Title

Neural network architectures using min-plus algebra for solving certain high-dimensional optimal control problems and Hamilton–Jacobi PDEs

Journal

Mathematics of Control, Signals, and Systems, 35(1)

ISSN

0932-4194

Authors

Darbon, Jérôme
Dower, Peter M
Meng, Tingwei

Publication Date

2023-03-01

DOI

10.1007/s00498-022-00333-2

Noname manuscript No.
(will be inserted by the editor)

Neural network architectures using min-plus algebra for solving certain high dimensional optimal control problems and Hamilton-Jacobi PDEs

Jérôme Darbon · Peter M. Dower · Tingwei Meng

Received: date / Accepted: date

Abstract Solving high dimensional optimal control problems and corresponding Hamilton-Jacobi PDEs are important but challenging problems in control engineering. In this paper, we propose two abstract neural network architectures which are respectively used to compute the value function and the optimal control for certain class of high dimensional optimal control problems. We provide the mathematical analysis for the two abstract architectures. We also show several numerical results computed using the deep neural network implementations of these abstract architectures. A preliminary implementation of our proposed neural network architecture on FPGAs shows promising speed up compared to CPUs. This work paves the way to leverage efficient dedicated hardware designed for neural networks to solve high dimensional optimal control problems and Hamilton-Jacobi PDEs.

Keywords Optimal control · Hamilton–Jacobi partial differential equations · Neural networks · Grid-free numerical methods

Research supported by AFOSR MURI FA9550-20-1-0358. Authors' names are given in last/family name alphabetical order.

Jérôme Darbon
Division of Applied Mathematics, Brown University
E-mail: jerome_darbon@brown.edu

Peter M. Dower
Department of Electrical and Electronic Engineering, The University of Melbourne
E-mail: pdower@unimelb.edu.au

Tingwei Meng
Department of Mathematics, UCLA
E-mail: tingwei@math.ucla.edu

1 Introduction

Optimal control problems are an important class of optimization problems that find many applications in engineering, such as trajectory planning [31,38, 74, 106, 131, 141], robot manipulator control [27, 61, 84, 98, 110] and humanoid robot control [39,54,55,60,64,99]. Under some assumptions, the optimal control problems are related to backward Hamilton–Jacobi (HJ) PDEs of the form

$$\begin{cases} -\dfrac{\partial V}{\partial t}(t,\boldsymbol{x}) + H(t,\boldsymbol{x},\nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})) = 0 & \boldsymbol{x}\in\mathbb{R}^n, t\in(0,T), \\ V(T,\boldsymbol{x}) = \Psi(\boldsymbol{x}) & \boldsymbol{x}\in\mathbb{R}^n, \end{cases} \tag{1}$$

where the function $H\colon [0,T]\times\mathbb{R}^n\times\mathbb{R}^n \ni (t,\boldsymbol{x},\boldsymbol{p}) \mapsto H(t,\boldsymbol{x},\boldsymbol{p}) \in \mathbb{R}$ is called the Hamiltonian, which is convex with respect to $\boldsymbol{p}$, and the continuous function $\Psi\colon \mathbb{R}^n \to \mathbb{R}$ specifies the terminal cost. This relation between optimal control problems and HJ PDEs has been widely studied in the literature (see [13] for instance). The value function of an optimal control problem may be characterized as the unique viscosity solution of the corresponding HJ PDE, while the optimal feedback control in the optimal control problems is related to the spatial gradient which is the Fréchet derivative of the value function with respect to the state variable $\boldsymbol{x}$. Therefore, computing the viscosity solution of an HJ PDE and its spatial gradient is an important problem in control engineering.

In many practical engineering problems, the dimensionality is often high. For instance, in robot manipulator control problems, there are multiple joints and end effectors in the manipulator. To control and measure the movement of each joint or end effector, several variables such as velocities, angles or positions are included in the state variable in the optimal control problems. As a result, the dimension of the state space is usually greater than five in practice. However, when the dimension is greater than five, standard grid-based numerical algorithms such as ENO [129], WENO [82], and DG [78] are infeasible to apply. This infeasibility is due to the curse of dimensionality [18], i.e., as the dimension grows, the number of grid points grows exponentially, and hence the memory requirement as well as the computational time also grow exponentially. Therefore, solving optimal control problems and HJ PDEs in high dimensions efficiently is an important but challenging problem. In the literature, several methods are proposed to overcome the curse of dimensionality when solving high dimensional HJ PDEs and optimal control problems. These methods include, but are not limited to, max-plus methods [2, 3, 48, 62, 66, 116, 117, 119, 120], optimization methods [32, 34, 36, 154], tensor decomposition techniques [44, 76, 148], sparse grids [21, 65, 92], polynomial approximation [88, 89], model order reduction [6, 101], dynamic programming and reinforcement learning [5, 20] and neural networks [8, 12, 33, 35, 42, 70, 79–81, 85, 86, 104, 111, 126–128, 138, 140, 144].

Recently, neural networks have been a successful tool in solving scientific computing problems involving PDEs. The related works include but are not limited to [4,8,12,15–17,19,26,30,41–43,51,53,59,63,68,70,71,77,79,80,83,91, 95, 96, 102–104, 107, 111–113, 115, 121–125, 128, 130, 132–138, 140, 142, 144, 146,

147, 149, 151–153, 157, 158]. Due to the success of neural networks, many new hardware designs have been proposed to efficiently (in terms of speed, latency, throughput or energy) implement neural networks. For instance, Google designed the "Tensor Processor Unit" [87] to accelerate inference using neural networks, and Intel developed new specific low-level instructions in their processors to accelerate machine learning applications [11]. Field programmable gate arrays (FPGAs) have been successfully used to implement neural networks for real-time applications, see e.g., [56–58]. There are also efforts for proposing completely new silicon designs [29,73] and efficient hardware designs for standard activation functions [100]. In addition, new computing architectures specialized for implementing neural network start to be available: for instance Xilinx recently launched a new computing architecture called Versal AI to efficiently implement neural networks. Note that these trends follow what LeCun suggested in [105, Sec. 3]. This dedicated hardware can in-principle be used for any algorithm that can be represented as a neural network architecture.

Realising this new hardware and silicon designs requires new dedicated software to implement neural networks on the new platforms. There are some available software development kits to convert neural network codes in standard frameworks such as PyTorch, TensorFlow, ONNX and HALO to executable codes on the aforementioned dedicated hardware. As long as an algorithm can be expressed in the neural network languages, it is possible to accelerate it with these new hardware. Therefore, an algorithm must be expressed as a neural network to leverage these new computational platforms.

In the literature, most neural network based algorithms regard the space of neural networks as a finite dimensional function space which approximates abstract functional spaces in the problems, and this approximation is guaranteed by the universal approximation theorems (see [28, 75, 97, 108, 114, 139] for instance). The output is given by a neural network whose parameters are trained using a problem-related optimization model. However, in general, there is no guarantee for the convergence of the neural networks, and hence the outputs are not guaranteed to solve the targeted problems. There is another research direction which focuses more on the neural network architectures, and provides theoretical guarantees for certain architectures. Along this research direction, [50, 52] proposed the connections between Resnet architectures and numerical solvers for ODEs, and [33, 35] presented several neural network architectures which express representation formulas for solving certain HJ PDEs.

In this work, we enlarge the class of HJ PDEs and optimal control problems which are solvable using neural network architectures by considering representation formulas for certain HJ PDEs with state and time dependent Hamiltonians. We design the neural network architectures such that they solve the optimal control problems and HJ PDEs of interest, with the neural network parameters assigned directly from the problem data, without the need for a training process. Without the training process, our neural network architectures are guaranteed to solve the optimal control problems and HJ PDEs.

Contributions of this paper. We present neural network architectures which solve certain high dimensional optimal control problems and the corresponding HJ PDEs. We consider the Hamiltonians H in (1) which are quadratic with respect to $(\boldsymbol{x}, \boldsymbol{p})$ with coefficients depending on t, and the initial data Ψ which is the minimum of finitely many quadratics. There are numerical solvers and theoretical analysis in the literature for these problems using linear-quadratic regulator and min-plus algebra. In this work, we present the neural network architectures according to these theories. Our contribution is three-fold.

- First, our work paves the way to leverage efficient dedicated hardware designed for neural networks to solve high dimensional optimal control problems and HJ PDEs. We present the neural network architectures based on the solid algorithms and theories in the literature for solving these problems. With the neural network architectures, it is possible to obtain efficient implementations in practice by converting the neural network codes to executable codes on the dedicated hardware. This facilitates future real-time implementations for solving high dimensional practical optimal control problems. Our work is easily implemented in standard frameworks, and we provide our implementations using TensorFlow in `https://github.com/TingweiMeng/NN_HJ_minplus`.
- Unlike most neural network algorithms in the literature, we provide theoretical guarantees to prove that our neural network architectures solve certain optimal control problems and HJ PDEs. These theoretical guarantees follow from the linear-quadratic control problems and min-plus algebra techniques in the theories of optimal controls and HJ PDEs. In this way, we show the correspondence between optimal control theories and certain neural network architectures. This correspondence also provides possibilities for new interpretations of certain neural network architectures from the optimal control perspective.
- We present an FPGA implementation of our proposed neural network architecture which shows that promising speed-ups can be expected compared to implementation on CPUs.

Organization of this paper. The mathematical background of optimal controls and min-plus algebra is given in Section 2. In Section 3, two abstract neural network architectures are presented, which solve the HJ PDEs and are used to compute the optimal controls in the optimal control problems, respectively. The first abstract architecture is shown in Section 3.1 and depicted in Fig. 3, which is a one-layer neural network architecture with abstract neurons. It solves the HJ PDEs and the optimal values in the corresponding optimal control problems. The second abstract architecture is shown in Section 3.2 and depicted in Fig. 4, which is a two-layer neural network architecture with abstract neurons. It can be used to compute the optimal controls in the optimal control problems. In Section 3.3, we consider more general terminal conditions and propose a numerical algorithm that combines our proposed neural network architecture and ADMM to solve the corresponding HJ PDEs and

optimal control problems. The implementations of our proposed two abstract architectures and their numerical results are presented in Section 4. There are different ways to implement the abstract architectures. Among these implementations we show the one using the fourth order Runge-Kutta method for illustration, which gives the deep Resnet-type implementations depicted in Figs. 6 and 7. The numerical solutions computed by the proposed neural network architectures and implementations for three optimal control problems are shown in Sections 4.1, 4.2 and 4.3, respectively. Section 4.4 shows one numerical result with a general terminal condition. An implementation of our proposed neural network on a FPGA is described in Section 4.5 and it shows promising speed-ups compared to a CPU implementation. Some conclusions are drawn in Section 5.

2 Mathematical background

Throughout, we use $\mathbb{R}^{n\times l}$ to denote the set of matrices with n rows and l columns with entries in $\mathbb{R}$, and use S^n to denote the set of real-valued symmetric matrices in $\mathbb{R}^{n\times n}$. Also, $S^n_{>0}$ denotes the set of symmetric positive definite matrices in $\mathbb{R}^{n\times n}$, and $S^n_{\geq 0}$ denotes the set of symmetric positive semi-definite matrices in $\mathbb{R}^{n\times n}$. We denote the identity matrix in $\mathbb{R}^{n\times n}$ by I_n, and the zero matrix in $\mathbb{R}^{n\times n}$ by O_n. Moreover, we use the bold character to denote a vector, and we use the capital character to denote a matrix, if not mentioned specifically. The ℓ^2-norm and ℓ^1-norm in $\mathbb{R}^n$ are denoted by $\|\cdot\|$ and $\|\cdot\|_1$, respectively.

2.1 Optimal control and min-plus algebra

First, we give a brief introduction to optimal control problems, HJ PDEs and their relation. An optimal control problem is formulated as follows

$$V(t_0,\boldsymbol{x}_0)\doteq\inf\left\{\int_{t_0}^{T}L(s,\boldsymbol{x}(s),\boldsymbol{u}(s))ds+\Psi(\boldsymbol{x}(T))\right\} \tag{2}$$

subject to

$$\begin{cases}\dot{\boldsymbol{x}}(s)=f(s,\boldsymbol{x}(s),\boldsymbol{u}(s)) \quad s\in(t_0,T),\\ \boldsymbol{x}(t_0)=\boldsymbol{x}_0,\end{cases} \tag{3}$$

where $T\in(0,+\infty)$ and $t_0\in[0,T]$ are scalars which denote the terminal time and initial time, $\boldsymbol{x}_0$ is a vector in $\mathbb{R}^n$ which denotes the initial position, the trajectory $\boldsymbol{x}\colon[t_0,T]\to\mathbb{R}^n$ is an absolutely continuous function solving the Cauchy problem (3) almost everywhere, and the control $\boldsymbol{u}\colon[t_0,T]\to\mathbb{R}^l$ is a function in a function space such as $L^p(t_0,T;\mathbb{R}^l)$ or the space of measurable functions. In the optimal control problem (2), the running cost is given by the function $L\colon[0,T]\times\mathbb{R}^n\times\mathbb{R}^l\to\mathbb{R}$, which is also called Lagrangian, while the

terminal cost is given by the function $\Psi\colon \mathbb{R}^n \to \mathbb{R}$. The optimal cost is denoted by $V(t_0, \boldsymbol{x}_0)$, which is a function of the initial time t_0 and the initial position $\boldsymbol{x}_0$ in the Cauchy problem (3).

Under suitable assumptions (see [13] for instance), the value function V is a viscosity solution of the corresponding backward HJ PDE (1). Viscosity solutions are known to be equivalent to minimax solutions (also known as minimal selections), which are defined via the associated characteristic inclusion, see [25,145]. In the corresponding HJ PDE, the Hamiltonian $H\colon [0,T]\times\mathbb{R}^n\times\mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$ is given by the function f and the Lagrangian L as follows

$$H(t,\boldsymbol{x},\boldsymbol{p}) = \sup_{\boldsymbol{u}\in\mathbb{R}^l}\{-\langle f(t,\boldsymbol{x},\boldsymbol{u}),\boldsymbol{p}\rangle - L(t,\boldsymbol{x},\boldsymbol{u})\} \quad \forall t\in[0,T], \boldsymbol{x},\boldsymbol{p}\in\mathbb{R}^n,$$

and the terminal data $\Psi\colon \mathbb{R}^n \to \mathbb{R}$ is given by the terminal cost in the optimal control problem. Given the solution V to the HJ PDE (1), the optimal control $\boldsymbol{u}^*\colon [0,T]\to\mathbb{R}^l$ in the problem (2) is characterized by Pontryagin maximum principle [13], which states that at almost every $s\in[0,T]$, the optimal control $\boldsymbol{u}^*(s)$ satisfies

$$\boldsymbol{u}^*(s) \in \arg\max_{\boldsymbol{u}\in\mathbb{R}^l}\{-\langle f(s,\boldsymbol{x}^*(s),\boldsymbol{u}),\boldsymbol{p}\rangle - L(s,\boldsymbol{x}^*(s),\boldsymbol{u})\}, \tag{4}$$

for each $\boldsymbol{p}\in D^+_{\boldsymbol{x}}V(s,\boldsymbol{x}^*(s))\cup D^-_{\boldsymbol{x}}V(s,\boldsymbol{x}^*(s))$, where $\boldsymbol{x}^*\colon [t_0,T]\to\mathbb{R}^n$ is the corresponding trajectory solved by (3) given the control $\boldsymbol{u}^*$. Here, $D^+_{\boldsymbol{x}}V$ and $D^-_{\boldsymbol{x}}V$ denote the set of the spatial components of the superdifferential and subdifferential of V, respectively. There are different sets of assumptions for the above relation (4) to hold. For details of the assumptions, see [159], [13, Section III.3.4] and the references in [13, Section III.6]. A verification theorem can alternatively be used to check whether a control is optimal, if a solution to the HJ PDE is known to exist.

We consider the HJ PDE (1) whose terminal data Ψ is the minimum of several functions $\Psi_i\colon \mathbb{R}^n\to\mathbb{R}$, i.e., we assume

$$\Psi(\boldsymbol{x}) = \min_{i\in\{1,\dots,m\}} \Psi_i(\boldsymbol{x}) \quad \forall\, \boldsymbol{x}\in\mathbb{R}^n. \tag{5}$$

Denote by $V_i\colon \mathbb{R}^n\times[0,+\infty)\to\mathbb{R}$ the viscosity solution to the corresponding backward HJ PDE with terminal data Ψ_i, which reads

$$\begin{cases} -\dfrac{\partial V_i}{\partial t}(t,\boldsymbol{x}) + H(t,\boldsymbol{x},\nabla_{\boldsymbol{x}}V_i(t,\boldsymbol{x})) = 0 & \boldsymbol{x}\in\mathbb{R}^n, t\in(0,T), \\ V_i(T,\boldsymbol{x}) = \Psi_i(\boldsymbol{x}) & \boldsymbol{x}\in\mathbb{R}^n. \end{cases} \tag{6}$$

If the HJ PDEs (1) and (6) are solved by the value function (2) with terminal costs Ψ and Ψ_i, respectively, then the solution operator in the HJ PDE (1) is linear with respect to the min plus algebra [116]. From straightforward calculation using (2), the value function V can be written as the minimum of V_i as follows

$$V(t,\boldsymbol{x}) = \min_{i\in\{1,\dots,m\}} V_i(t,\boldsymbol{x}), \quad \forall\, \boldsymbol{x}\in\mathbb{R}^n, t\in[0,T].$$

Consequently, V in this form also solves the HJ PDE (1) with terminal data Ψ.

2.2 Neural networks

We give a brief introduction of neural networks and refer the reader to [1] for a full introduction. A neural network architecture defines a space of functions which approximates the solution space in the target problem. A general neural network is the composition of several functions whose inputs and outputs are called layers. Each layer contains several variables or quantities which are called neurons. The input and output of the neural network function are called the input layer and the output layer, and all the other layers are called hidden layers. Different types of neural networks have been proposed in the literature [1].

A basic neural network architecture is called a feedforward neural network, whose hidden layer is the composition of an affine function and a non-linear function called activation function. An illustration of a feedforward neural network architecture with two hidden layers is shown in Fig. 1, where each blue box corresponds to a neuron, and the line connecting the neurons illustrates the dependency between different neurons. To our knowledge, in the machine learning community, there is no standardised form to represent a neural network architecture as a diagram.

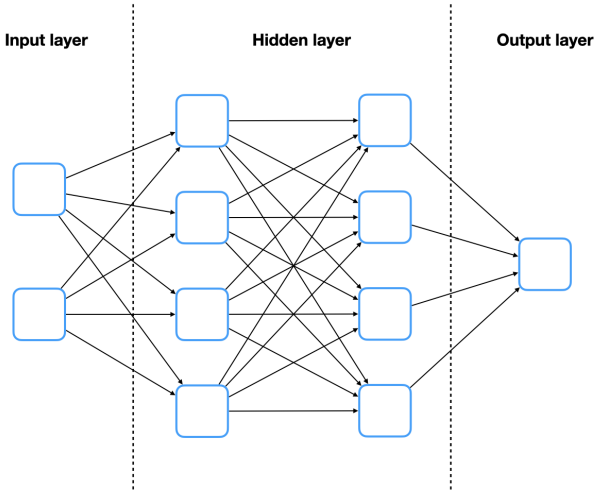


Fig. 1 An illustration of a feedforward neural network architecture with two hidden layers.

Another widely used neural network architecture is called residual neural network (Resnet) [1, 72]. A hidden layer in a Resnet involves more algebraic computations among compositions of affine functions and activation functions.

An illustration of a hidden layer in a standard Resnet architecture with activation function σ is shown in Fig. 2.

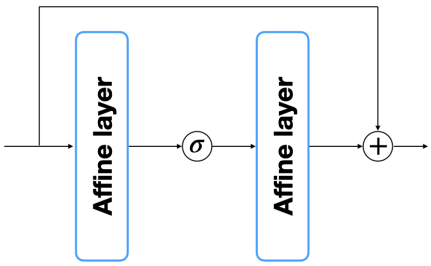


Fig. 2 An illustration of a hidden layer in the Resnet architecture.

3 Neural network architectures for solving certain HJ PDEs and optimal control problems

Fix a finite terminal time $T \in (0, +\infty)$, the following is assumed throughout:

(A1) Let $A \in C([0,T]; \mathbb{R}^{n\times n})$, $B \in C([0,T]; \mathbb{R}^{n\times l})$, $S \in C([0,T]; \mathbb{R}^{n\times l})$, $Q \in C([0,T]; S^n_{>0})$ and $R \in C([0,T]; S^l_{>0})$ be continuous functions. Let $G_i \in S^n_{\geq 0}$ be a constant matrix, $\boldsymbol{a}_i \in \mathbb{R}^n$ be a constant vector and $b_i \in \mathbb{R}$ be a constant scalar for each $i \in \{1, \dots, m\}$.

