Solving Elliptic Hamilton-Jacobi-Bellman Equations in a Value Space

Wenhao Qiu, Qingshuo Song, *Member, IEEE*, and George Yin, *Fellow, IEEE*

***Abstract*—In this letter, we study the numerical solutions of a class of elliptic type of Hamilton-Jacobi-Bellman (HJB) equations, which arise from stochastic control and multiplayer stochastic game problems with random terminal time. Two closely related algorithms are proposed. One of them is to use value iteration on approximating Markov decision processes, and the other is to use a deep-learning approach solving Bellman equations. The convergence is shown by using a viscosity solution approach.**

***Index Terms*—HJB equation, Markov chain approximation, reinforcement learning, viscosity solution, value iteration.**

I. Introduction

STOCHASTIC control theory has witnessed tremendous progress since 1960s; see the historical remarks in [7], [24] and references therein. The effort leads to a multidisciplinary study intertwined with fully nonlinear partial differential equations (for instance, Hamilton-Jacobi-Bellman (HJB) equations) and the use of nonlinear Feynman-Kac formulas; see [3] and [6]. Because the regularity of the system is often not known *a priori* due to the control processes, let alone the solutions of the HJB equations being usually highly nonlinear, closed-form solutions are rarely obtainable. Thus, numerical methods such as finite difference and finite element methods for solving the HJB equations become a viable alternative.

With the success of deep neural networks in the past few years, there are increasing interests of using this modern learning technologies to solve HJB equations; see [4], [10], [14], [18], and references therein. For instance, [14] shows that the solution of the deep-learning approach converges to the viscosity solution of HJB as long as the loss functions exhibit *contraction principle* in a certain sense.

In reference to the recent progress, this letter develops numerical methods using the deep learning approach based on the Markov chain approximation methods initiated by Kushner and colleagues ([16] and [17]). It entitles the following steps.

Step 1: Approximating the solutions of the underlying stochastic control problems and recast them into discrete-time Markov decision processes (MDPs).

Step 2: Solve the MDPs by using reinforcement learning. One of the main advantages of the Markov chain approximation methods is: one need not assume much of the regularity of the associate partial differential equations (which is generally not known *a priori*).

In the literature, convergence analysis of Markov chain approximation for stochastic controls has been a favorable choice; see [1], [16], [19], [20], [21], [22] with different applications. This letter distinguishes from the recent work [11], [12], and [15] in two aspects. (1) All the papers mentioned above focus on the parabolic type HJB equations, while this letter studies elliptic type partial differential equations. The main difficulty of computing elliptic type PDE is that in contrast to the parabolic PDE, there is no explicit scheme such as backward iteration available; see [7]. (2) We provide convergence of the deep learning approach to the solutions of the HJB equations. Deep learning approach distinguishes itself from the classical value iteration in that the values are approximated by parameters optimized with the stochastic gradient method.

In this letter, we work with a d-dimensional Euclidean space $\mathbb{R}^d$. We use $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$ to denote a filtered probability space with a d-dimensional Wiener Process W_t defined on it. Suppose that O is a domain (open set) with closure denoted by $\bar{O}$. An upper semicontinuous function f in the set $\bar{O}$ is denoted by $f \in \text{USC}(\bar{O})$, whereas a lower semicontinuous f is denoted by $f \in \text{LSC}(\bar{O})$. [Recall that $f \in \text{USC}(\bar{O})$, if $-f \in \text{USC}(\bar{O})$.] Moreover, we use f^* and f_* to denote the USC and LSC envelopes of f, respectively, and use $I_A(\cdot)$ to denote the indicator function of the set A. In this letter, we use K as a generic positive constant whose value may change for different appearances with the understanding of $K + K = K$ and $KK = K$, which is just a convention. That is, rather than using different constants K_1, K_2, K_3 etc., we use one K to represent all the constants whose values are not our main concern. Given a smooth function $f : \mathbb{R}^d \to \mathbb{R}$, its gradient and Hessian matrix are denoted by $Df(x) = (\partial_{x_i} f(x))_{i=1,\dots,d}$ and

Manuscript received March 2, 2020; revised May 12, 2020; accepted May 25, 2020. Date of publication June 3, 2020; date of current version June 19, 2020. The work of Wenhao Qiu and Qingshuo Song was supported in part by the RGC of Hong Kong CityU under Grant 11201518, and in part by TRIAD Seed Grant of WPI. The work of George Yin was supported in part by the Air Force Office of Scientific Research under Grant FA9550-18-1-0268. Recommended by Senior Editor J.-F. Zhang. *(Corresponding author: Qingshuo Song.)*

Wenhao Qiu and Qingshuo Song are with the Department of Mathematics, Worcester Polytechnic Institute, Worcester, MA 01748 USA (e-mail: wqiu@wpi.edu; qsong@wpi.edu).

