason, we chose a method inspired by kinodynamic RRT [21]. The selection procedure for this method ensures that the distribution of the chosen particles is more uniformly distributed in a user-supplied region of interest, is more likely to select particles that explore the empty space, and is, therefore, less likely to oversample dense clusters of particles.

With some probability $\varepsilon_i^{\text{rrt}} \in [0,1]$ we choose the RRT sampling procedure, but otherwise we choose a particle uniformly from $\{x_i^j\}_{j=1}^M$, each particle having an equal weight. This ensures that dense particle clusters will still receive more attention. The choice of the parameter $\varepsilon_i^{\text{rrt}}$ balances exploring the state space against refining the area around the current distribution.

For the drift values, that is, those sampled from the distribution κ left unspecified in Section IV-A, we again choose a random combination of exploration and exploitation. For exploitation we choose

$$K_i = f(t_i, X_i, \mu_i(X_i;\alpha_i)) \tag{39}$$

and for exploration we choose

$$K_i = f(t_i, X_i, u^{\text{rand}}) \tag{40}$$

where the control is sampled randomly from a user-supplied set $u^{\text{rand}} \sim U^{\text{rand}}$. For example, for minimum fuel (L_1 cost) problems where the control is bounded as $u \in [-1,1]$ and the running cost is $L = |u|$, we select $U^{\text{rand}} = \{-1,0,1\}$ because the policy (38) is guaranteed to only return values in this discrete set.

Algorithm 2 summarizes the implementation of the RRT-based sampling procedure that produces the forward sampling tree $\mathcal{G}$. The algorithm takes as input any tree with width $\widetilde{M}$ and adds nodes at each depth until the width is M, the desired tree width. In the first iteration, there are no value function estimated parameters available to exploit, so we set $\varepsilon^{\text{rrt}} = 1$ to maximize exploration using RRT sampling.

Algorithm 2: RRT Branched-Sampling.

```
1:  procedure ForwardPass ($\mathcal{G}, (\alpha_1,\ldots,\alpha_N)$)
2:    for $k = \widetilde{M}+1,\ldots,M$ do          ▷ Add node each loop
3:      for $i = 0,\ldots,N-1$ do                  ▷ For each time step
4:        $\{x_i^j\}_j \leftarrow \mathcal{G}$.nodesAtTime($i$)
5:        if $\varepsilon^{\text{rrt}} > \kappa^{\text{rrt}} \sim$ Uniform($[0,1]$) then
6:          $x_i^{\text{rand}} \sim$ Uniform($\mathcal{X}^{\text{roi}}$)
7:          $(x_i^{\text{near}}, j^{\text{near}}) \leftarrow$ Nearest($\{x_i^j\}_j, x_i^{\text{rand}}$)
8:        else
9:          $(x_i^{\text{near}}, j^{\text{near}}) \sim$ Uniform($\{x_i^j\}_j$)
10:       end if                ▷ $j^{\text{near}}$ is index of selected node
11:       if $\varepsilon^{\text{opt}} > \kappa^{\text{opt}} \sim$ Uniform($[0,1]$) then
12:         $u_i \leftarrow \mu_i(x_i^{\text{near}};\alpha_{i+1})$                ▷ (38)
13:       else
14:         $u_i \sim U^{\text{rand}}$
15:       end if
16:       $k_i \leftarrow f(t_i, x_i^{\text{near}}, u_i)$
17:       $w_i \sim \mathcal{N}(0, \Delta t I_n)$
18:       $x_{i+1}^{\text{next}} \leftarrow x_i^{\text{near}} + k_i\Delta t + \sigma(t_i, x_i^{\text{near}})w_i$
19:       $j^{\text{next}} \leftarrow \mathcal{G}$.addEdge($i, j^{\text{near}}, (x_i^{\text{near}}, k_i, x_{i+1}^{\text{next}})$)
20:       $\overrightarrow{\ell}_{0:i-1} \leftarrow \mathcal{G}$.getRunCost($i-1, j^{\text{near}}$)
21:       $\overrightarrow{\ell}_{0:i} \leftarrow \overrightarrow{\ell}_{0:i-1} + \ell_i(x_i^{\text{near}}, u_i)\Delta t$
22:       $\mathcal{G}$.setRunCost($i, j^{\text{next}}, \overrightarrow{\ell}_{0:i}$)
23:     end for
24:   end for
25:   return $\mathcal{G}$
26: end procedure
```

B. Path-Integral Dynamic Programming Heuristic

Next, we propose a heuristic design choice for the backward pass weighting variables ρ_{i+1}, and justify this choice with theoretical analysis. A good heuristic will give large weights to paths likely to have low values over the whole interval $[0,T]$. Thus, in the middle of the interval, we care both about the current running cost and the expected cost. A dynamic programming principle result following directly from [25, Ch. 4, Corollary 7.2] indicates that

$$V^*(0,x_0) = \min_{u[0,t_{i+1}]} E_{\mathsf{P}^u_{i+1}}\left[\int_0^{t_{i+1}} \ell(s,X_s,u_s)\,\mathrm{d}s + V^*(t_{i+1},X_{i+1})\right]$$

where $u_{[0,t_{i+1}]}$ is any control process in U on the interval $[0,t_{i+1}]$ and P^u_{i+1} is the measure produced by the drift $K_s = f(s,X_s,u_s)$. Following this minimization, we choose the heuristic to be the discrete approximation of

$$\rho_{i+1} = \int_0^{t_{i+1}} \ell(s,X_s,u_s)\,\mathrm{d}s + V^*(t_{i+1},X_{i+1}) \tag{41}$$

where $u_{[0,t_{i+1}]}$ is chosen identically to how the control for the drift is produced.

