DEEP NONLINEAR STOCHASTIC OPTIMAL CONTROL FOR SYSTEMS WITH MULTIPLICATIVE UNCERTAINTIES

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a deep recurrent neural network architecture to solve a class of stochastic optimal control problems described by fully nonlinear Hamilton Jacobi Bellman partial differential equations. Such PDEs arise when one considers stochastic dynamics characterized by uncertainties that are additive and control multiplicative. Stochastic models with the aforementioned characteristics have been used in computational neuroscience, biology, finance and aerospace systems and provide a more accurate representation of actuation than models with additive uncertainty. Previous literature has established the inadequacy of the linear HJB theory and instead rely on a non-linear Feynman-Kac lemma resulting in a second order forward-backward stochastic differential equations representation. However, the proposed solutions that use this representation suffer from compounding errors and computational complexity leading to lack of scalability. In this paper, we propose a deep learning based algorithm that leverages the second order Forward-Backward SDE representation and LSTM based recurrent neural networks to not only solve such Stochastic Optimal Control problems but also overcome the problems faced by previous approaches and scales well to high dimensional systems. The resulting control algorithm is tested on non-linear systems in robotics and biomechanics to demonstrate feasibility and out-performance against previous methods.

1 INTRODUCTION

Stochastic optimal control is the center of decision making under uncertainty with a history and extensive prior work both in terms of theory as well as algorithms (Stengel, 1994; Fleming & Soner, 2006). One of the most celebrated formulations of stochastic control is for linear dynamics and additive noise. This is the so-called Linear Quadratic Gaussian (LQG) case (Stengel, 1994). For stochastic systems that are nonlinear in the state and affine in control, stochastic control results in the Hamilton-Jacobi-Bellman (HJB) equation that is a backward nonlinear Partial Differential Equation (PDE). Solving the HJB equations for high dimensional systems is in general a challenging task and suffers from the curse of dimensionality.

Different algorithms have been derived to address stochastic control problems and solve the HJB equation. The algorithms could be classified into algorithms that rely on linearization and algorithms that rely on sampling. Linearization-based algorithms rely on first order Taylor's approximation of dynamics (iterative LQG or iLQG) or quadratic approximation of dynamics (Stochastic Differential Dynamic Programming), and quadratic approximation of the cost function (Todorov & Li, 2005; Theodorou et al., 2010b). Application of the aforementioned algorithms is not straightforward and requires very small time discretization or special linearization schemes especially for the cases of control and/or state dependent noise. It is worth also mentioning that the convergence properties of these algorithms has not been investigated and remains an open question. Sampling-based methods include the Markov-Chain Monte Carlo (MCMC) approximation of the HJB equation (Kushner & Dupuis, 1992; Huynh et al., 2016). MCMC-based algorithms rely on backward propagating the value function on a pre-specified grid. Recently researchers have incorporated tensor-train decomposition techniques to scale these methods (Gorodetsky et al., 2015). However, these techniques have been applied to special classes of systems and stochastic control problem formulations and have demonstrated limited applicability so far.

Alternative sampling-based methodologies rely on the probabilistic representation of backward PDEs and generalization of the so-called linear Feynman-Kac lemma (Karatzas & Shreve, 1991) to its nonlinear version (Pardoux & Rascanu, 2014). Application of the linear Feynman-Kac lemma requires the exponential transformation of the value function and certain assumptions related to control authority and the variance of the noise. Stochastic control then is computed using forward sampling of stochastic differential equations (Kappen, 2005; Todorov, 2007; 2009; Theodorou et al., 2010a). The nonlinear version of the Feynman-Kac lemma overcomes the aforementioned limitations. However it requires a more sophisticated numerical scheme than just forward sampling, which relies on the theory of Forward-Backward Stochastic Differential Equations (FBSDEs) and their connection to backward PDEs. The FBSDE formulation is very general and has been utilized in many problem formulations such as L_2 and L_1 stochastic control (Exarchos & Theodorou, 2018; 2016; Exarchos et al., 2018), min-max and risk-sensitive control (Exarchos et al., 2019) and control of systems with control multiplicative noise (Bakshi et al., 2017b). The major limitation of the algorithms that rely on FBSDEs, is the compounding of errors from Least Squares approximation used at every timestep of the Backward Stochastic Differential Equation (BSDE).

Recent efforts in the area of Deep Learning for solving non-linear PDEs have demonstrated encouraging results in terms of scalability and numerical efficiency. A Deep Learning-based algorithm was introduced by Han et al. (2018) to approximate the solution of non-linear parabolic PDEs through their connection to first order FBSDEs. Their framework relies on propagation of dynamics driven by white noise, which proves successful for simple linear systems, but suffers from insufficient exploration for non-linear dynamics. Thus, their approach is not directly applicable to many Stochastic Optimal Control (SOC) problems. One solution to this problem was proposed by Exarchos & Theodorou (2018) through the application of Girsanov's Theorem (Shreve, 2004, Chapter 5), leading to modification of the drift terms in the FBSDE to allow for sufficient exploration through controlled forward dynamics. This procedure is called importance sampling as trajectories are sampled from a distribution different from the original distribution of uncontrolled trajectories.

In this paper we develop a novel Deep Neural Network (DNN) architecture for PDEs that are fully nonlinear. Fully nonlinear PDEs appear in stochastic control problems in which noise is additive and control multiplicative. Such problem formulations are important in biomechanics and computational neuroscience, autonomous systems, and finance (Todorov, 2005; Mitrovic et al., 2010; Primbs, 2007; McLane, 1971; Phillis, 1985). Prior work, on stochastic control of such systems considers linear dynamics and quadratic cost functions. Attempts to generalize these linear methods to the case of stochastic nonlinear dynamics with control multiplicative noise are only preliminary and require special treatment in terms of methods to forward propagate and linearize the underlying stochastic dynamics (Torre & Theodorou., 2015).

Given the prior work in the core areas of stochastic control and deep learning, below we summarize the contributions of our work:

- We design a novel DNN architecture to represent and solve second-order FBSDEs (2FBSDEs). The neural network architecture consists of Fully Connected (FC) feed-forward and Long-Short Term Memory (LSTM) recurrent layers. The resulting Deep 2FBSDE network can be used to solve fully nonlinear PDEs for high dimensional nonlinear systems.

- We demonstrate the applicability and correctness of the proposed algorithm in four examples ranging from traditionally used non-linear systems in Control Theory to Robotics and Biomechanics. The proposed algorithm recovers analytical controls in the case of linear dynamics while it is also able to successfully control nonlinear dynamics with control-multiplicative and additive sources of stochasticity. Our simulations show the robustness of the Deep 2FBSDE algorithm and prove the importance of considering the nature of the stochastic disturbances in the problem formulation as well as in neural network representation.

The rest of the paper is organized as follows: in Section 2 we first introduce notation, some preliminaries and discuss the problem formulation. Next, in Section 3 we provide the 2FBSDE formulation. The Deep 2FBSDE algorithm is introduced in Section 4. Then we demonstrate and discuss results from our simulation experiments in Section 5. Finally we conclude the paper in Section 6 with discussion and future directions.

2 STOCHASTIC CONTROL

In this section we provide notation and definitions essential for the development of our proposed algorithm and then present the problem formulation. Note that hereon the bold faced notation will be abused for representing both vectors and matrices, while non-bold faced will be used for scalars.

2.1 PRELIMINARIES

We first introduce stochastic dynamical systems which have a *drift* component (i.e. the non-stochastic component of the dynamics) that is a nonlinear function of the state but affine with respect to the controls. The *diffusion* component (i.e. the stochastic component) is comprised of nonlinear functions of the state and affine control multiplicative matrix coefficients. Let $\boldsymbol{x} \in \mathbb{R}^{n_x}$ be the vector of state variables, and $\boldsymbol{u} \in \mathbb{R}^{n_u}$ be the vector of control variables taking values in the set of all admissible controls $\mathcal{U}([0,T])$, for a fixed finite time horizon $T \in [0,\infty)$. Let $\big([\boldsymbol{w}(t)^{\mathrm{T}}\ v(t)^{\mathrm{T}}]^{\mathrm{T}}\big)_{t\in[0,T]}$ be a Brownian motion in $\mathbb{R}^{n_w+1}$ on a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\in[0,T]}, \mathbb{Q})$, where $\boldsymbol{w}(t) \in \mathbb{R}^{n_w}$, $v(t) \in \mathbb{R}$ and the components of $\boldsymbol{w}(t)$ are mutually independent one dimensional standard Brownian motions. We now assume that functions $\boldsymbol{f} : [0,T] \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$, $\mathbf{G} : [0,T] \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x \times n_u}$, $\boldsymbol{\Sigma} : [0,T] \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x \times n_w}$ and $\sigma \in \mathbb{R}^+$ satisfy certain Lipschitz and growth conditions (refer to Assumption 1 in the supplementary material for more details).