We consider the optimal control problem (2) whose Lagrangian $L\colon [0,T] \times \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}$ and the function $f\colon [0,T] \times \mathbb{R}^n \times \mathbb{R}^l \to \mathbb{R}^n$ are defined by

$$\begin{aligned} L(t, \boldsymbol{x}, \boldsymbol{u}) &= \frac{1}{2}\boldsymbol{x}^T Q(t)\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^T R(t)\boldsymbol{u} + \boldsymbol{x}^T S(t)\boldsymbol{u}, \\ f(t, \boldsymbol{x}, \boldsymbol{u}) &= A(t)\boldsymbol{x} + B(t)\boldsymbol{u}, \end{aligned} \tag{7}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^l$ and $t \in [0,T]$. Define the terminal cost $\Psi\colon \mathbb{R}^n \to \mathbb{R}$ by

$$\Psi(\boldsymbol{x}) = \min_{i\in\{1,\dots,m\}} \left\{ \frac{1}{2}\boldsymbol{x}^T G_i \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + b_i \right\} \quad \forall \boldsymbol{x} \in \mathbb{R}^n. \tag{8}$$

The corresponding optimal control problem is defined via the value function

$$\begin{aligned} V(t_0, \boldsymbol{x}_0) = \inf_{(\boldsymbol{x},\boldsymbol{u})\in\mathcal{C}(t_0,\boldsymbol{x}_0)} \Bigg\{ \int_{t_0}^T \Big(&\frac{1}{2}\boldsymbol{x}(s)^T Q(s)\boldsymbol{x}(s) + \frac{1}{2}\boldsymbol{u}(s)^T R(s)\boldsymbol{u}(s) \\ &+ \boldsymbol{x}(s)^T S(s)\boldsymbol{u}(s) \Big) ds + \Psi(\boldsymbol{x}(T)) \Bigg\} \end{aligned} \tag{9}$$

where the constraint set $\mathcal{C}(t_0, \boldsymbol{x}_0)$ is defined to be the set of $(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)) \in L^2(t_0, T; \mathbb{R}^n) \times L^2(t_0, T; \mathbb{R}^l)$ which satisfies the following Cauchy problem

$$\begin{cases} \dot{\boldsymbol{x}}(s) = A(s)\boldsymbol{x}(s) + B(s)\boldsymbol{u}(s) & s \in (t_0, T), \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0. \end{cases} \tag{10}$$

For the corresponding HJ PDE, we consider the following standard assumption [150, 155].

(A2) Assume $C_{pp}\colon [0,T] \to S^n_{\geq 0}$, , $C_{xx}\colon [0,T] \to S^n_{>0}$ and $C_{xp}\colon [0,T] \to \mathbb{R}^{n\times n}$ are three functions defined by

$$\begin{cases} C_{pp}(t) = B(t)R(t)^{-1}B(t)^T, \\ C_{xx}(t) = Q(t) - S(t)R(t)^{-1}S(t)^T, \\ C_{xp}(t) = A(t) - B(t)R(t)^{-1}S(t)^T, \end{cases} \tag{11}$$

for all $t \in [0,T]$, where A, B, S, Q, R are the functions satisfying assumption (A1).

The Hamiltonian $H\colon [0,T] \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined by

$$H(t, \boldsymbol{x}, \boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^T C_{pp}(t)\boldsymbol{p} - \frac{1}{2}\boldsymbol{x}^T C_{xx}(t)\boldsymbol{x} - \boldsymbol{p}^T C_{xp}(t)\boldsymbol{x} \quad \forall t \in [0,T], \boldsymbol{x}, \boldsymbol{p} \in \mathbb{R}^n.$$

The corresponding HJ PDE reads

$$\begin{cases} -\dfrac{\partial V(t,\boldsymbol{x})}{\partial t} + H(t, \boldsymbol{x}, \nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})) = 0 & \boldsymbol{x} \in \mathbb{R}^n, t \in (0,T), \\ V(T,\boldsymbol{x}) = \Psi(\boldsymbol{x}) = \min\limits_{i\in\{1,\dots,m\}} \left\{ \dfrac{1}{2}\boldsymbol{x}^T G_i \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + b_i \right\} & \boldsymbol{x} \in \mathbb{R}^n. \end{cases} \tag{12}$$

This is the HJ PDE for a linear regulator problem with a min-of-quadratics terminal cost. Its solution can be represented via the solution to the Riccati equation (see [45, 47, 48, 116] for instance).

In the following two sections, we will present two abstract neural network architectures. The first one is shown in Section 3.1 which represents the viscosity solution to the HJ PDE (12). The same neural network architecture also represents the value function in the optimal control problem (9). The second abstract neural network architecture is shown in Section 3.2, and it can be used to compute the optimal control in the optimal control problem (9).

3.1 An abstract neural network architecture for solving the HJ PDE (12)

We present an abstract neural network architecture which represents the viscosity solution to the HJ PDE (12). The viscosity solution can be represented by a neural network V_{NN} defined as follows

$$V_{NN}(t,\boldsymbol{x}) \doteq \min_{i\in\{1,\dots,m\}} V_i(t,\boldsymbol{x}), \quad V_i(t,\boldsymbol{x}) \doteq \frac{1}{2}\boldsymbol{x}^T P_i(t)\boldsymbol{x} + \boldsymbol{q}_i(t)^T\boldsymbol{x} + r_i(t), \tag{13}$$

where the function $P_i \in C(0,T;S^n)$ solves the following Riccati final value problem (FVP)

$$\begin{cases} \dot{P}_i(t) = P_i(t)^T C_{pp}(t) P_i(t) - P_i(t)^T C_{xp}(t) - C_{xp}(t)^T P_i(t) - C_{xx}(t) & t\in(0,T) \\ P_i(T) = G_i, \end{cases} \tag{14}$$

the functions $\boldsymbol{q}_i \in C(0,T;\mathbb{R}^n)$ solves the following linear FVP

$$\begin{cases} \dot{\boldsymbol{q}}_i(t) = P_i(t)^T C_{pp}(t)\boldsymbol{q}_i(t) - C_{xp}(t)^T\boldsymbol{q}_i(t) & t\in(0,T), \\ \boldsymbol{q}_i(T) = \boldsymbol{a}_i, \end{cases} \tag{15}$$

and the function $r_i \in C(0,T;\mathbb{R})$ solves the following FVP

$$\begin{cases} \dot{r}_i(t) = \frac{1}{2}\boldsymbol{q}_i(t)^T C_{pp}(t)\boldsymbol{q}_i(t) & t\in(0,T), \\ r_i(T) = b_i. \end{cases} \tag{16}$$

An illustration for the neural network architecture (13) is shown in Fig. 3. This is a one-layer abstract architecture with a min-pooling activation function. The i-th abstract neuron is given by the function $V_i(t,\boldsymbol{x})$ in (13). The architecture and neurons are called abstract since some ODE solvers for (14), (15) and (16) are further required in order to evaluate each neuron and the neural network architecture. Later, in Section 4, we will provide a deep Resnet implementation (depicted in Fig. 6) for this abstract architecture.

The following proposition shows that this neural network architecture (13) provides the viscosity solution to the HJ PDE (12) and the value function in the optimal control problem (9).

Proposition 1 *Assume (A1)-(A2) hold. Let V_{NN} be the function defined by* (13)*. Then V_{NN} is the unique viscosity solution to the HJ PDE* (12)*. Moreover, V_{NN} equals the value function V in the optimal control problem* (9)*.*

Proof. First, we apply [150, Prop. 2.2] to prove that the unique viscosity solution to the HJ PDE (12) is given by the value function for the optimal control problem (9) under the assumptions (A1)-(A2). Most assumptions in [150, Prop. 2.2] are straightforward to check under this linear quadratic setting. Here, we only check the following three non-trivial assumptions:

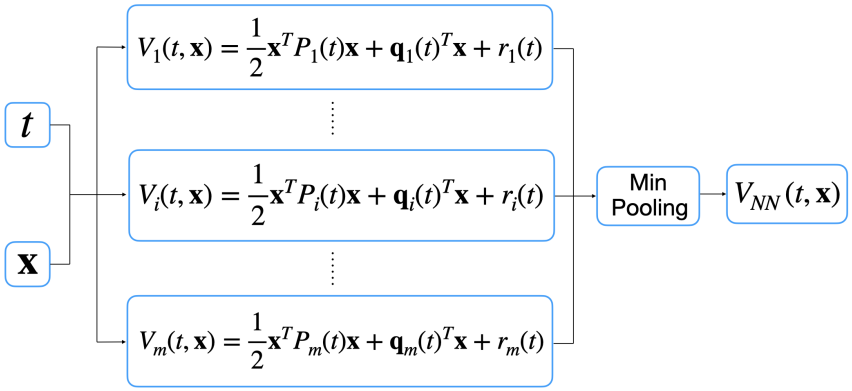


Fig. 3 Illustration of the abstract neural network architecture defined by (13) that represents the viscosity solution to the HJ PDE (12) and the value function in the optimal control problem (9).

1. There exists a positive constant $\rho > 0$ such that $R(t) - \rho I$ is positive semi-definite for all $t \in [0, T]$.
2. There exist some positive constants $C_L > 0$ and $\epsilon_0 \in (0, 1)$ such that $(1-\epsilon_0)\boldsymbol{x}^T Q(t)\boldsymbol{x} - \boldsymbol{x}^T S(t)R(t)^{-1}S(t)^T\boldsymbol{x} \geq -C_L$ holds for all $t \in [0, T]$ and $\boldsymbol{x} \in \mathbb{R}^n$.
3. There exists a constant $C_0 > 0$ such that $\Psi(\boldsymbol{x}) \leq C_0(1 + |\boldsymbol{x}|^2)$ holds for all $\boldsymbol{x} \in \mathbb{R}^n$.

Now, we check the first assumption. We are going to prove a slightly stronger version. We prove that there exists $\rho > 0$ such that the matrix $R(t) - \rho I$ is positive definite for all $t \in [0, T]$. Assume it does not hold. Then, there exist sequences $\{t_k\} \subset [0, T]$ and $\{\boldsymbol{x}_k\} \subset \mathbb{R}^n$ such that $\boldsymbol{x}_k^T R(t_k)\boldsymbol{x}_k \leq \frac{1}{k}\|\boldsymbol{x}_k\|^2$ holds for each $k \in \mathbb{N}$. After scaling, we assume $\|\boldsymbol{x}_k\| = 1$ holds for each $k \in \mathbb{N}$. By taking subsequences and still denoting the subsequences by $\{t_k\}$ and $\{\boldsymbol{x}_k\}$, we obtain the convergence of $\{t_k\}$ and $\{\boldsymbol{x}_k\}$, whose limits are denoted by $\bar{t} \in [0, T]$ and $\bar{\boldsymbol{x}} \in \mathbb{R}^n$, respectively. Note that $\|\boldsymbol{x}_k\| = 1$ for each $k \in \mathbb{N}$ implies $\|\bar{\boldsymbol{x}}\| = 1$. Since R is a continuous function, we obtain

$$\bar{\boldsymbol{x}}^T R(\bar{t})\bar{\boldsymbol{x}} = \lim_{k\to\infty}\left(\boldsymbol{x}_k^T R(t_k)\boldsymbol{x}_k - \frac{1}{k}\|\boldsymbol{x}_k\|^2\right) \leq 0,$$

which contradicts with the assumption that the matrix $R(\bar{t})$ is positive definite. Therefore, the first assumption holds. A similar argument proves that there exists a positive constant ϵ such that $C_{xx}(t) - \epsilon I$ is positive semi-definite for all $t \in [0, T]$. Also, the continuity of Q implies the existence of a uniform upper bound C for $\|Q(t)\|$ for all $t \in [0, T]$. Let ϵ_0 equal $\min\{\frac{\epsilon}{C}, 1\}$. Then, by (11),

we have

$$\begin{aligned}(1-\epsilon_0)\boldsymbol{x}^T Q(t)\boldsymbol{x} - \boldsymbol{x}^T S(t)R(t)^{-1}S(t)^T\boldsymbol{x} &= \boldsymbol{x}^T C_{xx}(x)\boldsymbol{x} - \epsilon_0 \boldsymbol{x}^T Q(t)\boldsymbol{x} \\ &\geq \boldsymbol{x}^T C_{xx}(x)\boldsymbol{x} - \epsilon_0 \|Q(t)\|\|\boldsymbol{x}\|^2 \\ &\geq \epsilon\|\boldsymbol{x}\|^2 - \epsilon_0 C\|\boldsymbol{x}\|^2 \geq 0.\end{aligned}$$

As a result, the second assumption holds. The third assumption follows from a straightforward computation which reads

$$\begin{aligned}\Psi(\boldsymbol{x}) &= \min_{i\in\{1,\dots,m\}} \left\{ \frac{1}{2}\boldsymbol{x}^T G_i \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + b_i \right\} \leq \frac{1}{2}\boldsymbol{x}^T G_1 \boldsymbol{x} + \boldsymbol{a}_1^T \boldsymbol{x} + b_1 \\ &\leq C_0(1+\|\boldsymbol{x}\|^2),\end{aligned}$$

for some positive constant C_0. Therefore, all the assumptions in [150, Prop. 2.2] are satisfied. Then, by [150, Prop. 2.2], the value function defined by (9) is the unique viscosity solution to the HJ PDE (12). (Note that if all the functions in (A1) and (A2) do not depend on time t, then this result follows from [14].) Therefore, it suffices to prove that V_{NN} defined by (13) is the value function in the optimal control problem (9), i.e., it suffices to prove that $V_{NN}(t_0, \boldsymbol{x}_0) = V(t_0, \boldsymbol{x}_0)$ holds for all $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $t_0 \in [0, T]$.

Define $\Psi_i \colon \mathbb{R}^n \to \mathbb{R}$ by

$$\Psi_i(\boldsymbol{x}) \doteq \frac{1}{2}\boldsymbol{x}^T G_i \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + b_i, \quad \forall\, \boldsymbol{x} \in \mathbb{R}^n, \tag{17}$$

for each $i \in \{1, \dots, m\}$. Then, by (8), the functions $\Psi_1, \dots, \Psi_m$ and Ψ satisfy (5). According to (9), (5) and the min-plus linearity of the dynamic programming evolution operator, we have

$$\begin{aligned}&V(t_0, \boldsymbol{x}_0) \\ &= \inf_{(\boldsymbol{x},\boldsymbol{u})\in\mathcal{C}(t_0,\boldsymbol{x}_0)} \left\{ \int_{t_0}^{T} \Big(\frac{1}{2}\boldsymbol{x}(s)^T Q(s)\boldsymbol{x}(s) + \frac{1}{2}\boldsymbol{u}(s)^T R(s)\boldsymbol{u}(s) \right. \\ &\qquad \left. + \boldsymbol{x}(s)^T S(s)\boldsymbol{u}(s) \Big) ds + \min_{i\in\{1,\dots,m\}} \Psi_i(\boldsymbol{x}(T)) \right\} \\ &= \min_{i\in\{1,\dots,m\}} \left\{ \inf_{(\boldsymbol{x},\boldsymbol{u})\in\mathcal{C}(t_0,\boldsymbol{x}_0)} \left\{ \int_{t_0}^{T} \Big(\frac{1}{2}\boldsymbol{x}(s)^T Q(s)\boldsymbol{x}(s) + \frac{1}{2}\boldsymbol{u}(s)^T R(s)\boldsymbol{u}(s) \right. \right. \\ &\qquad \left. \left. + \boldsymbol{x}(s)^T S(s)\boldsymbol{u}(s) \Big) ds + \Psi_i(\boldsymbol{x}(T)) \right\} \right\}.\end{aligned} \tag{18}$$

We define the value function in the last line of (18) to be $\tilde{V}_i(t_0, \boldsymbol{x}_0)$, i.e., we define the function $\tilde{V}_i \colon [0, T] \times \mathbb{R}^n \to \mathbb{R}$ by

$$\begin{aligned}\tilde{V}_i(t_0, \boldsymbol{x}_0) \doteq \inf_{(\boldsymbol{x},\boldsymbol{u})\in\mathcal{C}(t_0,\boldsymbol{x}_0)} \left\{ \int_{t_0}^{T} \Big(\frac{1}{2}\boldsymbol{x}(s)^T Q(s)\boldsymbol{x}(s) + \frac{1}{2}\boldsymbol{u}(s)^T R(s)\boldsymbol{u}(s) \right. \\ \left. + \boldsymbol{x}(s)^T S(s)\boldsymbol{u}(s) \Big) ds + \Psi_i(\boldsymbol{x}(T)) \right\}.\end{aligned} \tag{19}$$

It is well-known that $\tilde{V}_i$ can be solved by Riccati equation under the assumptions (A1)-(A2). One way to prove it is given as follows. First, [23] shows the existence of the global solution to the Riccati FVP (14) under the assumptions (A1)-(A2). Then, with a similar argument as in the proof of [155, Chap 6, Thm. 2.8], the value function $\tilde{V}_i$ is proved to satisfy

$$\tilde{V}_i(t,\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T P_i(t)\boldsymbol{x} + \boldsymbol{q}_i(t)^T\boldsymbol{x} + r_i(t) = V_i(t,\boldsymbol{x}), \tag{20}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$ and $t \in [0,T]$, where V_i is the function defined in (13), and P_i, $\boldsymbol{q}_i$ and r_i satisfy (14), (15), and (16), respectively.

Therefore, combining (18) and (20), we derive that $V_{NN} \equiv V$ in $\mathbb{R}^n \times [0,T]$ and the conclusion follows. □

3.2 An abstract neural network architecture for the optimal control problem (9)

We present an abstract neural network architecture for computing the optimal control in the optimal control problem (9). For any fixed index $j \in \{1,\dots,m\}$, define a function $\boldsymbol{u}_j\colon [t_0,T]\times\mathbb{R}^n \to \mathbb{R}^l$ by

$$\boldsymbol{u}_j(t,\boldsymbol{x}) \doteq -R(t)^{-1}\left(B(t)^T P_j(t)\boldsymbol{x} + B(t)^T\boldsymbol{q}_j(t) + S(t)^T\boldsymbol{x}\right), \tag{21}$$

for any $t \in [t_0,T]$ and $\boldsymbol{x} \in \mathbb{R}^n$. It is well-known that the function $\boldsymbol{u}_j$ is the feedback control for the optimal control problem (19) with index $i = j$. To solve the optimal control problem (9), we define the following function $\boldsymbol{u}_{NN}$ with specific selected index k_{NN}:

$$\boldsymbol{u}_{NN}(t_0,\boldsymbol{x}_0,t,\boldsymbol{x}) = \boldsymbol{u}_{k_{NN}(t_0,\boldsymbol{x}_0)}(t,\boldsymbol{x}), \quad \forall \boldsymbol{x}_0,\boldsymbol{x}\in\mathbb{R}^n, 0\le t_0\le t\le T, \tag{22}$$

where the index function $k_{NN}\colon [0,T]\times\mathbb{R}^n \to \{1,\dots,m\}$ is defined by

$$\begin{aligned} k_{NN}(t_0,\boldsymbol{x}_0) &\in \operatorname*{arg\,min}_{i\in\{1,\dots,m\}} \left\{\frac{1}{2}\boldsymbol{x}_0^T P_i(t_0)\boldsymbol{x}_0 + \boldsymbol{q}_i(t_0)^T\boldsymbol{x}_0 + r_i(t_0)\right\} \\ &= \operatorname*{arg\,min}_{i\in\{1,\dots,m\}} V_i(t_0,\boldsymbol{x}_0). \end{aligned} \tag{23}$$

When there is no ambiguity, we abuse the notation $k_{NN}(t_0,\boldsymbol{x}_0)$ with k_{NN}. Recall that the functions P_i, $\boldsymbol{q}_i$, r_i and V_i are the functions defined in (14), (15), (16) and (13), respectively. If there are more than one minimizer in the optimization problem in (23), we just select any of these minimizers and it will provide an optimal control. We will discuss more about this non-uniqueness later in Remark 1. Note that the function $\boldsymbol{u}_{NN}$ can be expressed using an abstract neural network architecture shown in Fig. 4. As it is discussed in Section 3.1, the evaluation of the neurons $\{V_i(t_0,\boldsymbol{x}_0)\}_{i=1}^m$, $P_{k_{NN}}(t)$ and $\boldsymbol{q}_{k_{NN}}(t)$ (where k_{NN} is the index defined in (23)) require further ODE solvers, and hence we call it an abstract architecture. An implementation of this abstract

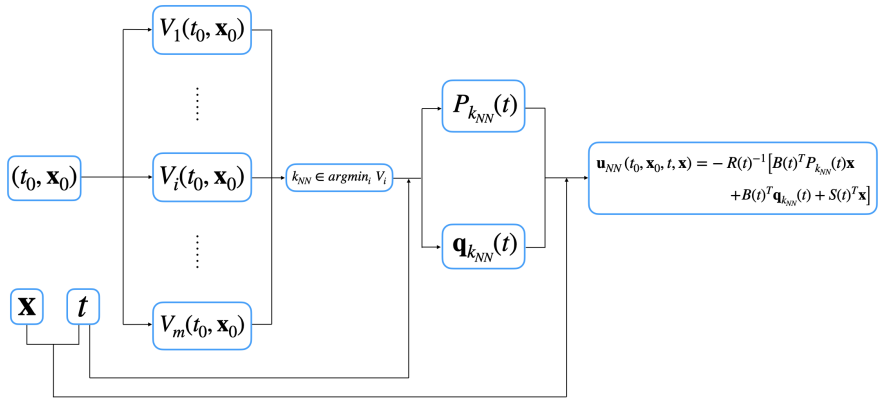


Fig. 4 Illustration of the abstract neural network architecture defined by (22) that can be used to compute the optimal control in the optimal control problem (9).

architecture using deep Resnet neural networks is provided in Section 4 and depicted in Fig. 7.