George Yin is with the Department of Mathematics, Wayne State University, Detroit, MI 48202 USA (e-mail: gyin@wayne.edu).

Digital Object Identifier 10.1109/LCSYS.2020.2999650

$D^2f(x) = (\partial_{x_i,x_j}f(x))_{i,j=1,\dots,d}$. Denote the trace of a square matrix A by $\mathrm{tr}(A)$. Then the Laplacian operator can be written as $\Delta f(x) = \mathrm{tr}(D^2f(x)) = \sum_{i=1}^{d} \partial_{x_ix_i}f(x)$.

The rest of this letter is arranged as follows. Section II presents stochastic control and game problems together with the associated HJB equation and Benchmark problems. Section III proposes two algorithms, one is value iteration as one of the tabular reinforcement learning, and the other uses a deep neural network approach. Corresponding convergence analysis is followed in Section IV. Finally, some further remarks are included in SectionVI.

II. Stochastic Control and Game Problems

We first present (A) one-player stochastic control problem and (B) Zero-sum stochastic differential game problem. Then we illustrate that both problems lead to certain elliptic Hamilton-Jacobi-Bellman (HJB) equations. The central issue of this letter is devoted to development of the computational methods for solving these equations and hence the stochastic control problems.

One-player stochastic control problem: We consider a stochastic dynamic system with drift vector $b : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$, and diffusion matrix $\sigma : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$, the running cost function $\ell : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$, and the terminal cost function $g : \mathbb{R}^d \to \mathbb{R}^d$. All the functions above are assumed to be Lipstchitz continuous on their respective domains. Suppose X_s satisfies controlled stochastic differential equation (SDE)

$$dX_s = b(X_s, \mathbf{a}_s)ds + dW_s, \;\; X_0 = x, \tag{1}$$

where $\mathbf{a} : [0, \infty) \times \Omega \to \mathbb{R}^d$ is an $\mathcal{F}_t$-progressively measurable control process taking values in A. The set of all admissible controls is denoted by $\mathcal{A}$ and $\{W_s : s \geq 0\}$ is a standard Brownian motion.

Define a stopping time $\tau = \inf\{s : X_s \notin O\}$ and a cost function with a given admissible control $\mathbf{a} \in \mathcal{A}$ as

$$J(x, \mathbf{a}) = \mathbb{E}\left[\int_t^\tau \ell(X_s, \mathbf{a}_s)ds + g(X_\tau)\right]. \tag{2}$$

Then the associated value function is defined as

$$v(x) = \inf_{\mathbf{a}\in\mathcal{A}} J(x, \mathbf{a}). \tag{3}$$

Zero-sum stochastic differential games: Let $A = A_1 \times A_2$ and $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2) \in \mathcal{A}$ be a pair of progressively measurable control processes. Suppose that X satisfies SDE (1), and $\mathbf{a}_1$ and $\mathbf{a}_2$ are controls offered by player 1 and player 2, respectively. The collection of admissible controls of player 1 and player 2 are denoted by $\mathcal{A}_1$ and $\mathcal{A}_2$. Player 1 (resp. player 2) wants to minimize (resp. maximize) the cost functional (2).

In what follows, we define the Elliott-Kalton type upper and lower values of differential games. For this purpose, we define strategy other than control process. An admissible strategy $\hat{a}_1$ (resp. $\hat{a}_2$) for player1 (resp. player2) is a progressively measurable mapping $\hat{a}_1 : \mathcal{A}_1 \to \mathcal{A}_2$ (resp. $\hat{a}_2 : \mathcal{A}_2 \to \mathcal{A}_1$). Let $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) denote the class of all admissible strategies of player 1 (resp. player 2).

The upper value $v^+(x)$ and lower value $v^-(x)$ are defined as

$$v^+(x) = \sup_{\hat{a}_1\in\mathcal{S}_1} \inf_{\mathbf{a}_1\in\mathcal{A}_1} J(x, \mathbf{a}_1, \hat{a}_1(\mathbf{a}_1)), \quad \text{and} \tag{4}$$

$$v^-(x) = \inf_{\hat{a}_2\in\mathcal{S}_2} \sup_{\mathbf{a}_2\in\mathcal{A}_2} J(x, \hat{a}_2(\mathbf{a}_2), \mathbf{a}_2), \tag{5}$$

respectively. If $v^+(x) = v^-(x)$ holds for all $x \in O$, then the differential game is said to have a saddle point, and its value is denoted by $v(x)$; see more detail in and [5] and [8].

