NEURAL IMPLICIT SOLUTION FORMULA FOR EFFICIENTLY SOLVING HAMILTON-JACOBI EQUATIONS*

YESOM PARK† AND STANLEY OSHER‡

Abstract. This paper presents an implicit solution formula for the Hamilton-Jacobi partial differential equation (HJ PDE). The formula is derived using the method of characteristics and is shown to coincide with the Hopf and Lax formulas in the case where either the Hamiltonian or the initial function is convex. It provides a simple and efficient numerical approach for computing the viscosity solution of HJ PDEs, bypassing the need for the Legendre transform of the Hamiltonian or the initial condition, and the explicit computation of individual characteristic trajectories. A deep learning-based methodology is proposed to learn this implicit solution formula, leveraging the mesh-free nature of deep learning to ensure scalability for high-dimensional problems. Building upon this framework, an algorithm is developed that approximates the characteristic curves piecewise linearly for state-dependent Hamiltonians. Extensive experimental results demonstrate that the proposed method delivers highly accurate solutions, even for nonconvex Hamiltonians, and exhibits remarkable scalability, achieving computational efficiency for problems up to 40 dimensions.

Key words. Hamilton-Jacobi equations, implicit solution formula, deep learning

MSC codes. 65M25, 68T07, 35C99

1. Introduction.

Hamilton-Jacobi partial differential equations (HJ PDEs) are of paramount importance in various fields of mathematics, physics, and engineering, including optimal control [27, 74, 4], mechanics [23, 21], and the study of dynamic systems [44, 73]. As they provide a powerful framework for modeling systems governed by physical laws, HJ PDEs have a wide range of applications in diverse areas such as geometric optics [63, 59], computer vision [8, 33, 65], robotics [53, 51, 3], trajectory optimization [22, 68], traffic flow modeling [37, 49], and financial strategies [32, 7]. These applications illustrate the versatility and significance of HJ PDEs, emphasizing the necessity for effective methods to solve them in both theoretical and practical contexts. It is well-known that the solutions to HJ PDEs are typically continuous but exhibit discontinuous derivatives, irrespective of the smoothness of the initial conditions or the Hamiltonian. Moreover, such solutions are typically non-unique. In this regard, viscosity solutions [14] are commonly considered as the appropriate notion of solution, as they reflect the physical characteristics inherent to the problem.

Numerical methods for solving HJ PDEs have been extensively developed, with numerous practical applications across various fields. The most prominent methods include essentially non-oscillatory (ENO) and weighted ENO (WENO) type schemes [67, 38, 6, 71], semi-Lagrangian methods [28, 15, 29], and level set approaches [66, 63, 64, 60, 2]. However, they encounter significant scalability challenges as the dimensionality of the state space increases. These methods rely on discretization of the state space with a grid and approximating the Hamiltonian in a discrete form. Consequently, the number of grid points required to obtain accurate solutions grows exponentially with the dimensionality of the problem, resulting in prohibitive computational costs. In high-dimensional settings, particularly those involving more than

*Submitted to the editors DATE

Funding: S. Osher was partially supported by MURI DOD-AFOSR-N00014-20-1-2787, NSF NSF-2208272, and NSF STROBE NSF-1554564. Y. Park was supported by the NRF Grant RS-2024-00343226.

†Department of Mathematics, University of California, Los Angeles (yeisom@math.ucla.edu).

‡Department of Mathematics, University of California, Los Angeles (sjo@math.ucla.edu).

four dimensions, this scaling issue renders the classical methods impractical for many real-world applications, where high-dimensional state spaces are prevalent.

Several approaches have been proposed to address the curse of dimensionality in solving HJ PDEs. Methods based on max-plus algebra [58, 1, 31] show promise but are restricted to specific classes of optimal control problems and encounter significant challenges in practical implementation due to their complexity. Another promising approach involves the use of Hopf or Lax formulas to represent solutions to HJ PDEs [20, 12, 13, 10]. These formulas offer a causality-free approach, where solutions at each spatial and temporal point can be computed by solving an optimization problem, thus enabling parallel computation. This approach eliminates the reliance on grid-based discretization, making it particularly well-suited for high-dimensional problems. However, these methods require computing the Legendre transform of the Hamiltonian or initial function and are generally applicable only under specific assumptions, such as convexity, or when the problem can be framed as a particular type of control problem. In parallel, algorithms based on optimal control theory have been developed. For problems with convex Hamiltonians, HJ equations are closely connected to optimal control formulations, where Pontryagin's maximum principle (PMP) provides necessary conditions for optimality. Several PMP-based methods have been proposed [41, 42, 82], which are essentially equivalent to employing the method of characteristics for solving the associated Hamiltonian system. Despite their theoretical appeal, the practical effectiveness of these methods is often limited by the need to solve a system of ordinary differential equations (ODEs) at each point. Additionally, some of these methods assume that multiple characteristics do not intersect, a condition that may not hold in general scenarios. Furthermore, alternative techniques, such as tensor decomposition [24] and polynomial approximation [40, 39], have been studied for specific control problems.