Given this assumption, it is known that for every initial condition $\xi \in \mathbb{R}^{n_x}$, there exists a unique solution $\big(\boldsymbol{x}(t)\big)_{t\in[0,T]}$ to the Forward Stochastic Differential Equation (FSDE),

$$\begin{cases} \mathrm{d}\boldsymbol{x}(t) &= \underbrace{\big(\boldsymbol{f}(t,\boldsymbol{x}(t)) + \mathbf{G}(t,\boldsymbol{x}(t))\boldsymbol{u}(t)\big)}_{\textit{drift}}\,\mathrm{d}t + \underbrace{C\big(t,\boldsymbol{x}(t),\boldsymbol{u}(t)\big)}_{\textit{diffusion}}\begin{bmatrix}\mathrm{d}\boldsymbol{w}(t)\\ \mathrm{d}v(t)\end{bmatrix} \\ \boldsymbol{x}(0) &= \xi, \end{cases} \tag{1}$$

where, $C : [0,T] \times \mathbb{R}^{n_x \times n_u} \to \mathbb{R}^{n_x}$ is defined as $C(t,\boldsymbol{x}(t),\boldsymbol{u}(t)) = \big[\sigma\mathbf{G}(t,\boldsymbol{x}(t))\boldsymbol{u}(t),\ \boldsymbol{\Sigma}(t,\boldsymbol{x}(t))\big]^{\mathrm{T}}$.

2.2 PROBLEM STATEMENT AND HJB PDE

For the controlled stochastic dynamical system above, we formulate the SOC problem as minimizing the following expected cost

$$J\big(t,\boldsymbol{x}(t);\boldsymbol{u}(t)\big) = \mathbb{E}_{\mathbb{Q}}\left[\int_t^T \ell\big(\boldsymbol{x}(s),\boldsymbol{u}(s)\big)\,\mathrm{d}s + \phi\big(\boldsymbol{x}(T)\big)\right] \tag{2}$$

where $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^+$ is the running cost and $C^{1,2} \ni \phi : \mathbb{R}^{n_x} \to \mathbb{R}^+$ is the terminal state cost. Here $C^{1,2}$ denotes functions differentiable and with continuous derivatives up to the second order. The expectation is taken with respect to the probability measure $\mathbb{Q}$ over the space of trajectories induced by the controlled stochastic dynamics. We can define the value function as

$$\begin{cases} V\big(t,\boldsymbol{x}(t)\big) &= \inf\limits_{\boldsymbol{u}(t)\in\mathcal{U}([t,T])} J\big(t,\boldsymbol{x}(t);\boldsymbol{u}(t)\big) \\ V\big(T,\boldsymbol{x}(T)\big) &= \phi\big(\boldsymbol{x}(T)\big). \end{cases} \tag{3}$$

Under the condition that the value function is in $C^{1,2}$ and considering a non-linear state cost and quadratic control cost such that the running cost, $\ell(\boldsymbol{x},\boldsymbol{u}) = q(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}R\boldsymbol{u}$, wherein the matrix R (control cost coefficients) is a positive definite matrix of size $n_u \times n_u$, we can follow the standard stochastic optimal control derivation to obtain the HJB PDE

$$\begin{cases} V_t + q + V_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{f} - \frac{1}{2}V_{\boldsymbol{x}}^{\mathrm{T}}\mathbf{G}\hat{R}^{-1}\mathbf{G}^{\mathrm{T}}V_{\boldsymbol{x}} + \frac{1}{2}\operatorname{tr}(V_{\boldsymbol{x}\boldsymbol{x}}\mathbf{C}\mathbf{C}^{\mathrm{T}}) = 0 \\ V(T,\boldsymbol{x}(T)) = \phi(\boldsymbol{x}(T)), \end{cases} \tag{4}$$

with the optimal control of the form,

$$\boldsymbol{u}^*(t,\boldsymbol{x}) = -\hat{R}^{-1}\mathbf{G}(t,\boldsymbol{x})^{\mathrm{T}}V_{\boldsymbol{x}}(t,\boldsymbol{x}). \tag{5}$$

where, $\hat{R} \triangleq (R + \sigma^2 G^{\mathrm{T}}V_{\boldsymbol{x}\boldsymbol{x}}G)$. The derivation for both equations 4 and 5 can be found in the supplementary materials (Section 2).

3 A FBSDE SOLUTION TO THE HJB PDE

The theory of BSDEs establishes a connection between solution of a parabolic PDE and a set of FBSDEs (El Karoui et al., 1997). This connection has been used to solve the HJB PDE in the context of SOC problems. Exarchos & Theodorou (2018) solved the HJB PDE in the absence of control multiplicative noise $\mathrm{d}v$ with a set of first order FBSDEs. Bakshi et al. (2017b) utilized the second order FBSDEs or 2FBSDEs to solve the fully nonlinear HJB PDE in the presence of control multiplicative noise, but did not consider any control in the drift of the FSDE. Lack of control leads to insufficient exploration and for highly nonlinear systems renders it impossible to find an optimal solution to complete the task. In light of this, we introduce a new set of 2FBSDEs

$$\begin{cases} \mathrm{d}\tilde{\mathbf{X}} &= \boldsymbol{f}(t,\tilde{\mathbf{X}})\mathrm{d}t + \underbrace{\mathbf{G}(t,\tilde{\mathbf{X}})(\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}t + \sigma\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}v)}_{control\ for\ exploration\ in\ FSDE} + \boldsymbol{\Sigma}(t,\tilde{\mathbf{X}})\mathrm{d}\tilde{\mathbf{W}}(t) \\ \mathrm{d}\tilde{Y} &= -h(t,\tilde{\mathbf{X}},\tilde{Y},\tilde{\mathbf{Z}},\boldsymbol{\Gamma})\mathrm{d}t + \underbrace{\tilde{\mathbf{Z}}^{\mathrm{T}}\mathbf{G}(t,\tilde{\mathbf{X}})(\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}t + \sigma\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}v)}_{compensation\ for\ control\ in\ BSDE} + \tilde{\mathbf{Z}}^{\mathrm{T}}\boldsymbol{\Sigma}(t,\tilde{\mathbf{X}})\mathrm{d}\tilde{\mathbf{W}}(t) \\ \mathrm{d}\tilde{\mathbf{Z}} &= \mathbf{A}(t,\tilde{\mathbf{X}})\mathrm{d}t + \underbrace{\boldsymbol{\Gamma}(t,\tilde{\mathbf{X}})\mathbf{G}(t,\tilde{\mathbf{X}})(\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}t + \sigma\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}v)}_{compensation\ for\ control\ in\ BSDE} + \boldsymbol{\Gamma}(t,\tilde{\mathbf{X}})\boldsymbol{\Sigma}(t,\tilde{\mathbf{X}})\mathrm{d}\tilde{\mathbf{W}}(t), \end{cases} \tag{6}$$

where

$$\begin{cases} h(t,\tilde{\mathbf{X}},\tilde{Y},\tilde{\mathbf{Z}},\boldsymbol{\Gamma}) &= q(t,\tilde{\mathbf{X}}) - \frac{1}{2}\big(\tilde{\mathbf{Z}}^{\mathrm{T}}\mathbf{G}\hat{R}^{-1}\mathbf{G}^{\mathrm{T}}\tilde{\mathbf{Z}}\big)(t,\tilde{\mathbf{X}}) \\ \big(\tilde{Y},\tilde{\mathbf{Z}},\boldsymbol{\Gamma},\mathbf{A}\big) &= \big(V, V_{\boldsymbol{x}}, V_{\boldsymbol{xx}}, \mathcal{H}(V_{\boldsymbol{x}})\big). \end{cases} \tag{7}$$

with $\boldsymbol{u}^*(t,\tilde{\mathbf{X}}) = -(R+\sigma^2\mathbf{G}^{\mathrm{T}}\boldsymbol{\Gamma}\mathbf{G})^{-1}\mathbf{G}^{\mathrm{T}}\tilde{\mathbf{Z}}$, and with initial and terminal conditions of $\tilde{\mathbf{X}}(0) = \xi$, $\tilde{Y}(T) = \phi(\tilde{\mathbf{X}}(T))$ and $\tilde{Z}(T) = \phi_{\boldsymbol{x}}(\tilde{\mathbf{X}}(T))$. Additionally, the HJB operator $\mathcal{H}$ is defined as

$$\mathcal{H}(\cdot) \triangleq \partial_t(\cdot) + \partial_{\boldsymbol{x}}(\cdot)^{\mathrm{T}}\boldsymbol{f} + \frac{1}{2}\mathrm{tr}(\partial_{\boldsymbol{xx}}(\cdot)\mathbf{C}\mathbf{C}^{\mathrm{T}}). \tag{8}$$

The equivalence of the backward process $\tilde{Y}$ and the value function V of our original problem (equation 3) is proven in the supplementary materials. Note that control appears in the forward process $\tilde{\mathbf{X}}$ as well as in both backward processes $\tilde{Y}$ and $\tilde{\mathbf{Z}}$. The inclusion of control allows guidance of the exploration needed for solving the 2FBSDEs and hence the SOC problem.