A. Hamilton-Jacobi-Bellman Equations

We introduce an operator $\otimes_a$ as defined in [19] that summarizes values over actions as a function of the state satisfying the following three conditions: For any real-valued functions ϕ_1, ϕ_2, and constant c, there exists a constant K

(C1) $\otimes_a[c\phi_1(x, a) + \phi_2(x)] = c \otimes_a [\phi_1(x, a)] + \phi_2(x)$.
(C2) $\otimes_a^x \phi_1(x, a) \leq \otimes_a \phi_2(x, a)$, whenever $\phi_1 \leq \phi_2$.
(C3) $|\otimes_a \phi_1(x, a) - \otimes_a \phi_2(x, a)| \leq K \max_a |\phi_1(x, a) - \phi_2(x, a)|$.

One can immediately verify that $\max_a \phi(x, a)$ (respectively $\min_a \phi(x, a)$) for a and $\min_{a_1} \max_{a_2} \phi(x, a_1, a_2)$ for $a = (a_1, a_2)$ satisfy all the three conditions.

Consider an elliptic nonlinear HJB equation

$$F(x, Du(x), D^2u(x)) = 0, \forall x \in O \tag{6}$$

with its boundary condition

$$u(x) = g(x), \forall x \in O^c. \tag{7}$$

where $F : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{d\times d} \to \mathbb{R}$ is a given continuous function of the type

$$F(x, p, q) = \otimes_a\{b(x, a) \cdot p + \frac{1}{2}\mathbf{tr}(q) + \ell(x, a)\}.$$

Note that solving HJB equation (6) together with the boundary condition (7) is exactly what is needed to solve the stochastic control and game problems posed in Section II. In fact, if $\otimes_a = \min_{a\in A}$, then $v(x)$ of (3) defined by stochastic control problem is the unique viscosity solution of HJB (6)-(7). On the other hand, if $\otimes_a$ is defined as $\min_{a_1\in A_1} \max_{a_2\in A_2}$ (resp. $\max_{a_2\in A_2} \min_{a_1\in A_1}$), then $v^+(x)$ (resp. $v^-(x)$) is the unique viscosity solution of HJB equation (6)-(7). We refer more details to [3], [9], and [7].

B. Benchmark Problem

In order to have a reference for the comparison of numerical solutions to that of exact (or analytic, or closed-form) solutions, we begin by considering a benchmark problem. For this purpose, we consider the following d-dimensional HJB equation:

- Domain $O = \{x \in \mathbb{R}^d : 0 < x_i < 1, i = 1, 2, \dots d\}$.
- Equation on O: $\frac{1}{2}\Delta u + \inf_{|a|\leq 3}(a \cdot \nabla u + \ell(x, a)) = 0, \; x \in O$. where $\ell(x, a) = d + 2|x|^2 + \frac{1}{2}|a|^2$.
- Dirichlet data on the boundary ∂O : $u(x) = -|x|^2$, $x \in \partial O$.

The exact solution is given by $u(x) = -|x|^2$, with $a = 2x$, and F in (6) can be written as

$$F(x, p, q) = \inf_{|a|\leq 3}\left\{\frac{1}{2}\mathrm{tr}(q) + a \cdot p + \ell(x, a)\right\}.$$

As will be seen later, this benchmark problem offers us an opportunity to compare the exact and numerical solutions.

III. Computational Methods

With mesh size $h > 0$, we set the interior and the boundary of the discrete domain as $O^h = O \cap h\mathbb{Z}^d$ with $\partial O^h = \{x \in h\mathbb{Z}^d \setminus O^h : \exists y \in Os.t. |x - y| < h\}$, respectively.

A. Markov Chain Approximation by Central Scheme

Motivated by [16], we use Markov chain approximation to construct u_h from the following Bellman equation to approximate the solution u of HJB (6)-(7) with a given discrete parameter $h > 0$:

$$u_h(x) = \tilde{F}_h[u_h](x), \ \forall x \in O^h, \quad \text{and} \tag{8}$$
$$u_h(x) = g(x), \ \forall x \in \partial O^h, \tag{9}$$

where $\tilde{F}_h$ is of the form

$$\tilde{F}_h[\phi](x) = \otimes_a \Big\{ \sum_{i=1}^{d} \Big(p^h(x + he_i|x, a)\phi(x + he_i) + p^h(x - he_i|x, a)\phi(x - he_i) \Big) + \ell^h(x, a) \Big\}, \tag{10}$$

for some carefully chosen controlled transition probabilities $p^h(y|x, a)$ and discrete running cost $\ell^h(x, a)$ for some Markov decision process (MDP) approximating the stochastic control problem associated to HJB (6) and boundary condition (7). Note that in the above (x, a) is the state and control pair. The controlled transition probabilities should be selected in such a way that the local mean and variance match that of the controlled diffusion, which is known as local consistency. To find appropriate transition probabilities $\{p^h(y|x, a), \ell^h(x, a) : x, y \in \mathbb{R}^h, a \in A\}$, the commonly used methods are originated from upwind finite difference scheme and central finite difference scheme. To proceed, the following finite difference operator:

$$\delta_{he_i}[\phi](x) = \frac{\phi(x + he_i) - \phi(x)}{h}.$$

The above operator is called forward finite difference operator, while δ_{-he_i} is called backward finite difference operator. The average of these two

$$\bar{\delta}_{he_i}[\phi](x) := \frac{1}{2}(\delta_{he_i} + \delta_{-he_i})[\phi](x) = \frac{\phi(x + he_i) - \phi(x - he_i)}{2h}.$$