For completeness, the following proposition proves that the function $\boldsymbol{u}_{NN}$ defined in (22) computes the optimal control in the problem (9) if $(t, \boldsymbol{x})$ is on the optimal trajectory with initial time t_0 and initial position $\boldsymbol{x}_0$.

Proposition 2 *Let $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $t_0 \in [0, T]$ be the initial position and initial time. Assume (A1)-(A2) hold. Let $\boldsymbol{u}^* \colon [t_0, T] \to \mathbb{R}^l$ be a feasible control, and $\boldsymbol{x}^* \in C([t_0, T]; \mathbb{R}^n)$ be the solution to the Cauchy problem (10) with the control $\boldsymbol{u}^*$. Then, the function $\boldsymbol{u}^*$ is an optimal control in the problem (9) if and only if there exists an index k in the set of minimizers of the optimization problem in (23), such that there holds*

$$\boldsymbol{u}^*(t) = \boldsymbol{u}_k(t, \boldsymbol{x}^*(t)) \quad \forall t \in [t_0, T], \tag{24}$$

where the function $\boldsymbol{u}_k \colon [t_0, T] \times \mathbb{R}^n \to \mathbb{R}^l$ is defined by (21) with the index k.

Proof. In this proof, we adopt the same notations as in the proof of Prop. 1, but we abuse the notation V_{NN} with V because they are shown to be equal in the proof of Prop. 1. Let Ψ_i be the function defined by (17). We consider the corresponding optimal control problem (19). It is well known that the optimal control of the linear quadratic optimal control problem (19) exists and is unique under the assumptions (A1)-(A2). We denote the optimal control by $\boldsymbol{v}_i^* \colon [t_0, T] \to \mathbb{R}^l$, and denote the corresponding trajectory by $\boldsymbol{y}_i^* \colon [t_0, T] \to \mathbb{R}^n$. Moreover, the feedback form of the optimal control is given by

$$\begin{aligned} \boldsymbol{v}_i^*(t) &= -R(t)^{-1}(B(t)^T \nabla_{\boldsymbol{x}} V_i(t, \boldsymbol{y}_i^*(t)) + S(t)^T \boldsymbol{y}_i^*(t)) \\ &= -R(t)^{-1}(B(t)^T P_i(t) \boldsymbol{y}_i^*(t) + B(t)^T \boldsymbol{q}_i(t) + S(t)^T \boldsymbol{y}_i^*(t)) \\ &= \boldsymbol{u}_i(t, \boldsymbol{y}_i^*(t)), \end{aligned} \tag{25}$$

for any $t \in [t_0, T]$, where V_i is defined by (20). This is proved, for instance, by [23] and a similar argument in the proof of [155, Chap 6, Thm. 2.8].

Now, we prove the first implication of the proposition statement. Let k be an index satisfying (23), and the function $\boldsymbol{u}_k \colon [t_0, T] \times \mathbb{R}^n \to \mathbb{R}^l$ be defined by (21) with the index k. From (25), we conclude that the open loop control $t \mapsto \boldsymbol{u}^*(t) = \boldsymbol{u}_k(t, \boldsymbol{x}^*(t))$ is the optimal control for the problem (19) with the index k. Moreover, by (18) and the definition of k, the optimal value $V(t_0, \boldsymbol{x}_0)$ in the problem (9) equals the optimal value $V_k(t_0, \boldsymbol{x}_0)$ in the problem (19) with the index k. Therefore, $t \mapsto \boldsymbol{u}^*(t)$ is also an optimal control for the problem (9).

Then, we prove the other implication. Assume $\boldsymbol{u}^*$ is an optimal control in the problem (9), and $\boldsymbol{x}^*$ is the corresponding optimal trajectory. Let $\tilde{k}$ be an index satisfying

$$\tilde{k} \in \operatorname*{arg\,min}_{i \in \{1,\dots,m\}} \Psi_i(\boldsymbol{x}^*(T)).$$

Then, we have

$$\begin{aligned} V(t_0, \boldsymbol{x}_0) &= \int_{t_0}^{T} \Big(\frac{1}{2} \boldsymbol{x}^*(t)^T Q(t) \boldsymbol{x}^*(t) + \frac{1}{2} \boldsymbol{u}^*(t)^T R(t) \boldsymbol{u}^*(t) \\ &\qquad + \boldsymbol{x}^*(t)^T S(t) \boldsymbol{u}^*(t) \Big) dt + \Psi(\boldsymbol{x}^*(T)) \\ &= \int_{t_0}^{T} \Big(\frac{1}{2} \boldsymbol{x}^*(t)^T Q(t) \boldsymbol{x}^*(t) + \frac{1}{2} \boldsymbol{u}^*(t)^T R(t) \boldsymbol{u}^*(t) \\ &\qquad + \boldsymbol{x}^*(t)^T S(t) \boldsymbol{u}^*(t) \Big) dt + \Psi_{\tilde{k}}(\boldsymbol{x}^*(T)) \\ &\geq V_{\tilde{k}}(t_0, \boldsymbol{x}_0) \geq V(t_0, \boldsymbol{x}_0), \end{aligned} \tag{26}$$

where the first equality holds since $\boldsymbol{u}^*$ is an optimal control of (9) with the trajectory $\boldsymbol{x}^*$, the second equality holds by definition of $\tilde{k}$, the first inequality holds since $V_{\tilde{k}}(t_0, \boldsymbol{x}_0)$ is the optimal value of the problem (19) with the index $\tilde{k}$, and the last inequality holds by (18). As a result, the two inequalities in (26) both become equalities, which implies that $\boldsymbol{u}^*$ is the optimal control of the problem (19) with the index $\tilde{k}$, and $\tilde{k}$ is a minimizer of the optimization problem in (23). Recall that the unique optimal control of the problem (19) with the index $\tilde{k}$ satisfies the feedback form (25), and hence we get (24). □

Remark 1 Note that the existence of the optimal control $\boldsymbol{u}^*$ is given by Prop. 2, the existence of k_{NN} in (23), and the existence of the optimal control with terminal cost $\Psi_{k_{NN}}$. However, such $\boldsymbol{u}^*$ may not be unique, since there may be more than one minimizer in (23). It can be seen in the above proposition any minimizer k in (23) can define an optimal control in the problem (9). As a result, if $\arg\min_{i \in \{1,\dots,m\}} V_i(t_0, \boldsymbol{x}_0)$ is not a singleton, then there may be more than one optimal control in the problem (9). This non-uniqueness is possible since the optimal control problem is a non-convex optimization problem, where the terminal condition Ψ is non-convex. For a fixed initial position $\boldsymbol{x}_0$ and initial time t_0, one candidate of open loop optimal control is $\boldsymbol{v}^*_{k_{NN}}$ in (25) with index $i = k_{NN}$. If there are more than one minimizer

in (23), we select k_{NN} to be one minimizer, and then an optimal control $\boldsymbol{u}^*$ is computed using $\boldsymbol{u}_{NN}$ in (22).

Remark 2 We can compute the open-loop optimal control using our proposed neural network architecture $\boldsymbol{u}_{NN}$. For a fixed initial time t_0 and initial position $\boldsymbol{x}_0$, we combine the function $\boldsymbol{u}_{NN}$ with the Cauchy problem (10) and obtain the following Cauchy problem

$$\begin{cases} \dot{\boldsymbol{x}}(s) = A(s)\boldsymbol{x}(s) + B(s)\boldsymbol{u}_{NN}(t_0, \boldsymbol{x}_0, s, \boldsymbol{x}(s)) \quad s \in (t_0, T), \\ \boldsymbol{x}(t_0) = \boldsymbol{x}_0. \end{cases} \tag{27}$$

By straightforward calculation using (11) and (22), the differential equation above becomes

$$\begin{aligned} \dot{\boldsymbol{x}}(s) &= A(s)\boldsymbol{x}(s) + B(s)\boldsymbol{u}_{NN}(t_0, \boldsymbol{x}_0, s, \boldsymbol{x}(s)) \\ &= A(s)\boldsymbol{x}(s) - B(s)R(s)^{-1}\left(B(s)^T P_{k_{NN}}(s)\boldsymbol{x}(s) + B(s)^T \boldsymbol{q}_{k_{NN}}(s) + S(s)^T \boldsymbol{x}(s)\right) \\ &= \left(C_{xp}(s) - C_{pp}(s)P_{k_{NN}}(s)\right)\boldsymbol{x}(s) - C_{pp}(s)\boldsymbol{q}_{k_{NN}}(s). \end{aligned}$$

The solution to this Cauchy problem is the optimal trajectory $\boldsymbol{x}^*$. From the optimal trajectory, we obtain the open loop optimal control $\boldsymbol{u}^*$ by $\boldsymbol{u}^* = \boldsymbol{u}_{NN}(t_0, \boldsymbol{x}_0, s, \boldsymbol{x}^*(s))$. Our numerical results in Section 4 are computed using this procedure.

Remark 3 Note that the function $\boldsymbol{u}_{NN}$ also computes the feedback optimal control. Consider the function $(t, \boldsymbol{x}) \mapsto \boldsymbol{u}_{NN}(t, \boldsymbol{x}, t, \boldsymbol{x})$. Applying Prop. 2 to $\boldsymbol{x} = \boldsymbol{x}_0$ and $t = t_0$, we conclude that the optimal control at t_0 with initial time t_0 and position $\boldsymbol{x}_0$ is $\boldsymbol{u}^*(t_0) = \boldsymbol{u}_{NN}(t_0, \boldsymbol{x}_0, t_0, \boldsymbol{x}_0)$. Since the initial time $t_0 \in [0, T]$ and initial position $\boldsymbol{x}_0 \in \mathbb{R}^n$ can be arbitrary, we conclude that function $(t, \boldsymbol{x}) \mapsto \boldsymbol{u}_{NN}(t, \boldsymbol{x}, t, \boldsymbol{x})$ is a feedback optimal control for problem (9).

3.3 The extension to general terminal costs

Now, we consider more general terminal costs Ψ of the form

$$\Psi(\boldsymbol{x}) = \min_{i \in \{1,\dots,m\}} \Psi_i(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \tag{28}$$

where $\Psi_1, \dots, \Psi_m$ are some convex functions whose proximal points are numerically computable. Note that we no longer assume $\Psi_1, \dots, \Psi_m$ to be quadratics. By min-plus linearity, the solution V is given by

$$V(\boldsymbol{x}, t) = \min_{i \in \{1,\dots,m\}} V_i(\boldsymbol{x}, t) \quad \forall \boldsymbol{x} \in \mathbb{R}^n, t \geq 0, \tag{29}$$

where each V_i is the value function of the optimal control problem with terminal cost Ψ_i. Therefore, to solve this problem, we need to solve the m subproblems. In the i-th subproblem, we solve the optimal control problem with terminal cost Ψ_i. If Ψ_i is in the quadratic form, we apply the aforementioned method and neural network architecture to solve it. Otherwise, we apply the ADMM method [22, 67] to solve the i-th subproblem, whose j-th iteration includes the following three steps:

1. Solve the optimal control problem

$$\min_{(\boldsymbol{x},\boldsymbol{u})\in\mathcal{C}(t_0,\boldsymbol{x}_0)} \left\{ \int_{t_0}^{T} \frac{1}{2}\boldsymbol{x}(s)^T Q(s)\boldsymbol{x}(s) + \frac{1}{2}\boldsymbol{u}(s)^T R(s)\boldsymbol{u}(s) + \boldsymbol{x}(s)^T S(s)\boldsymbol{u}(s)ds + \frac{\rho}{2}\|\boldsymbol{x}(T) - \boldsymbol{y}^{j,i} + \boldsymbol{w}^{j,i}\|^2 \right\} \tag{30}$$

   where the constraint set $\mathcal{C}(t_0, \boldsymbol{x}_0)$ contains the solutions to (10). Denote the optimal trajectory by $\boldsymbol{x}^{j+1,i}(\cdot)$. Note that we add the superscript i the emphasize that these minimizers are used to solve the i-th subproblem. Since the terminal cost in (30) is a quadratic function, this optimal control problem can be solved using Riccati equation, which can be represented by the neural network architecture depicted in Fig. 3 with one neuron (i.e., $m = 1$). The solution is given by $\frac{1}{2}\boldsymbol{x}^T P_i(t)\boldsymbol{x} + \boldsymbol{q}_i(t)^T\boldsymbol{x} + r_i(t)$, where $P_i(t)$, $\boldsymbol{q}_i(t)$ and $r_i(t)$ solve the FVPs (14), (15) and (16) whose terminal conditions are given by

$$P_i(T) = \rho I_n, \quad \boldsymbol{q}_i(T) = \rho(\boldsymbol{w}^{j,i} - \boldsymbol{y}^{j,i}), \quad r_i(T) = \frac{\rho}{2}\|\boldsymbol{w}^{j,i} - \boldsymbol{y}^{j,i}\|^2.$$

2. Solve the following proximal point problem

$$\min_{\boldsymbol{y}\in\mathbb{R}^n} \Psi_i(\boldsymbol{y}) + \frac{\rho}{2}\|\boldsymbol{x}^{j+1,i}(T) - \boldsymbol{y} + \boldsymbol{w}^{j,i}\|^2,$$

   and denote the minimizer by $\boldsymbol{y}^{j+1,i}$.
3. Update $\boldsymbol{w}$ by $\boldsymbol{w}^{j+1,i} = \boldsymbol{w}^{j,i} + \boldsymbol{x}^{j+1,i}(T) - \boldsymbol{y}^{j+1,i}$.

The ADMM algorithm terminates when the number of iteration exceeds the maximal number of iteration or the following inequality holds

$$\max\{\|\boldsymbol{x}^{j+1,i}(T) - \boldsymbol{x}^{j,i}(T)\|, \|\boldsymbol{y}^{j+1,i} - \boldsymbol{y}^{j,i}\|, \|\boldsymbol{w}^{j+1,i} - \boldsymbol{w}^{j,i}\|\} \leq \epsilon$$

for some positive threshold ϵ. If the ADMM algorithm for the i-th subproblem terminates at the N_i-th step, we get the output parameters $\bar{\boldsymbol{w}}^i$ and $\bar{\boldsymbol{y}}^i$ by

$$\bar{\boldsymbol{w}}^i = \boldsymbol{w}^{N_i,i}, \quad \bar{\boldsymbol{y}}^i = \boldsymbol{y}^{N_i,i}.$$

Then, the solution V to the HJ PDE with the general terminal condition in (28) is computed using the neural network architecture in Fig. 3, where the coefficients in the i-th neuron, denoted by $P_i(t)$, $\boldsymbol{q}_i(t)$ and $r_i(t)$, are the solutions to the FVPs (14), (15) and (16) with terminal condition

$$P_i(T) = \rho I_n, \quad \boldsymbol{q}_i(T) = \rho(\bar{\boldsymbol{w}}^i - \bar{\boldsymbol{y}}^i), \quad r_i(T) = \frac{\rho}{2}\|\bar{\boldsymbol{w}}^i - \bar{\boldsymbol{y}}^i\|^2.$$

4 Implementations of the abstract neural network architectures

In the neural network architectures depicted in Figs. 3 and 4, each neuron involves the functions $P_i(t), \boldsymbol{q}_i(t)$ and $r_i(t)$, which are the solutions to the FVPs (14), (15) and (16). As a result, the neural network architectures require a numerical solver for solving the matrix Riccati FVP (14) and the FVPs (15) and (16).

In the literature, there are many numerical methods developed for solving the matrix Riccati differential equation. In order to solve this equation with a general initial or terminal condition, different fundamental solutions are proposed, including but not limited to Davison-Maki fundamental solution [37, 94], symplectic fundamental solution [109], and min-plus fundamental solution [40, 46, 49, 118]. Recently, there are some non-traditional methods developed for solving Riccati equations using ant colony programming [90], genetic programming [10] and neural networks [9, 143].

Algorithm 1: The fourth order Runge-Kutta method for solving the FVP (31).

Input : Time $t \in [t_0, T)$ and the step number N.
Output: The solution $z(t)$ at time t.

Initialization: set $z_N \doteq z_T$, $t_N \doteq T$ and $\Delta t \doteq \frac{T-t}{N}$;
for $k = N, N-1, \ldots, 1$ **do**
  $\delta_1 \doteq -\Delta t g(t_k, z_k)$ and $w_1 \doteq z_k + \frac{1}{2}\delta_1$;
  $\delta_2 \doteq -\Delta t g(t_k - \frac{\Delta t}{2}, w_1)$ and $w_2 \doteq z_k + \frac{1}{2}\delta_2$;
  $\delta_3 \doteq -\Delta t g(t_k - \frac{\Delta t}{2}, w_2)$ and $w_3 \doteq z_k + \delta_3$;
  $\delta_4 \doteq -\Delta t g(t_k - \Delta t, w_3)$;
  Update $z_{k-1} \doteq z_k + \frac{\delta_1}{6} + \frac{\delta_2}{3} + \frac{\delta_3}{3} + \frac{\delta_4}{6}$ and $t_{k-1} \doteq t_k - \Delta t$;
end
The output $z(t)$ is given by z_0.

We adopt the fourth order Runge-Kutta method to solve the general FVP

$$\begin{cases} \dot{z}(t) = g(t, z(t)) \quad t \in [t_0, T], \\ z(T) = z_T, \end{cases} \tag{31}$$

where the function $z\colon [t_0, T] \to \mathbb{R}^\alpha$ (for a positive integer α) is an absolutely continuous function solving the FVP almost everywhere, and the source term $g\colon [t_0, T] \times \mathbb{R}^\alpha \to \mathbb{R}^\alpha$ is continuous with respect to t and uniformly Lipschitz with respect to z. The fourth order Runge-Kutta algorithm for solving this FVP is reviewed in Algorithm 1. Note that the Runge-Kutta solver can be expressed using a neural network architecture (see, for instance, [7]). For illustration, we show in Fig. 5 the architecture corresponding to one step of the fourth order Runge-Kutta solver. This architecture belongs to the class of Resnet architectures proposed in [72]. With this connection, the Runge-Kutta solver can be implemented using standard neural network languages, which

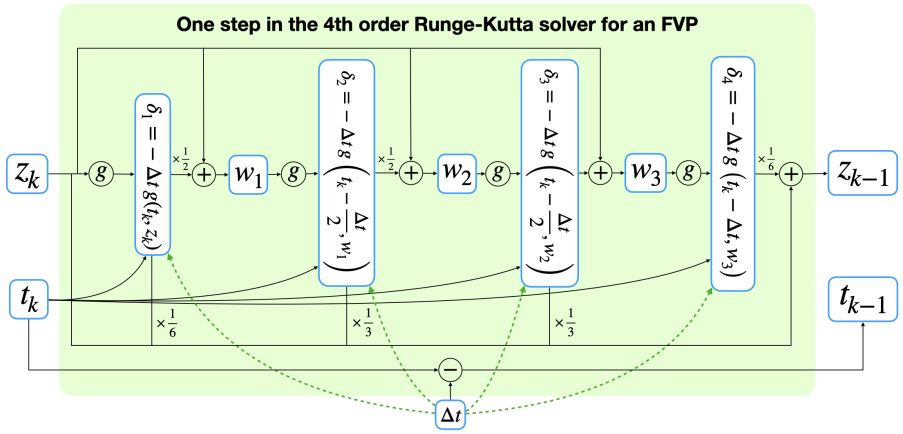


Fig. 5 Illustration of the Resnet architecture that represents one step in the fourth order Runge-Kutta solver shown in Algorithm 1 for solving a general FVP (31).

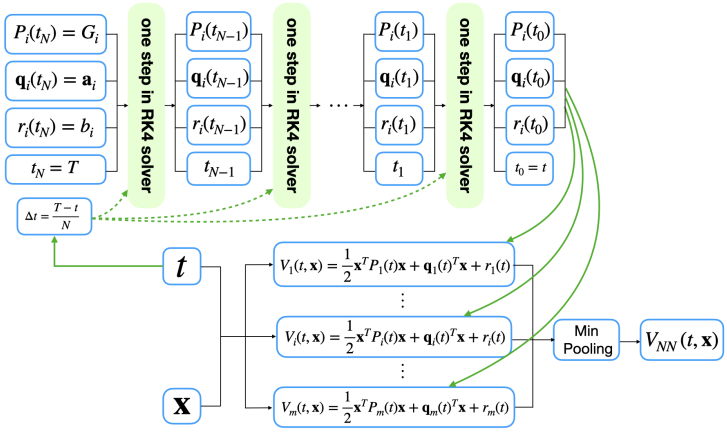


Fig. 6 An implementation of the abstract neural network architecture defined by (22) where $\{V_i(t, \boldsymbol{x})\}_{i=1}^m$ are computed using the Resnet neural network depicted in Fig. 5.

can be converted to executable codes on the dedicated hardware designed for neural networks. By employing the Runge-Kutta solver to evaluate each abstract neuron in the proposed architectures, we obtain their implementations using Resnet-type deep neural networks. The illustrations of the deep neural network implementations for V_{NN} and $\boldsymbol{u}_{NN}$ are shown in Figs. 6 and 7, respectively. From the neural network function $\boldsymbol{u}_{NN}$, we compute the optimal trajectory $\boldsymbol{x}^*$ by solving (27) using the fourth order Runge-Kutta method. Then, we get the open loop control $\boldsymbol{u}^*$ as described in Remark 2. The TensorFlow implementations of these architectures for our examples are given in `https://github.com/TingweiMeng/NN_HJ_minplus`.

