High-Dimensional HJBs: Mean-Field Limits, McKean-Vlasov Equations, and More

G. Yin

Wayne State University

IPAM Workshop I:
High Dimensional Hamilton-Jacobi Methods in Control and Differential Games

Outline

1. New Motivational Examples of High-Dimensional HJBs
2. Some Numerical Examples
3. Further Remarks

Why do we need to study high-dim HJBs?

- Stochastic control problems & high-dimensional HJBs
- Time inconsistent control problems
  - McKean-Vlasov control problems
- Mean-field type of controls for switching diffusions

New Motivational Examples of High-Dimensional HJBs

A non-classical “LQG” problem

joint work with Yun Li, Qingshuo Song, and Fuke Wu

A non-classical "LQG" problem

$$dX_t = (A_t X_t + B_t u_t)dt + \sigma_t dW_t,$$

- A_t, B_t, and σ_t are suitable functions of t
- W_t: standard real-valued Brownian motion
- u_t: control
- new angle: terminal cost not quadratic w.r.t. terminal state X_T, but in the distribution of X_T

$$\mathbb{E}[g_1(X_T)]\mathbb{E}[g_2(X_T)]$$

  - e.g., Markowitz mean-variance problem

$$D_1\mathbb{E}[X_T^2] + D_2(\mathbb{E}[X_T])^2 + D_3\mathbb{E}[X_T]$$

  - The $(\mathbb{E}X_T)^2$ in the cost makes it non-classical. The traditional HJB approach does not work.

Time Inconsistency

- Time inconsistency is a situation in which a decision-maker's preferences change over time in such a way that a preference can become inconsistent at another point in time.
  - an optimal policy/strategy made at a given moment is not optimal at a later time.
  - Which one do you prefer,
    (a) to be given 500 dollars today or 505 dollars tomorrow?
    (b) to be given 500 dollars 365 days from now or 505 dollars 366 days from now?

How do we solve the non-classical LQG?

- A game-theoretic framework (T. Björk, A. Murgoci, X.Y. Zhou (2014))
- The work of Yong (2015)

- Our idea is to treat the value function by increasing the dimension of the space with the addition of the probability law involved

For simplicity, we use a scalar system to illustrate

- μ: distribution on Borel sets $\mathscr{B}(\mathbb{R})$
- $f : \mathbb{R} \mapsto \mathbb{R}$
- $\langle f, \mu \rangle := \int_{\mathbb{R}} f(x)\mu(dx)$
- $[\mu]_m := \langle x^m, \mu \rangle$, mth moment
- $\mathscr{P}_m$: collection of μ having a finite mth moment $[\mu]_m$
- $\mu \in \mathscr{P}_m$: metrized with Wasserstein distance
- δ_x: Dirac measure, $\delta_x \in \mathscr{P}_m$ for any $m \geq 1$, since $[\delta_x]_m = x^m$
- S: metric space
- $\mathscr{M}(S)$: collection of of Borel measurable functions $f : S \mapsto \mathbb{R}$

HJB equations

value function $V(t,\mu)$ solves the HJB (μ_t probability law of X_t)

$$\begin{aligned} &\inf_{a\in\mathscr{M}(\mathbb{R})} \langle H(t,\cdot,\mu,v,a(\cdot)),\mu\rangle + \frac{1}{2}\sigma_t^2\langle \partial_{x\mu} v(t,\mu)(\cdot),\mu\rangle + \partial_t v(t,\mu) = 0 \\ &v(T,\mu) = g(\mu), \end{aligned} \tag{1.1}$$

Consider

$$\begin{aligned} b(t,x,u) &= A_t x + B_t u, \\ \ell(t,x,u) &= Q_t u^2, \\ g(\mu_T) &= D_1[\mu_T]_2 + D_2[\mu_T]_1^2 \\ &= D_1 \mathbb{E}[X_T^2] + D_2(\mathbb{E}[X_T])^2. \end{aligned} \tag{1.2}$$

Let $L = [L_1, L_2, L_3] : C^1((0,T), \mathbb{R}^3) \mapsto C((0,T), \mathbb{R}^3)$ be operators acted on the vector function $\phi = (\phi_1, \phi_2, \phi_3)$

$$\begin{aligned} L_1\phi(t) &= \phi_1'(t) - \frac{B_t^2}{Q_t}\phi_1^2(t) + 2A_t\phi_1(t), \\ L_2\phi(t) &= \phi_2'(t) - \frac{B_t^2}{Q_t}\phi_2^2(t) - \frac{2B_t^2}{Q_t}\phi_1(t)\phi_2(t) + 2A_t\phi_2(t), \\ L_3\phi(t) &= \phi_3'(t) + \sigma_t^2\phi_1(t). \end{aligned}$$

Then we have the Riccati equation

$$L\phi(t) = 0, \forall t \in (0,T), \text{ with } \phi(T) = (D_1, D_2, 0). \tag{1.3}$$

Theorem 1.1

Suppose $Q_t > 0$ *for all* t*, and there exists* $\phi \in C^1((0,T),\mathbb{R}^3)$ *for Ricatti system* (1.3)*. Then the pair* (v, a^*) *given by*

$$v(t,\mu) = \phi_1(t)[\mu]_2 + \phi_2(t)[\mu]_1^2 + \phi_3(t),$$
$$a^*(t,\mu,x) = -\frac{B_t}{Q_t}(\phi_1(t)x + \phi_2(t)[\mu]_1)$$

solves the HJB equation (1.1) *and the optimality condition. Moreover, if* $J(u_\cdot^*)$ *with* $b(\cdot)$ *and* $g(\cdot)$ *given as before &* $(X^*, u_\cdot^*)$ *satisfying*

$$X_t = x + \int_0^t b(s, X_s, u_s)ds + \int_0^t \sigma_s dW_s,$$
$$J(u_\cdot) = \mathbb{E}\Big[\int_0^T \ell(t, X_t, u_t)dt\Big] + g(\mu_T),$$

and

$$u_t^* = a^*(t, \mu_t^*, X_t^*),$$

then $(X^*, u_\cdot^*) =$ *(optimal trajectory, optimal control), and the optimal value is* $V^* = v(0, \delta_x)$.

An example

$$J(u) = \mathbb{E}\int_0^T u_s^2 ds + (\mathbb{E}X_T)^2$$
$$dX_t = u_t dt + dW_t,\ X_0 = x.$$

Riccati equation becomes

$$L\phi(t) = 0, t \in (0,T),\ (\phi_1(T), \phi_2(T), \phi_3(T)) = (0,1,0).$$

Solution:

$$\phi_1(t) = \phi_3(t) = 0,\ \phi_2(t) = \frac{1}{1+T-t}.$$

Optimal control and value function are

$$u_t^* = -\frac{\mathbb{E}[X_t^*]}{1+T-t},$$
$$v(t,\mu) = \frac{1}{1+T-t}\Big(\int_{\mathbb{R}} x\mu(dx)\Big)^2,$$