Recent advancements in deep learning have given rise to a growing interest in leveraging the extensive representational capabilities of neural networks to solve PDEs [75, 83, 72, 57, 52, 81]. The viscosity solution of HJ PDEs is challenging to obtain directly from the PDE itself, which underscores the development of alternative approaches beyond the established methods like physics-informed neural networks (PINNs) [72]. In response, data-driven methods have been proposed for solving HJ PDEs [61, 25, 16]; however, these methods face several challenges, including the need for large amounts of training data, the limitation that their performance cannot exceed the accuracy of the numerical methods used to generate the data, and concerns regarding their ability to generalize to unseen scenarios. Moreover, the integration of reinforcement learning techniques to solve HJ PDEs related to control problems has been studied [86, 54]. Another line of research has focused on the development of specialized neural network architectures that express representation formulas to specific HJ PDEs. Notable examples include architectures leveraging min-max algebra to model the value function in optimal control [17], as well as designs based on Hopf-type formulas [18, 19]. Although these approaches effectively bridge mathematical solution representations and network architecture, their applicability is limited to specific classes of problems. Additionally, for problems formulated as stochastic optimal control, deep learning methods leveraging the backward stochastic differential equation (BSDE) representations have been developed [36, 62, 70], providing a probabilistic framework for approximating solutions in high-dimensional settings. One of the most closely related prior works introduces a deep learning approach for learning implicit solution formulas along with characteristics for scalar conservation laws associated with one-dimensional HJ PDEs [84]. However, this method does not ensure

the attainment of an entropy solution.

This study presents a novel implicit solution formula for HJ PDEs. The proposed implicit solution formula is derived through the characteristics of the HJ PDE, with the costate identified as the gradient of the solution at the current spatio-temporal point, leading to an implicit representation formula for the solution. We demonstrate that this new formula coincides with the classical Hopf and Lax formulas, which provides the viscosity solution for HJ PDEs in the case where either the Hamiltonian or the initial function is convex (or concave). Notably, the implicit solution formula is simpler than both the Hopf and Lax formulas, as it does not require the Legendre transform of either the Hamiltonian or the initial function, thereby broadening its practical applicability. Furthermore, although being based on characteristics, the implicit solution formula alleviates the need to solve the system of characteristic ODEs from the initial state to the present time. From an optimal control perspective, we further explore the connection of the proposed formula with the Pontryagin's maximum principle and Bellman's principle, showing that the proposed implicit solution formula serves as an implicit representation of Bellman's principle.

Building on this foundation, we propose a deep learning-based approach to solve HJ PDEs by learning the implicit solution formula. This method approximates the solution as a Lipschitz continuous function, leveraging the powerful expressive capacity of neural networks. Unlike traditional grid-based methods, our approach does not require discretization of the domain, making it highly scalable and efficient, especially for high-dimensional problems. This effectively mitigates the curse of dimensionality, ensuring that computational time and memory usage scale efficiently with dimensionality. Thanks to the inherent simplicity of the implicit solution formula, it obviates the need for computing the Legendre transform and individual characteristic trajectories, thereby enhancing both its applicability and computational efficiency across a wide range of problems. Through extensive and rigorous experimentation, we show that the proposed algorithm provides accurate solutions even for problems with up to 40 dimensions with negligible increases in computational cost. Importantly, the method also shows robust performance on various nonconvex HJ PDEs, for which mathematical demonstration has not been established, underscoring its versatility and potential.

We extend our approach to handle HJ PDEs with state-dependent Hamiltonians. In such cases, where the characteristic curves are no longer linear, deriving an implicit solution formula becomes more intricate. To address this, we approximate the characteristic curves as piecewise linear segments over short time intervals, applying the proposed implicit solution formula at each interval. This leads to an efficient time-marching algorithm that can handle state-dependent Hamiltonians, which we validate through a series of experiments involving high-dimensional optimal control problems. The results demonstrate that the proposed method is not only simple and efficient but also effectively solving a wide range of high-dimensional, nonconvex HJ PDEs, highlighting its potential as a valuable tool for addressing complex optimal control problems and dynamic systems.

2. Implicit Solution formula of Hamilton-Jacobi Equations.

2.1. Implicit Solution Formula along Characteristics. In this subsection, we introduce a novel implicit solution formula for the Hamilton-Jacobi partial differ-

ential equation (HJ PDE):

$$\begin{cases} u_t + H(\nabla u) = 0 & \text{in } \Omega \times (0,T) \\ u = g & \text{on } \Omega \times \{t=0\}, \end{cases} \tag{2.1}$$

where $\Omega \subset \mathbb{R}^d$ is the spatial domain, $H : \mathbb{R}^d \to \mathbb{R}$ is the Hamiltonian and $g : \Omega \to \mathbb{R}$ is the initial function. System of characteristic ODEs for (2.1), also known as Hamilton's system, is given by the following:

$$\begin{cases} \dot{\mathbf{x}} = \nabla H & \text{(2.2a)} \\ \dot{u} = q + \mathbf{p}^{\mathrm{T}} \nabla H = -H + \mathbf{p}^{\mathrm{T}} \nabla H & \text{(2.2b)} \\ \dot{q} = 0 & \text{(2.2c)} \\ \dot{\mathbf{p}} = 0, & \text{(2.2d)} \end{cases}$$

where the variables q and $\mathbf{p}$ are shorthand for the partial derivatives $q = u_t$ and $\mathbf{p} = \nabla u$, respectively. From (2.2d) it is clear that the value of $\mathbf{p}$, which is the sole argument of the Hamiltonian, remains constant along the characteristic. Therefore, the characteristic emanated from $\mathbf{x}(0) = \mathbf{x}_0 \in \Omega$ is a straight line

$$\mathbf{x}(t) = t \nabla H(\mathbf{p}) + \mathbf{x}_0,$$

implying that

$$\begin{aligned} u(t, \mathbf{x}(t)) &= -tH(\mathbf{p}) + t\mathbf{p}^{\mathrm{T}} \nabla H(\mathbf{p}) + u(\mathbf{x}_0, 0) \\ &= -tH(\mathbf{p}) + t\mathbf{p}^{\mathrm{T}} \nabla H(\mathbf{p}) + g(\mathbf{x}_0). \end{aligned}$$

Given the constant nature of $\mathbf{p}$ along each characteristic line, its value can be determined at any intermediate time between the initial and current times. In this context, we adopt $\mathbf{p}$ as the gradient of the solution at the current time. Substituting $\mathbf{p} = \nabla u$ and expressing $\mathbf{x}(t) = \mathbf{x} \in \Omega$ induces that

$$\mathbf{x}_0 = \mathbf{x} - t \nabla H(\nabla u(\mathbf{x}, t)),$$

and hence we attain the following **implicit solution formula** for HJ PDEs (2.1):

$$u(\mathbf{x}, t) = -tH(\nabla u) + t \nabla u^{\mathrm{T}} \nabla H(\nabla u) + g(\mathbf{x} - t \nabla H(\nabla u)). \tag{2.3}$$

It is worth noting that this implicit solution formula expresses the solution without requiring the Legendre transform of the Hamiltonian H or the initial function g. Moreover, it does not require to compute individual characteristic trajectories by solving the system of characteristic ODEs. Therefore, it provides a highly practical and straightforward approach to solving HJ PDEs. Building upon this formula, we propose a highly simple and effective deep learning-based methodology for solving HJ PDEs in section 3.

A key distinction between conventional approaches based on characteristics and the proposed implicit solution formula lies in the treatment of $\mathbf{p}$, which is chosen as the gradient of the solution ∇u at the current time t. Since $\mathbf{p}$ remains constant along each characteristic line, it can be readily determined from the initial data. Consequently, conventional methods typically express $\mathbf{p}$ in terms of $\nabla g(\mathbf{x}_0)$. However, these approaches are limited in situations where no characteristic traces back to the

initial time $t = 0$, resulting in the gradient at the current time not being accessible from the initial data. In contrast, our approach employs the current value of $\mathbf{p}(t) = \nabla u(\mathbf{x}, t)$, allowing the implicit solution formula to effectively handle such scenarios.

It is well-established that under certain assumptions on the Hamiltonian H and the initial function g, a representation formula for the viscosity solution can be derived. The first is the *Hopf-Lax formula*

$$u(\mathbf{x}, t) = \inf_{\mathbf{y}} \left\{ tH^* \left(\frac{\mathbf{x} - \mathbf{y}}{t} \right) + g(\mathbf{y}) \right\}, \tag{2.4}$$

which holds for convex (or concave) H and Lipschitz g [35, 5, 55], or for Lipschitz and convex H and continuous g [77], or also for strictly convex H and lower semicontinuous (l.s.c.) g [46, 47]. Here, H^* is the Legendre transform of H. On the other hand, *Hopf formula*

$$u(\mathbf{x}, t) = -\inf_{\mathbf{z}} \left\{ g^*(\mathbf{z}) + tH(\mathbf{z}) - \mathbf{x}^{\mathrm{T}}\mathbf{z} \right\} \tag{2.5}$$

is valid for Lipschitz and convex (or concave) g and merely continuous H [35, 5], or for convex g and Lipschitz H [77]. In the following, we demonstrate that the proposed implicit solution formula (2.3) represents these two respective formulas under the conditions under which they hold.