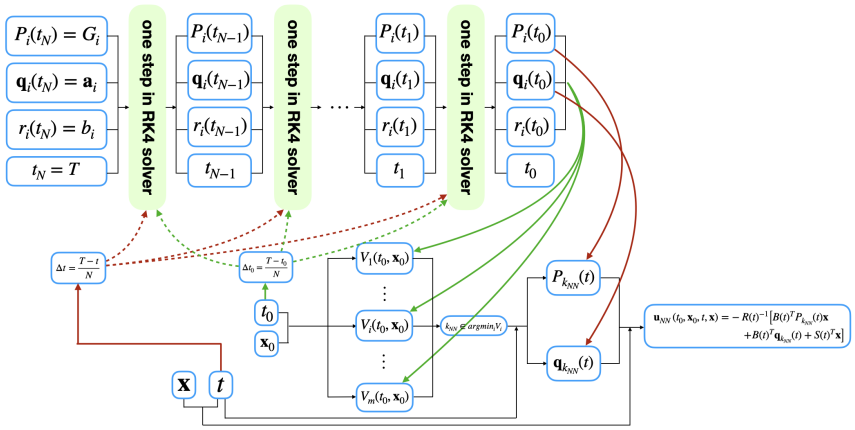


Fig. 7 An implementation of the abstract neural network architecture defined by (22) where the neurons $\{V_i(t_0, \boldsymbol{x}_0)\}_{i=1}^m$, $\boldsymbol{q}_{k_{NN}}(t)$ and $P_{k_{NN}}(t)$ (where k_{NN} is the index defined by (23)) are computed using a fourth order Runge-Kutta method depicted in Fig. 5.

Remark 4 It appears that the residual grows with $\|\boldsymbol{x}\|^2$ for a fixed time $t \in [0, T)$. It is expected because it follows from the Runge-Kutta error estimation (see, for instance, [24]). For a fixed time step (i.e., a fixed number of layers in our proposed neural network in Fig. 6), the error at $(\boldsymbol{x}, t)$ for a fixed time variable $t \in [0, T)$ and any spatial variable $\boldsymbol{x} \in \Omega$ is bounded in a compact set Ω, but not in the whole domain. However, this error will converge to zero as the number of layers goes to infinity.

Our implementations are based on the fourth order Runge-Kutta solver. Note that other ODE solvers can also be applied to compute the neurons in the abstract neural network architectures in Figs. 3 and 4. Each ODE solver which can be represented using neural network architectures provides possible neural network implementations for the two abstract architectures in Figs. 3 and 4, which therefore provide possibilities for leveraging different neural network architectures to solve high dimensional HJ PDEs (12) and corresponding optimal control problems (9).

We will show three numerical experiments. In these experiments, we assume the coefficients $C_{pp}, C_{xx}, C_{xp}, R, Q, S, A, B$ satisfy the assumptions (A1)-(A2). The first example is shown in Sec. 4.1, which has constant coefficients. The second example is shown in Sec. 4.2, whose coefficients depend on the time variable. And the third example is shown in Sec. 4.3, which is a slightly modified version of the HJ PDE (12) and the optimal control problem (9) considered in this paper.

In each example, we use the deep Resnet implementation depicted in Fig. 6 to solve the viscosity solution to the HJ PDEs and the value function in the optimal control problems at different time t. For different terminal time T, we solve the corresponding optimal controls and optimal trajectories with different initial position $\boldsymbol{x}_0$ by the method described in Remark 2 using the deep Resnet

neural network implementation depicted in Fig. 7. If not mentioned explicitly, we use 40 Runge-Kutta layers to compute the viscosity solution V_{NN} and 400 Runge-Kutta layers to compute the optimal controls and optimal trajectories.

To show the solution V_{NN} in high dimensional cases, we plot two dimensional slices of the function $\boldsymbol{x} \mapsto V_{NN}(t, \boldsymbol{x})$ for different time t. We consider the points $\boldsymbol{x} = (x_1, x_2, \mathbf{0}) \in \mathbb{R}^n$ where $(x_1, x_2) \in \mathbb{R}^2$ is any grid point in a two dimensional rectangular domain and $\mathbf{0}$ denotes the zero vector in $\mathbb{R}^{n-2}$. In each figure, the color is given by the function value $V_{NN}(t, x_1, x_2, \mathbf{0})$, and the x and y axes represent the variables x_1 and x_2, respectively. To show the errors of the viscosity solution, we compute the maximal absolute value of the residual (i.e., $\max_{i\in\{1,\dots,m\}} | - \frac{\partial V_i}{\partial t} + H(t, \boldsymbol{x}, \nabla_{\boldsymbol{x}} V_i)|$) in each example, where V_i is defined in (13). We set the number of Runge-Kutta layers to be 20, 40, and 80 to show the dependence of error on the number of layers. The residual values in different example, at different times t, and computed using different number of Runge-Kutta layers are shown in Table 1. From the table, we observe that the magnitude of the absolute values of the residuals is in general small (less than 10^{-6}), which provides a numerical validation that each V_i approximately satisfies the differential equation in (12). Since the solution operator of the HJ PDE (12) is linear with respect to the min-plus algebra, our proposed deep neural network architecture in Figure 6 indeed approximates the viscosity solution to the HJ PDE. The errors also decrease as the number of layers goes to infinity. This observation validates the error analysis of Runge-Kutta solvers (see, for instance, [24]).

	# RK layers	$t = 0.25$	$t = 0.5$	$t = 0.75$
Example 1 in Section 4.1	20	6.97E-06	1.45E-06	6.59E-08
	40	4.21E-07	8.93E-08	4.12E-09
	80	2.59E-08	5.54E-09	2.57E-10
Example 2 in Section 4.2	20	1.29E-07	3.28E-08	2.91E-09
	40	7.94E-09	2.02E-09	1.80E-10
	80	4.92E-10	1.25E-10	1.12E-11
Example 4 in Section 4.4	20	2.24E-07	4.49E-08	3.08E-09
	40	1.39E-08	2.77E-09	1.90E-10
	80	8.64E-10	1.72E-10	1.18E-11
	# RK layers	$t = 0.75$	$t = 0.95$	$t = 0.995$
Example 3 in Section 4.3	20	3.95E-04	1.21E-04	1.44E-08
	40	2.14E-05	7.10E-06	9.01E-10
	80	1.25E-06	4.30E-07	5.67E-11

Table 1 We show the maximal absolute residual $\max_{i\in\{1,\dots,m\}} | - \frac{\partial V_i}{\partial t} + H(t, \boldsymbol{x}, \nabla_{\boldsymbol{x}} V_i)|$ in each example, where V_i is defined in (13). The residual values are computed at different time t, with different Runge-Kutta (RK) layers (which is related to the number of layers in the proposed neural networks).

Also, to illustrate the optimal controls and optimal trajectories in each example, we consider different initial positions $\boldsymbol{x}_0 = (x, \mathbf{0}) \in \mathbb{R}^n$ (where x's are the grid points in a one-dimensional interval and $\mathbf{0}$ denotes the zero vector in $\mathbb{R}^{n-1}$) and a fixed initial time $t_0 = 0$. To avoid ambiguity, we use the notations

$\boldsymbol{u}^*(s|\boldsymbol{x}_0)$ and $\boldsymbol{x}^*(s|\boldsymbol{x}_0)$ to denote the optimal control and trajectory at time s with initial position $\boldsymbol{x}_0$. For each initial time $t_0 = 0$ and initial position $\boldsymbol{x}_0 \in \mathbb{R}^n$, we compute the corresponding optimal control, denoted by $[0,T] \ni s \mapsto \boldsymbol{u}^*(s|\boldsymbol{x}_0) \in \mathbb{R}^l$, and the corresponding optimal trajectory, denoted by $[0,T] \ni s \mapsto \boldsymbol{x}^*(s|\boldsymbol{x}_0) \in \mathbb{R}^n$. For the one dimensional problem, we choose each initial position $x_0 \in \mathbb{R}$ to be a grid point in a one-dimensional interval, and plot the graphs of $u^*(\cdot|x_0)$ or $x^*(\cdot|x_0)$ with different x_0 in one figure. For the high dimensional problem, we choose different initial positions $\boldsymbol{x}_0 = (x, \mathbf{0})$ where x's are the grid points in a one-dimensional interval and $\mathbf{0}$ denotes the zero vector in $\mathbb{R}^{n-1}$, and solve the optimal controls $\boldsymbol{u}^*(\cdot|\boldsymbol{x}_0) = (u_1^*(\cdot|\boldsymbol{x}_0), \ldots, u_l^*(\cdot|\boldsymbol{x}_0))$ and the optimal trajectories $\boldsymbol{x}^*(\cdot|\boldsymbol{x}_0) = (x_1^*(\cdot|\boldsymbol{x}_0), \cdots, x_n^*(\cdot|\boldsymbol{x}_0))$. Then, the graphs of the some components of the optimal controls $\boldsymbol{u}^*(\cdot|\boldsymbol{x}_0)$ or optimal trajectories $\boldsymbol{x}^*(\cdot|\boldsymbol{x}_0)$ are plotted in each figure.

4.1 An optimal control problem with constant coefficients

We consider the optimal control problem (9) with the following constant coefficients

$$\begin{cases} l = n, \\ R = Q = C_{pp} = C_{xx} = C_{xp} = A = B = I_n, \\ S = O_n, \end{cases}$$

where I_n denotes the identity matrix in $\mathbb{R}^{n\times n}$ and O_n denotes the zero matrix in $\mathbb{R}^{n\times n}$. With these coefficients, we solve the optimal control problem (2) whose Lagrangian L in (7) is defined by

$$L(t, \boldsymbol{x}, \boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{x}\|^2 + \frac{1}{2}\|\boldsymbol{u}\|^2 \quad \forall t \in [0,T], \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathbb{R}^l, \tag{32}$$

and the source term f in (7) is defined by

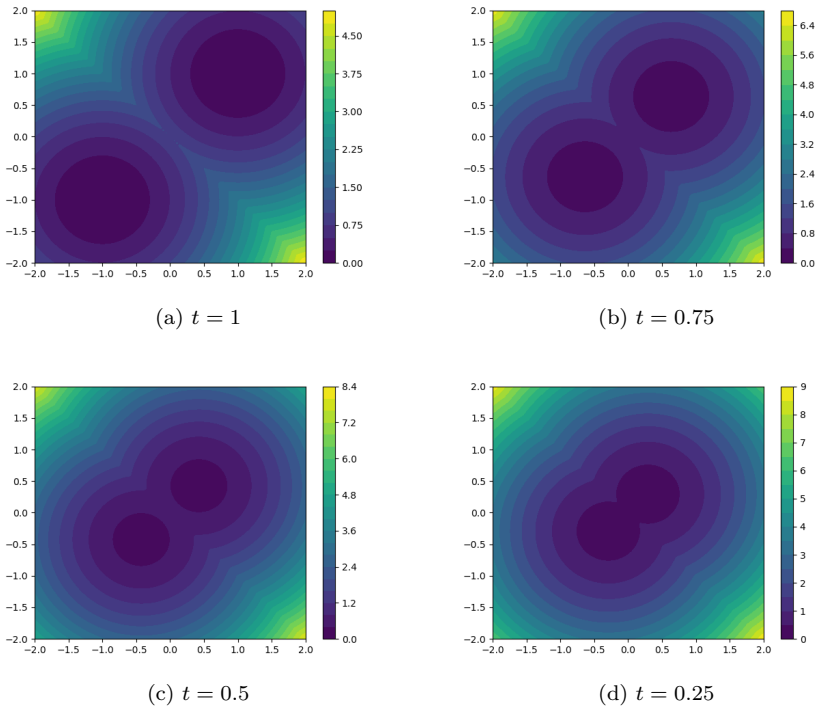
$$f(t, \boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x} + \boldsymbol{u} \quad \forall t \in [0,T], \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathbb{R}^l. \tag{33}$$

The corresponding HJ PDE is in the form of (12) where the Hamiltonian H is defined by

$$H(t, \boldsymbol{x}, \boldsymbol{p}) = \frac{1}{2}\|\boldsymbol{p}\|^2 - \frac{1}{2}\|\boldsymbol{x}\|^2 - \langle \boldsymbol{p}, \boldsymbol{x} \rangle \quad \forall t \in [0,T], \boldsymbol{x}, \boldsymbol{p} \in \mathbb{R}^n. \tag{34}$$

With these coefficients, the differential equations for P_i, $\boldsymbol{q}_i$ and r_i read

$$\begin{aligned} \dot{P}_i(t) &= P_i(t)^T P_i(t) - 2P_i(t) - I_n, \\ \dot{\boldsymbol{q}}_i(t) &= P_i(t)^T \boldsymbol{q}_i(t) - \boldsymbol{q}_i(t), \\ \dot{r}_i(t) &= \frac{1}{2}\|\boldsymbol{q}_i(t)\|^2, \end{aligned}$$



(a) $t = 1$ (b) $t = 0.75$

(c) $t = 0.5$ (d) $t = 0.25$

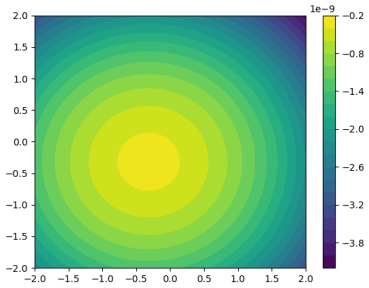
Fig. 8 The viscosity solution V_{NN} to the 16 dimensional HJ PDE (12) with Hamiltonian (34), terminal data (35) and terminal time $T = 1$ is computed using the proposed abstract neural network architecture (13) (depicted in Fig. 3) with the implementation depicted in Fig. 6. The two dimensional slices of V_{NN} at time $t = 1$ (i.e., the terminal cost), $t = 0.75$, $t = 0.5$ and $t = 0.25$ are shown in the subfigures (a), (b), (c) and (d), respectively. The color in each subfigure shows the solution value $V_{NN}(t, \boldsymbol{x})$, where the spatial variable $\boldsymbol{x}$ is in the form of $(x_1, x_2, \mathbf{0}) \in \mathbb{R}^{16}$ (with $\mathbf{0}$ denoting the zero vector in $\mathbb{R}^{14}$) for some points $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ which are represented by the x and y axes.

for each $t \in (0, T)$. We consider this high dimensional problem with $n = 16$ and $m = 2$, where the terminal data Ψ is defined by

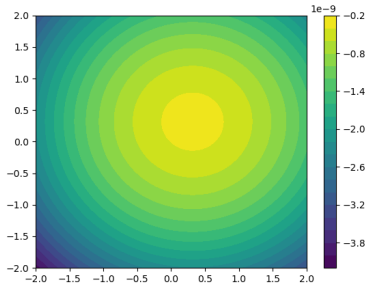
$$\Psi(\boldsymbol{x}) = \min\left\{\frac{1}{2}\left(\sum_{i=1}^{2}(x_i+1)^2 + \sum_{i=3}^{16} x_i^2\right), \frac{1}{2}\left(\sum_{i=1}^{2}(x_i-1)^2 + \sum_{i=3}^{16} x_i^2\right)\right\}, \tag{35}$$

for each $\boldsymbol{x} = (x_1, \ldots, x_{16}) \in \mathbb{R}^{16}$.

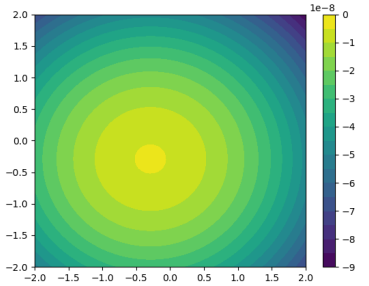
The viscosity solution to the HJ PDE (12) with Hamiltonian in (34) and terminal data in (35) is computed using the proposed abstract neural network (13) (depicted in Fig. 3) with the implementation depicted in Fig. 6. The two dimensional slices of the solution V_{NN} to the 16 dimensional problem with terminal time $T = 1$ is shown in Fig. 8. In this figure, the solution



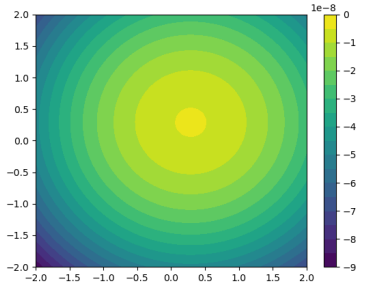
(a) $t = 0.75$



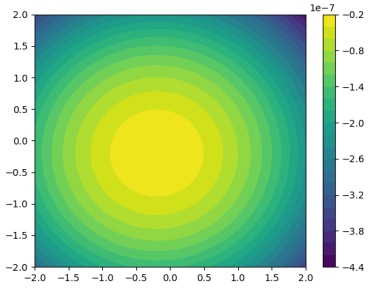
(b) $t = 0.75$



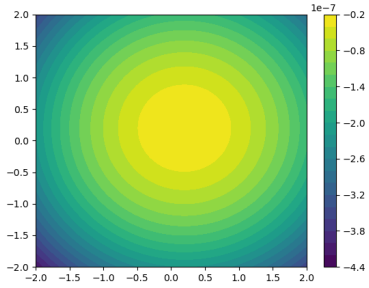
(c) $t = 0.5$



(d) $t = 0.5$

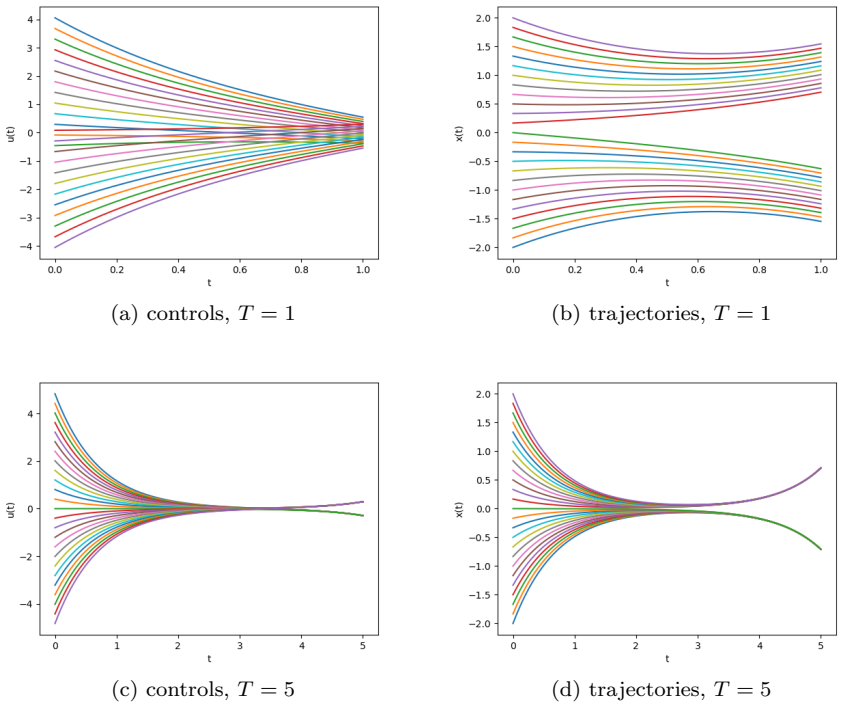


(e) $t = 0.25$



(f) $t = 0.25$

Fig. 9 The residual $-\frac{\partial V_i}{\partial t} + H(t, \boldsymbol{x}, \nabla_{\boldsymbol{x}} V_i)$ in the HJ PDE (12) with Hamiltonian (34) (with terminal time $T = 1$) is shown for each $i \in \{1, \ldots, m\}$ at different time t, where V_i is defined in (13). Figures (a), (c), (e) show the residuals for V_1 at time $t = 0.75$, $t = 0.5$ and $t = 0.25$, while figures (b), (d), (f) show the residuals for V_2 at time $t = 0.75$, $t = 0.5$ and $t = 0.25$, respectively. In each subfigure, we show the two dimensional slices of the residual function. The color shows the residual value at $(t, \boldsymbol{x})$, where the spatial variable $\boldsymbol{x}$ is in the form of $(x_1, x_2, \mathbf{0}) \in \mathbb{R}^{16}$ (with $\mathbf{0}$ denoting the zero vector in $\mathbb{R}^{14}$) for some points $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ which are represented by the x and y axes.