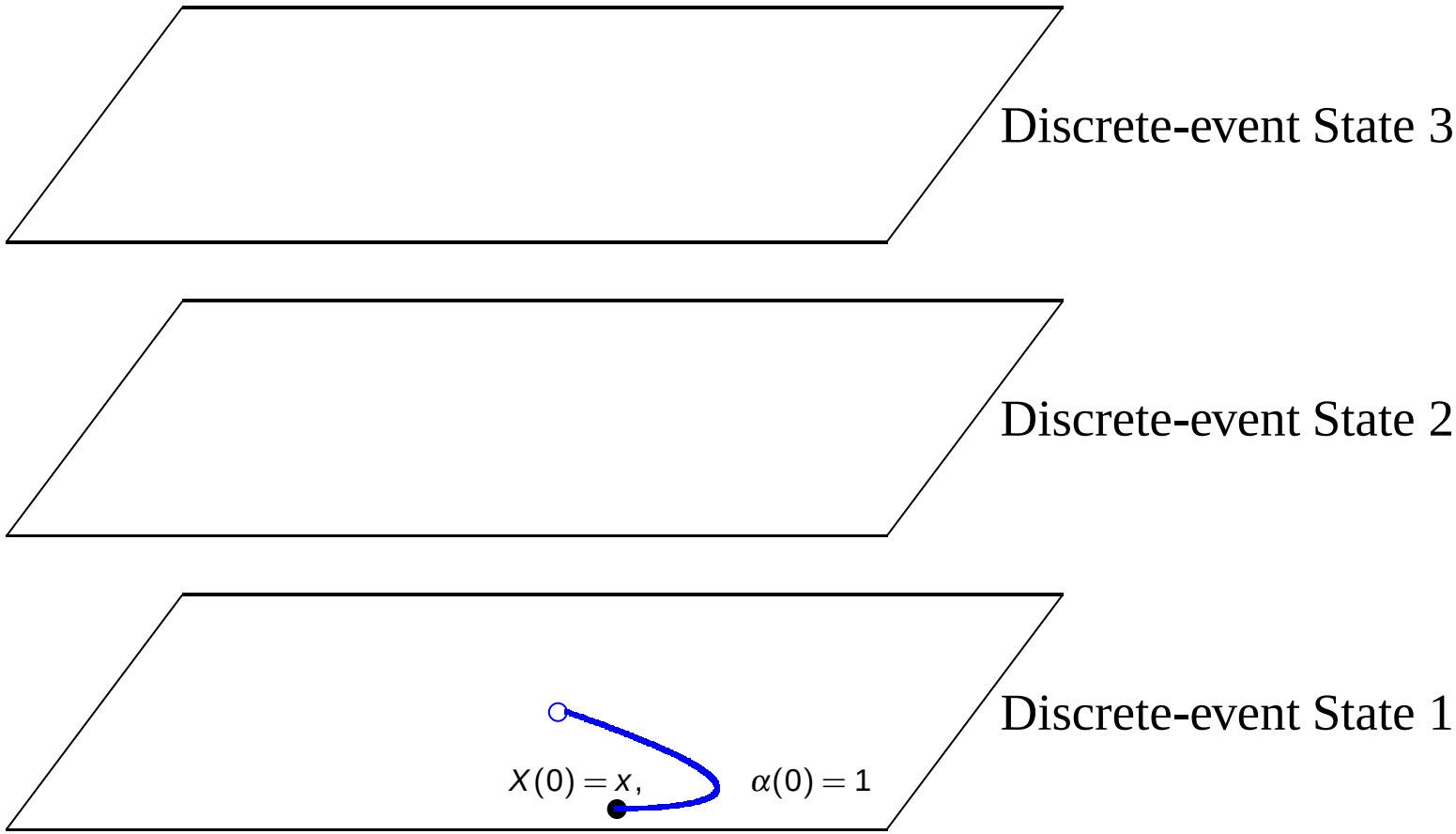
which implies the optimal value is

$$V^* = \frac{x^2}{1+T}.$$

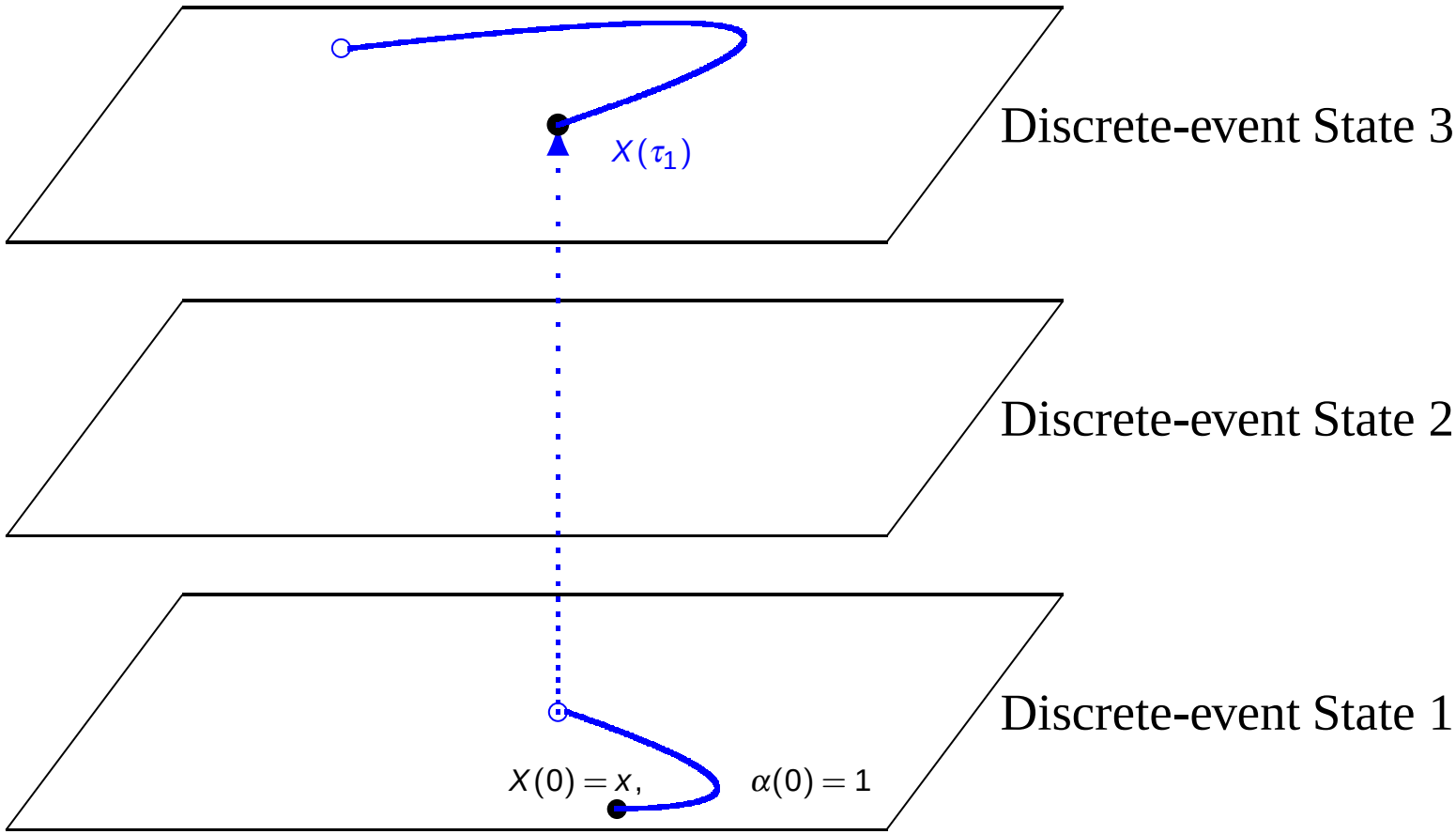
Switching Diffusions

joint work with Chao Zhu since (2007), book (2010);
with Dang Hai Nguyen (2016), (2018)