Although the theory up to this point does not require K_s to be a feasible drift under the dynamic constraints of the SOC problem, for the proposed FBRRT-SDE algorithm, the drift is always chosen as $K_s = f(s,X_s,u)$ for some randomly selected $u \in U$. In practice, the running cost is approximated by Euler-Maruyama, and the optimal value function is approximated by the latest estimate of the value function $V(x;\alpha_{i+1})$. The running cost is computed in the forward sampling in line 21 of Algorithm 2.

Algorithm 3 details the implementation of the backward pass with local entropy weighting. Line 18 does not, theoretically, have an effect on the optimization, since it will come out of the exponential as a constant multiplier, but it has the potential to improve the numerical conditioning of the exponential function computation as discussed in [31, Ch. 5, eq. (6.33)]. The λ value is, in general, a parameter which must be selected by the user. For some problems, we choose to search over a series of possible λ parameters, evaluating each one with a backward pass and using the one that produces the smallest expected cost over a batch of trajectory rollouts executing the computed policy.

Algorithm 3: Local Entropy Weighted LSMC Backward Pass.

```
 1: procedure BackwardPass ($\mathcal{G}$)
 2:   $\{\xi_N^j\}_j \leftarrow \mathcal{G}$.pathsAtTime($N$)
 3:   $\{x_N^j\}_j \leftarrow \{\xi_N^j\}_j$
 4:   $y_N \leftarrow [g(x_N^1)\ \cdots\ g(x_N^M)]^\top$
 5:   $\alpha_N \leftarrow \arg\min_\alpha \sum_j \Theta_N(\widehat{y}_N^j - \Phi(x_N^j)\alpha)^2$
 6:   for $i = N-1, \ldots, 1$ do                              ▷ For each time step
 7:     $\{\xi_{i+1}^j\}_j \leftarrow \mathcal{G}$.pathsAtTime($i+1$)
 8:     for $j = 1, \ldots, M$ do                               ▷ For each path
 9:       $(x_i^j, k_i^j, x_{i+1}^j) \leftarrow \xi_{i+1}^j$                ▷ $x_i^j = x_{i,i+1}^j$, etc.
10:       $y_{i+1}^j \leftarrow \Phi(x_{i+1}^j)\alpha_{i+1}$                         ▷ (27)
11:       $z_{i+1}^j \leftarrow \sigma_{i+1}^\top(x_{i+1}^j)\partial_x\Phi(x_{i+1}^j)\alpha_{i+1}$     ▷ (14)
12:       $\mu_i^j \leftarrow \mu_i(x_i^j; \alpha_{i+1})$                            ▷ (38)
13:       $d_i^j \leftarrow \sigma_{i+1}^{-1}(x_{i+1}^j)(f_i^\mu - k_i^j)$
14:       $\widehat{y}_i^j \leftarrow y_{i+1}^j + (\ell_i^\mu + z_{i+1}^{j\top} d_i^j)\Delta t$          ▷ (13)
15:       $\overrightarrow{\ell}_{0:i} \leftarrow \mathcal{G}$.getRunCost($i, j$)
16:       $\rho_{i+1}^j \leftarrow y_{i+1}^j + \overrightarrow{\ell}_{0:i}$                     ▷ (41)
17:     end for
18:     $\rho_{i+1} \leftarrow \rho_{i+1} - \min_j\{\rho_{i+1}^j\}$        ▷ exp conditioning
19:     $\Theta_{i+1} \leftarrow \exp(-\frac{1}{\lambda}\rho_{i+1})$                    ▷ (26)
20:     $\alpha_i \leftarrow \arg\min_\alpha \sum_j \Theta_{i+1}^j(\widehat{y}_i^j - \Phi(x_i^j)\alpha)^2$   ▷ (28)
21:   end for
22:   return $(\alpha_1, \ldots, \alpha_N)$
23: end procedure
```

C. Path Integral Erode

After the backward pass of the algorithm, we obtain updated approximations of the value function $\{V(\cdot;\alpha_i)\}_i$ along with the tree $\mathcal{G}$ that represents the forward sampling path measures $\{\overrightarrow{\mathsf{P}}_i\}_i$. To improve our approximation, we can use our value function estimates to create a new tree $\mathcal{G}'$ with new forward sampling measures $\{\overrightarrow{\mathsf{P}}'_i\}_i$ via the heuristic ρ_i in (41).

We have found experimentally that sampling a new tree from scratch is both wasteful and shows signs of catastrophic forgetting. That is, the subsequent backward pass performs worse, since it has lost data samples, which were important to form good function estimates. On the other hand, simply adding more samples to the current tree can prove to be unsustainable in the long run. To keep the time complexity constant between iterations, we propose to bound the number of samples at each time step. After each backward pass, we remove as many samples as were added in the forward pass, "eroding" the tree before the forward pass "expands" it.