(a) controls, $T = 1$ (b) trajectories, $T = 1$

(c) controls, $T = 5$ (d) trajectories, $T = 5$

Fig. 10 The open loop optimal controls and the corresponding optimal trajectories in the 16 dimensional optimal control problem (2) with Lagrangian (32), source term (33), terminal cost (35) and different terminal time $T = 1, 5$ are computed using the proposed abstract neural network architecture (22) with the implementation depicted in Fig. 7. Several graphs of the first component of the optimal controls with $T = 1$ are shown in (a), and the first component of the corresponding optimal trajectories are shown in (b). Several graphs of the first component of the optimal controls with $T = 5$ are shown in (c), and the first component of the corresponding optimal trajectories are shown in (d).

$V_{NN}(t, x_1, x_2, \mathbf{0})$ (where $\mathbf{0}$ denotes the zero vector in $\mathbb{R}^{n-2}$) at time $t = 1$, $t = 0.75$, $t = 0.5$ and $t = 0.25$ is shown in the subfigures (a), (b), (c) and (d), respectively. Recall that the x and y axes represent the first component x_1 and second component x_2 of the spatial variable, respectively. We also show in Figure 9 the (two-dimensional slices of) the residual $-\frac{\partial V_i}{\partial t} + H(t, \boldsymbol{x}, \nabla_{\boldsymbol{x}} V_i)$ for each $i \in \{1, \ldots, m\}$, where V_i is defined in (13). Note that the magnitude of the residuals is in general small (less than 10^{-6}), which provides a numerical validation that each V_i approximately satisfies the differential equation in (12). Since the solution operator of the HJ PDE (12) is linear with respect to the min-plus algebra, our proposed deep neural network architecture in Figure 6 indeed approximates the viscosity solution to the HJ PDE. It appears that the residual grows with $|\boldsymbol{x}|^2$ for a fixed time $t \in [0, T)$. It is expected because it follows from the Runge-Kutta error estimation. For a fixed time step (i.e., a fixed number of layers in our proposed neural network in Fig. 6), the error is

bounded in a compact set, but not in the whole domain. However, this error will converge to zero as the number of layers goes to infinity.

We compute optimal controls and optimal trajectories with different initial positions $\boldsymbol{x}_0 = (x, \mathbf{0}) \in \mathbb{R}^n$ (where $\mathbf{0}$ denotes the zero vector in $\mathbb{R}^{n-1}$) and a fixed initial time $t_0 = 0$. The first component of the optimal controls and trajectories is illustrated in Fig. 10. The optimal controls with terminal time $T = 1$ are shown in (a), the optimal trajectories with terminal time $T = 1$ are shown in (b), the optimal controls with terminal time $T = 5$ are shown in (c), and the optimal trajectories with terminal time $T = 5$ are shown in (d). There does appear to be a turnpike phenomenon for the longer time horizon (see, for instance, [69, 156] and the references in there).

4.2 An optimal control problem with time dependent coefficients

We consider the optimal control problem (9) whose coefficients depend on time. The coefficients are chosen to be

$$\begin{cases} l = n, \\ R(t) = 4e^{-t} I_n,\ Q(t) = \dfrac{e^{-t}}{2} I_n,\ S(t) \equiv e^{-t} I_n,\ A(t) \equiv \dfrac{1}{2} I_n,\ B(t) \equiv I_n, \\ C_{pp}(t) = \dfrac{e^{t}}{4} I_n,\ C_{xp}(t) = \dfrac{1}{4} I_n,\ C_{xx}(t) = \dfrac{e^{-t}}{4} I_n, \end{cases}$$

for each $t \in [0, T]$, where I_n denotes the identity matrix in $\mathbb{R}^{n \times n}$. With these coefficients, we solve the optimal control problem (2) whose Lagrangian L in (7) reads

$$L(t, \boldsymbol{x}, \boldsymbol{u}) = \frac{e^{-t}}{4} \|\boldsymbol{x}\|^2 + 2e^{-t} \|\boldsymbol{u}\|^2 + e^{-t} \boldsymbol{x}^T \boldsymbol{u} \quad \forall t \in [0, T], \boldsymbol{x}, \boldsymbol{u} \in \mathbb{R}^n, \tag{36}$$

and the source term f in (7) reads

$$f(t, \boldsymbol{x}, \boldsymbol{u}) = \frac{\boldsymbol{x}}{2} + \boldsymbol{u} \quad \forall t \in [0, T], \boldsymbol{x}, \boldsymbol{u} \in \mathbb{R}^n. \tag{37}$$

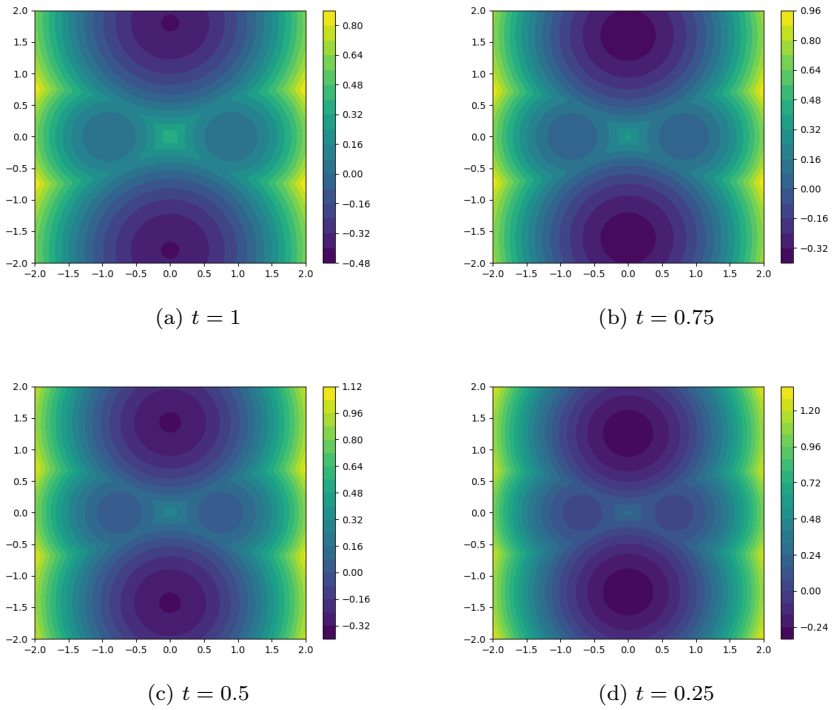
The corresponding HJ PDE is in the form of (12) where the Hamiltonian is defined by

$$H(t, \boldsymbol{x}, \boldsymbol{p}) = \frac{e^{t}}{8} \|\boldsymbol{p}\|^2 - \frac{e^{-t}}{8} \|\boldsymbol{x}\|^2 - \frac{1}{4} \boldsymbol{p}^T \boldsymbol{x} \quad \forall t \in [0, T], \boldsymbol{x}, \boldsymbol{p} \in \mathbb{R}^n. \tag{38}$$

With these coefficients, the differential equations for P_i, $\boldsymbol{q}_i$ and r_i read

$$\begin{aligned} \dot{P}_i(t) &= \frac{e^{t}}{4} P_i(t)^T P_i(t) - \frac{1}{2} P_i(t) - \frac{e^{-t}}{4} I_n, \\ \dot{\boldsymbol{q}}_i(t) &= \frac{e^{t}}{4} P_i(t)^T \boldsymbol{q}_i(t) - \frac{1}{4} \boldsymbol{q}_i(t), \\ \dot{r}_i(t) &= \frac{e^{t}}{8} \|\boldsymbol{q}_i(t)\|^2, \end{aligned}$$

for each $t \in (0, T)$. The running cost (36) involves a discount factor of 1. If the terminal cost was similarly discounted, we would expect to see this appear in the HJ PDE as a $-V$ term, see, e.g., [13, Section III.3.1].

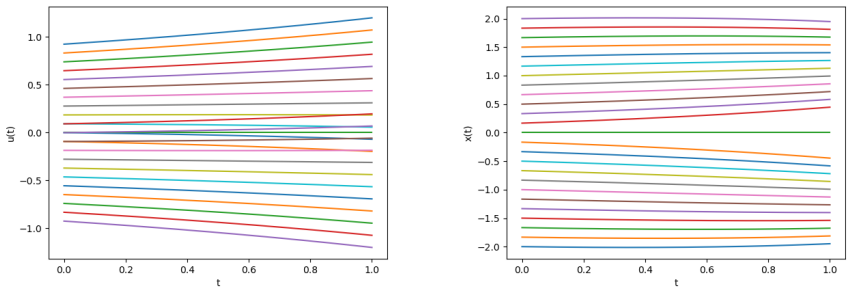


(a) $t = 1$ (b) $t = 0.75$

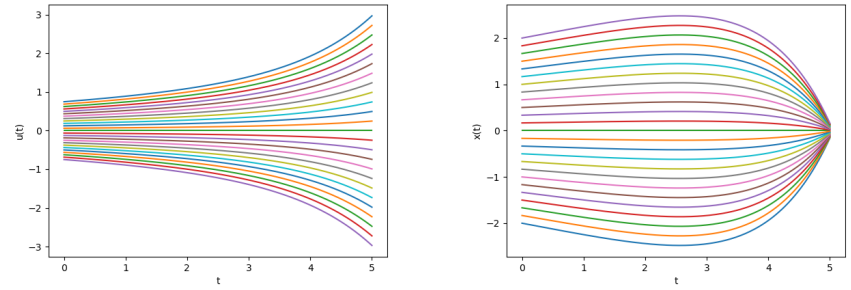
(c) $t = 0.5$ (d) $t = 0.25$

Fig. 11 The viscosity solution V_{NN} to the 16 dimensional HJ PDE (12) with Hamiltonian (38), terminal data (28) (where Ψ_i's are defined in (39)) and terminal time $T = 1$ is computed using the proposed abstract neural network architecture (13) with the implementation depicted in Fig. 6. The two dimensional slices of V_{NN} at time $t = 1$ (i.e., the terminal cost), $t = 0.75$, $t = 0.5$ and $t = 0.25$ are shown in the subfigures (a), (b), (c) and (d), respectively. The color in each subfigure shows the solution value $V_{NN}(t, \boldsymbol{x})$, where the spatial variable $\boldsymbol{x}$ is in the form of $(x_1, x_2, \mathbf{0}) \in \mathbb{R}^{16}$ (where $\mathbf{0}$ is the zero vector in $\mathbb{R}^{14}$) for some points $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ which are represented by x and y axes.

In what follows, we show the viscosity solution V_{NN}, the optimal controls $\boldsymbol{u}^*$, and the optimal trajectories $\boldsymbol{x}^*$ computed using the neural network implementations depicted in Figs. 6 and 7. We solve a 16 dimensional problem: $n = 16$, $m = 4$, and Ψ is defined by (28), where $\Psi_1, \Psi_2, \Psi_3, \Psi_4 \colon \mathbb{R}^{16} \to \mathbb{R}$ are



(a) controls, $T = 1$ (b) trajectories, $T = 1$



(c) controls, $T = 5$ (d) trajectories, $T = 5$

Fig. 12 The open loop optimal controls and the corresponding optimal trajectories in the 16 dimensional optimal control problem (2) with Lagrangian (36), source term (37), terminal cost (28) (where Ψ_i's are defined in (39)) and different terminal time $T = 1$, 5 are computed using the proposed abstract neural network architecture (22) with the implementation depicted in Fig. 7. Several graphs of the first component of the optimal controls with $T = 1$ and initial positions $\boldsymbol{x}_0 = (x, \mathbf{0}) \in \mathbb{R}^{16}$ (where $\mathbf{0}$ is the zero vector in $\mathbb{R}^{15}$) are shown in (a), and the first component of the corresponding optimal trajectories are shown in (b). Several graphs of the first component of the optimal controls with $T = 5$ and initial positions $\boldsymbol{x}_0 = (x, \mathbf{0}) \in \mathbb{R}^{16}$ (where $\mathbf{0}$ is the zero vector in $\mathbb{R}^{15}$) are shown in (c), and the first component of the corresponding optimal trajectories are shown in (d).

defined by

$$\begin{aligned} \Psi_1(\boldsymbol{x}) &= 0.5\|\boldsymbol{x}\|^2 + 0.9x_1 + 0.405, \\ \Psi_2(\boldsymbol{x}) &= 0.5\|\boldsymbol{x}\|^2 - 0.9x_1 + 0.405, \\ \Psi_3(\boldsymbol{x}) &= 0.25\|\boldsymbol{x}\|^2 + 0.9x_2 + 0.405, \\ \Psi_4(\boldsymbol{x}) &= 0.25\|\boldsymbol{x}\|^2 - 0.9x_2 + 0.405, \end{aligned} \tag{39}$$

for each $\boldsymbol{x} = (x_1, \ldots, x_{16}) \in \mathbb{R}^{16}$.

The HJ PDE (12) with Hamiltonian (38) and terminal data Ψ defined in (28) (where Ψ_i's are defined in (39)) is computed using the proposed abstract neural network (13) with the implementation depicted in Fig. 6. The terminal time is set to be $T = 1$. The two dimensional slices for the viscosity solution

are shown in Fig. 11. The subfigures (a), (b), (c), (d) show the solution at time $t = 1$, $t = 0.75$, $t = 0.5$, $t = 0.25$, respectively.

We also solve the optimal control problem with different terminal time $T = 1$ and $T = 5$ using the neural network given by the abstract neural network architecture (22) with the implementation depicted in Fig. 7. The open loop optimal controls $s \mapsto \boldsymbol{u}^*(s|\boldsymbol{x}_0)$ and the corresponding optimal trajectories $s \mapsto \boldsymbol{x}^*(s|\boldsymbol{x}_0)$ are then computed by solving (27) with the fourth order Runge-Kutta method whose one-step neural network representation is shown in Fig. 5. The graphs of the first component of the optimal controls $\boldsymbol{u}^*$ and the first component of the optimal trajectories $\boldsymbol{x}^*$ to the 16 dimensional problem with the initial position $\boldsymbol{x}_0 = (x, \mathbf{0}) \in \mathbb{R}^{16}$ (where $\mathbf{0}$ is the zero vector in $\mathbb{R}^{15}$) and the terminal cost (28) (where Ψ_i's are defined in (39)) are shown in Fig. 12. In the figure, (a) and (c) show the optimal controls $\boldsymbol{u}^*$ with $T = 1$ and $T = 5$, respectively, while (b) and (d) show the corresponding optimal trajectories $\boldsymbol{x}^*$ with $T = 1$ and $T = 5$, respectively.

4.3 An optimal control problem in Newton mechanics

We consider the optimal control problem (2) whose Lagrangian L reads

$$L(t, \boldsymbol{x}, \boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_r(t)\|^2 + \frac{1}{2000}\|\boldsymbol{u}\|^2 \quad \forall t \in [0, T], \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathbb{R}^l, \tag{40}$$

where we set $n = 2l$ and define the function $\boldsymbol{x}_r \colon [0, T] \to \mathbb{R}^n$ by

$$\boldsymbol{x}_r(t) = 5 \sin t \begin{pmatrix} I_l \\ O_l \end{pmatrix} + 5 \cos t \begin{pmatrix} O_l \\ I_l \end{pmatrix}.$$

The source term f in (3) is defined by

$$f(t, \boldsymbol{x}, \boldsymbol{u}) = \begin{pmatrix} O_l & I_l \\ O_l & O_l \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} O_l \\ I_l \end{pmatrix} \boldsymbol{u} \quad \forall t \in [0, T], \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{u} \in \mathbb{R}^l. \tag{41}$$

If we denote $\boldsymbol{x}(s) = (\boldsymbol{x}_1(s), \boldsymbol{x}_2(s))$, where $\boldsymbol{x}_1(s), \boldsymbol{x}_2(s) \in \mathbb{R}^l$ for each $s \in [t_0, T]$, then the Cauchy problem (3) becomes

$$\begin{cases} \dot{\boldsymbol{x}}_1(s) = \boldsymbol{x}_2(s) & s \in (t_0, T), \\ \dot{\boldsymbol{x}}_2(s) = \boldsymbol{u}(s) & s \in (t_0, T), \\ (\boldsymbol{x}_1(t_0), \boldsymbol{x}_2(t_0)) = \boldsymbol{x}_0. \end{cases}$$

This is the ODE in Newton mechanics, where $\boldsymbol{x}_1$ denotes the position of a particle, $\boldsymbol{x}_2$ denotes its velocity, and $\boldsymbol{u}$ denotes its acceleration. The corresponding HJ PDE is in the form of (12) where the Hamiltonian H is defined by

$$H(t, \boldsymbol{x}, \boldsymbol{p}) = \frac{1}{2}\langle \boldsymbol{p}, C_{pp}\boldsymbol{p}\rangle - \langle \boldsymbol{p}, C_{xp}\boldsymbol{x}\rangle - \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_r(t)\|^2 \quad \forall t \in [0, T], \boldsymbol{x}, \boldsymbol{p} \in \mathbb{R}^n, \tag{42}$$

where the coefficients $C_{pp}, C_{xp} \in \mathbb{R}^{n\times n}$ are constant matrices given by

$$C_{pp} = 1000 \begin{pmatrix} O_l & O_l \\ O_l & I_l \end{pmatrix} \quad \text{and} \quad C_{xp} = \begin{pmatrix} O_l & I_l \\ O_l & O_l \end{pmatrix}.$$

We consider the terminal data $\Psi\colon \mathbb{R}^n \to \mathbb{R}$ defined by

$$\Psi(\boldsymbol{x}) = \min\left\{ \frac{1}{320}\left((x_1+2)^2 + \sum_{i=2}^{n} x_i^2 \right), \frac{1}{320}\left((x_1-2)^2 + \sum_{i=2}^{n} x_i^2 \right) \right\}, \quad (43)$$

for each $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$.

Note that this problem requires a slight modification of the HJ PDE (12) and the optimal control problem (9) considered in this paper, because of the term $\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}_r(t)\|^2$. However, the two abstract neural network architectures (13) and (22) can still be used to compute the viscosity solution and the optimal control, where in the i-th neuron, the function $P_i \in C([0,T]; S^n)$ solves the Riccati FVP (14) which reads

$$\begin{cases} \dot{P}_i(t) = 1000 P_i(t)^T \begin{pmatrix} O_l & O_l \\ O_l & I_l \end{pmatrix} P_i(t) - P_i(t)^T \begin{pmatrix} O_l & I_l \\ O_l & O_l \end{pmatrix} \\ \qquad\qquad - \begin{pmatrix} O_l & O_l \\ I_l & O_l \end{pmatrix} P_i(t) - I_n \qquad\qquad t \in (0,T), \\ P_i(T) = \dfrac{1}{160} I_n, \end{cases}$$

the functions $\boldsymbol{q}_i \in C(0,T;\mathbb{R}^n)$ solves the modified FVP which reads

$$\begin{cases} \dot{\boldsymbol{q}}_i(t) = 1000 P_i(t)^T \begin{pmatrix} O_l & O_l \\ O_l & I_l \end{pmatrix} \boldsymbol{q}_i(t) - \begin{pmatrix} O_l & O_l \\ I_l & O_l \end{pmatrix} \boldsymbol{q}_i(t) + \boldsymbol{x}_r(t) \quad t \in (0,T), \\ \boldsymbol{q}_i(T) = \boldsymbol{a}_i, \end{cases}$$

and $r_i \in C(0,T;\mathbb{R})$ solves the modified FVP which reads

$$\begin{cases} \dot{r}_i(t) = 500 \boldsymbol{q}_i(t)^T \begin{pmatrix} O_l & O_l \\ O_l & I_l \end{pmatrix} \boldsymbol{q}_i(t) - \dfrac{25l}{2} \quad t \in (0,T), \\ r_i(T) = b_i. \end{cases}$$

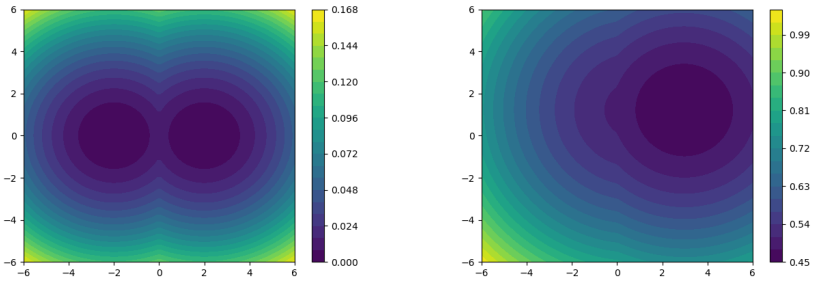
For the specific terminal data Ψ in (43), $\boldsymbol{a}_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ are given by

$$\boldsymbol{a}_1 = \frac{1}{80}(1, \boldsymbol{0})^T, \quad b_1 = \frac{1}{80}, \quad \boldsymbol{a}_2 = -\frac{1}{80}(1, \boldsymbol{0})^T, \quad b_2 = \frac{1}{80},$$

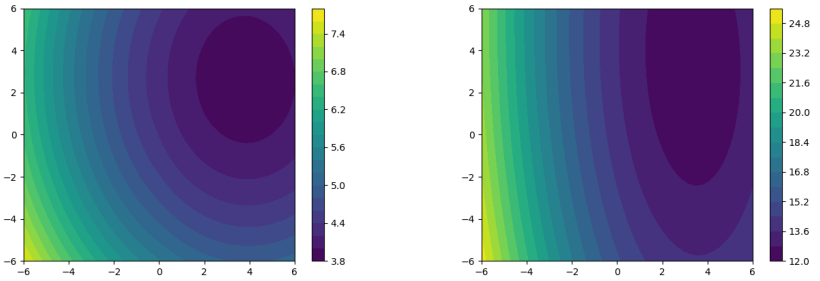
where $\boldsymbol{0}$ denotes the zero vector in $\mathbb{R}^{n-1}$.

