State Constrained Stochastic Optimal Control for Continuous and Hybrid Dynamical Systems Using DFBSDE ⋆

Bolun Dai [a], Prashanth Krishnamurthy [a], Andrew Papanicolaou [b], Farshad Khorrami [a]

[a] *New York University, Brooklyn, NY, 11201*

[b] *North Carolina State University, Raleigh, NC, 27695*

Abstract

We develop a computationally efficient learning-based forward-backward stochastic differential equations (FBSDE) controller for both continuous and hybrid dynamical (HD) systems subject to stochastic noise and state constraints. Solutions to stochastic optimal control (SOC) problems satisfy the Hamilton–Jacobi–Bellman (HJB) equation. Using current FBSDE-based solutions, the optimal control can be obtained from the HJB equations using deep neural networks (e.g., long short-term memory (LSTM) networks). To ensure the learned controller respects the constraint boundaries, we enforce the state constraints using a soft penalty function. In addition to previous works, we adapt the deep FBSDE (DFBSDE) control framework to handle HD systems consisting of continuous dynamics and a deterministic discrete state change. We demonstrate our proposed algorithm in simulation on a continuous nonlinear system (cart-pole) and a hybrid nonlinear system (five-link biped).

Key words: Stochastic control, Optimal control, Forward and backward stochastic differential equations

1 Introduction

Optimal control has been applied in various applications, e.g., robotic control [17] [5] and navigation [8]. Optimal control problems are often cast as the minimization of a cost function, which the solution can be found via numerical optimization techniques [9]. Recently, optimal control has been used to solve increasingly complex control problems [18] thanks to the enhanced computational abilities and optimization software and the adoption of neural networks [2]. Applying optimal control requires accurate modeling of the system of interest. However, unmodeled processes are common in real-world systems, e.g., measurement noise and external forces with unknown distributions [16] [10]. We adopt SOC [4] techniques to consider the uncontrolled inputs explicitly.

Recently, neural networks have been widely used to solve SOC problems [14] [15] [20] [7]. A promising direction for deep neural network (DNN) based SOC solutions is formulating the problem as an FBSDE [22]. FBSDE enables the numerical estimation of the solution to the HJB equation, i.e., the value function with respect to a cost function. The optimal control at each state then can be analytically obtained using the estimated value function. The introduction of deep neural networks [22] to FBSDEs improves the value function estimation. Compared to nonlinear model predictive control (MPC) approaches, DFBSDE-based approaches are computationally more efficient during inference by moving the heavy computation offline. On top of the vanilla DFBSDE, work has been done in introducing state constraints using penalty functions [6] and appending a differentiable quadratic program [19]. In this work, we explore a penalty function based approach since it is faster in training and inference.

Another key aspect of this paper is the control of HD systems. The type of HD systems considered consists of two phases: first, a phase of continuously evolving dynamics,

⋆ This work is supported in part by NSF grant DMS-1907518 and in part by the New York University Abu Dhabi (NYUAD) Center for Artificial Intelligence and Robotics, funded by Tamkeen under the NYUAD Research Institute Award CG010. An earlier version [6] of a portion of this paper was presented at the 2021 American Control Conference (Virtual), May 2021.

Email addresses: bd1555@nyu.edu (Bolun Dai), prashanth.krishnamurthy@nyu.edu (Prashanth Krishnamurthy), apapani@ncsu.edu (Andrew Papanicolaou), khorrami@nyu.edu (Farshad Khorrami).

and second, a phase of a discrete jump. Such HD systems are very common, e.g., walking robots [12]. Thus, it is worthwhile to understand how it is typically controlled. The control of walking robots is traditionally achieved with a hierarchical control architecture. A reduced-order model is used for motion generation. The generated motion is tracked via a tracking controller [17]. Another approach is to see the HD system as a combination of multiple phases of continuous dynamics [21]. Using this approach, the controller only needs to work well for the continuous dynamics and be robust to changes caused by discrete jumps. This formulation is end-to-end, which better suits DFBSDE.

This paper proposes a SOC setting for nonlinear systems that incorporates state constraints and adapts the method to handle HD systems. The main contribution of this paper is threefold: (1) proposed the state constraint DFBSDE algorithm and adapted it to handle HD systems; (2) devised a new training loss and controller ensemble setting for high-dimensional HD systems; (3) provided simulation studies on both continuous and HD systems to show the efficacy of our approach. In relationship with our previous work [6], this paper provides a more detailed development of the methodology, extension to hybrid systems, and application to bipeds. The remainder of this paper is structured as follows. In Section II, the state-constrained SOC formulation is given. Section III presents a derivation connecting the SOC formulation and FBSDEs, and how control saturation and state constraints can be incorporated. Then using the derived formulation, a deep neural network-based algorithm is presented to solve the FBSDE with state constraints and control saturation. The last part of Section III discusses adapting the DFBSDE algorithm to HD systems. In Section IV, simulation studies are presented for a continuous dynamical system, namely the cart-pole, and a five-link biped [13], which is HD.

2 Problem Formulation

In this section, we outline the SOC problem under state constraints. We consider a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P})$, where Ω is the sample space, $\mathcal{F}$ is the σ-algebra over Ω, $\mathbb{P}$ is a probability measure, and $\{\mathcal{F}_t\}_{t\geq 0}$ is a filtration with index t denoting time. A controlled affine system with noise represented by stochastic processes can be described using a stochastic differential equation (SDE)

$$d\mathbf{x}(t) = (\mathbf{F}(\mathbf{x}(t)) + \mathbf{G}(\mathbf{x}(t))\mathbf{u})dt + \Sigma(\mathbf{x}(t))d\mathbf{w}(t) \quad (1)$$

with initial state $\mathbf{x}_0 \in \mathbb{R}^n$ and $\mathbf{w}(t) \in \mathbb{R}^\nu$ an $\mathcal{F}_t$-adapted Brownian motion. Throughout this paper, we will write $\mathbf{x}(t)$ and other stochastic processes with argument t omitted whenever appropriate. The states are denoted by $\mathbf{x} \in \mathbb{R}^n$ and the control input by $\mathbf{u} \in \mathbb{R}^m$. In (1), $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ represents the drift, $\mathbf{G} : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ represents the control influence, and $\Sigma : \mathbb{R}^n \to \mathbb{R}^{n\times\nu}$ represents the diffusion (influence of the Brownian motion to the state). We assume that range($\mathbf{G}$) $\subseteq$ range(Σ), i.e., the noise enters wherever the control input appears (in addition to possibly elsewhere). The state constrained SOC problem is to find a controller $\mathbf{u}(\mathbf{x})$ that minimizes an objective function $J^{\mathbf{u}}(\mathbf{x}, t) \in \mathbb{R}_+$ under a set of state constraints. The objective function is denoted as