Using equation 6, we can forward sample the FSDE. The second-order BSDEs (2BSDEs), on the other hand, need to satisfy a terminal condition and therefore have to be propagated backwards in time. However, since the stochasticity that enters the dynamics evolves forward in time, only the conditional expectations of the 2BSDEs can be back-propagated (since $\mathbb{E}[\tilde{Y}|\mathcal{F}_t]$ and $\mathbb{E}[\tilde{Z}|\mathcal{F}_t]$ are $\mathcal{F}_t$-measurable not $\tilde{Y}$ and $\tilde{Z}$. For more details please refer to Shreve (2004, Chapter 2).) Bakshi et al. (2017a) back-propagate approximate conditional expectations of the two processes computed using regression. This method however, suffers from compounding errors introduced by least squares estimation at every time step. In contrast a Deep Learning (DL) based approach, first introduced by Han et al. (2018), mitigates this problem by using the terminal condition as the prediction target for a forward propagated BSDE. This is enabled by randomly initializing the initial condition and treating it as a trainable parameter of a self-supervised learning problem. In addition, the approximation errors at each time step are compensated for by backpropagation during training of the DNN. This allowed using FBSDEs to solve the HJB PDE for high-dimensional linear systems. However, similar to Bakshi et al. (2017b) this scheme also lacked sufficient exploration and relied purely on noise (uncontrolled dynamics) to guide learning. A more recent approach, the Deep FBSDE controller (Pereira et al., 2019), utilizes a controlled forward process for guiding exploration and has been successfully applied to systems in simulation that correspond to first order FBSDEs. Extending this work, we propose a new framework for solving SOC problems of systems with control multiplicative noise, for which the value function solutions correspond to 2FBSDEs.

4 DEEP 2FBSDE CONTROLLER

In this section, we introduce a new deep network architecture called the *Deep 2FBSDE Controller* and present a training algorithm to solve SOC problems with control multiplicative noise.

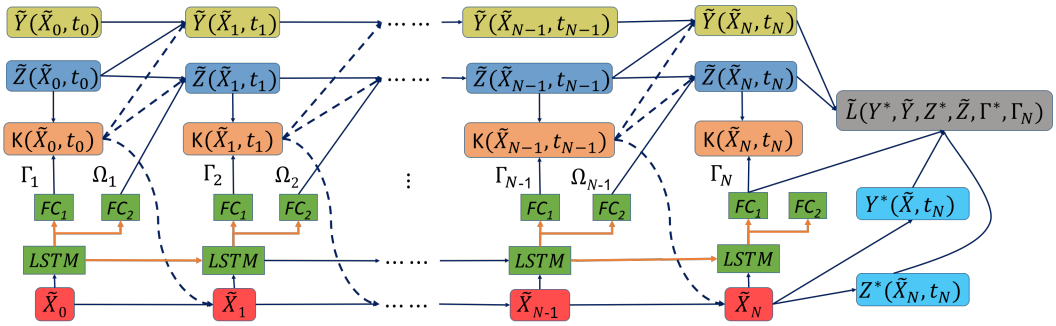

Figure 1: **Deep 2FBSDE neural network architecture.** The blocks $FC_{1,2}$ are fully connected layers with linear activations while the $LSTM$ block represents recurrent layers of stacked LSTM cells with standard nonlinear activations. These blocks alongwith $\tilde{Y}(\tilde{X}_0, t_0)$ and $\tilde{Z}(\tilde{X}_0, t_0)$ comprise the trainable parameters of the network which are shared temporally.

Algorithm 1: Finite Time Horizon Deep 2FBSDE Controller

Given: $\xi, \boldsymbol{f}, \mathbf{G}, \boldsymbol{\Sigma}, \sigma$: Initial state, drift, actuation dynamics, diffusion and noise std. deviation; ϕ, q, R: Cost function parameters; N: Task horizon, K: Number of iterations, M: Batch size; Δt: Time discretization; λ: weight-decay parameter for regularization;

Parameters: $\tilde{Y}_0 = V(\tilde{\mathbf{X}}_0; \psi)$: Value function at $t = 0$; $\tilde{\mathbf{Z}}_0 = V_{\tilde{\boldsymbol{x}}}(\tilde{\mathbf{X}}; \zeta)$: Gradient of value function at $t = 0$; θ: Weights and biases of all fully-connected and LSTM layers;

Initialize neural network parameters and states: $\theta^0, \psi^0, \zeta^0, \tilde{\mathbf{X}}_0 = \xi$

for $k = 1$ **to** K **do**
  for $i = 1$ **to** M **do**
    for $t = 0$ **to** $N - 1$ **do**
      Compute $\mathbf{G}$ matrix: $\mathbf{G}_t^i = \mathbf{G}\big(\tilde{\mathbf{X}}_t^i, t\big)$;
      Network prediction: $\boldsymbol{\Gamma}_t^i, \boldsymbol{\Omega}_t^i = f_{FC_1}\big(f_{LSTM}(\tilde{\mathbf{X}}; \theta^{k-1}); \big),\ f_{FC_2}\big(f_{LSTM}(\tilde{\mathbf{X}}; \theta^{k-1})\big)$
      Compute optimal control: $\boldsymbol{u}_t^{i*} = -(R + \sigma^2 \mathbf{G}_t^{i\mathrm{T}} \boldsymbol{\Gamma}_t^i \mathbf{G}_t^i)^{-1} \mathbf{G}_t^{i\mathrm{T}} \tilde{\mathbf{Z}}_t^i$;
      Sample Brownian noise: $\Delta \tilde{\mathbf{W}}_t^i \sim \mathcal{N}(0, I\Delta t); \Delta \tilde{v}_t^i \sim \mathcal{N}(0, \Delta t)$
      Compute control for exploration: $\mathbf{K}_t^i = \mathbf{K}(t, \tilde{\mathbf{X}}_t^i) = \mathbf{G}_t^i \left(\boldsymbol{u}_t^{i*} \Delta t + \sigma \boldsymbol{u}_t^{i*} \Delta \tilde{v}_t^i\right)$
      Forward propagate Stochastic Differential Equations (SDEs):
      $\tilde{\mathbf{X}}_{t+1}^i, \tilde{Y}_{t+1}^i, \tilde{\mathbf{Z}}_{t+1}^i = f_{FBSDE}(\tilde{\mathbf{X}}_t^i, \tilde{Y}_t^i, \tilde{\mathbf{Z}}_t^i, \mathbf{K}_t^i)$
    end for
  end for
  Compute mini-batch loss: $\tilde{\mathcal{L}} = f_{Loss}(\tilde{\mathbf{X}}_N, \tilde{Y}_N, \tilde{\mathbf{Z}}_N, \theta^{k-1})$
  Gradient update: $\theta^k, \psi^k, \zeta^k \leftarrow \text{Adam.step}(\tilde{\mathcal{L}}, \theta^{k-1}, \psi^{k-1}, \zeta^{k-1})$
end for
return $\theta^K, \psi^K, \zeta^K$

Time discretization: In order to approximate numerical solutions of the SDEs we choose the explicit Euler-Maruyama time-discretization scheme (Kloeden & Platen, 2013, Chapter 9) similar to (Pereira et al., 2019; Han et al., 2018). Here we overload t as both the continuous-time variable and discrete time index and discretize the task horizon $0 < t < T$ as $t = \{0, 1, \cdots, N\}$, where $T = N\Delta t$. This is also used to discretize all variables as step functions if their discrete time index t lies in the interval $[t\Delta t,\ (t + 1)\Delta t)$. We use subscript to denote the discretized variables.

Network architecture: Inspired by the LSTM-based recurrent neural network architecture introduced by Pereira et al. (2019) for solving first order FBSDEs, we propose the network in fig.1 adapted to the stochasticities in 2FBSDEs given by equation 6. Instead of predicting the gradient of the value function $V_{\boldsymbol{x}}$ at every time step, the output of the LSTM is used to predict the Hessian of value function $\boldsymbol{\Gamma}_t = V_{\boldsymbol{xx}}(t)$ and $\boldsymbol{\Omega}_t = \partial_t(V_{\boldsymbol{x}}) + \frac{1}{2}\mathrm{tr}(\partial_{\boldsymbol{xx}}(V_{\boldsymbol{x}})\mathbf{C}\mathbf{C}^{\mathrm{T}})$ using two separate FC layers with linear activations. Notice that $\partial_{\boldsymbol{xx}}(V_{\boldsymbol{x}})$ is a rank 3 tensor and using neural networks to predict this term explicitly would render this method unscalable. We however, bypass this problem by instead predicting the trace

of the tensor product which is a vector allowing linear growth in output size with state dimensionality. Of these, $\mathbf{\Gamma}_t$ is used to compute the control term $\mathbf{K}(t, \tilde{\mathbf{X}}) = \mathbf{G}(t, \tilde{\mathbf{X}})(\boldsymbol{u}^*(t, \tilde{\mathbf{X}})\mathrm{d}t + \sigma \boldsymbol{u}^*(t, \tilde{\mathbf{X}})\mathrm{d}v)$ for exploration. This in turn is used to propagate the stochastic dynamics $\tilde{\mathbf{X}}(t)$ and compensate for added control in the two forward-propagated backward processes $\tilde{\mathbf{Z}}(t)$ and $\tilde{Y}(t)$. Both $\mathbf{\Omega}_t$ and $\mathbf{\Gamma}_t$ are used to propagate $\tilde{\mathbf{Z}}(t)$ which is then used to propagate $\tilde{Y}(t)$. This is repeated until the end of the time horizon as shown in fig.1. Finally, the predicted values of $\tilde{Y}(t)$, $\tilde{\mathbf{Z}}(t)$ and $\mathbf{\Gamma}_t$ are compared with their targets computed using $\tilde{\mathbf{X}}(t)$ at the end of the horizon, to compute a loss function for backpropagation.