Here, we show the numerical results for $l = 8$ and $n = 2l = 16$. The viscosity solution V_{NN} with terminal time $T = 1$ is computed using the abstract neural network architecture (13) (depicted in Fig. 3) with the implementation shown in Fig. 6. The two dimensional slices of V_{NN} at $t = 1$, 0.995, 0.95, 0.75 are plotted in Fig. 13 (a), (b), (c), (d), respectively.

The optimal controls with different terminal time $T = 1, 5, 10$ are computed using the abstract neural network architecture (22) depicted in Fig. 4 with the implementation depicted in Fig. 7. The open loop optimal controls and the corresponding optimal trajectories are computed by solving (27) with the fourth order Runge-Kutta method whose one-step neural network representation is shown in Fig. 5. The graphs of the first components of the optimal trajectories with terminal time $T = 1$, $T = 5$, $T = 10$ and different initial positions $\boldsymbol{x}_0 = (x, \mathbf{0}) \in \mathbb{R}^{16}$ (where $\mathbf{0}$ is the zero vector in $\mathbb{R}^{15}$) are shown in Fig. 14 (a), (b), (c), respectively. The graphs of the first components of the corresponding optimal trajectories with $T = 1, 5, 10$ are shown in Fig. 15 (a), (c), (e), while the graphs of the ninth components of the optimal trajectories with $T = 1, 5, 10$ are shown in Fig. 15 (b), (d), (f). From the optimal controls and trajectories for longer time horizons, it appears there is a turnpike phenomenon.

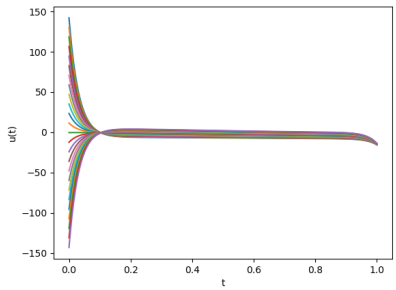


(a) $t = 1$ (b) $t = 0.995$

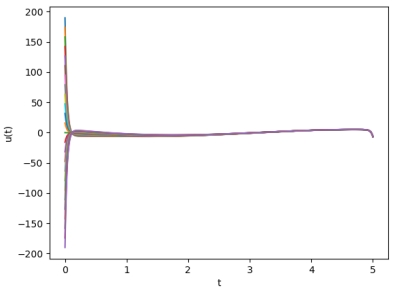


(c) $t = 0.95$ (d) $t = 0.75$

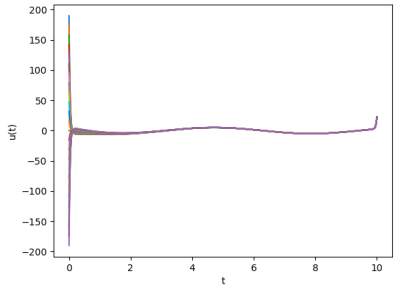
Fig. 13 The viscosity solution V_{NN} to the 16 dimensional HJ PDE (12) with Hamiltonian (42), terminal data (43) and terminal time $T = 1$ is computed using the proposed abstract neural network architecture (13) (depicted in Fig. 3) with the implementation depicted in Fig. 6. The two dimensional slices of V_{NN} at time $t = 1$ (i.e., the terminal cost), $t = 0.995$, $t = 0.95$ and $t = 0.75$ are shown in the subfigures (a), (b), (c) and (d), respectively. The color in each subfigure shows the solution value $V_{NN}(t, \boldsymbol{x})$, where the spatial variable $\boldsymbol{x}$ is in the form of $(x_1, \mathbf{0}, x_2, \mathbf{0}) \in \mathbb{R}^{16}$ (with $\mathbf{0}$ denoting the zero vector in $\mathbb{R}^7$) for some points $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ which are represented by the x and y axes.



(a) $T = 1$



(b) $T = 5$



(c) $T = 10$

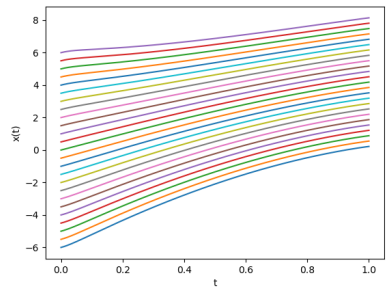
Fig. 14 The open loop optimal controls in the 16 dimensional optimal control problem (2) with Lagrangian (40), source term (41), terminal cost (43) and different terminal time $T = 1, 5, 10$ are computed using the proposed abstract neural network architecture (22) depicted in Fig. 4 with the implementation shown in Fig. 7. Several graphs of the first component of the optimal controls with $T = 1$, $T = 5$ and $T = 10$ are shown in (a), (b) and (c), respectively. In each figure, different trajectories correspond to different initial positions $\boldsymbol{x}_0 = (x, \mathbf{0}) \in \mathbb{R}^{16}$ where $\mathbf{0}$ is the zero vector in $\mathbb{R}^{15}$.

4.4 An optimal control problem with general terminal cost

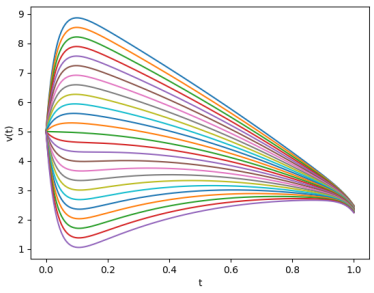
In this section, we consider a more general terminal cost Ψ in the form of (28) with $m = 2$, where Ψ_1 and Ψ_2 are defined by

$$\Psi_1(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_1\|_1, \quad \Psi_2(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_2\|, \tag{44}$$

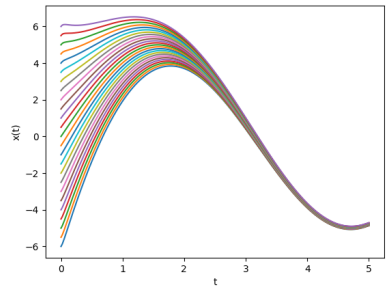
for all $\boldsymbol{x} \in \mathbb{R}^n$. Recall that $\|\cdot\|_1$ and $\|\cdot\|$ denote the ℓ^1-norm and ℓ^2-norm in $\mathbb{R}^n$, respectively. For illustration purposes, we set $\boldsymbol{x}_1 = (1, 1, \mathbf{0})$ and $\boldsymbol{x}_2 = (-1, -1, \mathbf{0})$. We solve the same optimal control problem as in Section 4.2, with the terminal cost (28) (where Ψ_i's are defined in (44)). The value function V_{NN} and the optimal control $\boldsymbol{u}$ are computed using the neural network architecture in Figs. 6 and 7, respectively, where the parameters $G_i, \boldsymbol{a}_i$ and b_i



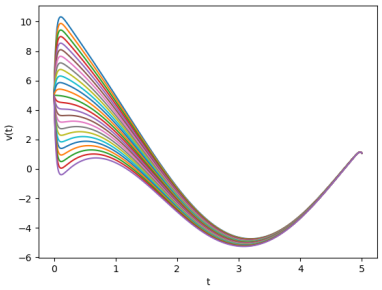
(a) 1st component x_1^*, $T = 1$



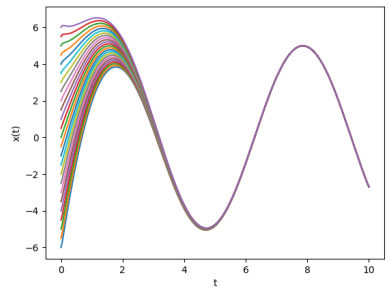
(b) 9th component x_9^*, $T = 1$



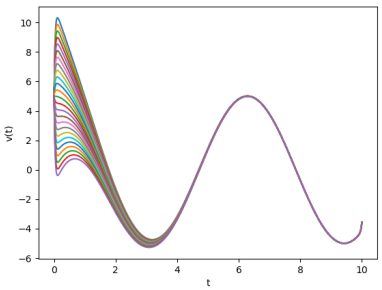
(c) 1st component x_1^*, $T = 5$



(d) 9th component x_9^*, $T = 5$



(e) 1st component x_1^*, $T = 10$



(f) 9th component x_9^*, $T = 10$

Fig. 15 The corresponding optimal trajectories $\boldsymbol{x}^*$ in the 16 dimensional optimal control problem (2) with Lagrangian (40), source term (41), terminal cost (43) and different terminal time $T = 1, 5, 10$ are computed using the proposed abstract neural network architecture (22) depicted in Fig. 4 with the implementation shown in Fig. 7. Several graphs of the first component x_1^* of the optimal trajectories $\boldsymbol{x}^*$ with $T = 1$, $T = 5$ and $T = 10$ are shown in (a), (c) and (e), respectively. The corresponding graphs of the ninth component x_9^* of the optimal trajectories $\boldsymbol{x}^*$ with $T = 1$, $T = 5$ and $T = 10$ are shown in (b), (d) and (f), respectively. In each figure, different trajectories correspond to different initial positions $\boldsymbol{x}_0 = (x, \mathbf{0}) \in \mathbb{R}^{16}$ where $\mathbf{0}$ is the zero vector in $\mathbb{R}^{15}$.

are trained using the ADMM algorithm described in Section 3.3. Note that in each iteration in the ADMM algorithm, we need to solve the optimal control problem (30) once using our proposed neural network architecture in Fig. 6.

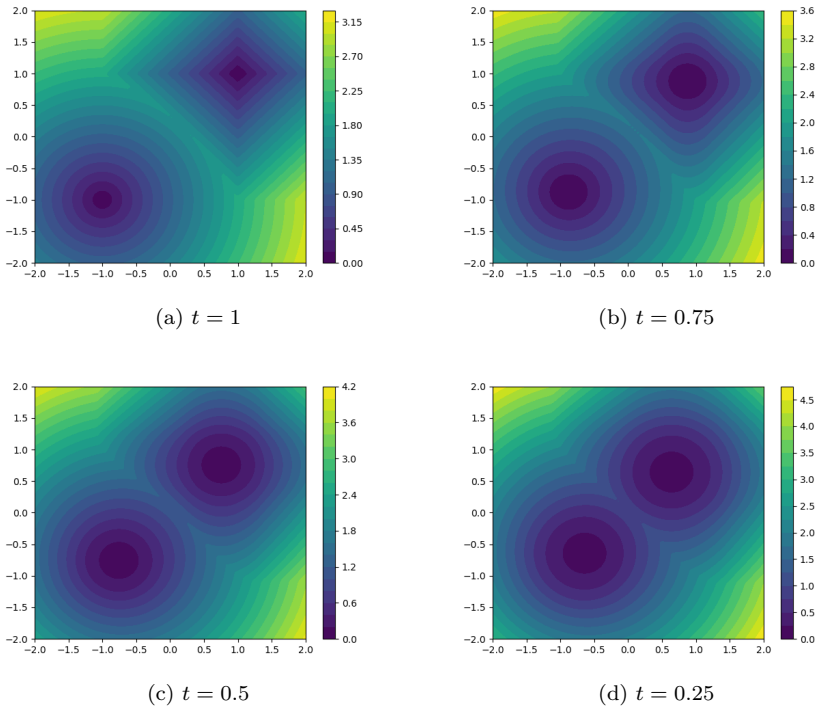
We show the numerical results for the 16-dimensional problem. The two dimensional slices of the viscosity solution V_{NN} with terminal time $T = 1$ at time $t = 1, 0.75, 0.5, 0.25$ are plotted in Fig. 16 (a), (b), (c), (d), respectively. The residual $-\frac{\partial V_i}{\partial t} + H(t, \boldsymbol{x}, \nabla_{\boldsymbol{x}} V_i)$ of the HJ PDE with terminal condition Ψ_1 and Ψ_2 at different time $t = 0.75, 0.5, 0.25$ are shown in Fig. 17. We can observe a small error from these error plots, which numerically validate that our proposed neural network architecture indeed solves the viscosity solution to the corresponding HJ PDE.

We also compute several optimal controls and trajectories with different initial position $\boldsymbol{x} = (x, \mathbf{0}) \in \mathbb{R}^n$ and fixed initial time $t_0 = 0$, and the graphs of their first components are shown in Fig. 18. The optimal controls with terminal time $T = 1, 5$ are shown in Fig. 18 (a), (c), while the optimal trajectories with terminal time $T = 1, 5$ are shown in Fig. 18 (b), (d), respectively.

4.5 An FPGA implementation and numerical results

We now briefly describe am implementation of our proposed neural network on FPGA to illustrate the performance that can be achieved using simple precision floating points. Specifically, we only present an FPGA implementation with low latency, where the latency corresponds to the amount of time the neural network takes to produce one result.

FPGAs are an array of programmable logic blocks and memory elements that are connected together using a programmable interconnect. FPGAs contain different types of logic resources. These resources include general purposes logics such as lookup tables (LUTs) and Flip-Flops (FFs), more specialized arithmetic units, such as digital signal processing units (DSPs), and memory such as Block Random Access Memory (BRAMs). We refer the reader to [93] for a concise description of FPGAs. We use the Xilinx Alveo U280 board with a target design running at 300 MHz. The main computational burden of our proposed neural network consists of matrix-matrix multiplications used in the fourth order Runge-Kutta method for solving an FVP as described in Fig. 5. Traditional non-parallel algorithms for performing matrix-matrix multiplications have an $O(n^3)$ time complexity. Using the parallel programming feature of FPGAs we can obtain a complexity of $O(n^2)$ for computing matrix-matrix multiplications (see [93] for instance). Therefore, we spend most of FPGA resources on performing these matrix-matrix multiplications in order to reduce the latency of the design. Note that the Alveo U280 board is composed of three "chiplets" and crossing chiplets consumes scarce routing resources that severely degrades performance and prevents scaling. Therefore, we only consider FPGA designs that use less that 30% of the FPGA resources so that no chiplet is crossed. Table 2 presents the FPGA resources and latencies to implement our proposed neural network depicted in Fig. 6 for various dimensions

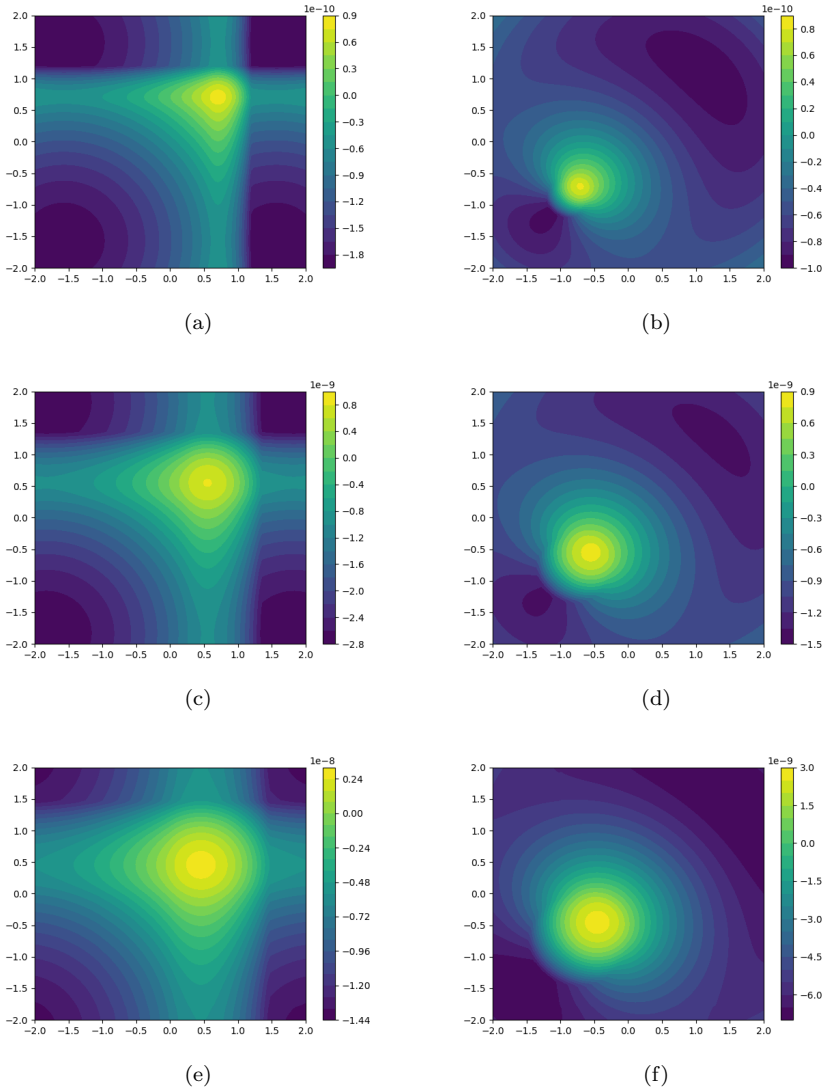


(a) $t = 1$ (b) $t = 0.75$

(c) $t = 0.5$ (d) $t = 0.25$

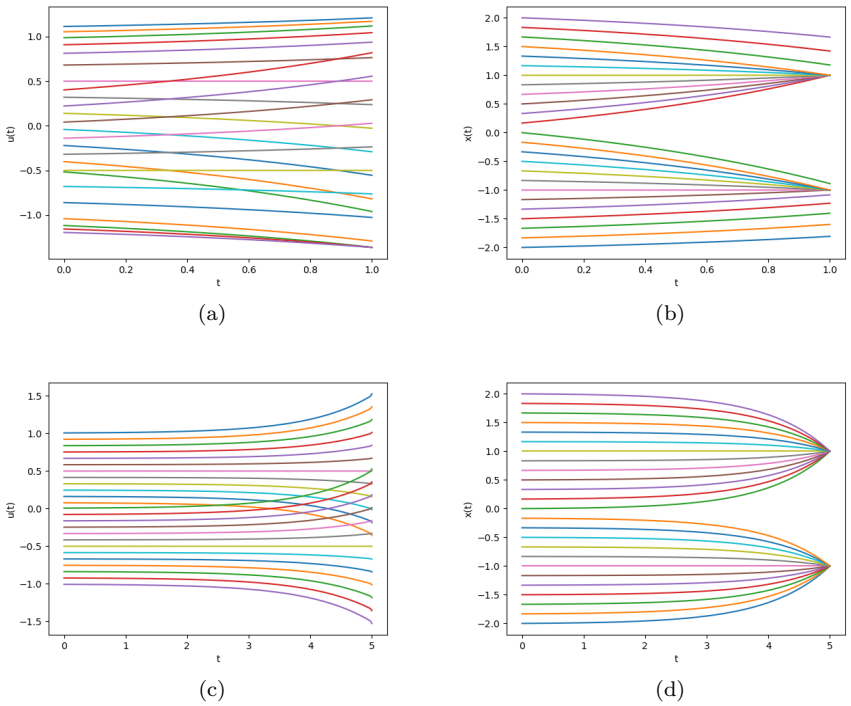
Fig. 16 The viscosity solution V_{NN} to the 16 dimensional HJ PDE with Hamiltonian (38), terminal data (28) (where Ψ_i's are defined in (44)) and terminal time $T = 1$ is computed using the proposed abstract neural network architecture depicted in Fig. 6, whose parameters are trained using the ADMM method. The two dimensional slices of V_{NN} at time $t = 1$ (i.e., the terminal cost), $t = 0.75$, $t = 0.5$ and $t = 0.25$ are shown in the subfigures (a), (b), (c) and (d), respectively. The color in each subfigure shows the solution value $V_{NN}(t, \boldsymbol{x})$, where the spatial variable $\boldsymbol{x}$ is in the form of $(x_1, x_2, \mathbf{0}) \in \mathbb{R}^{16}$ (with $\mathbf{0}$ denoting the zero vector in $\mathbb{R}^{14}$) for some points $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ which are represented by the x and y axes.

n and numbers of layers L. We observe from the table that the latencies for $(n, L) = (16, 8)$, $(32, 4)$, and $(64, 2)$ are 2.1110e-05s, 7.5150e-05s, and 2.8600e-04s, respectively. We also implemented our proposed neural network architecture on CPUs using C++ to highlight the boost of performance we can obtain using FPGAs. We perform 1,000,000 runs on a single Intel core I7-1165G7 and report the average time to produce a result for $(n, L) = (16, 8)$, $(32, 4)$, $(64, 2)$ in Table 3 as well as the speed-up compared to our FPGA implementation. We observe a speed-up from 12 to about 20 depending on the dimension n and the number of layers L. Our FPGA design also allows for larger number of layers than those reported here. We simply iterate the FPGA kernel that we designed here for the neural network with fewer layers. In these cases, the amount of FPGA resources remain the same but the latency is multiplied by the number of iterations of the FPGA kernel.