The algorithm starts with a tree of width M and ends with a tree of width $\widetilde{M}$ at every depth. We begin at the end of the trajectory $i = N$ and remove the nodes $\{x_N^j\}_{j=1}^M$ with highest ρ_N^j value until there are only $\widetilde{M}$ nodes left at depth N. We proceed in a similar fashion backwards down the tree, removing nodes with high ρ_i^j values. However, due to the tree structure of the path measures, if we remove nodes that have children, we disconnect the paths and ruin the assumed structure. Thus, we only remove nodes that have no children. The implementation of this algorithm is detailed in Algorithm 4.

Algorithm 4: Path Integral Erode.

```
 1: procedure Erode ($\mathcal{G}, (\alpha_i)_i$)
 2:   for $i = N, \ldots, 1$ do                    ▷ For each time step
 3:     $\{\rho_i^j\}_j \leftarrow \mathcal{G}$.getHeuristics($t_i$)
 4:     for all $j' \in$ sortDescending($\{\rho_i^j\}_j$) do
 5:       if $\mathcal{G}$.hasNoChildren($x_i^{j'}$) then
 6:         $\mathcal{G}$.removeParentEdge($x_i^{j'}$)
 7:         $\mathcal{G}$.removeNode($x_i^{j'}$)
 8:       end if
 9:       if $\mathcal{G}$.numNodes($t_i$) $= \widetilde{M}$ then
10:         break
11:       end if
12:     end for
13:   end for
14:   return $\mathcal{G}$
15: end procedure
```

D. Function Approximation

In our implementation of the FBRRT-SDE algorithm, the value function is represented by 2nd order multivariate Chebyshev polynomials. Specifically, we use all products of the basis functions $\bigcup_{j=1}^n\{1, x_j, 2x_j^2 - 1\}$ with polynomial degree 2 or lower, namely

$$\Phi(x) := (1, x_1, \ldots, x_n, 2x_1^2 - 1, \ldots, 2x_n^2 - 1$$
$$x_1x_2, \ldots, x_1x_n, x_2x_3, \ldots, x_2, x_n, \ldots, x_{n-1}x_n).$$

For better conditioning, points are first normalized to the interval $[-1, 1]^n$ based on a parameterized region of interest to obtain the basis functions $\Phi(\ldots, (x_j - a_j^{\text{offset}})/a_j^{\text{scale}}, \ldots)$.

E. Computational Complexity

The computational complexity of the forward pass is dominated by the nearest-neighbor search. In our implementation, we use brute force search so the complexity of Line 7 in Algorithm 2 is $\mathcal{O}(M)$, but the complexity can be lowered to $\mathcal{O}(\log(M))$ if a kd-tree is used for the nearest neighbor search, instead. Thus, the forward pass complexity is $\mathcal{O}(NM^2)$ (or $\mathcal{O}(NM\log(M))$ with a kd-tree).

The backward pass complexity is dominated by the least squares regression in Line 20 in Algorithm 3. A QR-decomposition solver is used for the regression, so if k is the number of basis functions used in $\Phi(x)$, and $M \gg k$, then its complexity is $\mathcal{O}(Mk^2)$. Thus, the complexity of the backward pass is $\mathcal{O}(NMk^2)$.

VI. Numerical Examples

We evaluated the FBRRT-SDE algorithm by applying it to three challenging nonlinear SOC problems. For all three problems, the terminal cost is taken to be a quadratic function centered at the origin and the running cost is taken to be an

L_1/min-fuel cost, which makes these problems challenging for traditional solution methods. For the first two problems the control input was restricted in the interval $U = [-1, 1]$. For the third problem the control is restricted to $U = [-1, 1]^2$.

A. L_1 Double Integrator

In order to compare the proposed FBRRT-SDE algorithm to the parallel sampled techniques in [6], which we denote below as parallel-sampled FBSDE, we considered the double integrator system with

$$\begin{bmatrix} \mathrm{d}X_s^{(1)} \\ \mathrm{d}X_s^{(2)} \end{bmatrix} = \begin{bmatrix} X_s^{(2)} \\ u \end{bmatrix} \mathrm{d}s + \begin{bmatrix} 0.01 & 0 \\ 0 & 0.1 \end{bmatrix} \begin{bmatrix} \mathrm{d}W_s^{(1)} \\ \mathrm{d}W_s^{(2)} \end{bmatrix}$$

with L_1 running cost, i.e.,

$$\inf_{u_{[0,T]}} \mathbf{E}_{\mathrm{Q}} \left[\int_0^T c_0 |u_s| \mathrm{d}s + \sum_{j=1}^{n} c_j (X_T^{(j)})^2 \right] \tag{42}$$

where c_0, c_1, c_2 are scalar parameters. When the system starts with positive position and velocity, the optimal policy is to decelerate to a negative velocity, coast for a period of time so that fuel is not used, and then accelerate to reach the origin. For this example, the number of particles per time step is $M = 1,024$, the number of time steps is $N = 64$, and the erode particle number is $\widetilde{M} = M/2$.

As shown in Fig. 6(c)–(d), the parallel-sampled FBSDE takes a significant number of iterations to begin converging to the near-optimal policy, while the proposed method produces a near-optimal policy at the first iteration. The algorithm converges to the optimal policy.