$$J^{\mathbf{u}}(\mathbf{x}, t) = \mathbb{E}\Big[\mathbf{q}_N(\mathbf{x}(T)) + \int_t^T \Big(\mathbf{q}(\mathbf{x}) + \mathbf{r}(\mathbf{u})\Big)d\tau\Big] \quad (2)$$

where $\mathbb{E}$ represents expectation, $T \in \mathbb{R}_+$ is the terminal time, $\mathbf{q}_N : \mathbb{R}^n \to \mathbb{R}_+$ is the terminal state cost, $\mathbf{q} : \mathbb{R}^n \to \mathbb{R}_+$ is the instantaneous state cost, and the instantaneous control cost is $\mathbf{r} : \mathbb{R}^m \to \mathbb{R}_+$. The controller $\mathbf{u}(\mathbf{x}(t))$ is a function of the state, this dependency will be omitted in the remainder of this paper for brevity, and the controller will be written as $\mathbf{u}$. Without loss of generality, we consider state constraints in the following form $\mathbf{c}(\mathbf{x}) \leq \mathbf{b}$, where $\mathbf{c}(\mathbf{x}) \in \mathbb{R}^r$ is a vector of functions of the state, and $\mathbf{b} \in \mathbb{R}^r$ represents the element-wise upper bound of $\mathbf{c}(\mathbf{x})$. Control saturation (with $\mathbf{U}_{\max} \in \mathbb{R}^m_+$) can also be introduced into the SOC formulation

$$\mathbf{u} \in \mathcal{U} = \{\mathbf{u} \mid |\mathbf{u}_i| \leq \mathbf{U}_{i,\max},\ i = 1, \ldots, m\} \quad (3)$$

where $\mathbf{u}_i$ and $\mathbf{U}_{i,\max}$ are the i^{th} element of $\mathbf{u}$ and $\mathbf{U}_{\max}$, respectively. Note that all operations with stochastic processes are understood a.s. (almost surely, i.e., with probability 1) unless specifically indicated. In summary, we have the SOC problem as

$$\min_{\mathbf{u}\in\mathcal{U}}\ J^{\mathbf{u}}(\mathbf{x}_0, t_0) \text{ subject to } (1) \text{ and } \mathbf{c}(\mathbf{x}) \leq \mathbf{b} \text{ a.s.} \quad (4)$$

3 Method

In this section, we first show how (4) can be written as an FBSDE. Then, state constraints are considered. Subsequently, adaptations for HD systems are presented. Finally, a DNN-based solution to the FBSDE is proposed.

3.1 SOC to FBSDE

The derivation presented in this section was originally proposed in [22], it is summarized here for completeness. Let $\mathbf{V}(\cdot, \cdot)$ be the value function $\mathbf{V}(\mathbf{x}, t) \in \mathbb{R}_+$ defined as

$$\mathbf{V}(\mathbf{x}, t) := \inf_{\mathbf{u}\in\mathcal{U}} J^{\mathbf{u}}(\mathbf{x}, t). \quad (5)$$

Using (2) and Bellman's principle [1] yields

$$\mathbf{V}_t(\mathbf{x}, t) + \mathcal{L}\mathbf{V}(\mathbf{x}, t) + \mathbf{h}(\mathbf{x}, \mathbf{V}_\mathbf{x}(\mathbf{x}, t)) = 0 \quad (6a)$$
$$\mathbf{V}(\mathbf{x}(T), T) = \mathbf{q}_N(\mathbf{x}(T)) \quad (6b)$$

where $\mathbf{h}$ denotes the Hamiltonian

$$\mathbf{h}(\mathbf{x}, \mathbf{V_x}) = \inf_{\mathbf{u}\in\mathcal{U}} \Big(\mathbf{q}(\mathbf{x}) + (\mathbf{G}(\mathbf{x})\mathbf{u})^T\mathbf{V_x} + \mathbf{r}(\mathbf{u})\Big), \quad (7)$$

the differential operator of $\mathbf{V}$ is given by

$$\mathcal{L}\mathbf{V}(\mathbf{x}, t) = \frac{1}{2}\text{trace}\Big(\Sigma\Sigma^T\mathbf{V_{xx}}(\mathbf{x}, t)\Big) + \mathbf{F}^T(\mathbf{x})\mathbf{V_x}(\mathbf{x}, t), \quad (8)$$

$\mathbf{V}_t(\mathbf{x}, t) \in \mathbb{R}$ is the partial derivative of $\mathbf{V}$ with respect to t, and $\mathbf{V_x}(\mathbf{x}, t) \in \mathbb{R}^{n\times 1}$ and $\mathbf{V_{xx}}(\mathbf{x}, t) \in \mathbb{R}^{n\times n}$ denote the first and second partial derivatives, respectively, of $\mathbf{V}$ with respect to $\mathbf{x}$. To consider control constraints [22], we define $\mathbf{r}(\mathbf{u})$ as

$$\mathbf{r}(\mathbf{u}) = \sum_{i=1}^{m} \mathbf{S}_i(\mathbf{u}_i) = \sum_{i=1}^{m} c_i \int_0^{\mathbf{u}_i} \text{sig}^{-1}\Big(\frac{v}{\mathbf{u}_i^{\max}}\Big)dv \quad (9)$$

with $\mathbf{u}_i^{\max}$ being the i-th element of $\mathbf{U}_{\max}$ and

$$\text{sig}(v) = \frac{2}{1 + e^{-v}} - 1. \quad (10)$$

In (9), c_i weights the importance between different control inputs. Using first-order conditions to solve for an analytic solution of the optimal control action that achieves the infimum of the Hamiltonian yields

$$\mathbf{G}^T(\mathbf{x})\mathbf{V_x} + \mathbf{R}\text{sig}^{-1}(\frac{\mathbf{u}^*}{\mathbf{U}_{\max}}) = 0, \quad (11)$$