THEOREM 2.1. *Assume the Hamiltonian H is differentiable and satisfies*

$$\begin{cases} \mathbf{p} \mapsto H(\mathbf{p}) \ \textit{is strictly convex or concave}, \\ \lim_{|\mathbf{p}| \to \infty} \frac{H(\mathbf{p})}{|\mathbf{p}|} = +\infty, \end{cases} \tag{2.6}$$

and the initial function g is l.s.c. Then, the continuous function u that satisfies the implicit solution formula (2.3) *is the viscosity solution of* (2.1) *a.e.*

Proof. The proof is provided in Appendix A.1. ∎

THEOREM 2.2. *Assume the initial function g satisfies*

$$\begin{cases} \mathbf{x} \mapsto g(\mathbf{x}) \ \textit{is convex or concave}, \\ \lim_{|\mathbf{x}| \to \infty} \frac{g(\mathbf{x})}{|\mathbf{x}|} = +\infty, \end{cases}$$

that the Hamiltonian H is continuous, and that either the H or g is Lipschitz continuous. Then, the continuous function u that satisfies the implicit solution formula (2.3) *is the viscosity solution of* (2.1) *a.e.*

Proof. The proof is provided in Appendix A.2. ∎

Theorems 2.1 and 2.2 offer the theoretical validation of the implicit solution formula (2.3) under the assumption of convexity of the Hamiltonian H or the initial function g. However, this result has not yet been extended to the nonconvex case. Nonetheless, as illustrated in subsection 4.2, we present robust empirical evidence demonstrating the performance of the proposed approach through extensive experiments on a diverse range of nonconvex examples, where neither the Hamiltonian nor the initial function is convex (or concave). These results suggest the potential applicability and validity of the proposed formula in such scenarios.

To facilitate comprehension of the implicit solution formula, a simple example is presented.

EXAMPLE 2.1. *Consider a one-dimensional example with a quadratic Hamiltonian and a homogeneous initial condition:*

$$\begin{cases} u_t + u_x^2 = 0 & \text{in } \mathbb{R} \times (0, \infty) \\ u = 0 & \text{on } \mathbb{R} \times \{t = 0\}. \end{cases} \tag{2.7}$$

The viscosity solution to this problem is $u^* = 0$. Note that there are infinitely many Lipschitz functions satisfying (2.7) *a.e. [26], for instance,*

$$v(x,t) = \begin{cases} 0 & \text{if } \mid x \mid \geq t \\ x - t & \text{if } 0 \leq x \leq t \\ -x - t & \text{if } -t \leq x \leq 0. \end{cases}$$

This example shows that, although there are infinitely many Lipschitz functions that satisfy the HJ PDE, the implicit solution formula uniquely characterizes the viscosity solution. The implicit solution formula (2.3) *corresponding to* (2.7) *is written as*

$$u = t u_x. \tag{2.8}$$

For $t = 0$, (2.8) *satisfies the initial condition $u = 0$. For a fixed time $t > 0$,* (2.8) *represents an ordinary differential equation (ODE) with respect to the variable x with a coefficient that depends on t, and the ODE admits infinitely many solutions*

$$u = Ce^{x/t}, \ \forall C \in \mathbb{R}.$$

However, in order to satisfy the initial condition $u = 0$ at $t = 0$, it follows that C must be zero. Therefore, the viscosity solution $u^* = 0$ is the unique continuous function that satisfies the implicit solution formula (2.8).

This example illustrates that, despite the existence of an infinitely many weak solutions to the governing HJ PDE, the continuous function that satisfies the implicit solution formula (2.3) is the unique viscosity solution. However, it also suggests that, at a fixed time $t > 0$, the implicit solution formula may admit multiple solutions. For a fixed $t > 0$, the implicit solution formula (2.3) describes a first-order nonlinear static PDE (or an ODE in the one-dimensional case) in $\mathbf{x}$, where the time variable t appears as coefficients. The absence of boundary conditions in this static PDE at fixed $t > 0$ naturally leads to the ill-posedness of the PDE with multiple solutions. Therefore, the condition that the implicit solution formula (2.3) satisfies the initial condition at $t = 0$ is crucial, and finding a continuous function that satisfies the implicit solution formula across the entire spacetime domain from the initial data is essential for obtaining the unique viscosity solution. It is noteworthy, however, that the above example is taken in the unbounded spatial domain $\mathbb{R}$. For the general case of HJ PDEs on a bounded domain Ω, boundary conditions are specified. In such cases, the given boundary condition serves as the boundary condition for the static PDE described by (2.3) at a fixed time, thereby ensuring the uniqueness of the solution.

Remark 2.3. (Level set propagation) If the Hamiltonian H is homogeneous of degree one in its gradient argument, i.e., for all $\lambda > 0$.

$$H(\lambda \mathbf{p}) = \lambda H(\mathbf{p}),$$

then the implicit solution formula (2.3) comes down to the following simple formula a.e.

$$u(\mathbf{x}, t) = g(\mathbf{x} - t \nabla H(\nabla u)), \tag{2.9}$$

where the solution u is constant along the characteristics.

2.2. Control Perspectives on the Implicit Solution Formula. In this subsection, we revisit the implicit solution formula (2.3) from the perspective of control theory, elucidating that it represents an implicit formulation of Bellman's principle. This perspective also enables a comprehensive exploration of the relationships between the implicit solution formula and Pontryagin's maximum principle (PMP) and the Hopf-Lax formula (2.4), while also highlighting the distinctions between these established approaches and the proposed implicit solution formula.