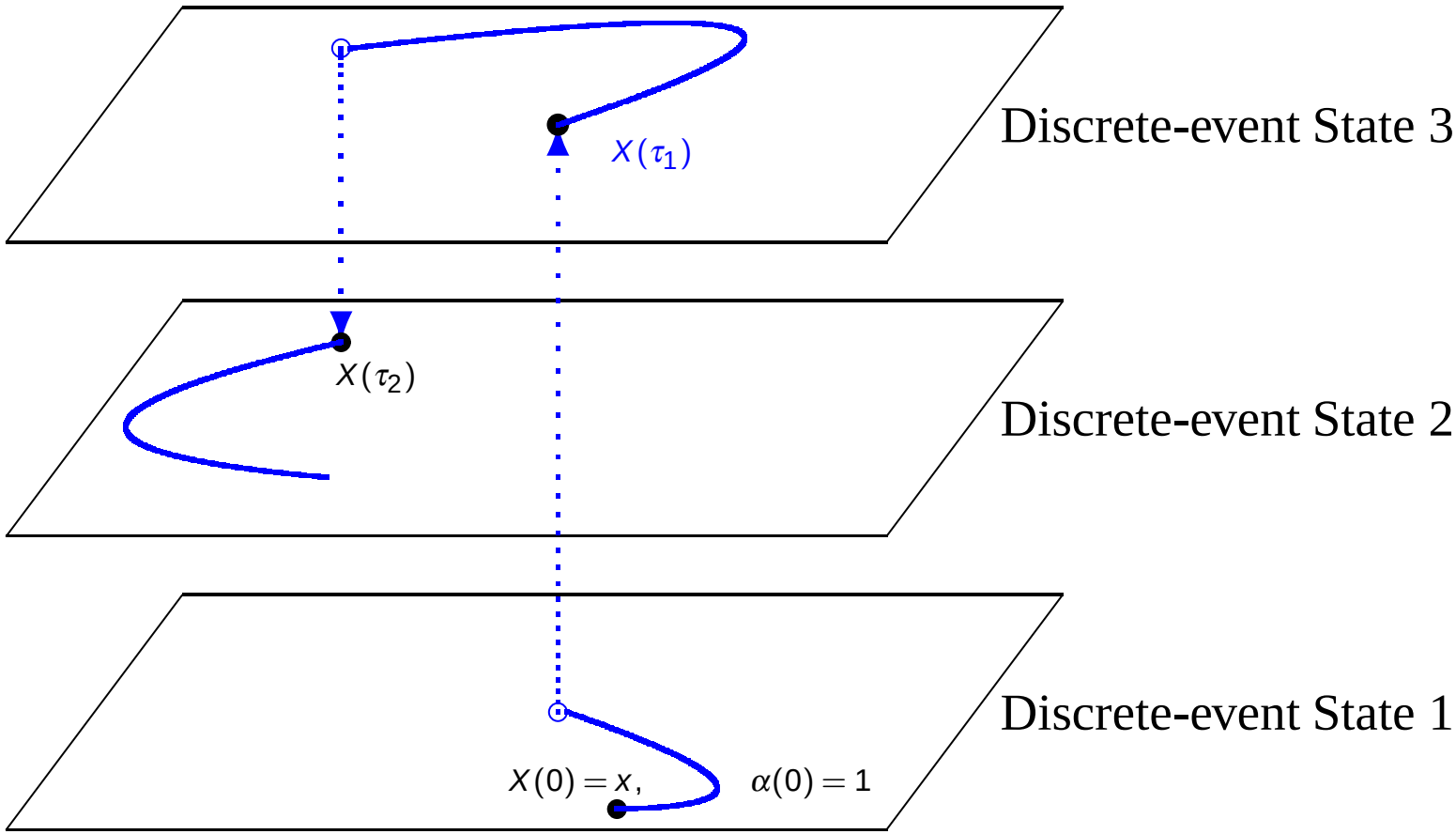
Switching Diffusion: An Illustration



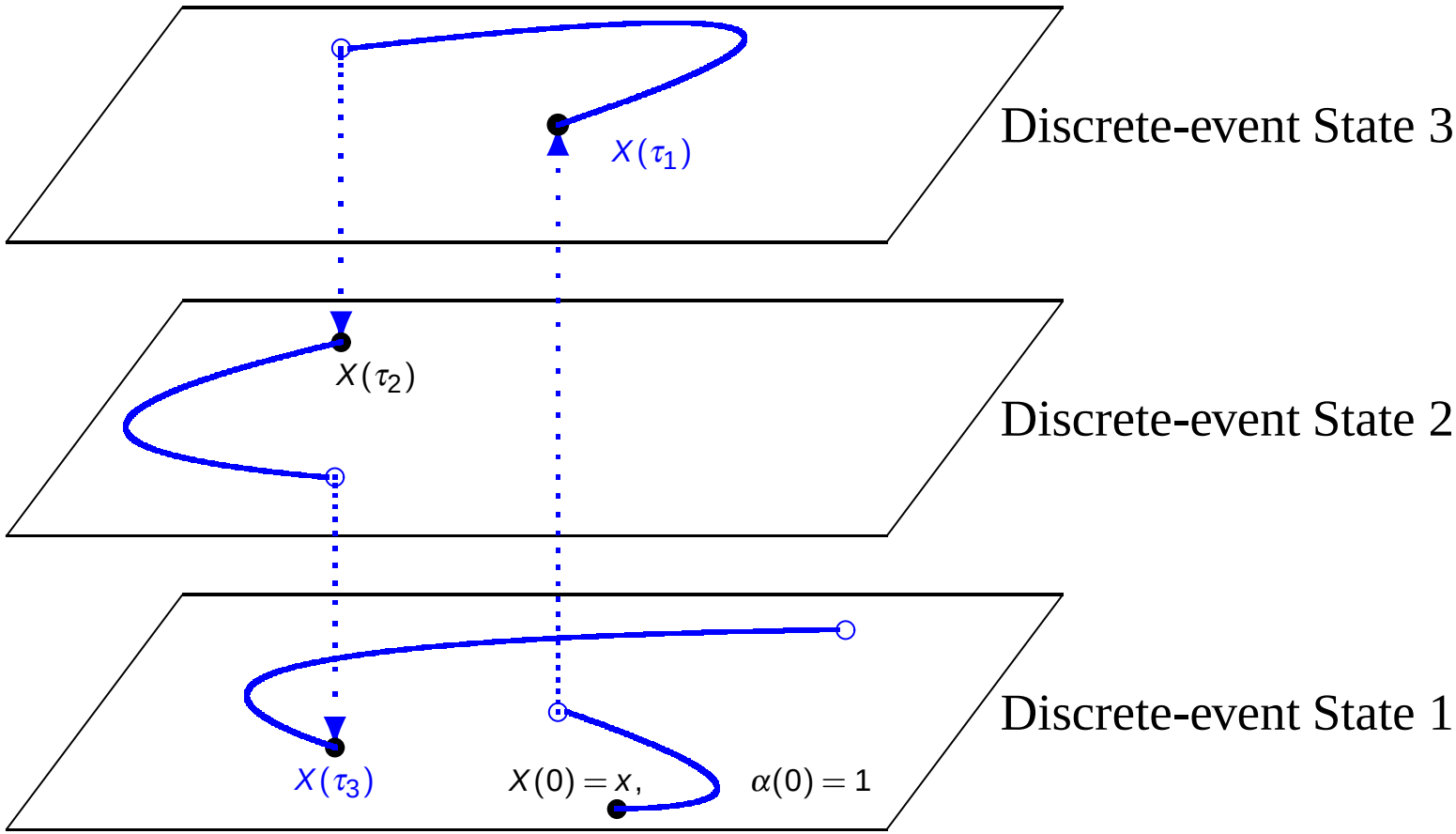
Switching Diffusion: An Illustration



Switching Diffusion: An Illustration



Switching Diffusion: An Illustration



Switching Diffusion: An Illustration

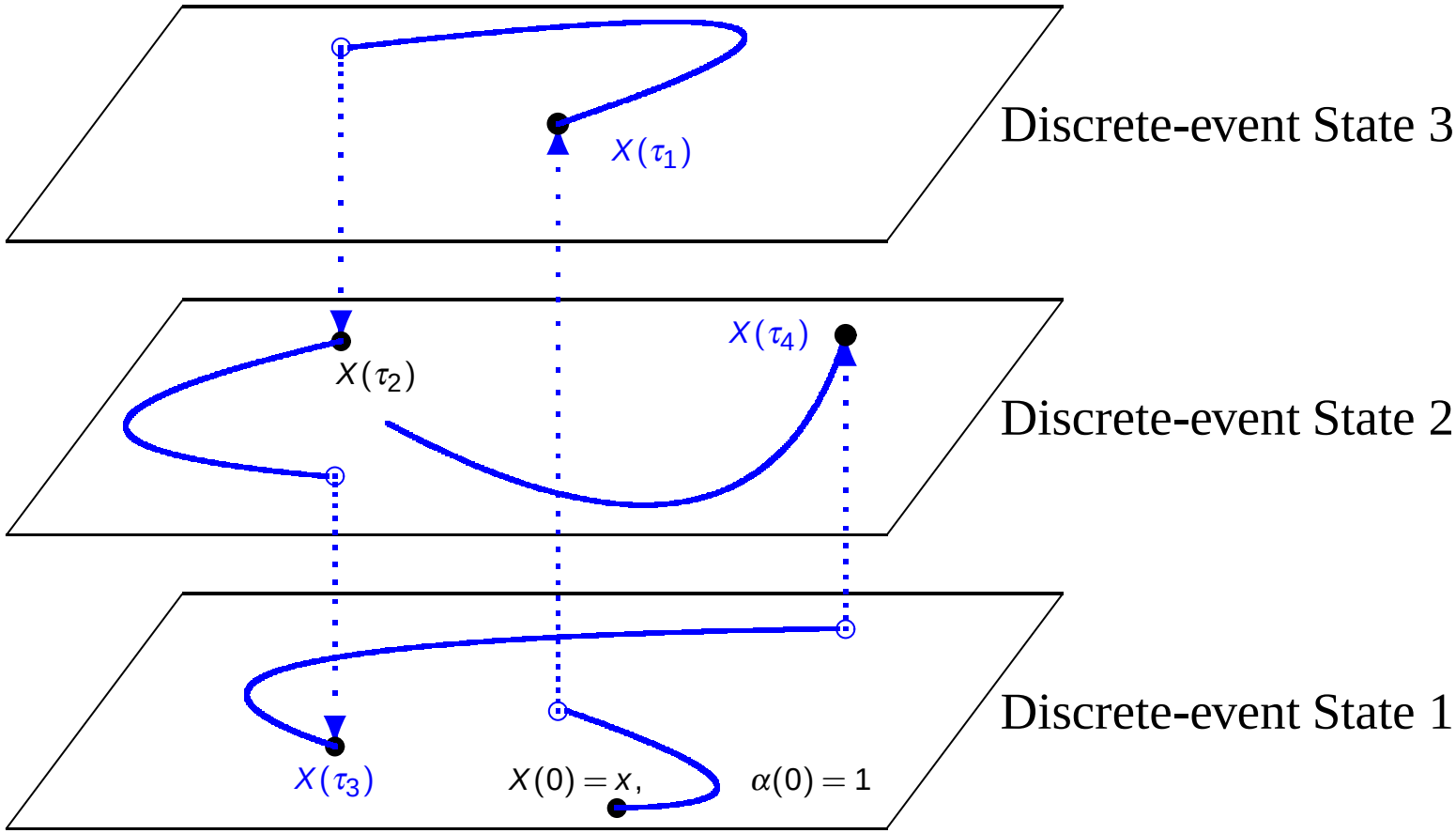


Figure 1: A "Sample Path" of Switching Diffusion $(X(t), \alpha(t))$.