Note that $\bar{\delta}_{he_i}[\phi](x) \to \partial_{x_i}\phi(x)$, as $h \to 0$ if $\partial_{x_i}\phi(x)$ exists. For the second-order differentiation, we use the approximation $\delta_{-he_i}\delta_{he_j}[\phi](x) \to \partial_{x_i x_j}\phi(x)$, which yields in particular for $i = j$

$$\delta_{-he_i}\delta_{he_i}[\phi](x) = \frac{\phi(x + he_i) - 2\phi(x) + \phi(x - he_i)}{h^2}.$$

The discrete version of HJB (6) is obtained by replacing

$$\partial_{x_i}u(x) \leftarrow \delta_{he_i}[u_h](x), \ \partial_{x_i x_j}u(x) \leftarrow \delta_{-he_i}\delta_{he_j}[u_h](x),$$

which yields

$$F(x, (\delta_{he_i}[u_h](x))_i, (\delta_{-he_i}\delta_{he_j}[u_h](x))_{i,j}) = 0.$$

By isolating $u_h(x)$ to one side, one obtains the Bellman equation (8) with $\tilde{F}_h$ given by

$$\tilde{F}_h[\phi](x) = \frac{h^2}{2d}F(x, (\delta_{he_i}[\phi](x))_i, (\delta_{-he_i}\delta_{he_j}[\phi](x))_{i,j}) + \phi(x).$$

Algorithm 1 Value Iteration of MDP(h)

```
procedure VALUE ITERATION(h)                    ▷ h: mesh size
    for x in ∂O^h do                            ▷ set boundary value
        u^h(x) ← 0
    for x in O^h do                             ▷ Init value
        u^h(x) ← g(x)
    tol ← 1e − 5                                ▷ set tolerance
    flag ← True
    while flag do                               ▷ value iteration
        err ← 0
        for x in O^h do
            û ← u^h(x)
            u^h(x) ← F̃_h[u^h](x) using (11)-(10)
            err ← err + (û − u^h(x))^2
        if err < tol then flag ← False
    return {u^h(x):x ∈ O^h}
```

By comparing the specific form (10) of $\tilde{F}_h$, it leads to

$$p^h(x \pm he_i|x, a) = \frac{1}{2d}(1 \pm hb_i(x, a)),$$
$$\ell^h(x, a) = \frac{h^2\ell(x, a)}{d}. \tag{11}$$

One can easily verify that

$$\sum_{i=1}^{d}(p^h(x + he_i|x, a) + p^h(x - he_i|x, a)) = 1 \tag{12}$$

for all $(x, a) \in O^h \times A$. Therefore, if h is small enough such that for all p^h takes non-negative values, then (11) exactly defines an MDP with stepsize or mesh h on O^h with absorbing state ∂O^h and transition probability p^h and running cost ℓ^h. We term this problem MDP(h).

To solve the MDP(h), one can use various methods in reinforcement learning, including value iteration, policy iteration, Q-learning, SARSA. For simplicity, we present a pseudocode for the Gauss-Seidel value iteration below; more details can be found in [23].

B. Deep Neural Network

The value iteration introduced in Algorithm 1 belongs to tabular method, which means all data will be saved in the form of a multidimensional array. For instance, in the benchmark problem, the array for v^h takes the memory of $O(1/h^d)$. For small $h > 0$, the memory can be huge. It renders the method being not practical for high-dimensional problems due to the exponential growth of memory consumption. A feasible way to overcome this difficulty is the use of a deep neural network. Let $\psi : \mathbb{R}^d \times \Theta \to \mathbb{R}$ be a given function generated by a certain neural network consisted of multiple layer neurons and some activation functions with some parameter set Θ.

Example 1: Suppose the network function $\psi : \mathbb{R}^2 \times \Theta \to \mathbb{R}$ is given by two fully connected hidden layers with three neurons for each layer attached with ReLU activation function, denoted by $\mathbf{r}(x) = (\max\{x_1, 0\}, \ldots, \max\{x_d, 0\})^T, \ \forall x \in \mathbb{R}^d$. Then, the first hidden layer stands for a function