Let $L = L(\mathbf{q}) : \mathbb{R}^d \to \mathbb{R}$ be the corresponding Lagrangian, that is, $L = H^*$, the Legendre transform of H. Under the assumption (2.6) on the Hamiltonian H, the Lagrangian satisfies

$$\begin{cases} \mathbf{q} \mapsto L(\mathbf{q}) \text{ is convex,} \\ \lim_{|\mathbf{q}|\to\infty} \frac{L(\mathbf{q})}{|\mathbf{q}|} = +\infty, \end{cases}$$

and $H(\mathbf{p}) = L^*(\mathbf{p}) = \sup_{\mathbf{q}} \left\{ \mathbf{p}^{\mathrm{T}}\mathbf{q} - L(\mathbf{q}) \right\}$.

It is well-known that the viscosity solution u of the HJ PDEs

$$\begin{cases} u_t + H(\nabla u) = u_t + \sup_{\mathbf{q}} \left\{ \nabla u^{\mathrm{T}}\mathbf{q} - L(\mathbf{q}) \right\} = 0, \\ u(\mathbf{x}, 0) = g(\mathbf{x}) \end{cases} \tag{2.10}$$

is represented by the value function of the following corresponding optimal control problem:

$$u(\mathbf{x}, t) = \inf_{\mathbf{q}} \left\{ \int_0^t L(\mathbf{q}(s))\, \mathrm{d}s + g(\mathbf{y}(0)) : \mathbf{y}(t) = \mathbf{x}, \dot{\mathbf{y}}(s) = \mathbf{q}(s), 0 \le s \le t \right\}. \tag{2.11}$$

Pontryagin's maximum principle (PMP) states that the optimal trajectory of state $\mathbf{y}(t)$ arriving at $\mathbf{y}(t) = \mathbf{x}$ and costate $\mathbf{p}(t)$ satisfies

$$\begin{cases} \dot{\mathbf{y}} = \mathbf{q}, \ \mathbf{y}(t) = \mathbf{x}, & \text{(2.12a)} \\ \dot{\mathbf{p}} = 0, \ \mathbf{p}(0) = \nabla g(\mathbf{y}(0)), & \text{(2.12b)} \\ \mathbf{q} = \operatorname{argmax}_{\mathbf{v}} \left\{ \mathbf{p}^{\mathrm{T}}\mathbf{v} - L(\mathbf{v}) \right\}. & \text{(2.12c)} \end{cases}$$

Note that this is identical to the characteristic ODEs for the state $\mathbf{x}$ (2.2a) and the gradient of the solution (2.2d) of the HJ PDEs. Therefore, PMP implies that the characteristic of the HJ PDEs corresponds to the optimal trajectory.

We now establish that both the Hopf-Lax formula and the implicit solution formula can be derived from the PMP in conjunction with the definition of the value function. This facilitates a comprehensive understanding of their relationships and differences.

Hopf-Lax Formula. Since the costate $\mathbf{p}$ is constant along the optimal trajectory (2.12b), the last condition (2.12c) of the PMP implies that $\mathbf{q}$ is also constant. Therefore, from (2.12a), it follows that the optimal trajectory of $\mathbf{y}$ is a straight line, whose solution is given by

$$\mathbf{y}(0) = \mathbf{y}(t) - t\mathbf{q} = \mathbf{x} - t\mathbf{q}. \tag{2.13}$$

Therefore, the optimal $\mathbf{q}$ is expressed by $\mathbf{y}(0)$ as follows:

$$\mathbf{q} = \frac{\mathbf{x} - \mathbf{y}(0)}{t}, \tag{2.14}$$

and hence, the minimization with respect to $\mathbf{q}$ can be transformed into a minimization with respect to $\mathbf{y}(0) \in \mathbb{R}^d$. Substituting this relation (2.14) into the definition of the value function (2.11) leads to the following Hopf-Lax formula:

$$\begin{aligned} u(\mathbf{x}, t) &= \inf_{\mathbf{y} \in \mathbb{R}^d} \left\{ \int_0^t L\left(\frac{\mathbf{x} - \mathbf{y}}{t}\right) \mathrm{d}s + g(\mathbf{y}) \right\} \\ &= \inf_{\mathbf{y} \in \mathbb{R}^d} \left\{ tL\left(\frac{\mathbf{x} - \mathbf{y}}{t}\right) + g(\mathbf{y}) \right\} \\ &= \inf_{\mathbf{y} \in \mathbb{R}^d} \left\{ tH^*\left(\frac{\mathbf{x} - \mathbf{y}}{t}\right) + g(\mathbf{y}) \right\}. \end{aligned}$$

In other words, the Hopf-Lax formula is derived by substituting the control $\mathbf{q}$ in terms of the initial state $\mathbf{y}(0) = \mathbf{y}$, leveraging the fact that the optimal trajectory is linear (2.14), as determined by the characteristic ODE of the PMP.