with $\mathbf{R} = \text{diag}(c_1, \cdots, c_m)$. Solving (11) for $\mathbf{u}^*$ in yields

$$\mathbf{u}^*(\mathbf{x}, t) = \mathbf{U}_{\max} * \text{sig}(-\mathbf{R}^{-1}\mathbf{G}^T(\mathbf{x})\mathbf{V_x}(\mathbf{x}, t)) \quad (12)$$

with "$*$" denoting element-wise multiplication. The control is saturated between $[-\mathbf{U}_{\max}, \mathbf{U}_{\max}]$. Define $\mathbf{y}(t) = \mathbf{V}(\mathbf{x}(t), t)$. From (7), we can write a stochastic system for the optimal value function

$$-d\mathbf{y}(t) = (\mathbf{q}(\mathbf{x}) + \mathbf{r}(\mathbf{u}^*))dt - \mathbf{V}_\mathbf{x}^T(\mathbf{x}, t)\Sigma^T(\mathbf{x})d\mathbf{w} \quad (13a)$$
$$d\mathbf{x}(t) = (\mathbf{F}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u}^*)dt + \Sigma(\mathbf{x})d\mathbf{w} \quad (13b)$$
$$\mathbf{y}(T) = \mathbf{q}_N(\mathbf{x}(T)) \quad (13c)$$
$$\mathbf{x}(0) = \mathbf{x}_0 \quad (13d)$$

which is in the form of a FBSDE (the forward part consists of (13b) (13d), the backward part consists of (13a) (13c)). The forward and backward parts are of the form of a forward SDE (FSDE) and a backward SDE (BSDE), respectively. For details regarding the derivation from (7) to (13), the reader is referred to [22][6].

3.2 Handling State Constraints

In this section, we consider incorporating state constraints into the FBSDE framework using a penalty function based approach. A perfect penalty function would be zero inside the constraint boundary and infinity outside. We propose two differentiable and numerically stable alternatives to a perfect penalty function. The first option is a penalty function based on logistic functions (PFL). When the state constraint has both an upper bound $\mathbf{b}_{\max}$ and lower bound $\mathbf{b}_{\min}$, the i-th element of PFL would be

$$\mathbf{p}_i(\mathbf{x}) = \frac{L}{1 + e^{-\mathbf{k}(\mathbf{c}_i(\mathbf{x}) - \mathbf{b}_{i,\max})}} - \frac{L}{1 + e^{-\mathbf{k}(\mathbf{c}_i(\mathbf{x}) - \mathbf{b}_{i,\min})}} + L - \frac{2L}{1 + e^{-\mathbf{k}(\mu_i - \mathbf{b}_{i,\max})}} \quad (14)$$

where $L \in \mathbb{R}^+$ determines the maximum value of the penalty, $\mathbf{k} \in \mathbb{R}^+$ determines the steepness of the boundary (larger k leads to steeper boundaries), and $\mu = (\mathbf{b}_{\min} + \mathbf{b}_{\max})/2$. The proposed penalty function consists of two parts: the first consists of two logistics functions, which give the valley shape; the second sets the minimum value of the penalty function to zero. The second option is a rectified-linear-unit (ReLU) based function which is defined as $\mathbf{ReLU}(x) = \max(x, 0)$. When an upper and lower bound exist, the penalty function is

$$\mathbf{p}_i(\mathbf{x}) = \mathbf{k}(\mathbf{ReLU}(\mathbf{b}_{i,\min} - \mathbf{c}_i(\mathbf{x})) + \mathbf{ReLU}(\mathbf{c}_i(\mathbf{x}) - \mathbf{b}_{i,\max})). \quad (15)$$

Similar to (14), $\mathbf{k}$ represents the steepness of the constraint boundaries. However, in (15), there is no bound on the penalty function magnitude. An example of these two types of penalty functions with varying steepness is shown in Fig. 1. When only an upper bound exists, i.e., $\mathbf{c}(\mathbf{x}) \leq \mathbf{b}$, the lower bound is set to be $-\infty$, and when only a lower bound exists, i.e., $\mathbf{c}(\mathbf{x}) \geq \mathbf{b}$, the upper bound is set to be ∞. The main difference between these two penalty functions is their values within the constraint boundary. For PFL, the value of the penalty function is only close to zero when a large $\mathbf{k}$ value is utilized. Otherwise, states within but close to constraint boundaries will have a positive penalty function value. The ReLU-based penalty function is always zero within the bound. Specific use cases are demonstrated in Section 4.

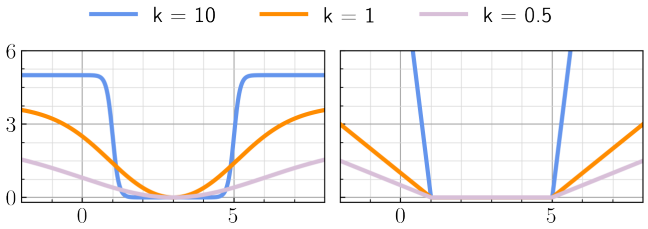

Fig. 1. The left figure shows $\mathbf{p}(\mathbf{x})$ in (14) under different values of k, with $\mu = 3$, $L = 5$, $\mathbf{c}(\mathbf{x}) = \mathbf{x}$, $\mathbf{b}_{\min} = 1$, $\mathbf{b}_{\max} = 5$ and $\mathbf{x}$ being a scalar. The right figure shows $\mathbf{p}(\mathbf{x})$ in (15) under different values of k, with the same parameters.

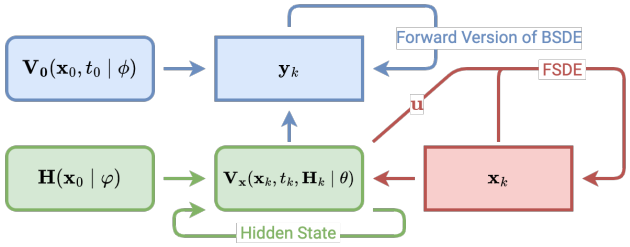

Fig. 2. The DFBSDE architecture for one time step. Curve-edged boxes contains DNNs, and sharp-edged boxes represent intermediate calculations. Everything in green contains the LSTM network, red and blue colors represent the FSDE and the forward version of the BSDE, respectively.