Algorithm 2 Deep Learning of MDP(h)

procedure DEEP LEARNING(h, w) ▷ h: mesh size
▷ w: weights
  Initialize neural network $\psi(x, \theta)$
  $tol \leftarrow 1e-5$ ▷ set tolerance
  $flag \leftarrow True$
  $r \leftarrow 0.01$ ▷ learning rate
  while flag **do** ▷ Gradient descent
    err $\leftarrow 0$
    for x in O^h **do**
      $err \leftarrow err + w_1(\tilde{F}_h[\psi(\cdot,\theta)](x) - \psi(x,\theta))^2$
    for x in ∂O^h **do**
      $err \leftarrow err + w_2(\psi(x,\theta) - g(x))^2$
    $\theta \leftarrow \theta - r \cdot \nabla_\theta err$
    if $err < tol$ **then** flag $\leftarrow$ False
  return $\phi^h(x, \theta)$

$\psi_1(\cdot, A_1, b_1) : \mathbb{R}^2 \to \mathbb{R}^3$ for some $A_1 \in \mathbb{R}^{3\times 2}$ and $b_1 \in \mathbb{R}^3$ such that $\psi_1(x, A_1, b_1) = \mathbf{r}(A_1 x + b_1)$, $\forall x \in \mathbb{R}^2$. Similarly, the second hidden layer is a function $\psi_2(\cdot, A_2, b_2) : \mathbb{R}^3 \to \mathbb{R}^3$ for some $A_2 \in \mathbb{R}^{3\times 3}$ and $b_2 \in \mathbb{R}^3$ such that $\psi_2(x, A_2, b_2) = \mathbf{r}(A_2 x + b_2)$, $\forall x \in \mathbb{R}^3$. The last layer is $\psi_3(\cdot, A_3, b_3) : \mathbb{R}^3 \to \mathbb{R}$ for some $A_3 \in \mathbb{R}^{1\times 3}$ and $b_3 \in \mathbb{R}$ such that $\psi_3(x, A_3, b_3) = \mathbf{r}(A_3 x + b_3)$, $\forall x \in \mathbb{R}^3$. Therefore, the network is indeed a function $\psi(\cdot, \theta) : \mathbb{R}^3 \to \mathbb{R}$ given by $\psi(x, \theta) = \psi_3 \circ \psi_2 \circ \psi_1(x, \theta)$ with $\theta = \{A_i, b_i : i = 1, 2, 3\}$. The parameter set Θ is isomorphic to $\mathbb{R}^{12}$. Under this network, to approximate the solution u, one finds 12 parameter values instead of $O(1/h^2)$ values. The universal approximation theorem (see [13]) asserts that one can learn any continuous function if there are sufficient number of neurons.

To proceed, we set the loss function as follows. For some given weight vector $w \in \mathbb{R}^{2,+}$,

$$\beta_h(\theta) = w_1 \sum_{x\in O_h} (\tilde{F}_h[\psi(\cdot,\theta)](x) - \psi(x,\theta))^2 + w_2 \sum_{x\in\partial O_h} (\psi(x,\theta) - g(x))^2.$$

In the above, the first summation is to gauge the loss in the interior and the second summation is the loss function at the boundary.

The numerical solution, denoted by u_h is given by a function $\psi(\cdot, \theta_h)$ for some parameter θ_h minimizing the loss function $\beta_h(\cdot)$ up to a certain tolerance ϵ_h, i.e., $u_h(x) = \psi(x, \theta_h)$, satisfying $\beta(\theta_h) < \arg\min_{\theta\in\Theta}\{\beta_h(\theta)\} + \epsilon_h$.

IV. Convergence Analysis

Our goal is to find sufficient conditions for the above computational methods so that the numerical solution converges to the solution of HJB equation as $h \to 0$, i.e., $\lim_{h\to 0} u_h(x) = u(x), \forall x \in O$. We focus on the convergence analysis for the deep neural network proposed in Algorithm 2. Note that the convergence analysis for the value iteration in Algorithm 1 is similar to that of Algorithm 2, and all conclusions obtained in this section for neural network also applies to value iteration only by dropping the assumption (A4).

A. Definition of Viscosity Solution

Due to the full nonlinearity, the solution of HJB equation often exhibits poor regularity. Thus it is not feasible to use the classical solution for the HJB equations. As a result, we use the notion of viscosity solution. For this purpose, we introduce super and sub test functions for each point in the domain.

1) For a given $u \in \text{USC}(\bar{O})$ and $x \in \bar{O}$, the space of super test functions is

$$J^+(u,x) = \{\phi \in C_0^\infty(\mathbb{R}^d), \text{ s.t. } \inf_{y\in\bar{O}}(\phi(y) - u(y)) = \phi(x) - u(x)\}.$$

2) For a given $u \in LSC(\bar{O})$ and $x \in \bar{O}$, the space of sub test functions is

$$J^-(u,x) = \{\phi \in C_0^\infty(\mathbb{R}^d), \text{ s.t. } \sup_{y\in\bar{O}}(\phi(y) - u(y)) = \phi(x) - u(x)\}.$$

We say that a function $u \in \text{USC}(\bar{O})$ satisfies the viscosity sub-solution property at some $x \in \bar{O}$, if for all $\phi \in J^+(u,x)$, $F(x, D\phi(x), D^2\phi(x)) \geq 0$. Similarly, a function $u \in \text{LSC}(\bar{O})$ satisfies the viscosity super-solution property at some $x \in \bar{O}$, if for all $\phi \in J^-(u,x)$, $F(x, D\phi(x), D^2\phi(x)) \leq 0$.