Algorithm: Algorithm. 1 details the training procedure of the deep 2FBSDE Stochastic Optimal Controller. Note that superscripts indicate batch index for variables and iteration number for trainable parameters. The value function $\tilde{Y}_0$ and its gradient $\tilde{\mathbf{Z}}_0$ (at time index $t = 0$), are randomly initialized and trainable. Functions f_{LSTM}, f_{FC_1} and f_{FC_2} denote the forward propagation equations of standard LSTM layers (Hochreiter & Schmidhuber, 1997) and FC layers with tanh and linear activations respectively, and f_{FBSDE} represents a discretized version of equation 6 using the explicit Euler-Maruyama time discretization scheme. The loss function ($\tilde{\mathcal{L}}$) is computed using the value function, its gradient and hessian at the final time index as well as an L_2 regularization term. A detailed justification of each loss term is included in the supplementary material. The network is trained using any variant of Stochastic Gradient Descent (SGD) such as Adam (Kingma & Ba, 2014) until convergence. The algorithm returns a trained network capable of computing an optimal feedback control at every timestep starting from the given initial state.

5 Simulation Results

In this section we demonstrate the capability of the Deep 2FBSDE Controller on systems ranging from traditional non-linear systems in Control Theory to Robotics and Biomechanics. To begin, we first compared the solution of the Deep 2FBSDE controller to the analytical solution for a scalar linear system with additive and control multiplicative stochasticities which validates the correctness of our proposed algorithm. We then considered the task of cart-pole swing-up from Control Theory followed by a quadcopter in simulation for a reaching task. Finally, we tested on a 2-link 6-muscle human arm model for a planar reaching task. The results were compared to the Deep FBSDE controller in Pereira et al. (2019) wherein the effect of control multiplicative noise was ignored by only considering first order FBSDEs. We also compared against the iLQG controller in Li & Todorov (2007).

All the comparison plots contain statistics gathered over 128 test trials. For the 2FBSDE and FBSDE controller, we used time discretization $\Delta t = 0.004$ seconds for the linear system problem, $\Delta t = 0.02$ seconds for the cart-pole and quadcopter simulations and $\Delta t = 0.01$ seconds for the human arm simulation. For the iLQG simulations, $\Delta t = 0.01$ seconds was used for cartpole and quadcopter and $\Delta t = 0.001$ seconds was used for the human arm to avoid numerical instability. In all plots, the solid line represents the mean trajectory, and the shaded region represents the 68% confidence region ($\mu \pm 1\sigma$). During comparison, we use blue for 2FBSDE, green for first order FBSDE and analytical solution, and red for iLQG. We use **black dotted line to denote target state**.

Linear System: We consider a scalar linear time-invariant system

$$\mathrm{d}x(t) = Ax(t)\mathrm{d}t + Bu(t)\mathrm{d}t + \sigma Bu(t)\mathrm{d}v(t) + C\mathrm{d}w(t), \tag{9}$$

along with quadratic running state cost and terminal state cost

$$q\big(x(t)\big) = \frac{1}{2}Qx(t)^2, \quad \phi\big(x(T)\big) = \frac{1}{2}Q_T x(t)^2. \tag{10}$$

The dynamics and cost function parameters are set as $A = 0.2, B = 1.0, C = 0.1, \sigma = 0.5, Q = 0, Q_T = 80, R = 2, x_0 = 1.0$. The task is to drive the state to 0. Note that the problem is different than the one for LQG due to the presence of control multiplicative noise (i.e. $v(t) \neq 0$). Let us assume that the value function (equation 4) has the form $V\big(t, x(t)\big) = \frac{1}{2}P(t)x^2(t) + S(t)x(t) + c(t)$, where $S(T) = 0, P(T) = Q_T, c(t) = \int_0^T \frac{1}{2}c^2 P(s)\mathrm{d}s$. The values of P and Q are obtained by solving corresponding Riccatti equations, using ODE solvers, that can be derived similarly to the LQG case

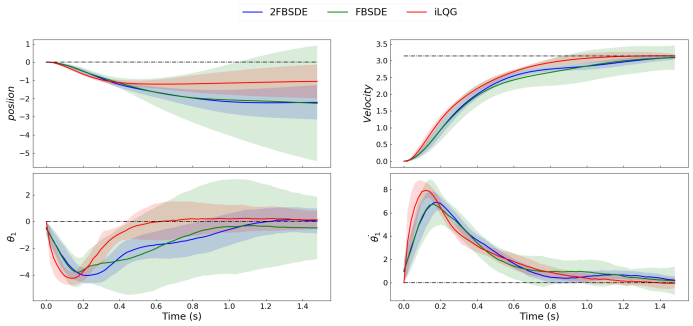

Figure 3: Comparison of 2FBSDE, FBSDE and iLQG on the cartpole

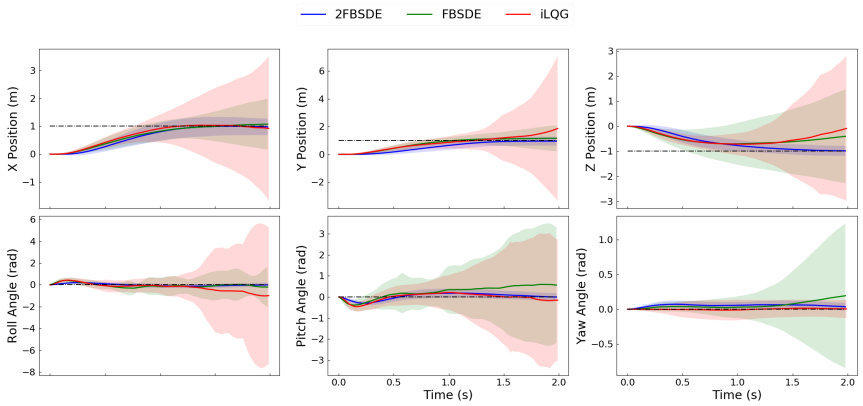

Figure 4: Comparison of 2FBSDE, FBSDE and iLQG on the quadcopter.

in Stengel (1994, Chapter 5) and are used to compute the optimal control equation 5 at every timestep. The solution obtained from the 2FBSDE controller is compared against the analytical solution in fig. 2. The resulting trajectories have matching mean and comparable variance, which verifies the effectiveness of the controller on linear systems.

Cartpole: We applied the controller to cart-pole dynamics (see supplementary) for a swing-up task with a time horizon of $T = 1.5$ s. The network was trained using a batch size of 256 each for 4000 iterations with a custom scheduled learning rate. It can be seen in Figure 3 that the FBSDE controller results in much higher variance in the trajectories than the 2FBSDE and iLQG, which take control multiplicative noise into account explicitly.

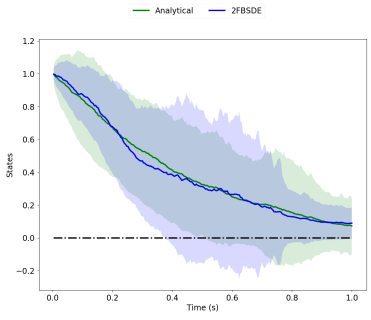

Figure 2: Comparison with analytical solution of a linear system.

Quadcopter: The controller was also tested on quadcopter dynamics (see supplementary) for a task of reaching and hovering at a target position with a time horizon of 2.0 seconds. The network was trained with a batch size of 512 for 6000 iterations. Only linear and angular states are included in fig. 4 since they most directly reflect the task performance (velocity plots included in the supplementary materals). The figure demonstrates superior performance of the 2FBSDE controller over the FBSDE and iLQG controller in reaching the target state faster and maintaining smaller state variance. Moreover, these results also convey the importance of explicitly taking into account multiplicative noise in the design of optimal controllers as the state dimensionality and system complexity increases.