Implicit Solution formula. The implicit solution formula (2.3) is derived in a manner analogous to the Hopf-Lax formula, but it differs by expressing $\mathbf{p}$ as the gradient of the value function ∇u and additionally removing the Legendre transform. From the optimality of $\mathbf{q}$ in (2.12c), the Hamiltonian is written as

$$H(\mathbf{p}) = \mathbf{p}^{\mathrm{T}}\mathbf{q} + L(\mathbf{q}). \tag{2.15}$$

It follows that

$$\nabla_{\mathbf{p}} H = \frac{\partial H}{\partial \mathbf{p}} + \frac{\partial H}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \mathbf{p}} = \frac{\partial H}{\partial \mathbf{p}} = \mathbf{q}, \tag{2.16}$$

where $\frac{\partial H}{\partial \mathbf{q}} = 0$ is induced from (2.12c). Let $\mathbf{q}^*$ be the optimal control. Putting (2.12c), (2.13), and (2.14) to the definition of the value function in (2.11) leads to the following formula of the value function:

$$\begin{aligned} u(\mathbf{x}, t) &= tL(\mathbf{q}^*) + g(\mathbf{x} - t\mathbf{q}^*) \\ &= t\left(H(\mathbf{p}) - \mathbf{p}^{\mathrm{T}}\mathbf{q}^*\right) + g(\mathbf{x} - t\mathbf{q}^*) \\ &= t\left(H(\mathbf{p}) - \mathbf{p}^{\mathrm{T}}\nabla_{\mathbf{p}} H(\mathbf{p})\right) + g(\mathbf{x} - t\nabla_{\mathbf{p}} H(\mathbf{p})), \end{aligned} \tag{2.17}$$

where the second and third equalities are derived from (2.15) and (2.16), respectively. In other words, by substituting these two expressions (2.15) and (2.16), we derive the formula for the value function u that is independent of both the control $\mathbf{q}$ and the Legendre transform. Since the optimal $\mathbf{p}$ is the gradient of the value function ∇u, the solution formula (2.17) derived from PMP is identical to the implicit solution formula (2.3). Furthermore, it is important to note that the definition of the value function (2.11) precisely encapsulates the integral of the characteristic ODE of u (2.2b); that is, it directly represents the solution to the characteristic ODE of u (2.2b). In other words, the characteristic ODE of u (2.2b), which is not explicitly included in the PMP formula (2.12), is inherently embedded within the construction of Bellman's value function.

Consequently, the PMP (2.12a)-(2.12c), the Bellman's value function (2.11), the Hopf-Lax formula (2.4), and the proposed implicit solution formula (2.3) are all interconnected. The PMP states that the characteristics of the HJ PDEs correspond to the optimal trajectory, and Bellman's principle expresses the value function in terms of the solution to the characteristic ODE of u (2.2b). Together, they imply that the viscosity solution to the HJ PDEs (2.1) is defined along the characteristics.

However, there are notable differences in how these formulas yield the solution to (2.1) from a practical perspective. The PMP necessitates the solution of a single trajectory of the characteristic ODEs, which implies that, when attempting to compute the value function, one must solve a system of ODEs for each trajectory, introducing significant computational complexity. The Hopf-Lax formula, by exploiting the linearity of the optimal trajectory, eliminates the need to solve such ODEs. However, it involves the computation of the Legendre transform of the Hamiltonian H, ultimately leading to a challenging min-max problem. In contrast, the implicit solution formula (2.3) alleviates both the ODE solving of PMP and the min-max problem from the Hopf-Lax formula by leveraging the fact that the optimal costate $\mathbf{p}$ is the gradient of the solution ∇u. Consequently, compared to the PMP and the Hopf-Lax formula, the proposed implicit solution formula provides a more practical and widely applicable approach for solving HJ PDEs.

3. Learning Implicit Solution with Neural Networks. In this section, we introduce a deep learning-based approach for solving the implicit solution formula (2.3). Building upon the implicit solution formula, we propose the following minimization problem:

$$\min_{u} \mathcal{L}(u) \coloneqq \int_0^T \int_\Omega \Big(u + tH(\nabla u) - t\nabla u^{\mathrm{T}} \nabla H(\nabla u) - g(\mathbf{x} - t\nabla H(\nabla u)) \Big)^2 \,\mathrm{d}\mathbf{x}\,\mathrm{d}t. \tag{3.1}$$

The minimization problem (3.1) is inherently complex to be efficiently solved using classical numerical methods. To address this challenge, we propose a deep learning framework that has shown significant effectiveness in optimizing complex problems. This approach enables the scalable learning of the implicit solution formula, even in high-dimensional settings, thereby allowing the solution of the HJ PDEs (2.1) to be represented by a neural network, which is a Lipschitz function.