We define the viscosity solution of (6)-(7) as follows.

Definition 1: 1) $u \in \text{USC}(\bar{O})$ is a viscosity subsolution of (6)-(7), if (a) u satisfies the viscosity subsolution property at each $x \in O$ and (b) $u(x) \leq g(x)$ at each $x \in \partial O$. 2) $u \in \text{LSC}(\bar{O})$ is a viscosity super-solution of (6)-(7), if (a) u satisfies the viscosity super-solution property at each $x \in O$ and (b) $u(x) \geq g(x)$ at each $x \in \partial O$. 3)$u \in C(\bar{O})$ is a viscosity solution of (6)-(7), if it is a viscosity sub-solution and super-solution simultaneously.

B. Convergence

We show the convergence of the numerical solution to the desired (viscosity) solution of the HJB (6)-(7). For this purpose, we assume the following conditions hold.

(A0) F is continuous, and comparison principle and unique solvability holds for (6)-(7) in the viscosity sense.

(A1) For any $\phi_1, \phi_2 \in C(\mathbb{R}^d, \mathbb{R})$ satisfying $\phi_1 \leq \phi_2$, the following inequality holds: $\tilde{F}_h[\phi_1] \leq \tilde{F}_h[\phi_2]$.

(A2) For any $c \in \mathbb{R}$ and $\phi \in C(\mathbb{R}^d, \mathbb{R})$, the following identity holds: $\tilde{F}_h[\phi + c] = \tilde{F}_h[\phi] + c$.

(A3) There exist γ and $K > 0$ such that the following identity holds for all $\phi \in C^\infty(\mathbb{R}^d, \mathbb{R})$:

$$\lim_{(h,y)\to(0,x)} \frac{\tilde{F}_h[\phi](y) - \phi(y)}{h^\gamma} = KF(x, \phi(x), D\phi(x), D^2\phi(x)),$$

(A4) $\inf_{\theta\in\Theta} \beta_h(\theta) + \epsilon_h = o(h^\gamma)$.

(A5) There exist K and $\alpha > 0$ such that the following identity holds for all $x, y \in \bar{O}$: $|u_h(x) - u_h(y)| \leq K|x - y|^\alpha$.

In the above, (A0)-(A3) are standard assumptions for any monotonic scheme, in particular, (A3) gives the consistency. (A4) is the required restriction on the loss function and neural network. (A5) is a Hölder condition, which can be verified for a wide range of problems.

Let u^* and u_* be the USC and LSC envelopes of u_h, i.e.,

$$u^*(x) = \limsup_{(h,y)\to(0^+,x)} u_h(y),$$
$$u_*(x) = \liminf_{(h,y)\to(0^+,x)} u_h(y). \tag{13}$$

In addition, the following assumptions are further needed.

Proposition 1: Let u^* and u_* be the USC and LSC envelopes of u_h as of (13). Under (A0)-(A5), u^* and u_* are sub and super solutions of (6)-(7), respectively.

Proof: We first prove that u^* is a sub solution. The boundary condition $u_h^*(x) \geq g(x)$ holds due to (C4) and (C5), and hence we will show its viscosity sub solution property for an arbitrarily fixed $x \in O$.

For any super test function $\phi \in J^+(u^*, x)$, since $u^* - \phi$ is USC envelope of $u_h - \phi$, we can choose $x_h \to x$ such that $\phi \in J^+(u_h, x_h)$, i.e., $u_h(x_h) - \phi(x_h) \geq u_h(y) - \phi(y), \forall y \in \bar{O}$, or equivalently $\phi(y) - \phi(x_h) \geq u_h(y) - u_h(x_h), \forall y \in \bar{O}$. Together with (C1)-(C2), we have $\tilde{F}_h[\phi](y) - \phi(x_h) \geq \tilde{F}_h[u_h](y) - u_h(x_h), \forall y \in \bar{O}$. Particularly, if $y = x_h$, then $\frac{\tilde{F}_h[\phi](x_h)-\phi(x_h)}{h^\gamma} \geq \frac{\tilde{F}_h[u_h](x_h)-\phi(x_h)}{h^\gamma}$. Note that (A4) implies that its right-hand side goes to zero as $h \to 0$. Thus, one can take $\lim_h$ on both sides and obtain by (A3) that $F(x, \phi(x), D\phi(x), D^2\phi(x)) \geq 0$. This establishes the sub-solution property of u^*. One can similarly show the super solution property of u_*. ■

In the context of the HJB (6)-(7), the following assumption is to replace (A1) in the above.