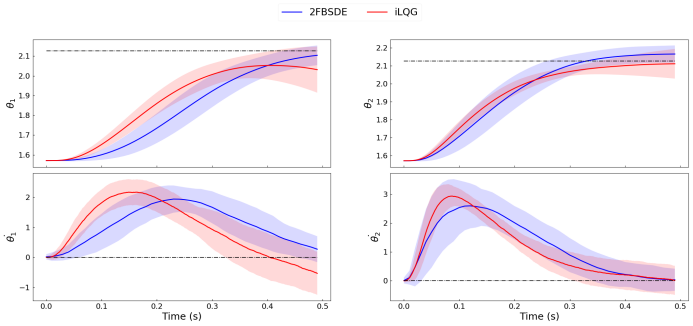

Figure 5: Comparison of 2FBSDE and iLQG on the 2-link human arm in phase plot.

2-link 6-muscle Human Arm: This is a bio-mechanical system wherein control multiplicative noise models have been found to closely mimic empirical observations. The controllers were tasked on reaching a target position in 0.5 seconds similar to Li & Todorov (2007) as depicted in Figures 5 and 6. The network was trained with a batch size of 256 for 6000 iterations. Both additive noise (in angular accelerations with $\Sigma_{ii} = 1.0$) and multiplicative noise (in muscle activations with $\sigma = 0.2$) were considered. As seen from the plots, in presense of higher σ, 2FBSDE demonstrates higher immunity whereas iLQG begins to diverge and miss the target as seen in the phase plot.

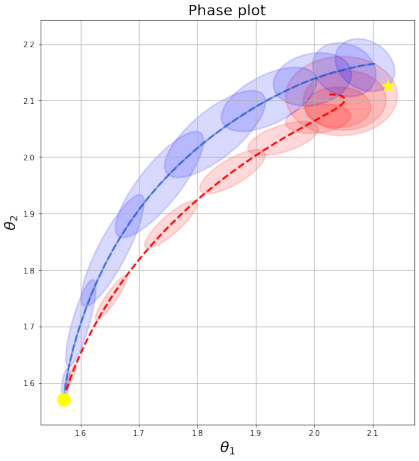

Figure 6: Phase plot with mean and uncertainty ellipses comparing 2FBSDE and iLQG controllers. The circle denotes the starting position and star denotes the target position of the end-effector.

Discussion: Although the performance of iLQG for the quadcopter and human arm tasks seem competitive to 2FBSDE, we would like to highlight the fact that iLQG becomes brittle as the standard deviation of the control multiplicative noise is increased and requires very fine time discretization ($\Delta t = 0.01$s and $\Delta t = 0.001$s for quadcopter and human arm simulations respectively) in order to be able to converge. This fact motivated the work by Torre & Theodorou. (2015) on incorporating stochastic variational integrators and structured linearization in Stochastic Differential Dynamic Programming. Moreover, the control cost (R) had to be tuned to a high value in order to prevent divergence at higher noise standard deviations. This is in contrast to the results in Li & Todorov (2007), on account of two main reasons: firstly they do not consider the presence of additive noise in the angular acceleration channels in addition to multiplicative noise in the muscle activation channels and secondly the paper does not provide all details to fully reproduce the published results nor is code with muscle actuation made publicly available. Some of the crucial elements to fully implement the human arm model such as relationships between muscle lengths and muscle velocities and the respective joint angles had to be obtained by studying the work done by Teka et al. (2017).

6 Conclusions

In this paper, we proposed the Deep 2FBSDE controller to solve the Stochastic Optimal Control problems for systems with both additive and multiplicative noise. The algorithm relies on the 2FBSDE formulation with control in the forward process for sufficient exploration. The effectiveness of the algorithm is demonstrated by comparing against analytical solution for a linear system, and against the first order FBSDE controller on systems of cartpole, quadcopter and 2-link human arm in simulation. Potential future directions of this work include application to financial models with intrinsic control multiplicative noise and theoretical analysis of error bounds on value function approximation.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

K. S. Bakshi, D. D. Fan, and E. A. Theodorou. Stochastic control of systems with control multiplicative noise using second order fbsdes. In *2017 American Control Conference (ACC)*, pp. 424–431, May 2017a. doi: 10.23919/ACC.2017.7962990.

K. S. Bakshi, D. D. Fan, and E. A. Theodorou. Stochastic control of systems with control multiplicative noise using second order fbsdes. In *2017 American Control Conference (ACC)*, pp. 424–431, May 2017b. doi: 10.23919/ACC.2017.7962990.

R. Bellman and R. Kalaba. *Selected Papers On mathematical trends in Control Theory*. Dover Publications, 1964.

Nicole El Karoui, Shige Peng, and Marie Claire Quenez. Backward stochastic differential equations in finance. *Mathematical finance*, 7(1):1–71, 1997.

I. Exarchos and E. A. Theodorou. Learning optimal control via forward and backward stochastic differential equations. In *American Control Conference (ACC), 2016*, pp. 2155–2161. IEEE, 2016.

I. Exarchos and E. A. Theodorou. Stochastic optimal control via forward and backward stochastic differential equations and importance sampling. *Automatica*, 87:159–165, 2018.

I. Exarchos, E. A. Theodorou, and P. Tsiotras. Stochastic L^1-optimal control via forward and backward sampling. *Systems & Control Letters*, 118:101–108, 2018.

Ioannis Exarchos, Evangelos Theodorou, and Panagiotis Tsiotras. Stochastic differential games: A sampling approach via fbsdes. *Dynamic Games and Applications*, 9(2):486–505, Jun 2019. ISSN 2153-0793. doi: 10.1007/s13235-018-0268-4. URL `https://doi.org/10.1007/s13235-018-0268-4`.

W. H. Fleming and H. M. Soner. *Controlled Markov processes and viscosity solutions*. Applications of mathematics. Springer, New York, 2nd edition, 2006.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Alex Gorodetsky, Sertac Karaman, and Youssef Marzouk. Efficient high-dimensional stochastic optimal motion control using tensor-train decomposition. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015. doi: 10.15607/RSS.2015.XI.015.

Maki K Habib, Wahied Gharieb Ali Abdelaal, Mohamed Shawky Saad, et al. Dynamic modeling and control of a quadrotor using linear and nonlinear approaches. 2014.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018. ISSN 0027-8424. doi: 10.1073/pnas.1718942115. URL `https://www.pnas.org/content/115/34/8505`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

V. A. Huynh, S. Karaman, and E. Frazzoli. An incremental sampling-based algorithm for stochastic optimal control. *I. J. Robotic Res.*, 35(4):305–333, 2016. doi: 10.1177/0278364915616866. URL `http://dx.doi.org/10.1177/0278364915616866`.

Kiyosi Itô. 109. stochastic integral. *Proceedings of the Imperial Academy*, 20(8):519–524, 1944.

H. J. Kappen. Linear theory for control of nonlinear stochastic systems. *Phys Rev Lett*, 95:200201, 2005. Journal Article United States.

I. Karatzas and S. E. Shreve. *Brownian Motion and Stochastic Calculus (Graduate Texts in Mathematics)*. Springer, 2nd edition, August 1991. ISBN 0387976558.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*, volume 23. Springer Science & Business Media, 2013.

H. J. Kushner and P. G. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time*. Springer-Verlag, London, UK, UK, 1992. ISBN 0-387-97834-8.

Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pp. 222–229, 2004.

Weiwei Li and Emanuel Todorov. Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system. *International Journal of Control*, 80(9):1439–1453, 2007.

P. McLane. Optimal stochastic control of linear systems with state- and control-dependent disturbances. *IEEE Transactions on Automatic Control*, 16(6):793–798, December 1971. ISSN 0018-9286. doi: 10.1109/TAC.1971.1099828.

Djordje Mitrovic, Stefan Klanke, Rieko Osu, Mitsuo Kawato, and Sethu Vijayakumar. A computational model of limb impedance control based on principles of internal model uncertainty. *PLOS ONE*, 5(10):1–11, 10 2010. doi: 10.1371/journal.pone.0013601. URL https://doi.org/10.1371/journal.pone.0013601.

Etienne Pardoux and Aurel Rascanu. *Stochastic Differential Equations, Backward SDEs, Partial Differential Equations*, volume 69. 07 2014. doi: 10.1007/978-3-319-05714-9.

Marcus A Pereira, Ziyi Wang, Ioannis Exarchos, and Evangelos A Theodorou. Learning deep stochastic optimal control policies using forward-backward sdes. In *Robotics: science and systems*, 2019.

Y. Phillis. Controller design of systems with multiplicative noise. *IEEE Transactions on Automatic Control*, 30 (10):1017–1019, October 1985. ISSN 0018-9286. doi: 10.1109/TAC.1985.1103828.

J. A. Primbs. Portfolio optimization applications of stochastic receding horizon control. In *2007 American Control Conference*, pp. 1811–1816, July 2007. doi: 10.1109/ACC.2007.4282251.

Steven E Shreve. *Stochastic calculus for finance II: Continuous-time models*, volume 11. Springer Science & Business Media, 2004.

R. F. Stengel. *Optimal control and estimation*. Dover books on advanced mathematics. Dover Publications, New York, 1994.