3.1. Implicit Neural Representation. We represent the solution u of the HJ PDEs (2.1) using a standard artificial neural network architecture, a multi-layer perceptron (MLP). The MLP is a function $u_\theta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ defined as the composition of functions, which can be expressed as follows:

$$u_\theta(\mathbf{x}, t) = W\left(h_L \circ \cdots \circ h_0(\mathbf{x}, t)\right) + \mathbf{b}, \ (\mathbf{x}, t) \in \mathbb{R}^n \times \mathbb{R}, \tag{3.2}$$

where $L \in \mathbb{N}$ is a given depth, $W \in \mathbb{R}^{1 \times d_L}$ is a weight of the output layer, $\mathbf{b} \in \mathbb{R}$ is an output bias and the perceptron (also known as the hidden layer) $h_\ell : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}$ is defined by

$$h_\ell(\mathbf{y}) = \sigma\left(W_\ell \mathbf{y} + \mathbf{b}_\ell\right), \ \mathbf{y} \in \mathbb{R}^{d_{\ell-1}}, \ \text{for all } \ell = 0, \ldots, L,$$

for weight matrices $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ with the input dimension $d_{-1} = d+1$, bias vectors $\mathbf{b}_\ell \in \mathbb{R}^{d_\ell}$, and a non-linear activation function σ. The dimensions d_ℓ of the hidden layers are also called by the width of the network. A shorthand notation θ is used to refer to all the parameters of the network, including the weights $\{W, W_0, \cdots, W_L\}$

and biases $\{\mathbf{b}, \mathbf{b}_0, \cdots, \mathbf{b}_L\}$. Since Lipschitz continuous activation functions σ are used, the MLP f_θ is a Lipschitz function and is also bounded on a bounded domain. Given the current parameter configuration, the parameters θ are successively adapted by minimizing an assigned loss function explained in the subsequent section.

Representing the solution of HJ PDEs using neural networks offers a scalable and efficient approach for modeling the spatio-temporal dependencies of the solution, offering several advantages over classical numerical schemes. Classical methods discretize the spatial vector field using primitives such as meshes, which scale poorly with the number of spatial samples. In contrast, representing the spatio-temporal function through networks known as implicit neural representations (INRs) encodes spatial and temporal dependencies through neural network parameters θ, each globally influencing the function. Consequently, the memory usage of INRs remains independent of the spatial sample size, being determined solely by the number of network parameters, which enables scalability in high-dimensional settings as evidenced in Section 4 for the proposed method. Additionally, INRs are adaptive, leveraging their capacity to represent arbitrary spatio-temporal locations of interest without requiring memory expansion or structural modifications. The expressivity of non-linear neural networks enables INRs to achieve superior accuracy compared to mesh-based and meshless methods, even under the same memory constraints [78, 9]. Furthermore, INRs represent the solution as a continuous function rather than at discrete points, with activation functions that can be tailored to the solution's regularity. Thanks to the architecture of MLPs, exact derivatives can be computed via the chain rule, eliminating the need for numerical differentiation methods such as finite differences. The partial derivatives of u_θ are efficiently computed using automatic differentiation library (`autograd`) [69].

3.2. Training. Given that the solution u is represented by the neural network u_θ, the minimization problem (3.1) reduces to finding the network parameters θ that minimize $\mathcal{L}$ in (3.1). For notational convenience, we denote

$$\mathcal{S}(u) = u + tH(\nabla u) - t\nabla u^{\mathrm{T}} \nabla H(\nabla u) - g(\mathbf{x} - t\nabla H(\nabla u)). \tag{3.3}$$

The integral of $\mathcal{L}$ is approximated using Monte Carlo methods

$$\hat{\mathcal{L}}(\theta) = \frac{1}{M} \sum_{j=1}^{M} \mathcal{S}(u_\theta(\mathbf{x}_j, t_j))^2, \tag{3.4}$$

with the M collocation points $\{(\mathbf{x}_j, t_j)\}_{j=1}^M$ randomly sampled from a uniform distribution $U(\Omega \times [0, T])$. This empirical loss $\hat{\mathcal{L}}$ serves as the loss function for training the neural network. The current network parameters are updated using a gradient-based optimizer to minimize the loss function $\hat{\mathcal{L}}$. During training, different random collocation points are employed in each iteration to ensure accurate learning across the entire domain. The partial derivatives of the network u_θ are computed through `autograd` when calculating the loss. The training procedure for optimization using gradient descent is summarized in Algorithm 3.1.

This algorithm is considerably simpler than existing methodologies in several respects and yields remarkable results, as demonstrated in section 4. Previous approaches [20, 12, 13, 10], which aimed to obtain the viscosity solution via the Hopf or Lax formula, involved calculating the Legendre transform of the Hamiltonian or the initial function. Therefore, these methods were restricted to problems where the

Algorithm 3.1 Algorithm for Learning Implicit Solution of HJ PDEs

1: Initialize the network u_θ with an initial network parameter θ_0.
2: **for** $n = 0, \cdots, N$ **do**
3: Randomly sample M collocations points $\{(\mathbf{x}_j, t_j)\}_{j=1}^M \sim U(\Omega \times [0, T])$.
4: Calculate the loss by Monte Carlo integration

$$\hat{\mathcal{L}}(\theta_n) = \frac{1}{M} \sum_{j=1}^{M} \mathcal{S}\left(u_{\theta_n}(\mathbf{x}_j, t_j)\right)^2.$$

5: Update θ_n by gradient descent with a step size $\alpha > 0$

$$\theta_{n+1} \leftarrow \theta_n - \alpha \nabla_\theta \hat{\mathcal{L}}(\theta_n).$$

6: **end for**
7: **return** u_{θ_N} as the predicted viscosity solution to the HJ PDEs (2.1).