Main Features

- continuous dynamics & discrete events coexist
- switching is used to model random environment or other random factors that cannot be formulated by the usual differential equations
- problems naturally arise in applications such as distributed, cooperative, and non-cooperative games, wireless communication, target tracking, reconfigurable sensor deployment, autonomous decision making, learning, etc.
- traditional ODE or SDE models are no longer adequate
- non-Gaussian distribution

Mean-Field Limits

joint work with Son Luu Nguyen and Tuan Anh Hoang

Some Previous Works of Mean-Field Limits etc.

- Kac (1956), time evolution of systems with long range weak interactions
- McKean (1966) Proc. Nat. Acad. Sci.
- Dawson (1975)
- Dawson and Vaillancourt (1995), studied so-called McKean-Vlasov limits of exchangeable infinite systems (diffusions) of infinite particles that arises from the original work of McKean related to the Boltzmann model of rarefied gases
- Graham (1990), system of diffusing particles alternate between two states, time-change, stopping times
- Kolokoltsov (2010), nonlinear Markov processes
- Kurtz and Xiong (1999), system of infinite many SDEs for the locations and weights of a collection of particles, a common space-time Gaussian white noise. Their model contains infinitely many exchangeable particles so ergodic theory can be applied

Limit measure?

A mean-field system of N particles ($N \gg 1$) with random switching,

$$dX_t^{i,N} = b\left(X_t^{i,N}, \frac{1}{N}\sum_{j=1}^{N}\delta_{X_t^{j,N}}, \alpha(t_-), u_t\right)dt + \sigma\left(X_t^{i,N}, \frac{1}{N}\sum_{j=1}^{N}\delta_{X_t^{j,N}}, \alpha(t_-), u_t\right)dW_t^i, \tag{1.4}$$

The idea of mean-field control is to replace $\frac{1}{N}\sum_{j=1}^{N}\delta_{X_t^{j,N}}$ by a limit measure. Note the dependence of $\alpha(t_-)$. What should the limit measure look like?

Limit: Stochastic McKean-Vlasov Equations (Nguyen, Yin, Hoang, SPA (2019)); Maximum Principle: Nguyen, Nguyen, Y, ESAIM COCV (2020))

Theorem 1.2

Under (A1) *and* (A2), *for* $f(\cdot,i_0)\in C_c^2(\mathbb{R}^d)$ *and* $i_0\in\mathbb{S}$, *the system*

$$\begin{aligned}\langle \mu(t), f(\cdot,\alpha(t))\rangle =& \langle \mu_0, f(\cdot,\alpha(0))\rangle + \int_0^t \Big\langle \mu(s), \mathscr{L}(\mu(s)) f(\cdot,\alpha(s_-))\Big\rangle ds \\ &+ \sum_{i_0,j_0\in\mathbb{S}} \int_0^t \Big\langle \mu(s), f(\cdot,j_0)-f(\cdot,i_0)\Big\rangle dM_{i_0j_0}(s), \end{aligned} \tag{1.5}$$

has a unique solution $\mathcal{L}\big(y(t)\big|\mathscr{F}_{t_-}^{\alpha}\big)$ *in* $D\big([0,T],\mathcal{M}_1\big)$ *for all* $0\le t\le T$.

$y(t)$ *is the unique solution of*

$$\begin{cases} dy(t) &= b\Big(y(t),\mu_\alpha(t),\alpha(t_-)\Big)dt+\sigma\Big(y(t),\mu_\alpha(t),\alpha(t_-)\Big)d\tilde{w}(t),\\ \mu_\alpha(t) &= \mathcal{L}\big(y(t)\big|\mathscr{F}_{t_-}^{\alpha}\big),\quad \mathcal{L}(y(0))=\mu_0, \end{cases}$$

where $\tilde{w}(\cdot)$ *is a standard Brownian motion independent of* $\alpha(\cdot)$.

What's new in our work?

- The limit of the empirical measures is not deterministic but a random measure that depends on the history of the Markovian switching process

What's new in our work?

- The limit of the empirical measures is not deterministic but a random measure that depends on the history of the Markovian switching process
  - the stochastic McKean-Vlasov equation in the limit is driven by martingales associate with the Markov switching process
  - Main difficulty: to characterize the limit using the martingale problem formulation does not work.
  - Compare to Kurtz and Xiong, we no longer have infinitely many exchangeable particles thus the existing ergodic theory is not applicable.

What's new in our work?

- The limit of the empirical measures is not deterministic but a random measure that depends on the history of the Markovian switching process
  - the stochastic McKean-Vlasov equation in the limit is driven by martingales associate with the Markov switching process
  - Main difficulty: to characterize the limit using the martingale problem formulation does not work.
  - Compare to Kurtz and Xiong, we no longer have infinitely many exchangeable particles thus the existing ergodic theory is not applicable.
- other works use conditional mean fields:
  - In Buckdahn, Li, and Ma, mean-field control with partial observations is treated. The idea of converting a partially observable system into a fully observable systems is used. Then naturally, a conditional mean-field is used conditioning on the observations.
  - In Carmona and Zhu, a probabilistic approach for mean-field games with major and minor players is provided. A conditional mean-field with condition on the major player is used.

Some Numerical Examples

Deep Filtering

joint work with Qing Zhang and Le Yi Wang

Deep Filtering

- A deep neural network (DNN) is a neural network (NN) with several hidden layers.
- The deepness of the network is measured by the number of layers used.
- We only consider a fully connected NN and there are no connections between nodes in the same layer.
- Backpropagation is an often used main driver in DNN training, which is an algorithm for supervised learning of artificial neural networks using gradient descent procedures.
- Given a NN and a loss or error function, the scheme calculates the gradient of the loss function w.r.t. the weights of the neural network.

Backpropagation requires

(a) A data set consisting of a fixed pairs of input and output variables;

(b) A feedforward NN with parameters given by the weights θ;

(c) A loss (error) function $L(\theta)$ defining the error between the desired output and the calculated output.

A typical Deep Neural Network diagram (Nielsen's book)

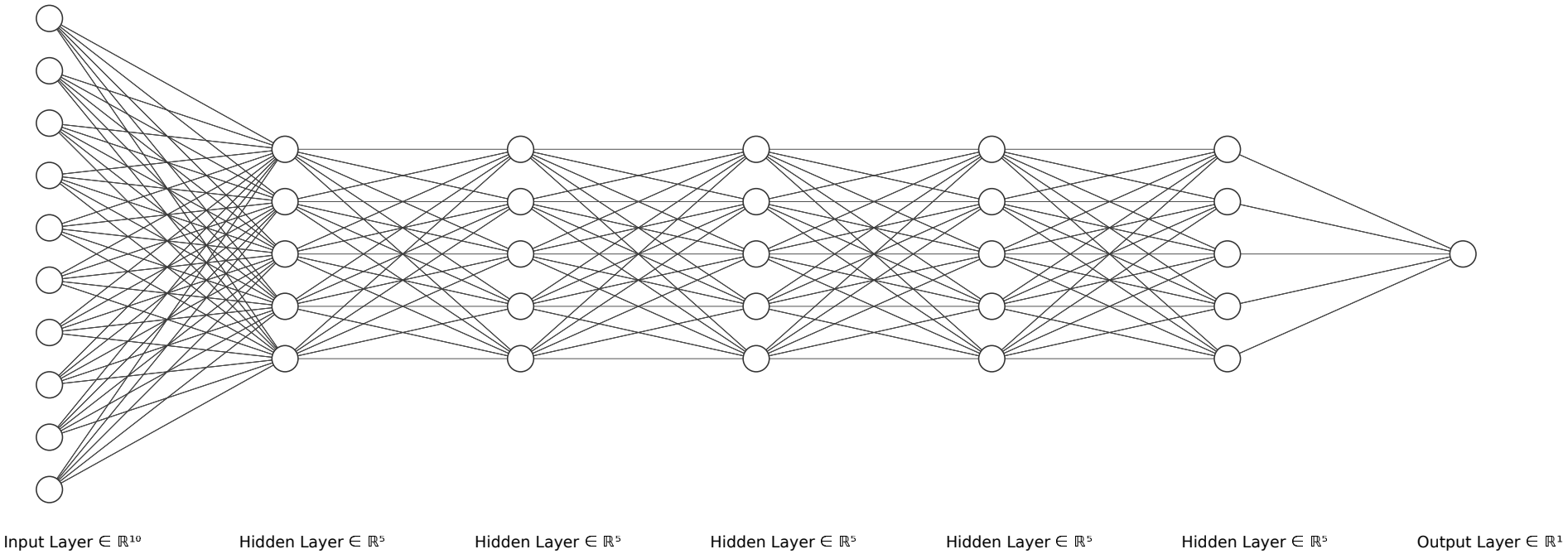


Figure 2: Deep Neutral Network

Deep Filtering

- generate Monte Carlo samples

Deep Filtering

- generate Monte Carlo samples
- use these samples to train a deep neutral network