After applying penalty function $\mathbf{p}(\mathbf{x})$, the instantaneous state cost is $\bar{\mathbf{q}}(\mathbf{x}) = \mathbf{q}(\mathbf{x}) + \mathbf{p}(\mathbf{x})$. Ideally, we would pick larger $\mathbf{k}$ and L to ensure a steeper boundary and larger penalty. While a controller without access to measurements of future noise can not *guarantee* that state constraints will be met for all time due to stochastic nature of the system dynamics, note that the penalty functions guide the learning towards a robust controller that does not violate state constraints (at least under the disturbances seen during training). The expected value of the integral of $\mathbf{p}(\mathbf{x})$ is at most $J^{\mathbf{u}}$. If $J^{\mathbf{u}}$ is finite and $\mathbf{p}(\mathbf{x})$ is a perfect penalty function, the constraint is satisfied except on a set of measure zero.

3.3 Deep FBSDE Algorithm

The solution of the FBSDE in (13) requires backward integration in time and the fact that $\mathbf{V_x}(\mathbf{x})$ is unknown creates additional difficulties in utilizing classical numerical integration approaches. To deal with backward integration in time, following the formulation in [22], we can estimate the value function at time zero using a DNN, which is parameterized by ϕ, i.e., $\mathbf{V_0}(\mathbf{x}, t_0)$. This allows the forward integration of the BSDE. We can then rewrite the FBSDE as two FSDEs. For unknown $\mathbf{V_x}(\mathbf{x}, t)$ values, it can also be estimated using a DNN parameterized by θ, i.e., $\mathbf{V_x}(\mathbf{x}, t \mid \theta)$.

We deploy an LSTM-based architecture [11] for $\mathbf{V_x}(\mathbf{x}, t \mid \theta)$, which has shown to be superior to dense-layer-based architectures [22]. Linear regression based approaches have also been considered, but are inferior due to compounding error. Assuming the initial state and time horizon are fixed [22], the initial value function is a learned fixed value $\mathbf{V_0}(\phi)$. Due to the use of LSTMs, two changes need to be made to the DFBSDE algorithm. First, a separate network $\mathbf{H}_0(\varphi)$ is used to estimate the initial hidden state values. Second, $\mathbf{y}_k$ will also depend on the hidden state values $\mathbf{H}_k$. The FBSDE can be forward integrated by discretizing the time as follows

$$\mathbf{y}_k = V_0(\mathbf{x}_0, t_0 \mid \phi) - \sum_{i=1}^{k} (\bar{\mathbf{q}}(\mathbf{x}_{i-1}) + \mathbf{r}(\mathbf{u}^*_{i-1}))\Delta t$$
$$+ \sum_{i=2}^{k} \mathbf{V}_{\mathbf{x}}^T(\mathbf{x}_{i-1}, t_{i-1}, \mathbf{H}_{i-1} \mid \theta)\Sigma^T(\mathbf{x}_{i-1})\Delta \mathbf{w}_{i-1}$$
$$+ \mathbf{V}_{\mathbf{x}}^T(\mathbf{x}_0, t_0, \mathbf{H}_0(\varphi) \mid \theta)\Sigma^T(\mathbf{x}_0)\Delta \mathbf{w}_0 \quad (16a)$$

$$\mathbf{x}_k = \mathbf{x}_0 + \sum_{i=1}^{k} \Delta \mathbf{x}_{i-1} \quad (16b)$$

where $k \geq 1$, $\Delta t = T/N$, $\Delta \mathbf{x}_k = \dot{\mathbf{x}}(k\Delta t)\Delta t$ and $\Delta \mathbf{w}_k = \dot{\mathbf{w}}(k\Delta t)\Delta t$. The calculation of $\mathbf{x}_k$ and $\mathbf{y}_k$ is illustrated in Fig. 2. To learn the weights, we minimize the least square loss between the estimated terminal value function and the measured terminal cost,

$$\mathbf{L}_{\text{FBSDE}}(\theta, \phi, \varphi) = \sum_{j=1}^{M} \|\mathbf{q}_N(\mathbf{x}_N^j) - \mathbf{y}_N^j\|^2 \quad (17)$$

where M is the batch size and the superscript j denotes the j-th set of data in the batch. The detailed DFBSDE algorithm is shown in Algorithm 1.

3.4 Penalty Function Update

For the penalty function, if $\mathbf{k}$ is large in the initial stage of training, numerical instabilities can arise due to large gradients generated by states within the steep region near the constraint boundary. On the other hand, if $\mathbf{k}$ is kept small, it wouldn't be effective since the penalty for constraint violation is small. Thus, we propose to update $\mathbf{k}$ during training so that it gradually increases.

The update scheme is as follows. For a given $\mathbf{k}$, the DNN is trained until convergence; then, $\mathbf{k}$ is updated. A metric to check for convergence is the variance of the episode-wise cost over a few episodes; the episode window size is denoted as η. If the iterates are converging, the variance will be small. Convergence is determined by comparing the square root of the variance with a threshold $\beta \in \mathbf{R}_+$; if it is smaller than β, $\mathbf{k}$ is updated as $\mathbf{k} \leftarrow \mathbf{k} + \delta$. This condition is checked every η iterations. Additionally, if the condition is not satisfied after η' iterations, then also $\mathbf{k}$ is updated since it could be stuck at a local minimum. Empirically, we have found $\eta = 500$, $\mathbf{k} = 1.5$, $\delta = 0.5$, and $\Delta_\delta = 0.25$ to be reasonable values to start with. Both δ and Δ_δ can be increased if training is stable and faster growth of $\mathbf{k}$ is desired; otherwise, δ and/or Δ_δ should be reduced. During training, the variance decreases. Thus, β should also decrease after each update: $\beta = \gamma\beta$, with $\gamma \in (0, 1)$. To make the decrease in β smoother, we gradually increase γ to 1 using $\gamma = \gamma + \Delta$ with $\Delta \in \mathbb{R}_+$. Similarly, the acceleration of k is made negative, i.e., $\delta = \delta + \Delta_\delta$, with $\Delta_\delta \in \mathbb{R}_-$, leading to finer-grained changes in k at later stages of training. The initial value of β can be determined after the training converges for the first iteration of Algorithm 2. From empirical evaluations, we find $\gamma = 0.9$ and $\Delta = 0.02$ to be reasonable values. The penalty function update scheme is shown in Algorithm 2.