Wondimu W Teka, Khaldoun C Hamade, William H Barnett, Taegyo Kim, Sergey N Markin, Ilya A Rybak, and Yaroslav I Molkov. From the motor cortex to the movement and back again. *PloS one*, 12(6):e0179288, 2017.

E.A. Theodorou, J. Buchli, and S. Schaal. A generalized path integral approach to reinforcement learning. *Journal of Machine Learning Research*, (11):3137–3181, 2010a.

E.A. Theodorou, Y. Tassa, and E. Todorov. Stochastic differential dynamic programming. In *American Control Conference*, pp. 1125–1132, 2010b.

E. Todorov. Linearly-solvable markov decision problems. In B. Scholkopf, J. Platt, and T. Hoffman (eds.), *Advances in Neural Information Processing Systems 19*, Vancouver, BC, 2007. Cambridge, MA: MIT Press.

E. Todorov. Efficient computation of optimal actions. *Proceedings of the national academy of sciences*, 106(28): 11478–11483, 2009.

E. Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pp. 300–306 vol. 1, June 2005. doi: 10.1109/ACC.2005.1469949.

Emanuel Todorov. Stochastic optimal control and estimation methods adapted to the noise characteristics of the sensorimotor system. *Neural Comput.*, 17:1084–1108, May 2005.

G. DeLa Torre and E.A. Theodorou. Stochastic variational integrators for system propagation and linearization. Sept 2015.

SUPPLEMENTARY MATERIALS

1 ASSUMPTIONS

Assumption 1. *There exists a constant $C > 0$ such that*

$$|(\boldsymbol{f} + \mathbf{G}\boldsymbol{u})(t, \boldsymbol{x}, \boldsymbol{u}) - (\boldsymbol{f} + \mathbf{G}\boldsymbol{u})(t, \boldsymbol{x}', \boldsymbol{u}')| \leq C(|\boldsymbol{x} - \boldsymbol{x}'| + |\boldsymbol{u} - \boldsymbol{u}'|) \tag{11}$$

$$|\mathbf{B}(t, \boldsymbol{x}, \boldsymbol{u}) - \mathbf{B}(t, \boldsymbol{x}', \boldsymbol{u}')| \leq C(|\boldsymbol{x} - \boldsymbol{x}'| + |\boldsymbol{u} - \boldsymbol{u}'|) \tag{12}$$

$$|(\boldsymbol{f} + \mathbf{G}\boldsymbol{u})(t, \boldsymbol{x}, \boldsymbol{u}) + \mathbf{B}(t, \boldsymbol{x})| \leq C(1 + |\boldsymbol{x}|), \tag{13}$$

where $\mathbf{B} = [\sigma\mathbf{G}\boldsymbol{u} \;\; \boldsymbol{\Sigma}]$, $\forall t \in [0, T]$, $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{n_x}$ *and* $\forall \boldsymbol{u}, \boldsymbol{u}' \in \mathbb{R}^{n_u}$.

2 HJB PDE DERIVATION

Applying the dynamic programming principle (Bellman & Kalaba, 1964) to the value function we have

$$V(t, \boldsymbol{x}(t)) = \inf_{\boldsymbol{u}(s) \in \mathcal{U}([t, t+\mathrm{d}t])} \mathbb{E}_{\mathbb{Q}}\Big[\int_{t}^{t+\mathrm{d}t} \ell(s, \boldsymbol{x}(s), \boldsymbol{u}(s))\mathrm{d}s + V(t + \mathrm{d}t, \boldsymbol{x}(t + \mathrm{d}t))\Big]. \tag{14}$$

Then, we can approximate the running cost integral with a step function and apply Ito's lemma (Itô, 1944) to obtain

$$\begin{aligned}
V(t, \boldsymbol{x}(t)) &= \inf_{\boldsymbol{u} \in \mathcal{U}} \mathbb{E}_{\mathbb{Q}}\Big[(\ell(\boldsymbol{x}(t), \boldsymbol{u}(t))\mathrm{d}t + V(t + \mathrm{d}t, \boldsymbol{x}(t + \mathrm{d}t))\Big] \\
&\overset{\text{Itô}}{=} \inf_{\boldsymbol{u} \in \mathcal{U}} \mathbb{E}_{\mathbb{Q}}\Big[(\ell(\boldsymbol{x}(t), \boldsymbol{u}(t))\mathrm{d}t + V(t, \boldsymbol{x}(t)) + V_t(t, \boldsymbol{x}(t))\mathrm{d}t \\
&+ V_{\boldsymbol{x}}^{\mathrm{T}}(t, \boldsymbol{x}(t))\Big(\boldsymbol{f}(t, \boldsymbol{x}(t))\mathrm{d}t + \mathbf{G}(t, \boldsymbol{x}(t))(\boldsymbol{u}\mathrm{d}t + \sigma\boldsymbol{u}(t)\mathrm{d}v(t) \\
&+ \boldsymbol{\Sigma}(t, \boldsymbol{x}(t))\mathrm{d}\boldsymbol{w}(t)\Big) + \frac{1}{2}\mathrm{tr}\big(V_{\boldsymbol{x}\boldsymbol{x}}(t, \boldsymbol{x}(t))\mathbf{C}(t, \boldsymbol{x}(t), \boldsymbol{u}(t))\mathbf{C}^{\mathrm{T}}(t, \boldsymbol{x}(t), \boldsymbol{u}(t))\big)\mathrm{d}t \\
&= \inf_{\boldsymbol{u} \in \mathcal{U}}\Big[\ell(\boldsymbol{x}(t), \boldsymbol{u}(t))\mathrm{d}t + V(t, \boldsymbol{x}(t)) + V_t(t, \boldsymbol{x}(t))\mathrm{d}t \\
&+ V_{\boldsymbol{x}}^{\mathrm{T}}(t, \boldsymbol{x}(t))\Big(\boldsymbol{f}(t, \boldsymbol{x}(t))\mathrm{d}t + \mathbf{G}(t, \boldsymbol{x}(t))\boldsymbol{u}(t)\mathrm{d}t\Big) \\
&+ \frac{1}{2}\mathrm{tr}\big(V_{\boldsymbol{x}\boldsymbol{x}}(t, \boldsymbol{x}(t))\mathbf{C}(t, \boldsymbol{x}(t), \boldsymbol{u}(t))\mathbf{C}^{\mathrm{T}}(t, \boldsymbol{x}(t), \boldsymbol{u}(t))\big)\mathrm{d}t
\end{aligned}$$

We can cancel $V(t, \boldsymbol{x}(t))$ on both sides, plug in the definition of running cost, and bring the terms not dependent on controls outside of the infimum to get

$$V_t + V_{\boldsymbol{x}}^{\mathrm{T}}\boldsymbol{f} + q + \inf_{\boldsymbol{u} \in \mathcal{U}[0,T]}\Big\{\frac{1}{2}\boldsymbol{u}^{\mathrm{T}}R\boldsymbol{u} + V_{\boldsymbol{x}}^{\mathrm{T}}\mathbf{G}\boldsymbol{u} + \frac{1}{2}\,\mathrm{tr}(V_{\boldsymbol{x}\boldsymbol{x}}\mathbf{C}\mathbf{C}^{\mathrm{T}})\Big\} = 0 \tag{15}$$

Now we can find the optimal control by taking gradient of the terms inside the infimum with respect to $\boldsymbol{u}$ and set it to zero as

$$\boldsymbol{u}^*(t, \boldsymbol{x}) = -\hat{R}^{-1}\mathbf{G}(t, \boldsymbol{x})^{\mathrm{T}}V_{\boldsymbol{x}}(t, \boldsymbol{x}). \tag{16}$$

Finally, the optimal control can be plugged back into equation 15 to obtain the HJB PDE.