Deep Filtering

- generate Monte Carlo samples
- use these samples to train a deep neutral network
- observation process is used as input to the DNN and the state from the Monte Carlo samples is used as target

Deep Filtering

- generate Monte Carlo samples
- use these samples to train a deep neutral network
- observation process is used as input to the DNN and the state from the Monte Carlo samples is used as target
- a least squares error of the target and calculated output is used as a loss function for network training to come up with weight vectors.

Deep Filtering

- generate Monte Carlo samples
- use these samples to train a deep neutral network
- observation process is used as input to the DNN and the state from the Monte Carlo samples is used as target
- a least squares error of the target and calculated output is used as a loss function for network training to come up with weight vectors.
- these weight vectors are used to another set of Monte Carlo samples of an actual dynamic model

Deep Filtering

- generate Monte Carlo samples
- use these samples to train a deep neutral network
- observation process is used as input to the DNN and the state from the Monte Carlo samples is used as target
- a least squares error of the target and calculated output is used as a loss function for network training to come up with weight vectors.
- these weight vectors are used to another set of Monte Carlo samples of an actual dynamic model
- the calculated DNN output is termed a deep filter (DF).

Set up

x_n: state process s.t.

$$x_{n+1} = F_n(x_n, u_n),\ x_0 = x,\ n = 0, 1, 2, \ldots,$$

for suitable F_n and system noise $\{u_n\}$.
y_n: observation process given by

$$y_n = H_n(x_n, v_n),$$

$\{v_n\}$: observation noise

- N_{seed}: number of training sample paths
- n_0: training window for each sample path
- For any fixed $\kappa = n_0, \ldots, N$ with a fixed ω, take $\{y_\kappa(\omega), y_{\kappa-1}(\omega), \ldots, y_{\kappa-n_0+1}(\omega)\}$ as the input vector to the neural network and $x_\kappa(\omega)$ as the target. In what follows, we suppress the ω dependence.
- Fix x_κ, let ξ_ℓ denote the neural network output at iteration ℓ, which depends on θ and noise $\{\zeta_\ell\}$
- goal: find NN weights θ to minimize the loss function

$$L(\theta) = \frac{1}{2} E |\xi_\ell(\theta, \zeta_\ell) - x_\kappa|^2. \tag{2.1}$$

- Use stochastic approximation algorithms (see Kushner and Yin (2013), 2nd Ed.)

$$\theta_{\ell+1} = \theta_\ell - \gamma \frac{\partial \xi_\ell(\theta_\ell, \zeta_\ell)}{\partial \theta} [\xi_\ell(\theta_\ell, \zeta_\ell) - x_\kappa]. \tag{2.2}$$

- Define $\theta^\gamma(t) = \theta_\ell$ for $t \in [\ell\gamma, \ell\gamma + \gamma)$. It can be shown that the weak limit of $\theta^\gamma(\cdot)$ is $\theta(\cdot)$ s.t.

$$\dot{\theta}(t) = -\frac{\partial \overline{\xi}(\theta(t))}{\partial \theta} [\overline{\xi}(\theta(t)) - x_\kappa], \tag{2.3}$$

  where $\overline{\xi}(\theta)$ is the "average" of $\xi_\ell(\theta, \zeta)$.

- Under suitable conditions, it can also be shown that $\theta^\gamma(\cdot + t_\gamma)$ converges to θ^*, which is a solution of $\overline{\xi}(\theta^*) - x_\kappa = 0$, as $\gamma \to 0$, where $t_\gamma \to \infty$ as $\gamma \to 0$.

Robustness of Deep Filtering

- nominal model (NM): an estimated model
- actual model (AM):

$$\begin{cases} (\mathrm{NM}): \begin{cases} x_{n+1} = x_n + 0.1\eta x_n + \sqrt{\eta}\ \sigma u_n,\ x_0 = x, \\ y_n = x_n + \sigma_0^{\mathrm{NM}} v_n,\ n = 0,1,2,\ldots, \end{cases} \\ (\mathrm{AM}): \begin{cases} x_{n+1} = x_n + 0.1\eta x_n + \sqrt{\eta}\ \sigma u_n,\ x_0 = x, \\ y_n = x_n + \sigma_0^{\mathrm{AM}} v_n,\ n = 0,1,2,\ldots, \end{cases} \end{cases}$$

- The recursion is so defined that under the interpolation $x^\eta(t) = x_n$ on $t \in [n\eta, b\eta + \eta)$, $x^\eta(\cdot)$ converges weakly to $x(\cdot)$ so that $x(\cdot)$ is a solution to the Kalman filter in continuous time.

σ_0^{NM}	0.1	0.5	1.0	1.5	2.0	2.5
DF	8.89	5.71	5.64	5.62	5.62	5.62
KF	6.54	4.08	4.59	5.33	6.04	6.70

Table 1: Error dependence on σ_0^{NM}.

- With fixed $\sigma_0^{\mathrm{AM}} = 0.5$, vary σ_0^{NM}.
  - The KF depends heavily on nominal observation noise. On the other hand, the DF is more robust when $\sigma_0^{\mathrm{NM}} \geq 0.5$.

σ_0^{AM}	0.1	0.5	1.0	1.5	2.0	2.5
DF	5.25	5.71	6.99	8.71	10.63	12.65
KF	2.90	4.08	6.49	9.15	11.88	14.59

Table 2: Error dependence on σ_0^{NM}.

- with $\sigma_0^{\mathrm{NM}} = 0.5$, vary σ_0^{AM}.
- As the actual observation noise increases, both DF and KF deteriorate.
- The DF appears to be more robust than the KF because it is less sensitive to such changes than the KF.

Nonlinear Models

In this section, we consider nonlinear (NL) models and comparison of the DF with the corresponding extended Kalman filter. We consider the two (NM and AM) models:

$$\begin{cases} (\mathrm{NM}): \begin{cases} x_{n+1} = x_n + \eta \sin(5x_n) + \sqrt{\eta}\ \sigma u_n,\ x_0 = x, \\ y_n = x_n + \sigma_0^{\mathrm{NM}} v_n,\ n = 0,1,2,\ldots, \end{cases} \\ (\mathrm{AM}): \begin{cases} x_{n+1} = x_n + \eta \sin(5x_n) + \sqrt{\eta}\ \sigma u_n,\ x_0 = x, \\ y_n = x_n + \sigma_0^{\mathrm{AM}} v_n,\ n = 0,1,2,\ldots, \end{cases} \end{cases}$$

σ_0^{NM}	0.1	0.5	1.0	1.5	2.0	2.5
DF	12.24	7.75	7.62	7.58	7.56	7.56
EKF	9.22	5.58	6.63	8.29	10.13	12.14

Table 3: (NM=NL, AM=NL): Error dependence on σ_0^{NM}.

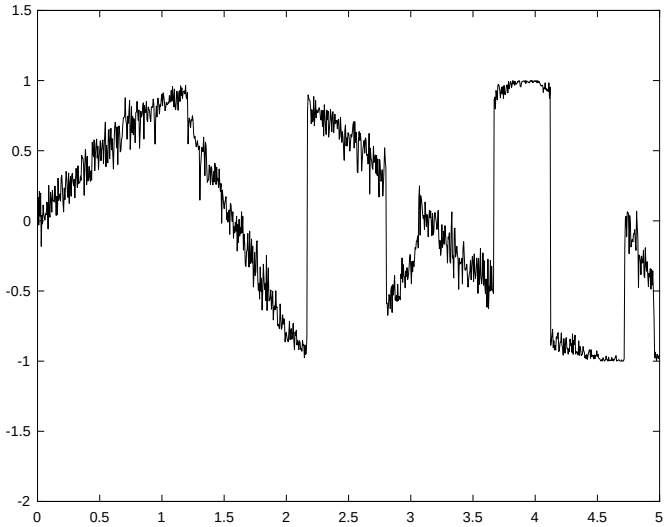
- The deep filter is more robust and less dependent on nominal observation noise changes when compared against the corresponding extended Kalman filter.
- when training the DF, the observation noise in training data should not be too small. This is typical in DNN training.