Algorithm 1 State Constrained DFBSDE Controller

Get Initial state and state dynamics, cost function parameters, penalty function, N: Time horizon, N_I: Number of iterations, M: Batch size, Δt: Time step, λ: Weight decay parameter, and state constraint parameters (refer to Algorithm 2);
Initialize θ, ϕ, φ, k, δ, β, γ;
for $n_I = 1$ to N_I, $m = 1$ to M, $k = 0$ to $N-1$ **do**
    Calculate $t_k = k\Delta t$ and $\mathbf{V}_{\mathbf{x}}^m(\mathbf{x}_k, t_k \mid \theta)$;
    Calculate optimal control as in (12);
    Sample Brownian noise: $\Delta \mathbf{w}_k^m \sim \mathcal{N}(0, \Delta t)$;
    Update value function $\mathbf{y}_k^m$ and system state $\mathbf{x}_k^m$;
    Compute target terminal cost;
    Compute $\mathbf{L}_{\text{FBSDE}}$ (17) or $\mathbf{L}_{\text{HFBSDE}}$ (20)
    Update θ, ϕ and φ and run Algorithm 2;

3.5 Adaptation to Hybrid Dynamics

This section will discuss the adaptations required for the DFBSDE algorithm to handle HD systems. HD systems consist of two phases: a continuous dynamic phase and a phase of a deterministic discrete jump. Such an HD system is very common, e.g., a biped. The FBSDE formulation in (13) is only for continuous dynamics. Inspired by the cyclic motion in human walking, we treat the hybrid dynamics as having multiple cycles, where the dynamics are continuous within each cycle. For the case of bipedal locomotion, each cycle will be a footstep; at the end of each cycle, the swing foot lands on the ground (more on this in Section 4). The main challenges in adapting the DFBSDE framework to HD systems are robustness to the initial state and assurance of cyclic motion. This requires the initial value function estimator to be robust to a wide range of states. The loss function (17) only trains the estimator for a fixed $\mathbf{x}_0$. To ensure that the learned value function estimator is robust to variations in the initial state, we can train it such that it provides an accurate estimation for all of the states recorded. The measured cost-to-go can approximate the value function

$$\tilde{\mathbf{V}}(\mathbf{x}_k, t_k) = \mathbf{q}_N(\mathbf{x}_N) + \sum_{i=k}^{N-1} (\bar{\mathbf{q}}(\mathbf{x}_i) + \mathbf{r}(\mathbf{u}_i^*))\Delta t - \mathbf{V}_{\mathbf{x}}^T(\mathbf{x}_i, t_i, \mathbf{H}_i \mid \theta)\Sigma^T(\mathbf{x}_i)\Delta \mathbf{w}_i. \quad (18)$$

Using (18), we can learn the initial value function using

$$\mathbf{L}_{\mathbf{V}} = \sum_{j=1}^{M}\sum_{i=0}^{N} \|\mathbf{V}_{\mathbf{0}}(\mathbf{x}_i^j, t_i \mid \phi) - \tilde{\mathbf{V}}(\mathbf{x}_i^j, t_i)\|^2. \quad (19)$$

Thus, the loss function for the Hybrid FBSDE (HFBSDE) becomes

$$\mathbf{L}_{\text{HFBSDE}} = \mathbf{L}_{\text{FBSDE}} + \lambda \mathbf{L}_{\mathbf{V}} \quad (20)$$

with the terminal cost calculated using the state after the jump $\mathbf{x}_N^+$ and $\lambda \in \mathbb{R}_+$ adjusts the weighting between the two losses. This terminal cost encourages cyclic motion.

Algorithm 2 Penalty Function Update

Given: $\mathbf{k}$: Boundary steepness, δ: Change of boundary steepness, β: Update threshold, γ: Threshold change ratio, n_I: Iteration number, Δ: Boundary steepness change acceleration, η: Update interval, η': Max interval, Δ_δ: Threshold change acceleration;
if state trajectory not inside constraint boundary **then**
    if $(n_I \bmod \eta) = 0$ **then**
        Calculate $\sigma_{\mathbf{C}}^2$ = variance of the costs $\{\mathbf{C}_1, \cdots, \mathbf{C}_\eta\}$ for the past η episodes;
        if $\sigma_{\mathbf{C}} < \beta$ or $(n_I \bmod \eta') = 0$ **then**
            $\mathbf{k} = \mathbf{k} + \delta$, $\delta = \delta + \Delta_\delta$, $\beta = \gamma\beta$, $\gamma = \gamma + \Delta$;
    if $\delta < 0$ **then**
        $\delta = 0$;
    if $\gamma > 1$ **then**
        $\gamma = 1$;
else
    Update penalty function with new parameters.

4 Simulations

In this section, we show the performance of our control algorithm using a continuous nonlinear system: the cart-pole for a swing-up task; and a hybrid nonlinear system: a five-link biped for walking. The trained model was evaluated over 256 trials for all systems. For the cart-pole experiments, two LSTM layers with size 16 are used at each time step, followed by a dense layer with an output size corresponding to the state dimension. For fixed initial states, the initial value function estimation uses a trainable weight of size one. The initial hidden state and cell state for the LSTM layers are estimated using a trainable weight of size 16. For the five-link biped experiments, we used two LSTM layers with size 32, followed by a dense layer with size 10. The initial value function network has four layers with size [8, 16, 8, 1]. The initial hidden state network consists of four different networks for estimating the initial hidden state and the initial cell state of the two LSTM layers; they all consist of two dense layers that have an output size of 8.