(a) (b) (c) (d) (e) (f)

Fig. 17 The residual $-\frac{\partial V_i}{\partial t} + H(t, \boldsymbol{x}, \nabla_{\boldsymbol{x}} V_i)$ in the HJ PDE with terminal time $T = 1$, Hamiltonian (38) and terminal condition (28) (where Ψ_i's are defined in (44)) is shown for each $i \in \{1, 2\}$ at different time t, where V_i is the solution to the i-th subproblem. Figures (a), (c), (e) show the residuals for V_1 at time $t = 0.75T$, $t = 0.5T$ and $t = 0.25T$, while figures (b), (d), (f) show the residuals for V_2 at time $t = 0.75T$, $t = 0.5T$ and $t = 0.25T$, respectively. In each subfigure, we show the two dimensional slices of the residual function. The color shows the residual value at $(t, \boldsymbol{x})$, where the spatial variable $\boldsymbol{x}$ is in the form of $(x_1, x_2, \mathbf{0}) \in \mathbb{R}^{16}$ (with $\mathbf{0}$ denoting the zero vector in $\mathbb{R}^{14}$) for some points $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ which are represented by the x and y axes.



(a) (b)

(c) (d)

Fig. 18 The open loop optimal controls and the optimal trajectories in the 16 dimensional optimal control problem with Lagrangian (36), source term (37), terminal cost (28) (where Ψ_i's are defined in (44)) and different terminal time $T = 1, 5$ are computed using the proposed abstract neural network architecture (22) with the implementation depicted in Fig. 7. Several graphs of the optimal controls with $T = 1$ are shown in (a), whose corresponding optimal trajectories are shown in (b). Several graphs of the optimal controls with $T = 5$ are shown in (c), whose corresponding optimal trajectories are shown in (d).

n/L	Latency (ns)	BRAMs	DSPs	FFs	LUTs
16/8	6,345 (2.111E4)	601(14%)	2402(26%)	354,013(13%)	258,710(19%)
32/4	22,547(7.515E4)	602(14%)	2482(27%)	353,225(13%)	248,369(19%)
64/2	85,547 (2.860E5)	608(15%)	2522(27%)	352,715(13%)	242,886(19%)

Table 2 FPGA resources and latencies in cycles and nanoseconds (ns) to implement L layers of the neural network for various dimensions n using simple precision floating point on a Xilinx Alveo U280 board with a frequency of 300 MHz.

n/L	CPU time	FPGA time	speed up
16/8	2.6310e-04s	2.1110e-05s	12.463
32/4	1.2021e-03s	7.5150e-05s	15.996
64/2	5.9730e-03s	2.8600e-04s	20.885

Table 3 Comparison of the average time for 1,000,000 runs for various dimensions and number of layers on a single Intel Core I7-1165G7 and our FPGA implementation on a Xilinx Alveo U280 board running at 300 MHz. The speed-up using FPGA compared to the CPU is presented in the last column.

5 Conclusion

We propose two abstract neural network architectures depicted in Figs. 3 and 4, which respectively solve certain high dimensional HJ PDEs and are used to compute the optimal controls in the corresponding optimal control problems. To implement these abstract architectures, we present two Resnet-type deep neural network implementations and show several numerical results in Section 4. These architectures pave the way to leverage dedicated hardware designed for neural networks to obtain efficient implementations of the numerical algorithms for certain optimal control problems and HJ PDEs. It has potential in real-time computations for these high dimensional problems. Moreover, these architectures are designed based on the theories of linear-quadratic controls and min-plus algebra, and hence there are theoretical guarantees for these neural network architectures. A preliminary implementation of our proposed neural network architecture on FPGAs shows promising speed up compared to CPUs. Beyond the numerical experiments in Section 4, we also tried some examples where the assumption (A2) is not satisfied. In these examples, we observed that our proposed neural network architectures also provide reasonable numerical outputs. These observations suggest that the assumption (A2) is sufficient but not necessary for our proposed architectures.

Acknowledgements This research is supported by AFOSR MURI FA9550-20-1-0358. The authors also thank the Xilinx Center of Excellence at the University of Illinois, Urbana-Champaign UIUC to provide access to Xilinx Alveo boards and computing resources.

References

1. Aggarwal, C.C., et al.: Neural networks and deep learning. Springer **10**, 978–3 (2018) (Cited on page 7.)
2. Akian, M., Bapat, R., Gaubert, S.: Max-plus algebra. Handbook of linear algebra **39** (2006) (Cited on page 2.)
3. Akian, M., Gaubert, S., Lakhoua, A.: The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis. SIAM Journal on Control and Optimization **47**(2), 817–848 (2008) (Cited on page 2.)
4. Albi, G., Bicego, S., Kalise, D.: Gradient-augmented supervised learning of optimal feedback laws using state-dependent riccati equations. arXiv preprint arXiv:2103.04091 (2021) (Cited on page 2.)
5. Alla, A., Falcone, M., Saluzzi, L.: An efficient DP algorithm on a tree-structure for finite horizon optimal control problems. SIAM Journal on Scientific Computing **41**(4), A2384–A2406 (2019) (Cited on page 2.)
6. Alla, A., Falcone, M., Volkwein, S.: Error analysis for POD approximations of infinite horizon problems via the dynamic programming approach. SIAM Journal on Control and Optimization **55**(5), 3091–3115 (2017) (Cited on page 2.)
7. Anastassi, A.A.: Constructing Runge–Kutta methods with the use of artificial neural networks. Neural Computing and Applications **25**(1), 229–236 (2014) (Cited on page 18.)
8. Bachouch, A., Huré, C., Langrené, N., Pham, H.: Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications. arXiv preprint arXiv:1812.05916 (2018) (Cited on page 2.)

9. Balasubramaniam, P., Abdul Samath, J., Kumaresan, N., Vincent Antony Kumar, A.: Solution of matrix Riccati differential equation for the linear quadratic singular system using neural networks. Applied Mathematics and Computation **182**(2), 1832 – 1839 (2006). DOI https://doi.org/10.1016/j.amc.2006.06.020. URL `http://www.sciencedirect.com/science/article/pii/S0096300306005327` (Cited on page 18.)
10. Balasubramaniam, P., Vincent Antony Kumar, A.: Solution of matrix Riccati differential equation for nonlinear singular system using genetic programming. Genetic Programming and Evolvable Machines **10**(1), 71–89 (2009). DOI 10.1007/s10710-008-9072-z (Cited on page 18.)
11. Banerjee, K., Georganas, E., Kalamkar, D., Ziv, B., Segal, E., Anderson, C., Heinecke, A.: Optimizing deep learning RNN topologies on Intel architecture. Supercomputing Frontiers and Innovations **6**(3) (2019) (Cited on page 3.)
12. Bansal, S., Tomlin, C.: Deepreach: A deep learning approach to high-dimensional reachability. arXiv preprint arXiv:2011.02082 (2020) (Cited on page 2.)
13. Bardi, M., Capuzzo-Dolcetta, I.: Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations. Systems & Control: Foundations & Applications. Birkhäuser Boston, Inc., Boston, MA (1997). DOI 10.1007/978-0-8176-4755-1. URL `https://doi.org/10.1007/978-0-8176-4755-1`. With appendices by Maurizio Falcone and Pierpaolo Soravia (Cited on pages 2, 6, and 27.)
14. Bardi, M., Da Lio, F.: On the Bellman equation for some unbounded control problems. NoDEA Nonlinear Differential Equations Appl. **4**(4), 491–510 (1997). DOI 10.1007/s000300050027. URL `https://doi.org/10.1007/s000300050027` (Cited on page 12.)
15. Beck, C., Becker, S., Cheridito, P., Jentzen, A., Neufeld, A.: Deep splitting method for parabolic PDEs. arXiv preprint arXiv:1907.03452 (2019) (Cited on page 2.)
16. Beck, C., Becker, S., Grohs, P., Jaafari, N., Jentzen, A.: Solving stochastic differential equations and Kolmogorov equations by means of deep learning. arXiv preprint arXiv:1806.00421 (2018) (Cited on page 2.)
17. Beck, C., E, W., Jentzen, A.: Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. Journal of Nonlinear Science **29**(4), 1563–1619 (2019) (Cited on page 2.)
18. Bellman, R.E.: Adaptive control processes: a guided tour. Princeton university press (1961) (Cited on page 2.)
19. Berg, J., Nyström, K.: A unified deep artificial neural network approach to partial differential equations in complex geometries. Neurocomputing **317**, 28 – 41 (2018). DOI 10.1016/j.neucom.2018.06.056 (Cited on page 2.)
20. Bertsekas, D.P.: Reinforcement learning and optimal control. Athena Scientific, Belmont, Massachusetts (2019) (Cited on page 2.)
21. Bokanowski, O., Garcke, J., Griebel, M., Klompmaker, I.: An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations. Journal of Scientific Computing **55**(3), 575–605 (2013) (Cited on page 2.)
22. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011). DOI 10.1561/2200000016. URL `https://doi.org/10.1561/2200000016` (Cited on page 16.)
23. Bucy, R.: Global theory of the Riccati equation. Journal of Computer and System Sciences **1**(4), 349 – 361 (1967). DOI https://doi.org/10.1016/S0022-0000(67)80025-4. URL `http://www.sciencedirect.com/science/article/pii/S0022000067800254` (Cited on pages 13 and 15.)
24. Butcher, J.C.: Numerical methods for ordinary differential equations. John Wiley & Sons (2016) (Cited on pages 20 and 21.)
25. Cannarsa, P., Sinestrari, C.: Semiconcave functions, Hamilton-Jacobi equations, and optimal control, *Progress in Nonlinear Differential Equations and their Applications*, vol. 58. Birkhäuser Boston, Inc., Boston, MA (2004) (Cited on page 6.)
26. Chan-Wai-Nam, Q., Mikael, J., Warin, X.: Machine learning for semi linear PDEs. Journal of Scientific Computing **79**(3), 1667–1712 (2019) (Cited on page 2.)

27. Chen, M., Hu, Q., Fisac, J.F., Akametalu, K., Mackin, C., Tomlin, C.J.: Reachability-based safety and goal satisfaction of unmanned aerial platoons on air highways. Journal of Guidance, Control, and Dynamics **40**(6), 1360–1373 (2017). DOI 10.2514/1.G000774. URL `https://doi.org/10.2514/1.G000774` (Cited on page 2.)
28. Chen, T., Chen, H.: Approximations of continuous functionals by neural networks with application to dynamic systems. IEEE Transactions on Neural Networks **4**(6), 910–918 (1993). DOI 10.1109/72.286886 (Cited on page 3.)
29. Chen, T., van Gelder, J., van de Ven, B., Amitonov, S.V., de Wilde, B., Euler, H.C.R., Broersma, H., Bobbert, P.A., Zwanenburg, F.A., van der Wiel, W.G.: Classification with a disordered dopant-atom network in silicon. Nature **577**(7790), 341–345 (2020) (Cited on page 3.)
30. Cheng, T., Lewis, F.L.: Fixed-final time constrained optimal control of nonlinear systems using neural network HJB approach. In: Proceedings of the 45th IEEE Conference on Decision and Control, pp. 3016–3021 (2006). DOI 10.1109/CDC.2006.377523 (Cited on page 2.)
31. Coupechoux, M., Darbon, J., Kélif, J., Sigelle, M.: Optimal trajectories of a UAV base station using Lagrangian mechanics. In: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 626–631 (2019). DOI 10.1109/INFCOMW.2019.8845287 (Cited on page 2.)
32. Darbon, J.: On convex finite-dimensional variational methods in imaging sciences and Hamilton–Jacobi equations. SIAM Journal on Imaging Sciences **8**(4), 2268–2293 (2015). DOI 10.1137/130944163 (Cited on page 2.)
33. Darbon, J., Langlois, G.P., Meng, T.: Overcoming the curse of dimensionality for some Hamilton-Jacobi partial differential equations via neural network architectures. Res. Math. Sci. **7**(3), 20 (2020). DOI 10.1007/s40687-020-00215-6. URL `https://doi.org/10.1007/s40687-020-00215-6` (Cited on pages 2 and 3.)
34. Darbon, J., Meng, T.: On decomposition models in imaging sciences and multi-time Hamilton-Jacobi partial differential equations. arXiv preprint arXiv:1906.09502 (2019) (Cited on page 2.)
35. Darbon, J., Meng, T.: On some neural network architectures that can represent viscosity solutions of certain high dimensional Hamilton–Jacobi partial differential equations. Journal of Computational Physics **425**, 109907 (2021). DOI https://doi.org/10.1016/j.jcp.2020.109907. URL `http://www.sciencedirect.com/science/article/pii/S0021999120306811` (Cited on pages 2 and 3.)
36. Darbon, J., Osher, S.: Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere. Research in the Mathematical Sciences **3**(1), 19 (2016). DOI 10.1186/s40687-016-0068-7 (Cited on page 2.)
37. Davison, E., Maki, M.: The numerical solution of the matrix Riccati differential equation. IEEE Transactions on Automatic Control **18**(1), 71–73 (1973) (Cited on page 18.)
38. Delahaye, D., Puechmorel, S., Tsiotras, P., Feron, E.: Mathematical models for aircraft trajectory design: A survey. In: Air Traffic Management and Systems, pp. 205–247. Springer Japan, Tokyo (2014) (Cited on page 2.)
39. Denk, J., Schmidt, G.: Synthesis of a walking primitive database for a humanoid robot using optimal control techniques. In: Proceedings of IEEE-RAS International Conference on Humanoid Robots, pp. 319–326 (2001) (Cited on page 2.)
40. Deshpande, A.S.: Max-plus representation for the fundamental solution of the time-varying differential Riccati equation. Automatica **47**(8), 1667 – 1676 (2011). DOI https://doi.org/10.1016/j.automatica.2011.05.009. URL `http://www.sciencedirect.com/science/article/pii/S0005109811002822` (Cited on page 18.)
41. Dissanayake, M.W.M.G., Phan-Thien, N.: Neural-network-based approximations for solving partial differential equations. Communications in Numerical Methods in Engineering **10**(3), 195–201 (1994). DOI 10.1002/cnm.1640100303 (Cited on page 2.)
42. Djeridane, B., Lygeros, J.: Neural approximation of PDE solutions: An application to reachability computations. In: Proceedings of the 45th IEEE Conference on Decision and Control, pp. 3034–3039 (2006). DOI 10.1109/CDC.2006.377184 (Cited on page 2.)
43. Dockhorn, T.: A discussion on solving partial differential equations using neural networks. arXiv preprint arXiv:1904.07200 (2019) (Cited on page 2.)

44. Dolgov, S., Kalise, D., Kunisch, K.: A tensor decomposition approach for high-dimensional Hamilton-Jacobi-Bellman equations. arXiv preprint arXiv:1908.01533 (2019) (Cited on page 2.)
45. Dower, P., McEneaney, W., Cantoni, M.: A dynamic game approximation for a linear regulator problem with a log-barrier state constraint. In: Proc. 22nd International Symposium on Mathematical Theory of Networks and Systems, pp. 297–304 (2016) (Cited on page 9.)
46. Dower, P.M., McEneaney, W.M.: A max-plus dual space fundamental solution for a class of operator differential Riccati equations. SIAM Journal on Control and Optimization **53**(2), 969–1002 (2015). DOI 10.1137/120879312. URL `https://doi.org/10.1137/120879312` (Cited on page 18.)
47. Dower, P.M., McEneaney, W.M., Cantoni, M.: A game representation for state constrained linear regulator problems. In: 2016 IEEE 55th Conference on Decision and Control (CDC), pp. 1074–1079 (2016). DOI 10.1109/CDC.2016.7798410 (Cited on page 9.)
48. Dower, P.M., McEneaney, W.M., Zhang, H.: Max-plus fundamental solution semigroups for optimal control problems. In: 2015 Proceedings of the Conference on Control and its Applications, pp. 368–375. SIAM (2015) (Cited on pages 2 and 9.)
49. Dower, P.M., Zhang, H.: A new fundamental solution for differential Riccati equations arising in L2-gain analysis. In: 2015 5th Australian Control Conference (AUCC), pp. 65–68 (2015) (Cited on page 18.)
50. E, W.: A proposal on machine learning via dynamical systems. Commun. Math. Stat. **5**(1), 1–11 (2017). DOI 10.1007/s40304-017-0103-z. URL `https://doi.org/10.1007/s40304-017-0103-z` (Cited on page 3.)
51. E, W., Han, J., Jentzen, A.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Communications in Mathematics and Statistics **5**(4), 349–380 (2017). DOI 10.1007/s40304-017-0117-6 (Cited on page 2.)
52. E, W., Han, J., Li, Q.: A mean-field optimal control formulation of deep learning. Res. Math. Sci. **6**(1), Paper No. 10, 41 (2019). DOI 10.1007/s40687-018-0172-y. URL `https://doi.org/10.1007/s40687-018-0172-y` (Cited on page 3.)
53. E, W., Yu, B.: The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics **6**(1), 1–12 (2018) (Cited on page 2.)
54. El Khoury, A., Lamiraux, F., Taïx, M.: Optimal motion planning for humanoid robots. In: 2013 IEEE International Conference on Robotics and Automation, pp. 3136–3141 (2013). DOI 10.1109/ICRA.2013.6631013 (Cited on page 2.)
55. Fallon, M., Kuindersma, S., Karumanchi, S., Antone, M., Schneider, T., Dai, H., D'Arpino, C.P., Deits, R., DiCicco, M., Fourie, D., et al.: An architecture for online affordance-based perception and whole-body planning. Journal of Field Robotics **32**(2), 229–254 (2015) (Cited on page 2.)
56. Farabet, C., LeCun, Y., Kavukcuoglu, K., Culurciello, E., Martini, B., Akselrod, P., Talay, S.: Large-scale FPGA-based convolutional networks. In: R. Bekkerman, M. Bilenko, J. Langford (eds.) Scaling up Machine Learning: Parallel and Distributed Approaches. Cambridge University Press (2011) (Cited on page 3.)
57. Farabet, C., poulet, C., Han, J., LeCun, Y.: CNP: An FPGA-based processor for convolutional networks. In: International Conference on Field Programmable Logic and Applications. IEEE, Prague (2009) (Cited on page 3.)
58. Farabet, C., Poulet, C., LeCun, Y.: An FPGA-based stream processor for embedded real-time vision with convolutional networks. In: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, pp. 878–885. IEEE Computer Society, Los Alamitos, CA, USA (2009). DOI 10.1109/ICCVW.2009.5457611. URL `https://doi.ieeecomputersociety.org/10.1109/ICCVW.2009.5457611` (Cited on page 3.)
59. Farimani, A.B., Gomes, J., Pande, V.S.: Deep Learning the Physics of Transport Phenomena. arXiv e-prints (2017) (Cited on page 2.)
60. Feng, S., Whitman, E., Xinjilefu, X., Atkeson, C.G.: Optimization based full body control for the atlas robot. In: 2014 IEEE-RAS International Conference on Humanoid

Robots, pp. 120–127 (2014). DOI 10.1109/HUMANOIDS.2014.7041347 (Cited on page 2.)
61. Feng Lin, Brandt, R.D.: An optimal control approach to robust control of robot manipulators. IEEE Transactions on Robotics and Automation **14**(1), 69–77 (1998). DOI 10.1109/70.660845 (Cited on page 2.)
62. Fleming, W., McEneaney, W.: A max-plus-based algorithm for a Hamilton–Jacobi–Bellman equation of nonlinear filtering. SIAM Journal on Control and Optimization **38**(3), 683–710 (2000). DOI 10.1137/S0363012998332433 (Cited on page 2.)
63. Fujii, M., Takahashi, A., Takahashi, M.: Asymptotic expansion as prior knowledge in deep learning method for high dimensional BSDEs. Asia-Pacific Financial Markets **26**(3), 391–408 (2019). DOI 10.1007/s10690-019-09271-7 (Cited on page 2.)
64. Fujiwara, K., Kajita, S., Harada, K., Kaneko, K., Morisawa, M., Kanehiro, F., Nakaoka, S., Hirukawa, H.: An optimal planning of falling motions of a humanoid robot. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 456–462 (2007). DOI 10.1109/IROS.2007.4399327 (Cited on page 2.)
65. Garcke, J., Kröner, A.: Suboptimal feedback control of PDEs by solving HJB equations on adaptive sparse grids. Journal of Scientific Computing **70**(1), 1–28 (2017) (Cited on page 2.)
66. Gaubert, S., McEneaney, W., Qu, Z.: Curse of dimensionality reduction in max-plus based approximation methods: Theoretical estimates and improved pruning algorithms. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference, pp. 1054–1061. IEEE (2011) (Cited on page 2.)
67. Glowinski, R.: On Alternating Direction Methods of Multipliers: A Historical Perspective, pp. 59–82. Springer Netherlands, Dordrecht (2014). DOI 10.1007/978-94-017-9054-3_4. URL `https://doi.org/10.1007/978-94-017-9054-3_4` (Cited on page 16.)
68. Grohs, P., Jentzen, A., Salimova, D.: Deep neural network approximations for Monte Carlo algorithms. arXiv preprint arXiv:1908.10828 (2019) (Cited on page 2.)
69. Grüne, L., Schaller, M., Schiela, A.: Exponential sensitivity and turnpike analysis for linear quadratic optimal control of general evolution equations. Journal of Differential Equations **268**(12), 7311–7341 (2020). DOI https://doi.org/10.1016/j.jde.2019.11.064. URL `https://www.sciencedirect.com/science/article/pii/S0022039619305984` (Cited on page 26.)
70. Han, J., Jentzen, A., E, W.: Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences **115**(34), 8505–8510 (2018). DOI 10.1073/pnas.1718942115 (Cited on page 2.)
71. Han, J., Zhang, L., E, W.: Solving many-electron Schrödinger equation using deep neural networks. Journal of Computational Physics p. 108929 (2019) (Cited on page 2.)
72. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) (Cited on pages 7 and 18.)
73. Hirjibehedin, C.: Evolution of circuits for machine learning. Nature **577**, 320–321 (2020). DOI 10.1038/d41586-020-00002-x (Cited on page 3.)
74. Hofer, M., Muehlebach, M., D'Andrea, R.: Application of an approximate model predictive control scheme on an unmanned aerial vehicle. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 2952–2957 (2016). DOI 10.1109/ICRA.2016.7487459 (Cited on page 2.)
75. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Networks **4**(2), 251–257 (1991). DOI https://doi.org/10.1016/0893-6080(91)90009-T. URL `https://www.sciencedirect.com/science/article/pii/089360809190009T` (Cited on page 3.)
76. Horowitz, M.B., Damle, A., Burdick, J.W.: Linear Hamilton Jacobi Bellman equations in high dimensions. In: 53rd IEEE Conference on Decision and Control, pp. 5880–5887. IEEE (2014) (Cited on page 2.)
77. Hsieh, J.T., Zhao, S., Eismann, S., Mirabella, L., Ermon, S.: Learning neural PDE solvers with convergence guarantees. In: International Conference on Learning Representations (2019) (Cited on page 2.)