A Random-Switching Model

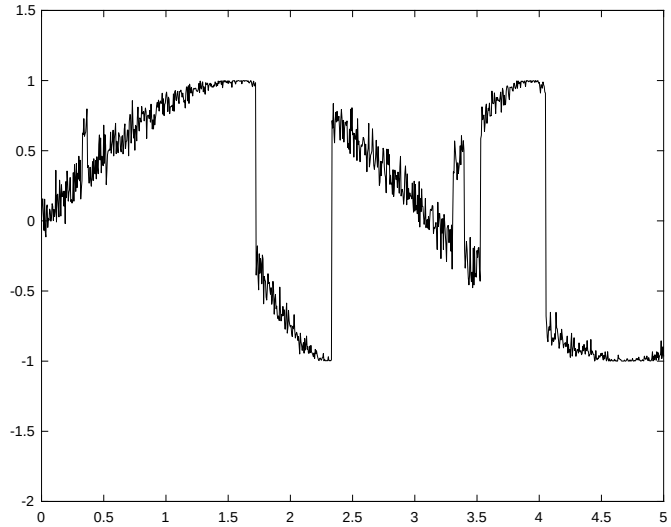
Consider $\alpha(t) \in \{1,2\}$ to be a continuous-time Markov chain with generator $Q = \begin{pmatrix} -2 & 2 \\ 2 & -2 \end{pmatrix}$ and

$$\begin{cases} (\mathrm{NM}): \begin{cases} x_n = \sin(n\eta\alpha_n + \sigma u_n),\ n = 0,1,2,\ldots, \\ y_n = x_n + \sigma_0^{\mathrm{NM}} v_n, \end{cases} \\ (\mathrm{AM}): \begin{cases} x_n = \sin(n\eta\alpha_n + \sigma u_n),\ n = 0,1,2,\ldots. \\ y_n = x_n + \sigma_0^{\mathrm{AM}} v_n. \end{cases} \end{cases}$$

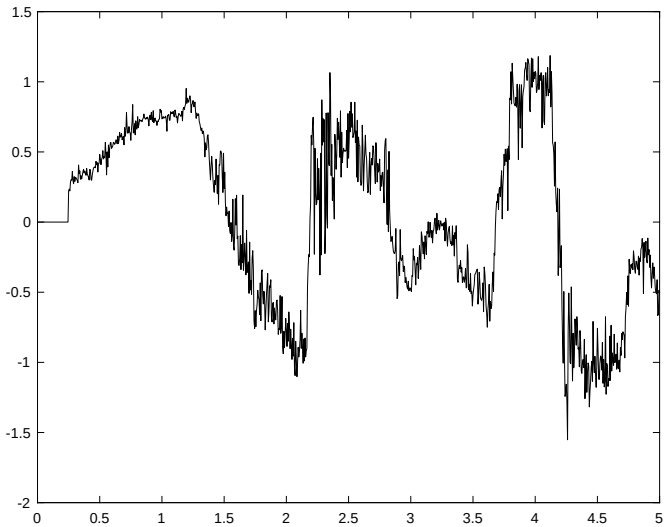
Take $\sigma = 0.1$ and $\sigma_0 = 0.3$. Two sample paths of x_n, $\widetilde{x}_n$ are generated. DF appears to be effective and catches up quickly the jumps of x_n.



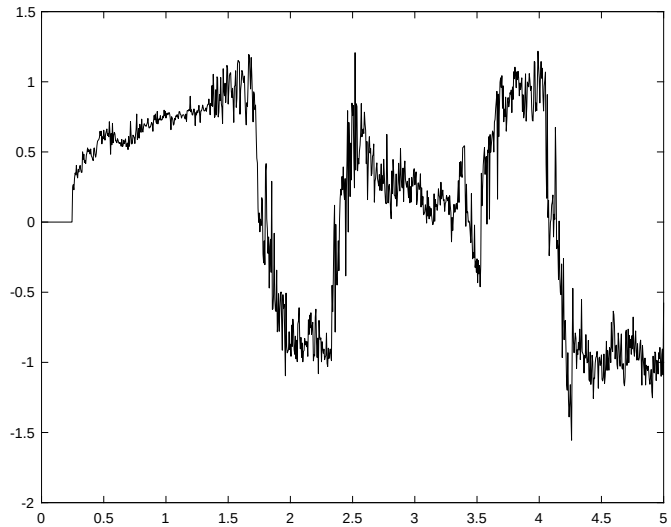
(a) State x_n (sample path 1)



(b) State x_n (sample path 2)

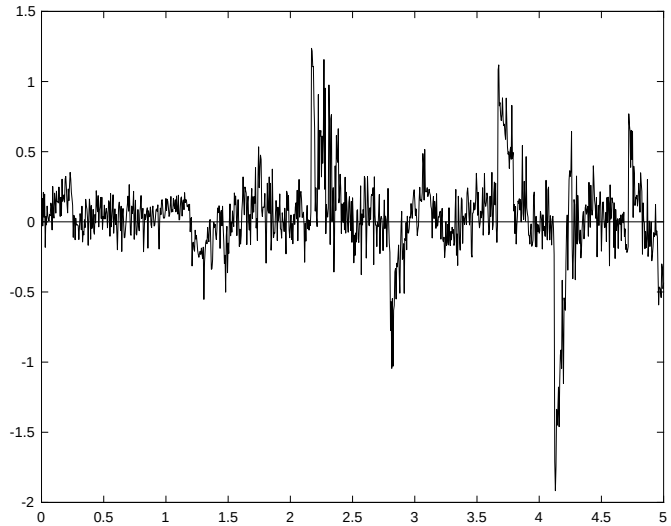


(c) DF $\widetilde{x}_n$ (sample path 1)

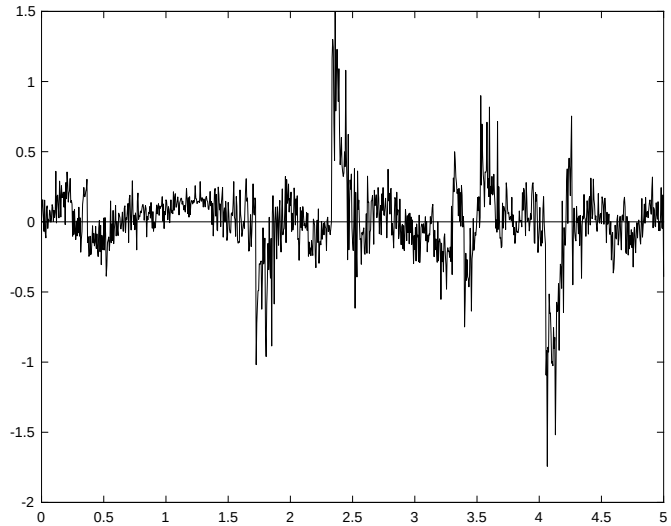


(d) DF $\widetilde{x}_n$ (sample path 2)

Figure 3: Two sample paths of x_n and $\widetilde{x}_n$.



(a) Error $x_n - \widetilde{x}_n$ (sample path 1)



(b) Error $x_n - \widetilde{x}_n$ (sample path 2)

Figure 4: Errors of two sample paths of x_n, $\widetilde{x}_n$.