4.1 Cart-pole Swing-Up Task I

The task for the cart-pole system is to swing the pole up and stabilize it at the top. The cart-pole dynamics are

$$(M+m)\ddot{x} - m\ell\sin\theta\dot{\theta}^2 + m\ell\cos\theta\ddot{\theta} = u \quad (21)$$

$$m\ell^2\ddot{\theta} + m\ell\cos\theta\ddot{x} + mg\ell\sin\theta = 0. \quad (22)$$

The cart position and pendulum angle are represented by x and θ, respectively. The pendulum angle is zero when it is pointed downwards. The state of the cart-pole system is $[x,\ \theta,\ \dot{x},\ \dot{\theta}]^T$. In our experiments, we set the cart mass

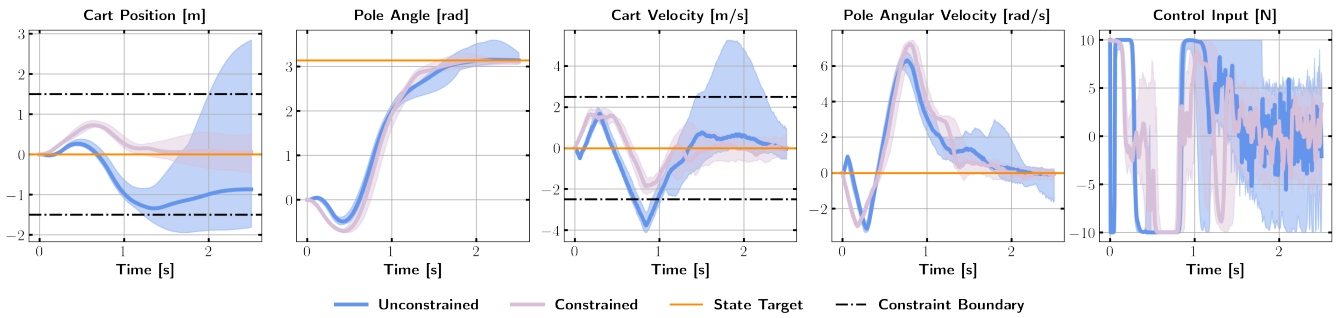

Fig. 3. This figure compares the performance between the state constrained (purple) and unconstrained (blue) controller. The demonstrated performance is from 256 trials. The darker blue and purple curves show a sampled trajectory from each setting. The light blue and purple regions show the state space each controller covers. The black dashed lines are the state constraints, and the orange lines give the state targets.

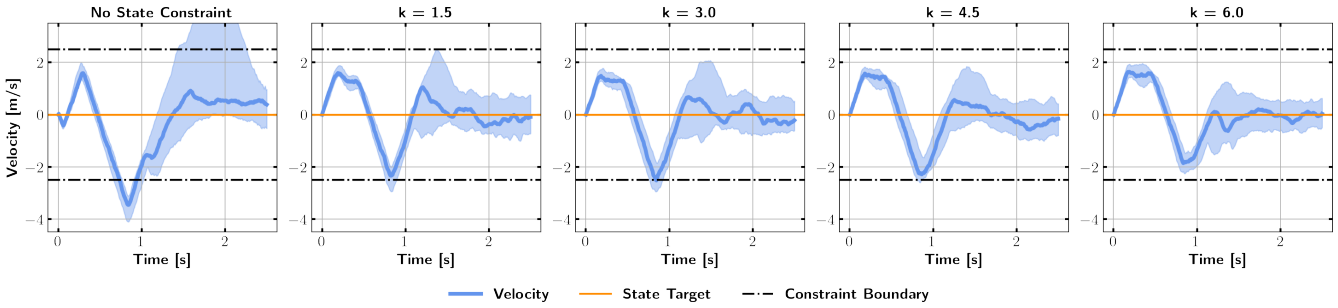

Fig. 4. The color scheme follows Figure 3. The leftmost figure shows the trajectory generated by the unconstrained controller violates the constraints. By applying the penalty function and following the adaptive update scheme, the velocity trajectory retreats inside the constraint boundaries and eventually satisfies the constraints.

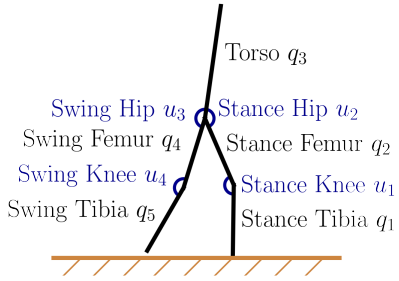

Fig. 5. Illustration of five-link biped model.

to $M = 1.0$kg, the pole mass to $m = 0.01$kg (point mass at the tip), the pole length to $\ell = 0.5$m, $\mathbf{x}_0 = \mathbf{0}_{4\times1}$, and the target state as $[0, \pi, 0, 0]^T$. The control is saturated at ±10N, the cart position is constrained between ±1.5m, and the cart velocity is constrained between ±2.5m. The time horizon is chosen to be 2.5 sec and the time step is $\Delta t = 1/110$ sec (this specific value of Δt is chosen randomly). We use the state cost function

$$\bar{\mathbf{q}}(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{Q}(\mathbf{x} - \bar{\mathbf{x}}) + \mathbf{p}(\mathbf{x}) \tag{23a}$$

$$\mathbf{q}_N(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{Q}_N(\mathbf{x} - \bar{\mathbf{x}}) \tag{23b}$$

where $\bar{\mathbf{x}}$ is the target state, $\mathbf{Q}$ is the cost weight matrix for the state, and $\mathbf{Q}_N$ is the terminal state cost matrix. We use the control cost defined in (9). The cost matrices are chosen as $\mathbf{Q} = \mathbf{Q}_N = \text{diag}([0.5,\ 1.0,\ 0.1,\ 0.1])$ and $\mathbf{R} = [0.1]$. $\mathbf{p}(\cdot)$ is the penalty function, in this example $\mathbf{p}(\cdot)$ is a logistics function-based penalty function (14), with $\mathbf{c}(\mathbf{x}) = [x, \dot{x}]^T$ and $\mathbf{b}_{\max} = -\mathbf{b}_{\min} = [1.5, 2.5]^T$. Fig. 3 shows the state trajectories generated by the trained controller. The state trajectory generated by the trained constrained DFBSDE controller satisfies the state constraints, while they are violated significantly under the trained unconstrained controller. The effect of the penalty function can be immediately seen in Fig. 4, even with a small k value. With $k = 1.5$, the part of the trajectory that lies outside the constraint boundaries is greatly reduced. After gradually increasing k to 6.0, following Algorithm 2, the entire cart velocity trajectory lies within the constraint boundaries. We also tested directly training with $k = 6.0$ without the adaptive update. The algorithm becomes numerically unstable after one epoch due to large gradients. This demonstrates that our method provides a stable training scheme.