We also compared the convergence speed and robustness of the two methods by randomly sampling different starting states and evaluated their relative performance over a number of trials. For each of 30 random initial states x_0, we ran 20 trials of each method for a number of iterations, each iteration producing an expected cost for the computed policy. We normalized the final costs across the initial states by dividing all costs for a particular initial state by the largest cost obtained across both methods. For each iteration, we assign the value of the accumulated minimum value across previous iterations for that trial, i.e., the value is the current best cost after running that many iterations, regardless of the current cost. We aggregated these values across initial states and trials into the box plots in Fig. 6. Since the FBRRT-SDE is significantly slower than the parallel-sampled FBSDE per iteration due to the nearest neighbors calculation, we scale each iteration by the runtime. Note that every iteration of FBRRT-SDE after the first one requires approximately half the runtime, since only half of the eroded tree needs resampling. In summary, the FBRRT-SDE converges faster using fewer iterations than the parallel-sampled FBSDE, and does so with half as many particle samples.

B. L_1 Double Inverted Pendulum

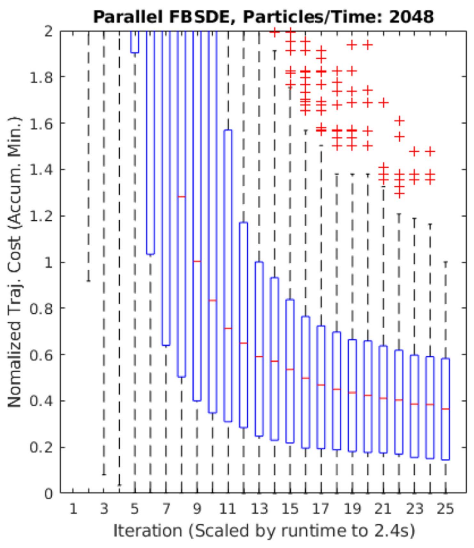
In order to study the proposed FBRRT-SDE algorithm on a highly nonlinear system in higher dimensions, we considered the double inverted pendulum with state space dimension $n = 4$ presented in [37], but with added damping friction to the joints. Thus, the dynamics are in the form $\mathrm{d}X_s = f(X_s, u_s)\,\mathrm{d}s + \sigma\,\mathrm{d}W_s$, where f is given in (44) shown at the bottom of this page, where $d_0 = 10, d_1 = 0.37, d_2 = 0.14, d_3 = 0.14, f_1 = 4.9, f_2 = 5.5, f_3 = 0.1, f_4 = 0.1$ are scalar parameters of the system, and where $\sigma = \mathrm{diag}[0.03, 0.03, 0.18, 0.18]$. The associated optimal control problem is

$$\inf_{u_{[0,T]}} \mathbf{E}_{\mathrm{Q}} \left[\int_0^T c_0 |u_s| \mathrm{d}s + \sum_{j=1}^{n} c_j (X_T^{(j)})^2 \right] \tag{43}$$

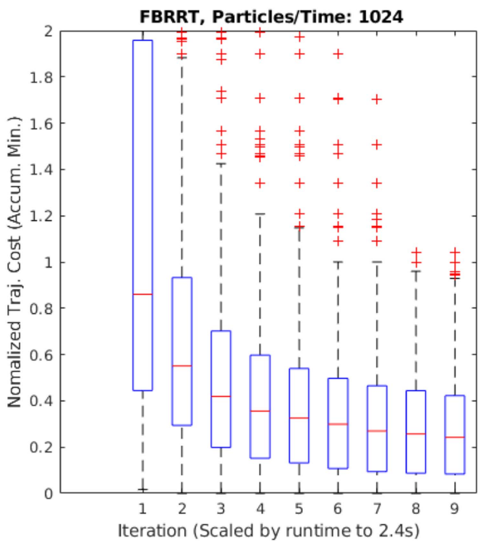
where c_0, c_1, c_2, c_3, c_4 are scalar parameters. With our unoptimized implementation (where brute force search is used for nearest neighbors) this example takes approximately 10 s to complete the first iteration and 80 s to complete 10 iterations for $M = 1 \times 1024$ (11 and 85 s, respectively, for $M = 3 \times 1024$). Two initial conditions were evaluated, $x_0^{\text{vert}} = [0, 0, 0, 0]^\top$, where the bars are vertically down and motionless, and $x_0^{\text{off}} = [\pi/10, \pi/10, 0, 0]^\top$, where the angles of both bars are slightly perturbed from x_0^{vert} by 18°. The number of time steps is taken to be $N = 80$ and the erode particle number is selected as $\widetilde{M} = (3/4)M$. The evaluation of these conditions over 30 trials with differing numbers of particles M is provided in Fig. 7.

Since the initial conditions of the two experiments are close, their optimal values should also be close. Despite having comparable optimal values, the x_0^{off} condition converges far more rapidly than the x_0^{vert} condition. Slightly perturbing the initial condition vastly improved the performance of the algorithm for this problem. The reason the x_0^{vert} condition performs poorly is likely because the system is very sensitive in that region and a localized policy results in a bifurcation of trajectory densities. If the differing groups of trajectories have similar heuristic values, the value function approximation tries to fit a function to groups of particles in different sides of the state space, resulting in poor accuracy for either group. When the x_0^{off} condition is used, there is less ambiguity in which trajectory distributions are near-optimal, resulting in better performance.