3 PROOF OF 2FBSDE'S EQUIVALENCE TO HJB PDE

Firstly, given the definition of $\big(\tilde{Y}, \tilde{\mathbf{Z}}, \boldsymbol{\Gamma}, \mathbf{A}\big)$, we can apply Ito's differentiation rule to $\tilde{Y}$:

$$\mathrm{d}\tilde{Y}(t) \overset{\text{Itô}}{=} \tilde{Y}_t\mathrm{d}t + \tilde{Y}_{\boldsymbol{x}}^{\mathrm{T}}\mathrm{d}\tilde{\mathbf{X}}(t) + \frac{1}{2}\,\mathrm{tr}\left(\tilde{Y}_{\boldsymbol{x}\boldsymbol{x}}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\mathrm{T}}\right)\mathrm{d}t. \tag{17}$$

Since $\tilde{Y}$ is the value function, we can substitute in the HJB PDE for $\tilde{Y}_t$ and forward dynamics for $\mathrm{d}\tilde{\mathbf{X}}$ to get:

$$\begin{aligned}\mathrm{d}\tilde{Y}(t) = &-\Big(q(\tilde{\mathbf{X}}(t)) + \tilde{\mathbf{Z}}^{\mathrm{T}}(t,\tilde{\mathbf{X}}(t))\boldsymbol{f}(t,\tilde{\mathbf{X}}(t)) - \frac{1}{2}\tilde{\mathbf{Z}}^{\mathrm{T}}(t,\tilde{\mathbf{X}}(t))\mathbf{G}(t,\tilde{\mathbf{X}}(t))\hat{R}^{-1}(t,\tilde{\mathbf{X}}(t))\mathbf{G}^{\mathrm{T}}(t,\tilde{\mathbf{X}}(t))\tilde{\mathbf{Z}}(t,\tilde{\mathbf{X}}(t)) \\ &+ \frac{1}{2}\operatorname{tr}\left(\boldsymbol{\Gamma}(t,\tilde{\mathbf{X}}(t))\boldsymbol{\Sigma}(t,\tilde{\mathbf{X}}(t))\boldsymbol{\Sigma}^{\mathrm{T}}(t,\tilde{\mathbf{X}}(t))\right)\Big)\mathrm{d}t \qquad (18) \\ &+ \tilde{\mathbf{Z}}^{\mathrm{T}}(t,\tilde{\mathbf{X}}(t))\Big(\boldsymbol{f}(t,\tilde{\mathbf{X}}(t))\mathrm{d}t + \mathbf{G}(t,\tilde{\mathbf{X}})(\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}t + \sigma\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}V) + \boldsymbol{\Sigma}(t,\tilde{\mathbf{X}}(t))\mathrm{d}\tilde{\mathbf{W}}(t)\Big) \\ &+ \frac{1}{2}\operatorname{tr}\left(\boldsymbol{\Gamma}(t,\tilde{\mathbf{X}}(t))\boldsymbol{\Sigma}(t,\tilde{\mathbf{X}}(t))\boldsymbol{\Sigma}^{\mathrm{T}}(t,\tilde{\mathbf{X}}(t))\right)\Big)\mathrm{d}t \\ &= -h(t,\tilde{\mathbf{X}},\tilde{Y},\tilde{\mathbf{Z}},\boldsymbol{\Gamma})\mathrm{d}t + \tilde{\mathbf{Z}}^{\mathrm{T}}\mathbf{G}(t,\tilde{\mathbf{X}})(\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}t + \sigma\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}V) + \tilde{\mathbf{Z}}^{\mathrm{T}}\boldsymbol{\Sigma}(t,\tilde{\mathbf{X}})\mathrm{d}\tilde{\mathbf{W}}(t). \qquad (19)\end{aligned}$$

For $\tilde{\mathbf{Z}}$, we can again apply Ito's differentiation rule:

$$\mathrm{d}\tilde{\mathbf{Z}}(t) \overset{\text{Itô}}{=} \tilde{\mathbf{Z}}_t\mathrm{d}t + \tilde{\mathbf{Z}}_{\boldsymbol{x}}^{\mathrm{T}}\mathrm{d}\tilde{\mathbf{X}}(t) + \frac{1}{2}\operatorname{tr}\left(\tilde{\mathbf{Z}}_{\boldsymbol{xx}}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\mathrm{T}}\right). \qquad (20)$$

We can again substitute in the forward dynamics for $\mathrm{d}\tilde{\mathbf{X}}$ and get:

$$\begin{aligned}\mathrm{d}\tilde{\mathbf{Z}}(t) = &\ \partial_t\tilde{\mathbf{Z}}(t,\tilde{\mathbf{X}}(t))\mathrm{d}t \qquad (21) \\ &+ \partial_{\boldsymbol{x}}\tilde{\mathbf{Z}}^{\mathrm{T}}(t,\tilde{\mathbf{X}}(t))\Big(\boldsymbol{f}(t,\tilde{\mathbf{X}}(t))\mathrm{d}t + \mathbf{G}(t,\tilde{\mathbf{X}})(\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}t + \sigma\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}V) + \boldsymbol{\Sigma}(t,\tilde{\mathbf{X}}(t))\mathrm{d}\tilde{\mathbf{W}}(t)\Big) \\ &+ \frac{1}{2}\operatorname{tr}\left(\partial_{\boldsymbol{xx}}\tilde{\mathbf{Z}}(t,\tilde{\mathbf{X}}(t)\boldsymbol{\Sigma}(t,\tilde{\mathbf{X}}(t))\boldsymbol{\Sigma}^{\mathrm{T}}(t,\tilde{\mathbf{X}}(t))\right)\mathrm{d}t \\ &= \mathbf{A}(t,\tilde{\mathbf{X}})\mathrm{d}t + \boldsymbol{\Gamma}(t,\tilde{\mathbf{X}})\mathbf{G}(t,\tilde{\mathbf{X}})(\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}t + \sigma\boldsymbol{u}^*(t,\tilde{\mathbf{X}})\mathrm{d}V) + \boldsymbol{\Gamma}(t,\tilde{\mathbf{X}})\boldsymbol{\Sigma}(t,\tilde{\mathbf{X}})\mathrm{d}\tilde{\mathbf{W}}(t). \qquad (22)\end{aligned}$$

Note that the transpose on $\boldsymbol{\Gamma}$ is dropped since it is symmetric.

4 Loss Function

The loss function used in this work builds on the loss functions used in Han et al. (2018) and Pereira et al. (2019). Because the Deep 2FBSDE Controller propagates 2 BSDEs, in addition to using the propagated value function $(\tilde{Y}_t)$, the propagated gradient of the value function $(\tilde{Z}_t)$ can also be used in the loss function to enforce that the network meets both the terminal constraints i.e. $\phi(\tilde{X}_N)$ and $\phi_{\boldsymbol{x}}(\tilde{X}_N)$ respectively. Moreover, since the terminal cost function is known, its hessian can be computed and used to enforce that the output of the FC_1 layer at the terminal time index (Γ_N) be equal to the target hessian of the terminal cost function $\phi_{\boldsymbol{xx}}(\tilde{\mathbf{X}}_N)$. Although this is enforced only at the terminal time index (Γ_N), because the weights of a recurrent neural network are shared across time, in order to be able to predict Γ_N accurately all of the prior predictions Γ_t will have to be adjusted accordingly, thereby representing some form of propagation of the hessian of the value function.

$$\tilde{\mathcal{L}} = ||\tilde{Y}^* - \tilde{Y}_N||_2^2 + ||\tilde{Z}^* - \tilde{Z}_N||_2^2 + ||\tilde{\Gamma}^* - \tilde{\Gamma}_N||_2^2 + \lambda||\theta||_2^2$$

where, $\tilde{Y}^* = \phi(\tilde{\mathbf{X}}_N),\ \tilde{Z}^* = \phi_{\boldsymbol{x}}(\tilde{\mathbf{X}}_N)$ and $\tilde{\Gamma}^* = \phi_{\boldsymbol{xx}}(\tilde{\mathbf{X}}_N)$

Additionally importance sampling introduces an additional gradient path through the system dynamics at every time step. Although this makes training difficult (gradient vanishing problem) it allows the weights to now influence what the next state (i.e. at the next time index) will be. As a result, the weights can control the state at the end time index and hence the target $\left(\tilde{Y}^*(\tilde{\mathbf{X}}_N)\right)$ for the neural network prediction itself. This can be added to the loss function which in addition to importance sampling allows to accelerate the minimization of the terminal cost to achieve the task objectives. So, the above loss function becomes,

$$\tilde{\mathcal{L}} = ||\tilde{Y}^* - \tilde{Y}_N||_2^2 + ||\tilde{Z}^* - \tilde{Z}_N||_2^2 + ||\tilde{\Gamma}^* - \tilde{\Gamma}_N||_2^2 + ||\tilde{Y}^*||_2^2 + \lambda||\theta||_2^2$$

Finally, for high-dimensional systems, we observed that the training process was very sensitive to weight initialization. Unlike Pereira et al. (2019), the 2FBSDE network propagates the gradient of the value function, rather than predicting it at every time step. Because the initial weights of the network are random and the number of trainable parameters in the FC_1 layer is $\mathcal{O}(n_x^2)$, the poor initial predictions leads to a snowballing effect causing $\tilde{Z}_t$ and hence the loss to diverge. This makes training very unstable and sometimes impossible. Therefore, to regulate the growth of error in $\tilde{Z}_t$, in addition to proper initialization, we added a term to the loss function to penalize the growth of the gradient of the value function. We define the loss term as follows,

$$Z_{\text{loss}} = \begin{cases} ||\tilde{Z}_t||_2^2, & \text{if } |\tilde{Z}_t| \geq \tilde{Z}_{\max} \\ 0, & \text{otherwise} \end{cases}$$

We used $\tilde{Z}_{\max} = 10.0$ for our quadcopter experiments. Finally, the entire loss function is as follows,

$$\tilde{\mathcal{L}} = ||\tilde{Y}^* - \tilde{Y}_N||_2^2 + ||\tilde{Z}^* - \tilde{Z}_N||_2^2 + ||\tilde{\Gamma}^* - \tilde{\Gamma}_N||_2^2 + ||\tilde{Y}^*||_2^2 + Z_{\text{loss}} + \lambda||\theta||_2^2$$

5 NON-LINEAR SYSTEM DYNAMICS AND SYSTEM PARAMETERS

5.1 CARTPOLE

The equations of motion for the cartpole are given by

$$(m_c + m_p)\ddot{x} + m_p l\ddot{\theta}\cos\theta - m_p l\dot{\theta}^2 \sin\theta = u$$
$$m_p l\ddot{x}\cos\theta + m_p l^2\ddot{\theta} + m_p gl\sin\theta = 0.$$

The model and dynamics parameters are set as $m_p = 0.01$ kg, $m_c = 1$ kg, $l = 0.5$ m, $\sigma = 0.125$, $\mathbf{\Sigma} = \begin{bmatrix} \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & I_{2\times2} \end{bmatrix}$. The initial pole and cart position are 0 rad and 0 m with no velocity, and the target state is a pole angle of π rad and zeros for all other states.