Elliptic HJBs

joint work with Wenhao Qiu and Qingshuo Song

Elliptic HJBs

O: a domain with closure $\overline{O}$. Consider an elliptic nonlinear HJB equation

$$F(x, Du(x), D^2u(x)) = 0, \forall x \in O \tag{2.4}$$

with boundary condition

$$u(x) = g(x), \forall x \in \partial O. \tag{2.5}$$

One-player stochastic control problem.

$$dX_s = b(X_s, \mathbf{a}_s)ds + dW_s,\ X_t = x, \tag{2.6}$$

where $\mathbf{a}_s$ is the control.

$$J(x, \mathbf{a}) = \mathbb{E}\Big[\int_t^\tau \ell(X_s, \mathbf{a}_s)ds + g(X_\tau)\Big]. \tag{2.7}$$

Then the associated value function is defined as

$$v(x) = \inf_{\mathbf{a} \in \mathscr{A}} J(x, \mathbf{a}). \tag{2.8}$$

Differential games

- $A = A_1 \times A_2$, $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2) \in \mathscr{A}$ be a pair of progressively measurable control processes.
- An admissible strategy $\widehat{a}_1$ (resp. $\widehat{a}_2$) for player 1 (resp. player 2) is a progressively measurable mapping $\widehat{a}_1 : \mathscr{A}_1 \to \mathscr{A}_2$ (resp. $\widehat{a}_2 : \mathscr{A}_2 \to \mathscr{A}_1$).
- Let $\mathscr{S}_1$ (resp. $\mathscr{S}_2$) denote the class of all admissible strategies of player 1 (resp. player 2).
- The upper and lower values $v^+(x)$ and $v^-(x)$ are

$$v^+(x) = \sup_{\widehat{a}_1 \in \mathscr{S}_1} \inf_{\mathbf{a}_1 \in \mathscr{A}_1} J(x, \mathbf{a}_1, \widehat{a}_1(\mathbf{a}_1)), \quad \text{and} \tag{2.9}$$

$$v^-(x) = \inf_{\widehat{a}_2 \in \mathscr{S}_2} \sup_{\mathbf{a}_2 \in \mathscr{A}_2} J(x, \widehat{a}_2(\mathbf{a}_2), \mathbf{a}_2). \tag{2.10}$$

Benchmark problem

- Domain $O = \{x \in \mathbb{R}^d : 0 < x_i < 1, i = 1, 2, \ldots d\}$.
- Equation on O: $\frac{1}{2}\Delta u + \inf_{|a|\leq 3}\left(a \cdot \nabla u + \ell(x,a)\right) = 0$, $x \in O$. where $\ell(x,a) = d + 2|x|^2 + \frac{1}{2}|a|^2$.
- Dirichlet data on the boundary ∂O: $u(x) = -|x|^2$, $x \in \partial O$.
- Exact solution is given by $u(x) = -|x|^2$, and $a = 2x$, with

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

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

Construct u_h using Markov chain $\approx$ methods (Kushner and Dupuis)

$$u_h(x) = \tilde{F}_h[u_h](x),\ \forall x \in O_h, \quad \text{and} \tag{2.11}$$

$$u_h(x) = g(x),\ \forall x \in \partial O^h, \tag{2.12}$$

$$\begin{aligned} \tilde{F}_h[\phi](x) = \quad & \otimes_a \Big\{ \sum_{i=1}^{d} \Big(p^h(x + he_i | x, a)\phi(x + he_i) \\ & + p^h(x - he_i | x, a)\phi(x - he_i) \Big) + \ell^h(x, a) \Big\}, \end{aligned} \tag{2.13}$$

- $p^h(y|x,a)$ controlled transition probabilities
- running cost $\ell^h(x,a)$

$\otimes_a$ is an operator, $\forall \phi_1,\ \phi_2$, & constant c, $\exists\ K > 0$,

(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) $|\otimes_a \phi_1(x,a) - \otimes_a \phi_2(x,a)| \leq K \max_a |\phi_1(x,a) - \phi_2(x,a)|$.

Notation

- $f \in \mathrm{USC}(\overline{O})$: an upper semicontinuous function f in the set $\overline{O}$
- $f \in \mathrm{LSC}(\overline{O})$: a lower semicontinuous f. [Recall that $f \in \mathrm{USC}(\overline{O})$, if $-f \in \mathrm{USC}(\overline{O})$.]
- f^* & f_*: USC and LSC envelopes of f
- $I_A(\cdot)$: indicator function of the set A

Definition of viscosity solution

1. For $u \in \mathrm{LSC}(\overline{O})$ & $x \in \overline{O}$, the space of super test functions is

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

2. For $u \in \mathrm{LSC}(\overline{O})$ and $x \in \overline{O}$, the space of sub test functions is

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

Definition (cont.)

- $u \in \mathrm{USC}(\overline{O})$ satisfies the viscosity subsolution property at some $x \in \overline{O}$, if for all $\phi \in J^+(u,x)$, $F(x, D\phi(x), D^2\phi(x)) \geq 0$.
- $u \in \mathrm{LSC}(\overline{O})$ satisfies the viscosity supersolution property at some $x \in \overline{O}$, if for all $\phi \in J^-(u,x)$, $F(x, D\phi(x), D^2\phi(x)) \leq 0$.

1. $u \in \mathrm{USC}(\overline{O})$ is a viscosity subsolution of (2.4)-(2.5), 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. $u \in \mathrm{LSC}(\overline{O})$ is a viscosity supersolution of (2.4)-(2.5), if (a) u satisfies the viscosity supersolution property at each $x \in O$ and (b) $u(x) \geq g(x)$ at each $x \in \partial O$.
3. $u \in C(\overline{O})$ is a viscosity solution of (2.4)-(2.5), if it is a viscosity subsolution and supersolution simultaneously.

Algorithm 1 Value iteration of MDP(h)

procedure VALUE ITERATION(h) ▷ h: mesh size
  for x in ∂O^h **do** ▷ set boundary value
    $u^h(x) \leftarrow 0$
  for x in O^h **do** ▷ Init value
    $u^h(x) \leftarrow g(x)$
  $tol \leftarrow 1e-5$ ▷ set tolerance
  $flag \leftarrow True$
  while flag **do** ▷ value iteration
    err $\leftarrow 0$
    for x in O^h **do**
      $\widehat{u} \leftarrow u^h(x)$
      $u^h(x) \leftarrow \tilde{F}_h[u^h](x)$ using (2.1)-(2.2)
      err $\leftarrow err + (\widehat{u} - u^h(x))^2$
    if $err < tol$ **then** flag $\leftarrow$ False
  return $\{u^h(x) : x \in O^h\}$

Deep neural network

- Value iteration belongs to tabular method; all data will be saved in the form of a multidimensional array.
  - 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.
- A feasible way is to use deep neural network.
  - Let $\psi : \mathbb{R}^d \times \Theta \mapsto \mathbb{R}$ be a function generated by a neural network consisting of multiple layer neurons and some activation functions with a parameter set Θ.

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 $\phi^h(x, \theta)$

Assumptions

- (A0) F is continuous, and comparison principle and unique solvability holds for (2.4)-(2.5) 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) $\inf_{\theta\in\Theta} \beta_h(\theta) + \varepsilon_h = o(h^\gamma)$.
- (A5) There exist K and $\alpha > 0$ such that the following identity holds for all $x, y \in \overline{O}$: $|u_h(x) - u_h(y)| \leq K|x - y|^\alpha$.

Convergence Analysis

Proposition 1

Let u^* and u_* be the USC *and* LSC *envelopes of u_h. Under* (A0)-(A5), *u^* and u_* are sub and super solutions of* (2.4)-(2.5), *respectively.*

Theorem 2

Let u_h be the solution obtained from neural network algorithm. Suppose (A0), (A4), (A5) *hold, and* $\inf_{(x,a,i)} |b_i(x,a)| > 0$. *Then,* $\lim_{h\to 0} u_h(x) = u(x)$, *where u is the solution of* (2.4)-(2.5).

1-d Example benchmark problem: value iteration with $h = 1/8$

The approximate solution is sufficiently close to the exact solution.

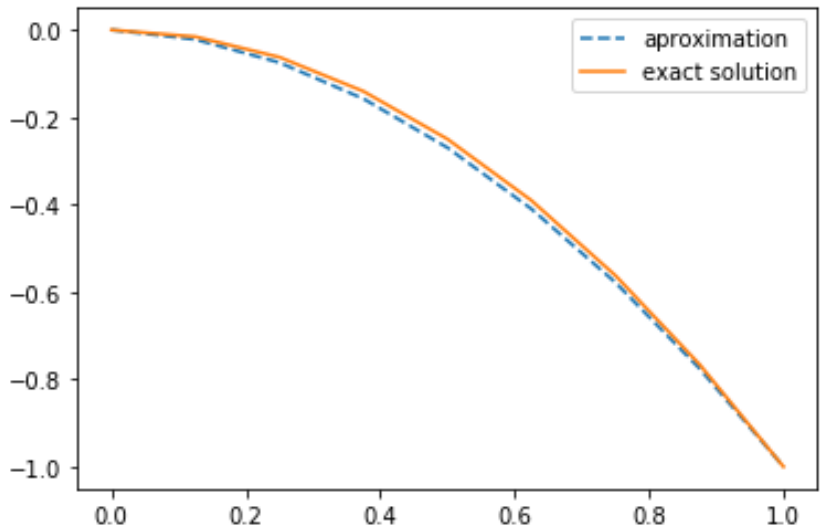


Figure 1: Approximate & exact solutions with value iteration. The numerical solution well $\approx$ exact solution.