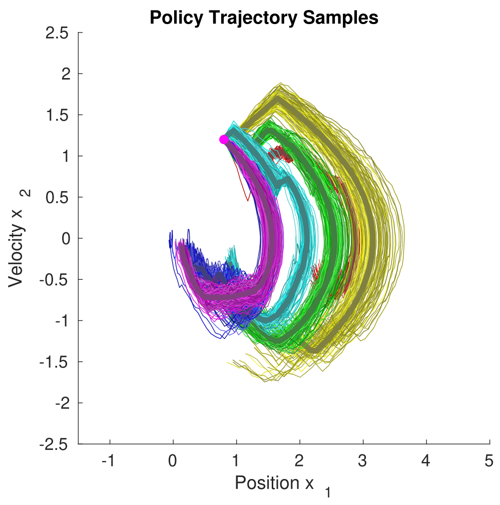
$$\begin{aligned} f(x,u) &\equiv f\left(\begin{bmatrix}\alpha & \beta & \omega & \psi\end{bmatrix}^\top, u\right) \\ &= \begin{bmatrix} \omega \\ \psi \\ \dfrac{d_3\big(d_2\psi^2\sin\beta + 2d_2\omega\psi\sin\beta - f_3\omega + f_2\sin(\alpha+\beta) - f_1\sin\alpha\big) + d_2\cos\beta\big(d_2\omega^2\sin\beta + f_4\psi - f_2\sin(\alpha+\beta)\big) + d_0 d_3\,u}{d_1 d_3 + 2d_2 d_3\cos\beta - d_2^2\cos^2\beta} \\ \dfrac{-(d_1 + 2d_2\cos\beta)\big(d_2\omega^2\sin\beta + f_4\psi - f_2\sin(\alpha+\beta)\big) - d_2\cos\beta\big(d_2\psi^2\sin\beta + 2d_2\omega\psi\sin\beta - f_3\omega + f_2\sin(\alpha+\beta) - f_1\sin\alpha\big) - d_0 d_2\cos\beta\,u}{d_1 d_3 + 2d_2 d_3\cos\beta - d_2^2\cos^2\beta} \end{bmatrix}. \end{aligned} \tag{44}$$



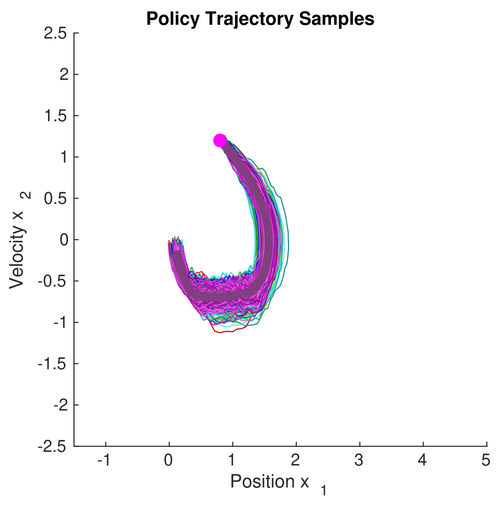
(a)



(b)



(c)



(d)

Fig. 6. (a) and (b) Comparison of parallel-sampled FBSDE [6] and FBRRT-SDE for the L_1 double integrator problem for random initial states. Expected trajectory costs for the computed policies are normalized across different initial conditions. The relative timings of iterations are visualized by their placement on the x-axis, were the right edge of each figure represents 2.4 s into runtime. (c) and (d) Trajectory samples from policies generated after the first 6 iterations. The first iteration is colored red, followed by yellow, green, cyan, dark blue, and magenta. Thick lines are mean trajectories. (a) Parallel-sampled FBSDE [6]. (b) FBRRT-SDE. (c) Trajectory samples for parallel-sampled FBSDE. (d) Trajectory samples for FBRRT-SDE.

C. L_1 Linearized Quadcopter

In order to demonstrate the proposed algorithm on a higher-dimensional system, we considered the linearized dynamics of a quadcopter having state space dimension $n = 8$, adapted from [38]. The dynamics are in the form $\mathrm{d}X_s = f(X_s, u_s)\,\mathrm{d}s + \sigma\,\mathrm{d}W_s$, where

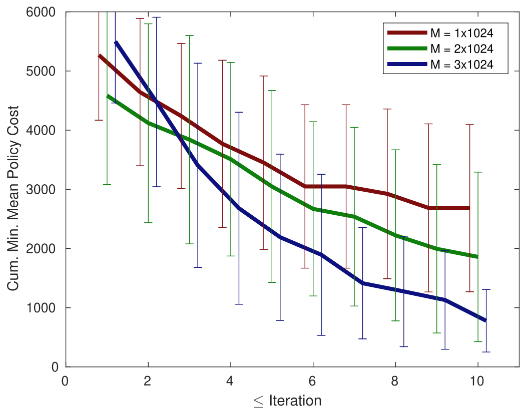
$$f(x,u) = f\left(\begin{bmatrix}\phi & \theta & p & q & u & v & x & y\end{bmatrix}^\top, \begin{bmatrix}\tau_x & \tau_y\end{bmatrix}^\top\right)$$
$$= \begin{bmatrix}p & q & d\tau_x & d\tau_y & -g\theta & g\phi & u & v\end{bmatrix}^\top$$

where $d = 4.1, g = 9.8$ are scalar parameters of the system, and where $\sigma = \mathrm{diag}[10^{-5}, 10^{-5}, 0.2, 0.2, 0.002, 0.002, 10^{-5}, 10^{-5}]$. The associated optimal control problem is