(A1') $\inf_{(x,a,i)} |b_i(x, a)| > 0$ holds.

Theorem 1: Let u_h be the solution obtained from neural network in Algorithm 2. Suppose (A0), (A1'), (A4), and (A5) holds. Then, the convergence takes place in the sense that $\lim_{h\to 0} u_h(x) = u(x)$, where u is the solution of (6)-(7).

Proof: In Proposition 1, we showed that u^* and u_* are sub and super solution of (6)-(7) under assumptions (A0)-(A5). Under the same set of assumptions, one can conclude the desired convergence due to the following reasons. (a) $u^* \geq u_*$ by definition of the envelopes; (b) $u^* \leq u_*$ by comparison principle, i.e., super-solution dominates sub-solution; (c) it yields that both envelops are equal to the solution u by comparison principle of (A0); (d) since the numerical solution u_h is sandwitched between two envelops, it must converge to the solution u, i.e., $u^*(x) = u_*(x) = \lim_{(h,y)\to(0,x)} u_h(y) = \lim_{h\to 0} u_h(x)$.

It is enough to verify (A1)-(A3). The monotonicity (A1) can be verified from the representation (9) of $\tilde{F}$ and the property of $\otimes$ given by (C1)-C(2) provided that p^h takes non-negative values. This is valid for small enough h due to (A1'). (A2) is valid since p^h forms a probability measure from (12). (A3) can be verified directly by computations with $K = \frac{1}{2d}$ and $\gamma = 2$. ■

The validity of (A0) has been well studied and we will not discuss here; see [2], [3], [7]. (A4) is a technical assumption about loss function adopted for the neural network and it is only needed for the convergence on the neural network approach. By the universal approximation theorem, if the number of neurons are large enough, (A4) shall be valid. More importantly, (A4) provides the stop criterion for minimization of the loss function. (A5) is a rather difficult for its verification in theory but one can use numerically demonstrate that it is satisfied with $K = \|\ell\|_\infty$ and $\alpha = 1$.

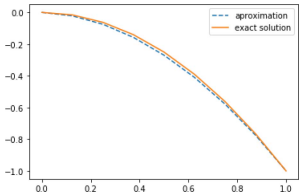

Fig. 1. Exact solution and approximation using iteration in value space.

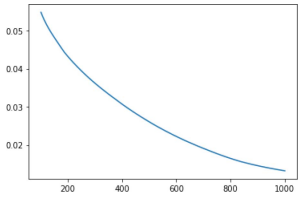

Fig. 2. Epoch number vs Loss function with $d = 1$, $h = 1/8$.

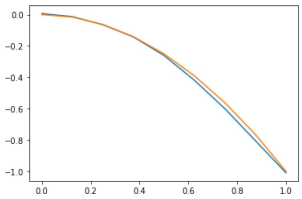

Fig. 3. Learned function from neural network with $d = 1$, $h = 1/8$.

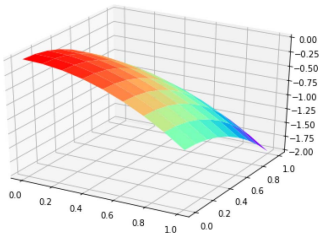

Fig. 4. Exact solution for $d = 2$.

V. Computation Examples

This section includes some numerical results of the benchmark problem in Section II-B, and the codes are listed at https://github.com/songqsh/LCSS2020.

d = 1. We first implement one-dimensional benchmark problem with $h = 1/8$. From Figure 1, it can be seen that the approximate solution by value iteration is sufficiently close to the exact solution.

The same problem has been carried out with neural network given by 32 neurons and ReLU activation function. Figure 2 records the number of epochs versus loss function.

Figure 3 records a learned function from neural network.

d = 2. In this case, the exact solution is given in Figure 4.

For neural network computation, we adopt 2 hidden layers, with 18 neurons for each layer and ReLU activation function. Figures 5 and 6 are the solutions computed from value iteration and neural network.

VI. Further Remarks

We developed approximation methods for solutions of elliptic HJB equations arising from certain stochastic control

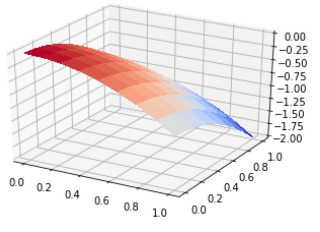

Fig. 5. Numerical solution with $d = 2$, $h = 1/8$ using value iteration.