4.2 Five-Link Biped Walking Task

The biped model of interest is shown in Fig. 5, derived from [3]. The leg on the ground is the stance leg, the

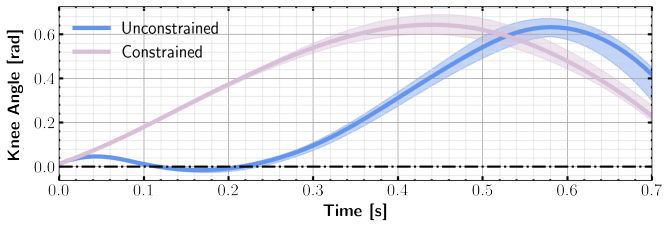

Fig. 6. Comparison of knee angles for constrained and unconstrained systems. To avoid hyperextending, the knee angle should be positive, i.e., above the black dashed line.

leg in the air is the swing leg, and the link representing the upper body is the torso. The biped state $\mathbf{x} \in \mathbb{R}^{10\times 1}$ consists of the link angles with respect to the horizontal plane $\mathbf{q} \in \mathbb{R}^{5\times 1}$ and the link angular velocities $\dot{\mathbf{q}} \in \mathbb{R}^{5\times 1}$, i.e., $\mathbf{x} = [\mathbf{q}^T,\ \dot{\mathbf{q}}^T]^T$. Note that the $\mathbf{q}$ denoted here is different from the state cost function $\mathbf{q}(\mathbf{x})$. The biped model has four actuated joints at the knee and hip joints. The control $\mathbf{u} \in \mathbb{R}^{5\times 1}$ consists of the applied torque at the actuated joints $\mathbf{u} = [0,\ u_1,\ u_2,\ u_3,\ u_4]^T$, where the 0 corresponds to the unactuated stance ankle. The output of the controller will only consist of $\tilde{\mathbf{u}} = [u_1,\ u_2,\ u_3,\ u_4]$, then a mapping matrix $T = [\mathbf{0},\ \mathbf{I}_{4\times 4}]$ is used to obtain $\mathbf{u} = T\tilde{\mathbf{u}}$. The parameters of the biped model can be found in [13]. The five-link biped dynamics is

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) = \mathbf{u}, \tag{24}$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{5\times 5}$ is the inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{5\times 5}$ the Coriolis matrix, and $G(\mathbf{q}) \in \mathbb{R}^{5\times 1}$ the gravitational terms. The dynamics is in the form of (1) after considering stochastic noise. A heel strike (HS) occurs when the biped comes into contact with the ground, which signals the termination of the current step, and initiation of the next step. During this transition, the swing and stance legs are swapped. We assume this transition is instantaneous, the biped is symmetric, and the new swing leg leaves the ground once the HS occurs (i.e., no double support phase). The joint indexing depends on the current swing and stance leg definition. Defining the joint angles right before HS as $\mathbf{q}^- \in \mathbb{R}^{5\times 1}$ and the joint angles right after HS as $\mathbf{q}^+ \in \mathbb{R}^{5\times 1}$, we have $\mathbf{q}^+ = \tilde{\mathbf{I}}_{5\times 5}\mathbf{q}^-$, where $\tilde{\mathbf{I}}_{5\times 5} \in \mathbb{R}^{5\times 5}$ is the anti-diagonal matrix. The HS creates an instantaneous change in angular velocity. Defining the angular velocity before HS as $\dot{\mathbf{q}}^- \in \mathbb{R}^{5\times 1}$ and the angular velocity after HS as $\dot{\mathbf{q}}^+ \in \mathbb{R}^{5\times 1}$, we have $\mathbf{x}^+ = \mathbf{f}_H(\mathbf{x}^-)$, with $\mathbf{f}_H : \mathbb{R}^{10\times 1} \to \mathbb{R}^{10\times 1}$ being the deterministic HS transition map [13].

To generate realistic walking motion, we need to avoid hyperextension in the knee. We transform this constraint into a variant of the ReLU-based penalty function in (15): $\mathbf{p}(\mathbf{x}) = \alpha\mathbf{ReLU}(-(q_4 - q_5))$ where α is a weighting coefficient for the penalty function, and q_4 and q_5 are defined in Fig. 5. A quadratic cost is applied to the control. We choose $\mathbf{R} = \text{diag}([2,\ 0.2,\ 0.2,\ 2])$. This choice of $\mathbf{R}$ implies that hip motors are more powerful than knee motors, which is true for many robots. The state cost $\mathbf{q}(\mathbf{x})$ is the weighted distance between the state $\mathbf{x}$ and a nominal terminal state $\bar{\mathbf{x}}$ plus the penalty

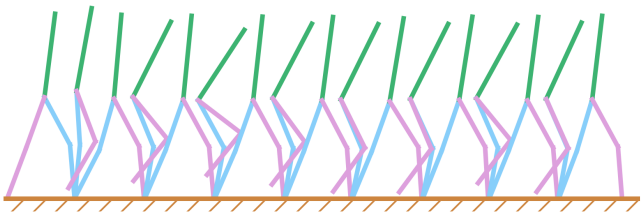

Fig. 7. In this figure, the motion generated by our proposed controller ensemble is shown. The torso is depicted in green, the swing leg in purple, and the stance leg in blue.

$$\bar{\mathbf{q}}(\mathbf{x}) = 1/2(\mathbf{x} - \bar{\mathbf{x}})^T\mathbf{Q}(\mathbf{x} - \bar{\mathbf{x}}) + \mathbf{p}(\mathbf{x}) \tag{25}$$

with $\mathbf{Q} = 10\mathbf{I}_{10\times 10}$. The terminal state cost has the same form as (25), the sole difference is replacing the state cost matrix with $\mathbf{Q}_N = 10\mathbf{Q}$. The target state is chosen to be

$$\bar{\mathbf{x}} = \Big[0.10, 0.50, -0.10, -0.35, -0.40, -1.50 \\ -0.50, 0.00, -0.55, -2.00\Big]^T. \tag{26}$$

A comparison between the motion generated by the constrained and unconstrained DFBSDE controller is given in Fig. 6. The unconstrained controller hyperextends the swing knee, while the constrained controller does not. Since the chosen penalty function could constrain the system directly, the k value doesn't need to be updated. In previous examples, the penalty functions are logistic function based. This works well when the constraint set is large since it creates a buffer between the interior of the constraint set and its boundary. However, the constraint set for locomotion tasks is relatively small, making ReLU functions a better candidate. For the five-link biped experiments, we use a time step of 0.01s.