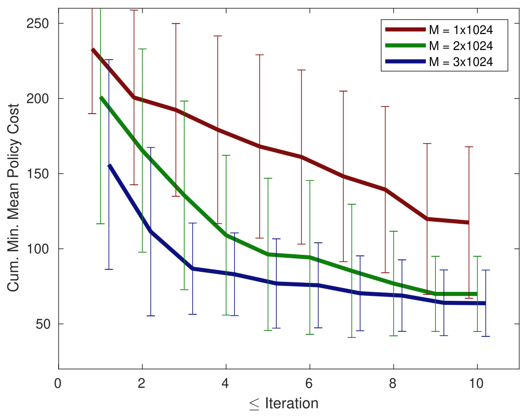
$$\inf_{u_{[0,T]}} \mathbf{E}_{\mathrm{Q}}\left[\int_0^T |u_s^{(1)}| + |u_s^{(2)}|\mathrm{d}s + \sum_{j=1}^n c_j (X_T^{(j)})^2\right] \qquad (45)$$

where $c_1 = c_2 = c_3 = c_4 = c_5 = c_6 = 1, c_7 = c_8 = 100$ are scalar parameters. With our unoptimized implementation (where brute force search is used for nearest neighbors) this example takes approximately 5 s to complete the first iteration and 37 s to complete ten iterations for $M = 1 \times 1024$ (18 and 110 s, respectively, for $M = 3 \times 1024$). The results are shown in Fig. 8.

For this example, progress in terms of convergence primarily occurs in the first iteration. Adding more particles significantly improved progress both in the first iteration and after several



(a)



(b)

Fig. 7. Mean policy cost statistics for L_1 double inverted pendulum problem. The mean bars and standard deviation whiskers characterize the distribution over 30 trials, where the value for each iteration is the accumulated minimum of the values over all previous iterations in that trial up to and including that iteration. M particles are used per time step in each condition. (a) Mean cost distribution for x_0^{vert}. (b) Mean cost distribution for x_0^{off}.

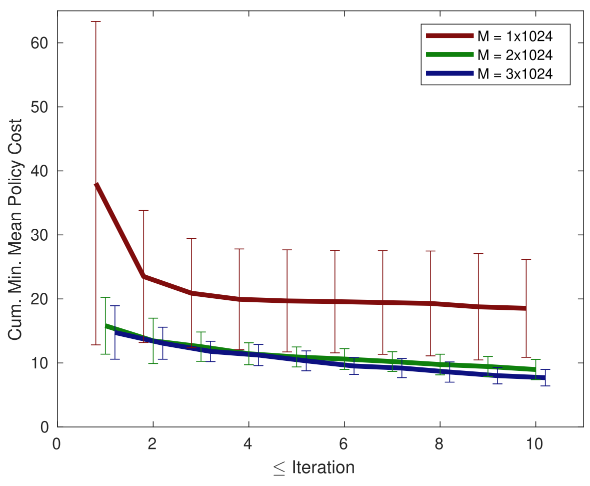


Fig. 8. Mean policy cost statistics for L_1 linearized quadcopter problem. The mean bars and standard deviation whiskers characterize the distribution over 30 trials, where the value for each iteration is the accumulated minimum of the values over all previous iterations in that trial up to and including that iteration. M particles are used per time step in each condition.

iterations. However, there were diminishing returns as the number of particles increased from 2×1024 to 3×1024, likely due to the fact that the first iteration is already near optimal. Of interest is also the observation that the time required to solve this 8-D problem was not significantly longer than the previous 4-D example. This suggests that the time-complexity of the proposed method is more related to the complexity of the control problem itself rather than the dimensionality of the state space.

VII. Conclusion

We have proposed a novel generalization of the FBSDE approach to solve SOC problems, combining branched sampling techniques with weighted least squares function approximation to greatly expand the flexibility of these methods. By leveraging the efficient space-filling properties of RRT methods, we have demonstrated that our method significantly improves the convergence properties of previous FBSDE numerical methods. We have shown how the proposed method works hand-in-hand with a local entropy-weighted LSMC method, concentrating function approximation in the regions where optimal trajectories are most likely to be dense. We have demonstrated that FBRRT-SDE can generate feedback control policies for high-dimensional nonlinear SOC problems.

Several of the design choices exposed by our approach offer significant opportunities for further research. First, although in this article, we have employed the most basic of the RRT algorithms, there has been almost two decades of development in this field. Employing more elaborate methods may improve the forward sampling even further. In addition, in this article, we did not discuss state constraints or obstacles. Since RRT methods are naturally designed to accommodate obstacles, the methods proposed here should be extendable to those problems as well.