78. Hu, C., Shu, C.: A discontinuous Galerkin finite element method for Hamilton–Jacobi equations. SIAM Journal on Scientific Computing **21**(2), 666–690 (1999). DOI 10.1137/S1064827598337282 (Cited on page 2.)
79. Huré, C., Pham, H., Bachouch, A., Langrené, N.: Deep neural networks algorithms for stochastic control problems on finite horizon, part I: convergence analysis. arXiv preprint arXiv:1812.04300 (2018) (Cited on page 2.)
80. Huré, C., Pham, H., Warin, X.: Some machine learning schemes for high-dimensional nonlinear PDEs. arXiv preprint arXiv:1902.01599 (2019) (Cited on page 2.)
81. Jiang, F., Chou, G., Chen, M., Tomlin, C.J.: Using neural networks to compute approximate and guaranteed feasible Hamilton-Jacobi-Bellman PDE solutions. arXiv preprint arXiv:1611.03158 (2016) (Cited on page 2.)
82. Jiang, G., Peng, D.: Weighted ENO schemes for Hamilton–Jacobi equations. SIAM Journal on Scientific Computing **21**(6), 2126–2143 (2000). DOI 10.1137/S106482759732455X (Cited on page 2.)
83. Jianyu, L., Siwei, L., Yingjian, Q., Yaping, H.: Numerical solution of elliptic partial differential equation using radial basis function neural networks. Neural Networks **16**(5-6), 729–734 (2003) (Cited on page 2.)
84. Jin, L., Li, S., Yu, J., He, J.: Robot manipulator control using neural networks: A survey. Neurocomputing **285**, 23 – 34 (2018). DOI https://doi.org/10.1016/j.neucom.2018.01.002. URL `http://www.sciencedirect.com/science/article/pii/S0925231218300158` (Cited on page 2.)
85. Jin, P., Zhang, Z., Kevrekidis, I.G., Karniadakis, G.E.: Learning Poisson systems and trajectories of autonomous systems via Poisson neural networks. arXiv preprint arXiv:2012.03133 (2020) (Cited on page 2.)
86. Jin, P., Zhang, Z., Zhu, A., Tang, Y., Karniadakis, G.E.: SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. Neural Networks **132**, 166–179 (2020). DOI https://doi.org/10.1016/j.neunet.2020.08.017. URL `https://www.sciencedirect.com/science/article/pii/S0893608020303063` (Cited on page 2.)
87. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al.: In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA '17, p. 1–12. Association for Computing Machinery, New York, NY, USA (2017). DOI 10.1145/3079856.3080246. URL `https://doi.org/10.1145/3079856.3080246` (Cited on page 3.)
88. Kalise, D., Kundu, S., Kunisch, K.: Robust feedback control of nonlinear PDEs by numerical approximation of high-dimensional Hamilton-Jacobi-Isaacs equations. arXiv preprint arXiv:1905.06276 (2019) (Cited on page 2.)
89. Kalise, D., Kunisch, K.: Polynomial approximation of high-dimensional Hamilton–Jacobi–Bellman equations and applications to feedback control of semilinear parabolic PDEs. SIAM Journal on Scientific Computing **40**(2), A629–A652 (2018) (Cited on page 2.)
90. Kamali, M.: A Study on Solution of Matrix Riccati Differential Equations Using Ant Colony Programming and Simulink. Institut Sains Matematik, Fakulti Sains, Universiti Malaya (2015). URL `https://books.google.com/books?id=0D8_nQAACAAJ` (Cited on page 18.)
91. Kang, W., Gong, Q.: Neural network approximations of compositional functions with applications to dynamical systems. arXiv preprint arXiv:2012.01698 (2020) (Cited on page 2.)
92. Kang, W., Wilcox, L.C.: Mitigating the curse of dimensionality: sparse grid characteristics method for optimal feedback control and HJB equations. Computational Optimization and Applications **68**(2), 289–315 (2017) (Cited on page 2.)
93. Kastner, R., Matai, J., Neuendorffer, S.: Parallel Programming for FPGAs. ArXiv e-prints (2018) (Cited on page 34.)
94. Kenney, C., Leipnik, R.: Numerical integration of the differential matrix Riccati equation. IEEE Transactions on Automatic Control **30**(10), 962–970 (1985) (Cited on page 18.)
95. Khoo, Y., Lu, J., Ying, L.: Solving parametric PDE problems with artificial neural networks. arXiv preprint arXiv:1707.03351 (2017) (Cited on page 2.)

96. Khoo, Y., Lu, J., Ying, L.: Solving for high-dimensional committor functions using artificial neural networks. Research in the Mathematical Sciences **6**(1), 1 (2019) (Cited on page 2.)
97. Kidger, P., Lyons, T.: Universal Approximation with Deep Narrow Networks. In: J. Abernethy, S. Agarwal (eds.) Proceedings of Thirty Third Conference on Learning Theory, *Proceedings of Machine Learning Research*, vol. 125, pp. 2306–2327. PMLR (2020). URL `http://proceedings.mlr.press/v125/kidger20a.html` (Cited on page 3.)
98. Kim, Y.H., Lewis, F.L., Dawson, D.M.: Intelligent optimal control of robotic manipulators using neural networks. Automatica **36**(9), 1355 – 1364 (2000). DOI https://doi.org/10.1016/S0005-1098(00)00045-5. URL `http://www.sciencedirect.com/science/article/pii/S0005109800000455` (Cited on page 2.)
99. Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., Tedrake, R.: Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. Autonomous robots **40**(3), 429–455 (2016) (Cited on page 2.)
100. Kundu, A., Srinivasan, S., Qin, E.C., Kalamkar, D., Mellempudi, N.K., Das, D., Banerjee, K., Kaul, B., Dubey, P.: K-tanh: Hardware efficient activations for deep learning. arXiv **1909.07729** (2019) (Cited on page 3.)
101. Kunisch, K., Volkwein, S., Xie, L.: HJB-POD-based feedback design for the optimal control of evolution problems. SIAM Journal on Applied Dynamical Systems **3**(4), 701–722 (2004) (Cited on page 2.)
102. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks **9**(5), 987–1000 (1998). DOI 10.1109/72.712178 (Cited on page 2.)
103. Lagaris, I.E., Likas, A.C., Papageorgiou, D.G.: Neural-network methods for boundary value problems with irregular boundaries. IEEE Transactions on Neural Networks **11**(5), 1041–1049 (2000). DOI 10.1109/72.870037 (Cited on page 2.)
104. Lambrianides, P., Gong, Q., Venturi, D.: A new scalable algorithm for computational optimal control under uncertainty. arXiv preprint arXiv:1909.07960 (2019) (Cited on page 2.)
105. LeCun, Y.: 1.1 deep learning hardware: Past, present, and future. In: 2019 IEEE International Solid- State Circuits Conference - (ISSCC), pp. 12–19 (2019). DOI 10.1109/ISSCC.2019.8662396 (Cited on page 3.)
106. Lee, D., Tomlin, C.J.: A Hopf-Lax formula in Hamilton–Jacobi analysis of reach-avoid problems. IEEE Control Systems Letters **5**(3), 1055–1060 (2021). DOI 10.1109/LCSYS.2020.3009933 (Cited on page 2.)
107. Lee, H., Kang, I.S.: Neural algorithm for solving differential equations. Journal of Computational Physics **91**(1), 110–131 (1990) (Cited on page 2.)
108. Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Networks **6**(6), 861–867 (1993). DOI https://doi.org/10.1016/S0893-6080(05)80131-5. URL `https://www.sciencedirect.com/science/article/pii/S0893608005801315` (Cited on page 3.)
109. Levin, J.J.: On the matrix Riccati equation. Proc. Amer. Math. Soc. **10**, 519–524 (1959). DOI 10.2307/2033645. URL `https://doi.org/10.2307/2033645` (Cited on page 18.)
110. Lewis, F., Dawson, D., Abdallah, C.: Robot Manipulator Control: Theory and Practice. Control engineering. Marcel Dekker (2004). URL `https://books.google.com/books?id=BDS_PQAACAAJ` (Cited on page 2.)
111. Li, A., Bansal, S., Giovanis, G., Tolani, V., Tomlin, C., Chen, M.: Generating robust supervision for learning-based visual navigation using Hamilton-Jacobi reachability. In: A.M. Bayen, A. Jadbabaie, G. Pappas, P.A. Parrilo, B. Recht, C. Tomlin, M. Zeilinger (eds.) Proceedings of the 2nd Conference on Learning for Dynamics and Control, *Proceedings of Machine Learning Research*, vol. 120, pp. 500–510. PMLR, The Cloud (2020). URL `http://proceedings.mlr.press/v120/li20a.html` (Cited on page 2.)
112. Long, Z., Lu, Y., Dong, B.: PDE-net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. Journal of Computational Physics **399**, 108925 (2019). DOI 10.1016/j.jcp.2019.108925 (Cited on page 2.)

113. Long, Z., Lu, Y., Ma, X., Dong, B.: PDE-net: Learning PDEs from data. arXiv preprint arXiv:1710.09668 (2017) (Cited on page 2.)
114. Lu, L., Jin, P., Karniadakis, G.E.: Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. arXiv preprint arXiv:1910.03193 (2019) (Cited on page 3.)
115. Lye, K.O., Mishra, S., Ray, D.: Deep learning observables in computational fluid dynamics. arXiv preprint arXiv:1903.03040 (2019) (Cited on page 2.)
116. McEneaney, W.: Max-plus methods for nonlinear control and estimation. Springer Science & Business Media (2006) (Cited on pages 2, 6, and 9.)
117. McEneaney, W.: A curse-of-dimensionality-free numerical method for solution of certain HJB PDEs. SIAM Journal on Control and Optimization **46**(4), 1239–1276 (2007). DOI 10.1137/040610830 (Cited on page 2.)
118. McEneaney, W.M.: A new fundamental solution for differential Riccati equations arising in control. Automatica **44**(4), 920 – 936 (2008). DOI https://doi.org/10.1016/j.automatica.2007.08.019. URL `http://www.sciencedirect.com/science/article/pii/S0005109807003895` (Cited on page 18.)
119. McEneaney, W.M., Deshpande, A., Gaubert, S.: Curse-of-complexity attenuation in the curse-of-dimensionality-free method for HJB PDEs. In: 2008 American Control Conference, pp. 4684–4690. IEEE (2008) (Cited on page 2.)
120. McEneaney, W.M., Kluberg, L.J.: Convergence rate for a curse-of-dimensionality-free method for a class of HJB PDEs. SIAM Journal on Control and Optimization **48**(5), 3052–3079 (2009) (Cited on page 2.)
121. McFall, K.S., Mahan, J.R.: Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. IEEE Transactions on Neural Networks **20**(8), 1221–1233 (2009). DOI 10.1109/TNN.2009.2020735 (Cited on page 2.)
122. Meade, A., Fernandez, A.: The numerical solution of linear ordinary differential equations by feedforward neural networks. Mathematical and Computer Modelling **19**(12), 1 – 25 (1994). DOI 10.1016/0895-7177(94)90095-7 (Cited on page 2.)
123. Meng, X., Karniadakis, G.E.: A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. arXiv preprint arXiv:1903.00104 (2019) (Cited on page 2.)
124. Meng, X., Li, Z., Zhang, D., Karniadakis, G.E.: PPINN: Parareal physics-informed neural network for time-dependent PDEs. arXiv preprint arXiv:1909.10145 (2019) (Cited on page 2.)
125. van Milligen, B.P., Tribaldos, V., Jiménez, J.A.: Neural network differential equation and plasma equilibrium solver. Phys. Rev. Lett. **75**, 3594–3597 (1995). DOI 10.1103/PhysRevLett.75.3594 (Cited on page 2.)
126. Nakamura-Zimmerer, T., Gong, Q., Kang, W.: Adaptive deep learning for high-dimensional Hamilton-Jacobi-Bellman equations. arXiv preprint arXiv:1907.05317 (2019) (Cited on page 2.)
127. Nakamura-Zimmerer, T., Gong, Q., Kang, W.: QRnet: Optimal regulator design with LQR-augmented neural networks. IEEE Control Systems Letters **5**(4), 1303–1308 (2021). DOI 10.1109/LCSYS.2020.3034415 (Cited on page 2.)
128. Niarchos, K.N., Lygeros, J.: A neural approximation to continuous time reachability computations. In: Proceedings of the 45th IEEE Conference on Decision and Control, pp. 6313–6318 (2006). DOI 10.1109/CDC.2006.377358 (Cited on page 2.)
129. Osher, S., Shu, C.: High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations. SIAM Journal on Numerical Analysis **28**(4), 907–922 (1991). DOI 10.1137/0728049 (Cited on page 2.)
130. Pang, G., Lu, L., Karniadakis, G.E.: fPINNs: Fractional physics-informed neural networks. SIAM Journal on Scientific Computing **41**(4), A2603–A2626 (2019) (Cited on page 2.)
131. Parzani, C., Puechmorel, S.: On a Hamilton-Jacobi-Bellman approach for coordinated optimal aircraft trajectories planning. In: CCC 2017 36th Chinese Control Conference, Control Conference (CCC), 2017 36th Chinese, pp. ISBN: 978–1–5386–2918–5. IEEE, Dalian, China (2017). DOI 10.23919/ChiCC.2017.8027369. URL `https://hal-enac.archives-ouvertes.fr/hal-01340565` (Cited on page 2.)

132. Pham, H., Pham, H., Warin, X.: Neural networks-based backward scheme for fully nonlinear PDEs. arXiv preprint arXiv:1908.00412 (2019) (Cited on page 2.)
133. Raissi, M.: Deep hidden physics models: Deep learning of nonlinear partial differential equations. The Journal of Machine Learning Research **19**(1), 932–955 (2018) (Cited on page 2.)
134. Raissi, M.: Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. arXiv preprint arXiv:1804.07010 (2018) (Cited on page 2.)
135. Raissi, M., Perdikaris, P., Karniadakis, G.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics **378**, 686 – 707 (2019). DOI 10.1016/j.jcp.2018.10.045 (Cited on page 2.)
136. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:1711.10561 (2017) (Cited on page 2.)
137. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. arXiv preprint arXiv:1711.10566 (2017) (Cited on page 2.)
138. Reisinger, C., Zhang, Y.: Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems. arXiv preprint arXiv:1903.06652 (2019) (Cited on page 2.)
139. Rossi, F., Conan-Guez, B.: Functional multi-layer perceptron: a non-linear tool for functional data analysis. Neural Networks **18**(1), 45–60 (2005). DOI https://doi.org/10.1016/j.neunet.2004.07.001. URL `https://www.sciencedirect.com/science/article/pii/S0893608004001418` (Cited on page 3.)
140. Royo, V.R., Tomlin, C.: Recursive regression with neural networks: Approximating the HJI PDE solution. arXiv preprint arXiv:1611.02739 (2016) (Cited on page 2.)
141. Rucco, A., Sujit, P.B., Aguiar, A.P., de Sousa, J.B., Pereira, F.L.: Optimal rendezvous trajectory for unmanned aerial-ground vehicles. IEEE Transactions on Aerospace and Electronic Systems **54**(2), 834–847 (2018). DOI 10.1109/TAES.2017.2767958 (Cited on page 2.)
142. Rudd, K., Muro, G.D., Ferrari, S.: A constrained backpropagation approach for the adaptive solution of partial differential equations. IEEE Transactions on Neural Networks and Learning Systems **25**(3), 571–584 (2014). DOI 10.1109/TNNLS.2013.2277601 (Cited on page 2.)
143. Samath, J.A., Selvaraju, N.: Solution of matrix Riccati differential equation for nonlinear singular system using neural networks. International Journal of Computer Applications **1**(29), 49–55 (2010). DOI 10.5120/575-181 (Cited on page 18.)
144. Sirignano, J., Spiliopoulos, K.: DGM: A deep learning algorithm for solving partial differential equations. Journal of Computational Physics **375**, 1339 – 1364 (2018). DOI 10.1016/j.jcp.2018.08.029 (Cited on page 2.)
145. Subbotin, A.I.: Minimax solutions of first-order partial differential equations. Russian Mathematical Surveys **51**(2), 283–313 (1996). DOI 10.1070/RM1996v051n02ABEH002773 (Cited on page 6.)
146. Tang, W., Shan, T., Dang, X., Li, M., Yang, F., Xu, S., Wu, J.: Study on a Poisson's equation solver based on deep learning technique. In: 2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS), pp. 1–3 (2017). DOI 10.1109/EDAPS.2017.8277017 (Cited on page 2.)
147. Tassa, Y., Erez, T.: Least squares solutions of the HJB equation with neural network value-function approximators. IEEE Transactions on Neural Networks **18**(4), 1031–1041 (2007). DOI 10.1109/TNN.2007.899249 (Cited on page 2.)
148. Todorov, E.: Efficient computation of optimal actions. Proceedings of the national academy of sciences **106**(28), 11478–11483 (2009) (Cited on page 2.)
149. Uchiyama, T., Sonehara, N.: Solving inverse problems in nonlinear PDEs by recurrent neural networks. In: IEEE International Conference on Neural Networks, pp. 99–102. IEEE (1993) (Cited on page 2.)
150. Wang, Yuanchang, Yong, Jiongmin: A deterministic affine-quadratic optimal control problem. ESAIM: COCV **20**(3), 633–661 (2014). DOI 10.1051/cocv/2013078. URL `https://doi.org/10.1051/cocv/2013078` (Cited on pages 9, 10, and 12.)

151. Yadav, N., Yadav, A., Kumar, M.: An introduction to neural network methods for differential equations. SpringerBriefs in Applied Sciences and Technology. Springer, Dordrecht (2015). DOI 10.1007/978-94-017-9816-7 (Cited on page 2.)
152. Yang, L., Zhang, D., Karniadakis, G.E.: Physics-informed generative adversarial networks for stochastic differential equations. arXiv preprint arXiv:1811.02033 (2018) (Cited on page 2.)
153. Yang, Y., Perdikaris, P.: Adversarial uncertainty quantification in physics-informed neural networks. Journal of Computational Physics **394**, 136–152 (2019) (Cited on page 2.)
154. Yegorov, I., Dower, P.M.: Perspectives on characteristics based curse-of-dimensionality-free numerical approaches for solving Hamilton–Jacobi equations. Applied Mathematics & Optimization pp. 1–49 (2017) (Cited on page 2.)
155. Yong, J., Zhou, X.Y.: Stochastic controls, *Applications of Mathematics (New York)*, vol. 43. Springer-Verlag, New York (1999). DOI 10.1007/978-1-4612-1466-3. URL `https://doi.org/10.1007/978-1-4612-1466-3`. Hamiltonian systems and HJB equations (Cited on pages 9, 13, and 15.)
156. Zaslavski, A.J.: Turnpike Theory of Continuous-Time Linear Optimal Control Problems. Springer International Publishing (2015). DOI 10.1007/978-3-319-19141-6. URL `https://doi.org/10.1007/978-3-319-19141-6` (Cited on page 26.)
157. Zhang, D., Guo, L., Karniadakis, G.E.: Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks. arXiv preprint arXiv:1905.01205 (2019) (Cited on page 2.)
158. Zhang, D., Lu, L., Guo, L., Karniadakis, G.E.: Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. Journal of Computational Physics **397**, 108850 (2019) (Cited on page 2.)
159. Zhou, X.Y.: Maximum principle, dynamic programming, and their connection in deterministic control. J. Optim. Theory Appl. **65**(2), 363–373 (1990). DOI 10.1007/BF01102352. URL `https://doi.org/10.1007/BF01102352` (Cited on page 6.)