Variations in the initial states make dissecting a multi-step walking problem into multiple single-step walking problems a challenging task. We train a robust controller to deal with this issue. We find it difficult for a single controller to handle all possible initial configurations. Thus, we use an ensemble of controllers, where each controller handles a range of initial configurations around a nominal state. For the duration of one footstep, only one controller is used, which is the controller in the ensemble that has the shortest distance between the initial state and its nominal state. The nominal state of a controller is known a priori. After training, the motion generated by this approach is shown in Fig. 7, where the ensemble size is three. The corresponding swing knee angle is shown in Fig. 8. It can be seen that no hyperextension occurred, which shows the effectiveness of enforcing the state constraints. On average, the initial range

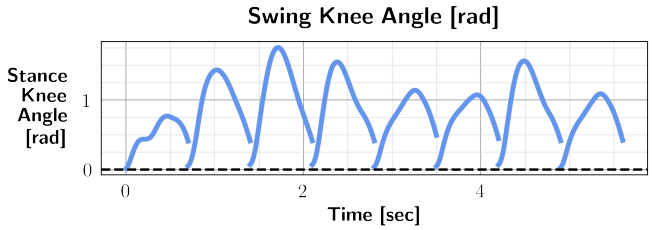

Fig. 8. This figure shows the swing knee angles for the motion depicted in Fig. 7.

each controller can handle spans 6.03 deg for the joint angles and 19.01 deg/sec for the joint velocities. Fig. 7 shows a tucking motion generated by the torso and the swing leg. This is due to our biped model's relatively small control authority over the torso. Without the forward angular momentum generated by the tucking motion, the torso gradually tilts back over multiple steps. Since there is no direct control over the torso, it is difficult for the DFBSDE controller to recover. We also compared the computation time of our proposed method for obtaining the control for one time step with trajectory optimization [13] (TO). As expected, our approach generates comparable results while being computationally more efficient. Solving for the control action requires 0.96s for a Hermite-Simpson TO approach, compared with 5.6ms for our proposed method.

5 Conclusion

In this paper, we solved SOC problems with state constraints using an LSTM-based DFBSDE framework which alleviates the curse of dimensionality and numerical integration issues. The state constraints are applied using an adaptive update scheme, significantly improving training stability. We also show how to adapt the algorithm to handle HD systems in addition to the continuous dynamics setting. The efficacy of our approach is demonstrated on a cart-pole system and a five-link biped which has hybrid dynamics.

References

[1] R.E. Bellman. Dynamic programming. *Dover, New York*, 2003.

[2] Y. Chen, Y. Shi, and B. Zhang. Optimal control via neural networks: A convex approach. In *Proceedings of International Conference on Learning Representations, New Orleans, LA*, May 2019.

[3] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E.R. Westervelt, C. Canudas-De-Wit, and J.W. Grizzle. RABBIT: a testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, 2003.

[4] L.G. Crespo and J. Sun. Stochastic optimal control via bellman's principle. *Automatica*, 39(12):2109–2114, 2003.

[5] B. Dai, P. Krishnamurthy, and F. Khorrami. Learning a better control barrier function. In *Proceedings of IEEE Conference on Decision and Control, Cancun, Mexico*, pages 945–950, December 2022.

[6] B. Dai, P. Krishnamurthy, A. Papanicolaou, and F. Khorrami. State constrained stochastic optimal control using LSTMs. In *Proceedings of the American Control Conference, New Orleans, LA*, pages 1294–1299, May 2021.

[7] B. Dai, V.R. Surabhi, P. Krishnamurthy, and F. Khorrami. Learning locomotion controllers for walking using deep FBSDE. *CoRR*, abs/2107.07931, 2021.

[8] M. Demetriou and E. Bakolas. Navigating over 3d environments while minimizing cumulative exposure to hazardous fields. *Automatica*, 115:108859, 2020.

[9] M. Diehl, H.J. Ferreau, and N. Haverbeke. Efficient numerical methods for nonlinear mpc and moving horizon estimation. *Nonlinear model predictive control*, pages 391–417, 2009.

[10] L. Hewing and M.N. Zeilinger. Scenario-based probabilistic reachable sets for recursively feasible stochastic model predictive control. *IEEE Control Systems Letters*, 4(2):450–455, 2019.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[12] R. Sinnet J. Grizzle, C. Chevallereau and A. Ames. Models, feedback control, and open problems of 3d bipedal robotic walking. *Automatica*, 50(8):1955–1988, 2014.

[13] M. Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[14] N. Kumaresan and P. Balasubramaniam. Optimal control for stochastic nonlinear singular system using neural networks. *Computers & Mathematics with Application*, 56(9):2145–2154, 2008.

[15] N Kumaresan and P Balasubramaniam. Optimal control for stochastic linear quadratic singular system using neural networks. *Journal of Process Control*, 19(3):482–488, 2009.

[16] M. Lorenzen, F. Dabbene, R. Tempo, and F. Allgöwer. Stochastic mpc with offline uncertainty sampling. *Automatica*, 81:176–183, 2017.

[17] M.N. Mistry, J. Buchli, and S. Schaal. Inverse dynamics control of floating base systems using orthogonal decomposition. In *Proceedings of the IEEE International Conference on Robotics and Automation, Anchorage, AK*, pages 3406–3412, May 2010.

[18] M. Neunert, M. Stäuble, M. Giftthaler, C.D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, 2018.

[19] M. Pereira, Z. Wang, I. Exarchos, and E.A. Theodorou. Safe optimal control using stochastic barrier functions and deep forward-backward sdes. In *Proceedings of Conference on Robot Learning, Cambridge, MA*, volume 155, pages 1783–1801, November 2020.

[20] L. Di Persio and M. Garbelli. Deep learning and mean-field games: A stochastic optimal control perspective. *Symmetry*, 13(1):14, 2021.

[21] J. Viereck and L. Righetti. Learning a centroidal motion planner for legged locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation, Xi'an, China*, pages 4905–4911, May 2021.

[22] Z. Wang, M.A. Pereira, and E.A. Theodorou. Learning deep stochastic optimal control policies using forward-backward SDEs. In *Proceedings of Robotics: Science and Systems XV, Freiburg im Breisgau, Germany*, June 2019.