Moreover, time discretization plays a significant role on the accuracy of the numerical solutions of Feynman-Kac forward backward SDEs. In this work, we used the conventional Euler-Maruyama method for time discretizations; however, as is shown in [39] and [40], the numerical accuracy of backward integration as well as the iterative robustness of the algorithms can be significantly improved using Taylor-based estimators for numerically solving Feynman-Kac FBSDEs.

Another area of research worth investigating is to find other methods of value function representation. In this article, we use a rather simple parameterization of the value functions, though this simplicity offers some distinct benefits. Specifically, quadratic basis functions result in gradients that are linear, and policies that are typically stable. In regions where particles are sparse, the convexity of the value function representation naturally drives the system back toward the particle distribution. Further investigations might reveal other value function parameterizations with potentially better representation power than quadratic functions, while maintaining the benefits and the nice properties of the latter.

References

[1] J. Yong and X. Y. Zhou, *Stochastic Controls: Hamiltonian Systems and HJB Equations*. Berlin, Germany: Springer, 1999.

[2] E. Pardoux and S. G. Peng, "Adapted solution of a backward stochastic differential equation," *Syst. Control Lett.*, vol. 14, no. 1, pp. 55–61, 1990.
[3] S. Peng, "Backward stochastic differential equations and applications to optimal control," *Appl. Math. Optim.*, vol. 27, no. 2, pp. 125–144, 1993.
[4] N. El Karoui, S. Peng, and M. C. Quenez, "Backward stochastic differential equations in finance," *Math. Finance*, vol. 7, no. 1, pp. 1–71, 1997.
[5] I. Exarchos and E. A. Theodorou, "Stochastic optimal control via forward and backward stochastic differential equations and importance sampling," *Automatica*, vol. 87, pp. 159–165, 2018.
[6] I. Exarchos, E. A. Theodorou, and P. Tsiotras, "Stochastic L^1-optimal control via forward and backward sampling," *Syst. Control Lett.*, vol. 118, pp. 101–108, 2018.
[7] I. Exarchos, E. A. Theodorou, and P. Tsiotras, "Game-theoretic and risk-sensitive stochastic optimal control via forward and backward stochastic differential equations," in *Proc. 55th IEEE Conf. Decis. Control*, 2016, pp. 6154–6160.
[8] I. Exarchos, E. A. Theodorou, and P. Tsiotras, "Stochastic differential games: A sampling approach via FBSDEs," *Dyn. Games Appl.*, vol. 9, pp. 486–505, 2019.
[9] H. M. Soner and N. Touzi, "A stochastic representation for the level set equations," *Commun. Partial Differ. Equ.*, vol. 27, no. 9/10, pp. 2031–2053, 2002.
[10] C. Bender and R. Denk, "A forward scheme for backward SDEs," *Stochastic Processes Appl.*, vol. 117, no. 12, pp. 1793–1812, 2007.
[11] F. A. Longstaff and E. S. Schwartz, "Valuing American options by simulation: A simple least-squares approach," *Rev. Financial Stud.*, vol. 14, no. 1, pp. 113–147, 2001.
[12] J. Ma and J. Yong, *Forward-Backward Stochastic Differential Equations and Their Applications*. Berlin, Germany: Springer, 2007.
[13] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. New York, NY, USA: North-Holland, 1970.
[14] E. A. Theodorou, Y. Tassa, and E. Todorov, "Stochastic differential dynamic programming," in *Proc. Amer. Control Conf.*, 2010, pp. 1125–1132.
[15] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2008, vol. 20, pp. 1465–1472.
[16] Y. Tassa, "Theory and implementation of biomimetic motor controllers," Ph.D. dissertation, Hebrew Univ. Jerusalem, Jerusalem, Israel, 2011.
[17] G. Williams et al., "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1714–1721.
[18] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice-A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.
[19] S. Yensiri, "An investigation of radial basis function-finite difference (RBF-FD) method for numerical solution of elliptic partial differential equations," *Mathematics*, vol. 5, 2017, Art. no. 54. [Online]. Available: http://www.mdpi.com/2227-7390/5/4/54
[20] M. Li, W. Chen, and C. S. Chen, "The localized RBFs collocation methods for solving high dimensional PDEs," *Eng. Anal. Boundary Elements*, vol. 37, no. 10, pp. 1300–1304, 2013, doi: 10.1016/j.enganabound.2013.06.001.
[21] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.
[22] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using RRT* based approaches: A survey and future directions," *Int. J. Adv. Comput. Sci. Appl*, vol. 7, no. 11, pp. 97–107, 2016.
[23] O. Arslan, E. A. Theodorou, and P. Tsiotras, "Information-theoretic stochastic optimal control via incremental sampling-based algorithms," in *Proc. IEEE Symp. Adaptive Dyn. Program. Reinforcement Learn.*, 2014, pp. 1–8.
[24] K. P. Hawkins, A. Pakniyat, E. Theodorou, and P. Tsiotras, "Forward-backward rapidly-exploring random trees for stochastic optimal control," in *Proc. 60th IEEE Conf. Decis. Control*, 2021, pp. 912–917.
[25] W. H. Fleming and H. M. Soner, *Controlled Markov Processes and Viscosity Solutions*. Berlin, Germany: Springer, 2006.
[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[27] G. Lowther, "Girsanov transformations," 2010. [Online]. Available: https://almostsure.wordpress.com/2010/05/03/girsanov-transformations/
[28] W. H. Fleming and R. W. Rishel, *Deterministic and Stochastic Optimal Control*. Berlin, Germany: Springer, 1975.
[29] C. Bender and T. Moseler, "Importance sampling for backward SDEs," *Stochastic Anal. Appl.*, vol. 28, no. 2, pp. 226–253, 2010.
[30] E. A. Theodorou and E. Todorov, "Relative entropy and free energy dualities: Connections to path integral and KL control," in *Proc. 51st IEEE Conf. Decis. Control*, 2012, pp. 1466–1473.
[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
[32] A. Pascucci, *PDE and Martingale Methods in Option Pricing*. Berlin, Germany: Springer, 2011.
[33] S. Resnick, *A Probability Path*. Basel, Switzerland: Birkhäuser Verlag AG, 2003.
[34] P. Del Moral, *Mean Field Simulation for Monte Carlo Integration*. Edward Chapman, William Hall: Chapman and Hall/CRC, 2013.
[35] V. I. Bogachev, *Measure Theory*, vol. 1. Berlin, Germany: Springer, 2007.
[36] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, vol. 23. Berlin, Germany: Springer, 2013.
[37] R. Tedrake, "Underactuated robotics: Learning, planning, and control for efficient and agile machines: Course notes for MIT 6.832," *Work. Draft Ed.*, vol. 3, 2009, Art. no. 2.
[38] F. Sabatino, "Quadrotor control: Modeling, nonlinear control design, and simulation," M.S. thesis, KTH Royal Inst. Technol., Stockholm, Sweden, Jun. 2015.
[39] K. P. Hawkins, A. Pakniyat, and P. Tsiotras, "On the time discretization of the Feynman-KAC forward-backward stochastic differential equations for value function approximation," in *Proc. 60th IEEE Conf. Decis. Control*, 2021, pp. 892–897.
[40] K. P. Hawkins, A. Pakniyat, and P. Tsiotras, "Value function estimators for Feynman-Kac forward-backward SDEs in stochastic optimal control," *Automatica*, vol. 158, 2023, Art. no. 111281.