1-d Benchmark example: Using neural network

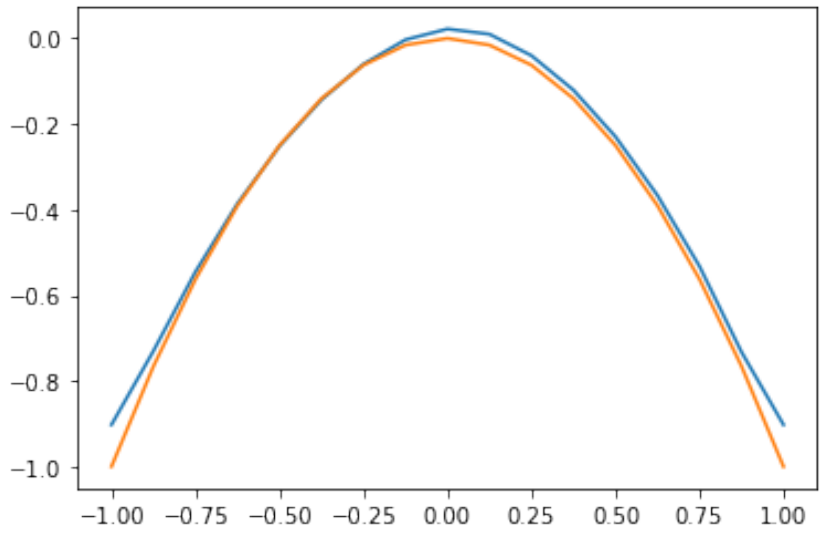


Figure 2: Approximation using neural network with $d = 1$, $h = 1/8$.

Using neural network: Boundary loss

Specifications: $w = 100$, 3 hidden layers, 8 neurons for each layer, ReLU activation function.

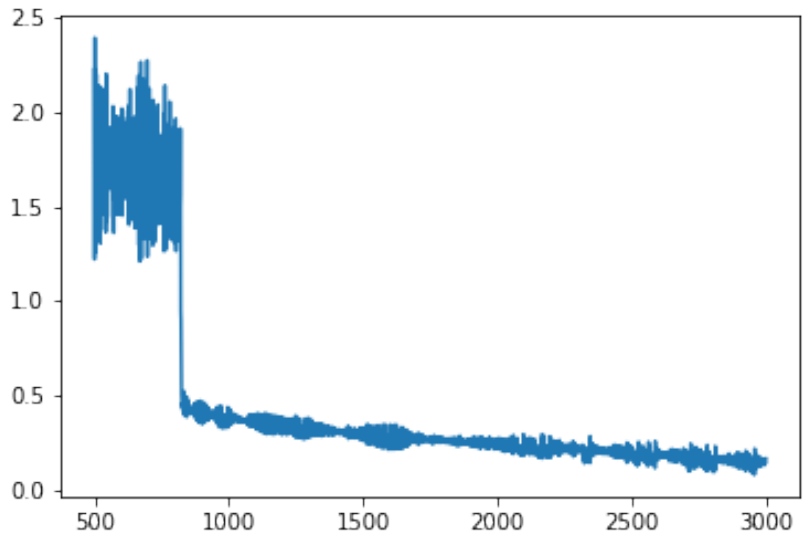


Figure 3: Epoch number vs Loss function with $d = 1$, $h = 1/8$.

2-d Benchmark example: Exact solution

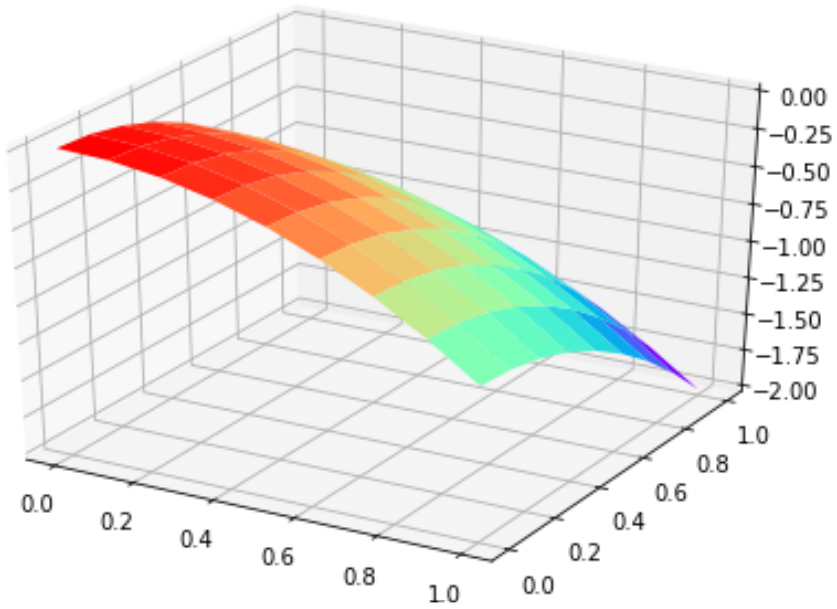


Figure 4: Exact solution for $d = 2$

Numerical solution: Value iteration

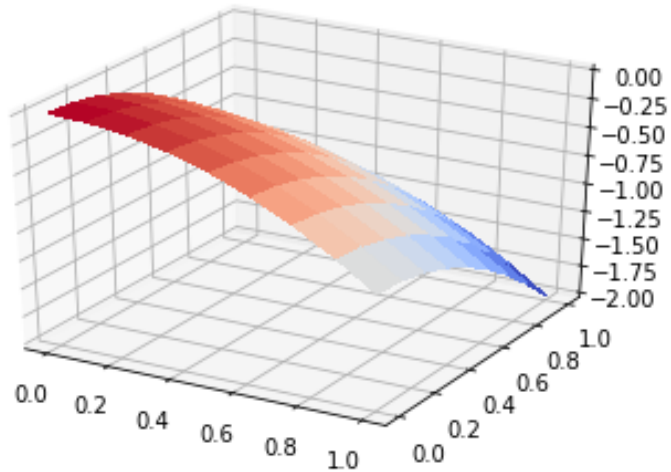


Figure 5: Numerical solution with $d = 2, h = 1/8$ from value iteration.

2-d Benchmark example: Approximation using neural network

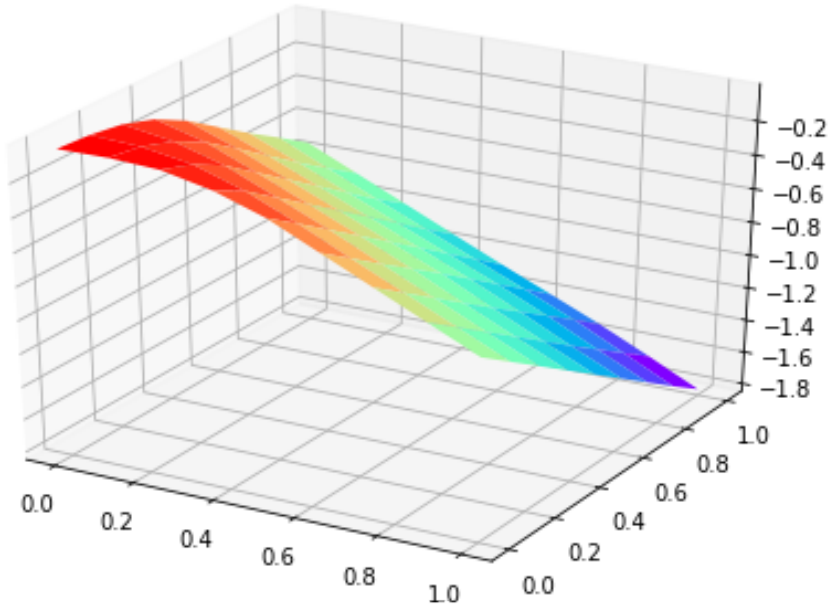


Figure 6: Numerical solution with $d = 2, h = 1/8$ from neural network.

Computational results

- 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.
- For 1-d and 2-d systems, they work well. For multi-dimensional systems, the robustness still needs to be tested.

Further Remarks

Future Study

- In this talk, we first presented two examples that lead to high-dimensional HJBs.
- We looked at a couple of examples of filtering and control using the idea of deep learning.
- We explored deep neural networks by providing preliminary experiments on various dynamic models.
- Extensive numerical tests on high-dimensional models with possible high nonlinearity, any genuine real world applications should be conducted.
- All in all, there are more questions than answers.

Thank you