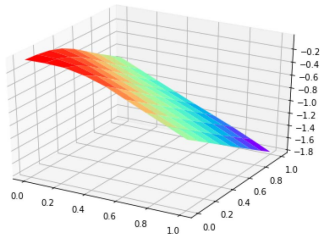

Fig. 6. Numerical solution with $d = 2$, $h = 1/8$ using neural network.

problems. We obtained convergence for both value iteration and deep learning methods on approximating MDPs under several assumptions. There are pros and cons for the deep learning based approach. The deep learning based approach opens up the possibility for solving a class of high-dimensional HJB equations. However, one of the main problems is that it does not provide a sufficient robustness. A slight change in the values of parameter can render the approximate solutions deviate from the true solutions considerably. Our current effort is devoted to improve the robustness of the approximation. A few other issues to be explored are (1) the verification of the assumption (A5), (2) convergence of MDP approximation based on upwind scheme, and (3) the cost functional with positive discount rate, etc.

REFERENCES

[1] G. Barles and P. E. Souganidis, "Convergence of approximation schemes for fully nonlinear second order equations," *Asymptotic Anal.*, vol. 4, no. 3, pp. 271–283, 1991.

[2] E. Bayraktar and Q. Song, "Solvability of Dirichlet problem with nonlinear integro-differential operator," *SIAM J. Control Optim.*, vol. 56, no. 1, pp. 292–315, 2018.

[3] M. G. Crandall, H. Ishii, and P.-L. Lions, "User's guide to viscosity solutions of second order partial differential equations," *Bull. Amer. Math. Soc. (N.S.)*, vol. 27, no. 1, pp. 1–67, 1992.

[4] W. E. J. Han and A. Jentzen, "Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations," *Commun. Math. Statist.*, vol. 5, no. 4, pp. 349–380, 2017.

[5] R. J. Elliott and N. J. Kalton, *The Existence of Value in Differential Games*. Providence, RI, USA: Amer. Math. Soc., 1972.

[6] L. C. Evans, *Partial Differential Equations*. Providence, RI, USA: Amer. Math Soc., 1998.

[7] W. H. Fleming and H. M. Soner, *Controlled Markov Processes and Viscosity Solutions*, 2nd ed. New York, NY, USA: Springer, 2006.

[8] W. H. Fleming and P. E. Souganidis, "On the existence of value functions of two-player, zero-sum stochastic differential games," *Indiana Univ. Math. J.*, vol. 38, no. 2, pp. 293–314, 1989.

[9] W. H. Fleming and R. W. Rishel, *Deterministic and Stochastic Optimal Control*. Berlin, Germany: Springer, 1975.

[10] J. Han, A. Jentzen, and E. Weinan, "Solving high-dimensional partial differential equations using deep learning," *Proc. Nat. Acad. Sci.*, vol. 115, no. 34, pp. 8505–8510, 2018.

[11] A. Hening and D. H. Nguyen, "The competitive exclusion principle in stochastic environments," 2018. [Online]. Available: arXiv:1810.00954.

[12] A. Hening and K. Tran, "Harvesting and seeding of stochastic populations: Analysis and numerical approximation," 2019. [Online]. Available: arXiv:1911.00734.

[13] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Netw.*, vol. 3, no. 5, pp. 551–560, 1990.

[14] C. Huré, H. Pham, and X. Warin, "Some machine learning schemes for high-dimensional nonlinear PDEs," 2019. [Online]. Available: arXiv:1902.01599.

[15] Z. Jin, G. Yin, and F. Wu, "Optimal reinsurance strategies in regime-switching jump diffusion models: Stochastic differential game formulation and numerical methods," *Insur. Math. Econ.*, vol. 53, no. 3, pp. 733–746, 2013.

[16] H. J. Kushner and P. Dupuis, *Numerical Methods for Stochastic Control Problems in Continuous Time*. New York, NY, USA: Springer, 2001.

[17] H. J. Kushner, "Numerical methods for stochastic control problems in continuous time," *SIAM J. Control Optim.*, vol. 28, no. 5, pp. 999–1048, 1990.

[18] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," *J. Comput. Phys.*, vol. 375, pp. 1339–1364, Dec. 2018.

[19] Q. Song, "Convergence of Markov chain approximation on generalized HJB equation and its applications," *Automatica*, vol. 44, no. 3, pp. 761–766, 2008.

[20] Q. Song and G. Yin, "Rates of convergence of numerical methods for controlled regime-switching diffusions with stopping times in the costs," *SIAM J. Control Optim.*, vol. 48, no. 3, pp. 1831–1857, 2009.

[21] Q. Song, G. Yin, and Z. Zhang, "Numerical methods for controlled regime-switching diffusions and regime-switching jump diffusions," *Automatica*, vol. 42, no. 7, pp. 1147–1157, 2006.

[22] Q. Song, G. Yin, and Z. Zhang, "Numerical solutions for stochastic differential games with regime switching," *IEEE Trans. Autom. Control*, vol. 53, no. 2, pp. 509–521, Mar. 2008.

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[24] J. Yong and X. Y. Zhou, *Stochastic Controls*. New York, NY, USA: Springer, 1999.