Kelsey P. Hawkins received the B.S. degree in applied mathematics and computer science from North Carolina State University, Raleigh, NC, USA, in 2010, and the Ph.D. degree in robotics from the Georgia Institute of Technology, Atlanta, GA, USA, in 2021.

He now develops L4 autonomous vehicle software with Woven by Toyota, Tokyo, Japan. His research interests include numerical methods for trajectory planning and robust collision avoidance using reachability analysis with applications in autonomous driving.

Ali Pakniyat (Member, IEEE) received the B.Sc. degree in mechanical engineering from Shiraz University, Shiraz, Iran, in 2008, the M.Sc. degree in mechanical engineering from the Sharif University of Technology, Tehran, Iran, in 2010, and the Ph.D. degree in electrical engineering from McGill University, Montreal, Canada, in 2016.

After holding two postdoctoral research positions with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA, and with the Institute for Robotics and Intelligent Machines, Georgia Tech, Atlanta, GA, USA, he joined the University of Alabama, Tuscaloosa, AL, USA, in 2021. He is currently an Assistant Professor with the Department of Mechanical Engineering, University of Alabama. His research interests include deterministic and stochastic optimal control, nonlinear and hybrid systems, multiagent systems and mean field games, with applications in the automotive industry, robotics, and mathematical finance.

Evangelos Theodorou received the bachelor's and master's degrees in electrical engineering from the Technical University of Crete, Chania, Greece, the master's degree in computer science and engineering from the University of Minnesota, Minneapolis, MN, USA, and the Ph.D. degree in computer science from the University of Southern California, Los Angeles, CA, USA.

He is currently an Associate Professor with the Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, USA. He is also affiliated with the Institute of Robotics and Intelligent Machines. His research interests span the areas of stochastic optimal control, robotics, machine learning, and computational neuroscience.

Panagiotis Tsiotras (Fellow, IEEE) received the Dipl.Eng. degree in mechanical engineering from the National Technical University of Athens, Athens, Greece, in 1986, the M.S. degree in aerospace engineering from Virginia Tech, Blacksburg, VA, USA, in 1987, the M.S. degree in mathematics from Purdue University, West Lafayette, IN, USA, in 1992, and the Ph.D. degree in aeronautics and astronautics from from Purdue University, West Lafayette, IN, USA, in 1993.

He has held visiting research appointments with MIT, JPL, INRIA Rocquencourt, and Mines ParisTech. He is the David and Andrew Lewis Chair Professor with the Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology (Georgia Tech), Atlanta, GA, USA. His research interests include optimal control of nonlinear systems and ground, aerial and space vehicle autonomy.

Dr. Tsiotras has served in the Editorial Boards of IEEE TRANSACTIONS ON AUTOMATIC CONTROL, *IEEE Control Systems Magazine*, the *AIAA Journal of Guidance, Control and Dynamics*, the *Dynamic Games and Applications*, and *Dynamics and Control*. He was the recipient of the NSF CAREER award, the Outstanding Aerospace Engineer award from Purdue, and the Technical Excellence Award in Aerospace Control from IEEE. He is a Fellow of AIAA and AAS.