5.2 QUADCOPTER

The quadcopter dynamics used can be found in Habib et al. (2014). The model and dynamics parameters are set as $m = 0.47$ kg, $J_x = J_y = 4.86 \times 10^{-3}$ kg m^2, $J_z = 8.8 \times 10^{-3}$kgm^2, $l = 0.225$ m, $\sigma = 0.2$, $\mathbf{\Sigma} = \begin{bmatrix} \mathbf{0}_{6\times6} & \mathbf{0}_{6\times6} \\ \mathbf{0}_{6\times6} & I_{6\times6} \end{bmatrix}$. Additionally, we used an exploration factor of 1.5 on the additive noise during training to facilitate convergence (effectively sampling $\Delta\tilde{W}(t) \sim \mathcal{N}(0, 1.5^2 I\Delta t)$). The initial state conditions used are all zeros, and the target state is 1 meter each in the north-east-down directions and all other states zero.

5.3 2-LINK 6-MUSCLE HUMAN ARM

The forward dynamics of the human arm as stated in Li & Todorov (2004) is given by:

$$\ddot{\theta} = \mathcal{M}(\theta)^{-1}(\tau - \mathcal{C}(\theta, \dot{\theta}) - \mathcal{B}\dot{\theta})$$

where $\theta \in \mathbb{R}^2$ represents the joint angle vector consisting of θ_1 (shoulder joint angle) and θ_2 (elbow joint angle). $\mathcal{M}(\theta) \in \mathbb{R}^{2\times2}$ is the inertia matrix which is positive definite and symmetric. $\mathcal{C}(\theta, \dot{\theta}) \in \mathbb{R}^2$ is the vector centripetal and Coriolis forces. $\mathcal{B} \in \mathbb{R}^{2\times2}$ is the joint friction matrix. $\tau \in \mathbb{R}^2$ is the joint torque applied on the system. The torque is generated by activation of 6 muscle groups. Following are equations for components of each of the above matrices and vectors:

$$\mathcal{M} = \begin{pmatrix} a_1 + 2a_2 cos\theta_2 & a_3 + a_2 cos\theta_2 \\ a_3 + a_2 cos\theta_2 & a_3 \end{pmatrix}$$
$$\mathcal{C} = \begin{pmatrix} -\dot{\theta_2}(2\dot{\theta_1} + \dot{\theta_2}) \\ \dot{\theta_1} \end{pmatrix} a_2 sin\theta_2$$
$$\mathcal{B} = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$
$$a_1 = I_1 + I_2 + M_2 l_1^2$$
$$a_2 = m_2 L_1 s_2$$
$$a_3 = I_2$$

Where $b_{11} = b_{22} = 0.05$, $b_{12} = b21 = 0.025$, m_i is the mass (1.4kg,1kg), l_i is the length of link i (30cm, 33cm), s_i is the distance between joint center and mass of link i (11cm, 16cm), I_i represents for the moment of inertia ($0.025kgm^2$,$0.045kgm^2$).
The activation dynamics for each muscle is given by:

$$\dot{a} = \frac{(1+\sigma)u - a}{t}$$
$$t = t_{deact}$$

Where $t_{deact} = 66msec$, $a \in \mathbb{R}^6$ is a vector of muscle activations, $u \in \mathbb{R}^6$ is a vector of instantaneous neural inputs. As a result of these dynamics, six new state variables ($\boldsymbol{a}$) will be incorporated into the dynamical system. With the muscle activation dynamics proposed above, the joint torque vector can be computed as follows:

$$\tau = M(\theta)T(a, l(\theta), v(\theta, \dot{\theta}))$$

wherein $T(a, l, v) \in \mathbb{R}$ is the tensile force generated by each muscle, whose expression is given by:

$$T(a,l,v) = A(a,l)(F_l(l)F_V(l,v) + F_P(l))$$
$$A(a,l) = 1 - exp\left(-\left(\frac{a}{0.56N_f(l)}\right)^{N_f(l)}\right)$$
$$N_f(l) = 2.11 + 4.16\left(\frac{1}{l} - 1\right)$$
$$F_L(l) = exp\left(-\left|\frac{l^{1.93} - 1}{1.03}\right|^{1.87}\right)$$
$$F_V(l,v) = \begin{cases} \frac{-5.72-v}{-5.72+(1.38+2.09l)v}, & \text{if } v \leq 0 \\ \frac{0.62-(-3.12+4.21l-2.67l^2)v}{0.62+v}, & \text{otherwise} \end{cases}$$
$$F_p(l) = -0.02exp(13.8 - 18.7l)$$

The equation for $M(\theta)$ is obtained from Teka et al. (2017, Equation 3).

The parameters used in the equations above are mostly obtained by work in Teka et al. (2017), which we provide here in the following table:

Muscle	**Maximal Force,** F_{max}**(N)**	**Optimal Length,** L_{opt}**(m)**	**Velocity,** v**(m/sec)**
SF	420	0.185	$-\dot{\theta_1}R_{SF}$
SE	570	0.170	$-\dot{\theta_1}R_{SE}$
EF	1010	0.180	$-(\dot{\theta 2} - \dot{\theta_1})R_{EF}$
EE	1880	0.055	$-(\dot{\theta 2} - \dot{\theta_1})R_{EE}$
BF	460	0.130	$-\dot{\theta_1}R_{BFS} - (\dot{\theta_2} - \dot{\theta_1})R_{BFE}$
BE	630	0.150	$-\dot{\theta_1}R_{BFS} - (\dot{\theta_2} - \dot{\theta_1})R_{BEE}$

Where the last column consists of moment arms which are approximated according to (Li & Todorov, 2004, figure B). $l \in \mathbb{R}^{6\times 1}$ is the length of each muscle, which is calculated by approximating the range of each muscle group in Li & Todorov (2004) using the data of joint angle ranges and linear relationships with corresponding joint angles as in Teka et al. (2017). Each of the fitted linear functions for muscle lengths and cosine functions for moment arms are available in the provided MATLAB code.

6 Network Initialization Strategies

For linear and low-dimensional systems (Pendulum and Cart Pole) we used the Xavier initialization strategy Glorot & Bengio (2010). This is considered to be standard method for recurrent networks that use tanh activations. This strategy was crucial to allow using large learning rates and allow convergence in $\sim 2000 - 4000$ iterations. Without this strategy, convergence is extremely slow and prohibits the use for large learning rates.

On the other hand, for high dimensional systems like the quadcopter, this strategy failed to work. The reason is partially explained in the loss function section of this supplementary text. Essentially, the FC layers having $\mathcal{O}(n_x^2)$ trainable parameters and random initialization values cause a snowballing effect on the propagation of $\tilde{\mathbf{Z}}_t$ causing the loss function to diverge and make training impossible as the gradient step in never reached. A simple solution to this problem was to use zero initialization for all weights and biases. This prevents $\tilde{\mathbf{Z}}_t$ from diverging as network predictions are zero and the state trajectory propagation is purely noise driven. This allows computing the loss function without diverging and taking gradient steps to start making meaningful predictions at every time step. Although this allows for training the network for high-dimensional systems, it slows down the process and convergence requires many more iterations. Therefore, further investigation into initialization strategies or other recurrent network architectures would allow improving convergence speed.

7 MACHINE INFORMATION

We used Tensorflow Abadi et al. (2015) extensively for defining the computational graph in fig. 1 and model training. Because our current implementation involves many consecutive CPU-GPU transfers, we decided to implement the CPU version of Tensorflow as the data transfer overhead was very time consuming. The models were trained on multiple desktops computers / laptops to evaluate diffent models and hyperparameters. An Alienware laptop was used with the following specs:
Processor: Intel Core i9-8950HK CPU @ 2.90GHz$\times$12, Memory: 32GiB.
The desktop computers used have the following specs:
Processor: Intel Xeon(R) CPU E5-1607v3 @ 3.10GHz$\times$4 Memory: 32GiB