Legendre transform was easily computable or required solving numerically intensive min-max problems for each spatio-temporal point, limiting their general applicability. In contrast, our approach bypasses the Legendre transform by using an implicit formula, enabling us to handle a broader class of Hamiltonians and initial functions. Moreover, while prior methods based on characteristics or PMP [41, 42, 82] necessitated solving a system of ODEs to track individual trajectories, the proposed method eliminates the requirement for explicit trajectory computation.

The proposed method also overcomes the limitations of classical grid-based numerical methods, which face challenges when dealing with high-dimensional or large-scale problems due to the increasing number of grid points required as the dimension or domain size grows. Unlike classical methods, the deep learning approach is characterized by its mesh-free nature, which precludes the necessity for a grid discretization of the computational domain. The mesh-free approach allows for the random selection of collocation points in each iteration, with the selected points gradually covering the domain as iterations proceed. Consequently, the computational and memory requirements do not increase significantly with higher dimensions, as evidenced in **??** for the proposed method. Furthermore, under certain mild assumptions, it has been demonstrated that this stochastic gradient descent applied using randomly sampled collocation points converges to the minimizer of the original expectation loss [43]. The absence of mesh generation also simplifies the practical implementation of the method.

This approach also offers distinct advantages over existing deep learning methods for solving PDEs. As an unsupervised learning method, it solves HJ PDEs given the Hamiltonian and initial condition, without requiring solution data for training. This addresses the limitations of supervised learning methods [61, 25, 16], which rely on extensive solution data and do not guarantee generalization to unseen problems. The proposed approach also offers strengths compared to the established unsupervised methods, such as PINNs [72] and the DeepRitz method [83]. DeepRitz, which is based on a variational formulation, is not suitable for HJ PDEs. PINNs, on the other hand, use the residual of the PDE itself as the loss function, which cannot guarantee obtaining the viscosity solution for HJ PDEs among multiple solutions. Since the viscosity solution cannot be directly derived from the PDE itself, there are

inherent challenges in obtaining it from the PDE residual loss used in PINNs. In comparison, the proposed method learns an implicit formula for the solution that naturally inherits the properties of the viscosity solution through the characteristic equation, enabling effective solutions to HJ PDEs. Furthermore, most deep learning methods for solving PDEs, including PINNs and DeepRitz method, use a training loss function that is the linear sum of the loss term corresponding to the PDE and the loss term for the initial condition. This requires a regularization parameter to balance the two loss terms, which is highly sensitive and difficult to optimize [79]. In contrast, the proposed method employs an implicit solution formula, whereby the initial condition is automatically incorporated by substituting $t = 0$ into (3.4). As a result, our approach eliminates the need for a regularization parameter, using only a single loss function and obviating the distinction between the initial condition and the PDE.

When boundary conditions are specified in the spatial domain Ω, we incorporate additional loss terms to enforce these conditions. For instance, when a Dirichlet boundary condition is imposed with the boundary function $h : \partial\Omega \to [0, T] \to \mathbb{R}$, the following loss function is used:

$$\frac{1}{M_b} \sum_{j=1}^{M_b} \left(u\left(\mathbf{x}_j^b, t_j^b\right) - h\left(\mathbf{x}_j^b, t_j^b\right) \right)^2,$$

where the M_b boundary collocation points $\left(\mathbf{x}_j^b, t_j^b\right) \in \partial\Omega \times [0, T]$ are randomly sampled from a uniform distribution. Similarly, for a periodic boundary condition, the boundary loss term is given as follows:

$$\frac{1}{M_b} \sum_{j=1}^{M_b} \left(u\left(\mathbf{x}_j^b, t_j^b\right) - u\left(\mathbf{y}_j^b, t_j^b\right) \right)^2,$$

where $\mathbf{y}_i^b \in \partial\Omega$ represents the point on the opposite side of the domain Ω corresponding to $\mathbf{x}_i^b$. This boundary loss, weighted by the regularization parameter $\lambda > 0$, is then added to the implicit solution loss $\hat{\mathcal{L}}$ (3.4) to form the total training loss.

Remark 3.1. If the goal is to obtain a solution at a specific time $t = T$ rather than over the entire temporal evolution, integrating over time in the loss function may not be necessary. However, as shown in Example 2.1 in subsection 2.1, when the PDE lacks boundary conditions, the implicit solution formula at a fixed t results in a differential equation without boundary conditions, leading to the possibility of multiple spurious solutions. To address this, it is preferable to incorporate an integral over the entire temporal domain in the loss function, thereby training a continuous network to find a continuous solution that satisfies (2.3) across the entire spacetime domain. On the other hand, when boundary conditions are given in the HJ PDEs (2.1), these can serve as the boundary condition for the differential equation (2.3) at the fixed time, ensuring the uniqueness of the solution. In such cases, training the model exclusively with respect to the terminal time $t = T$ may suffice.

3.3. State-dependent Hamiltonian. In this subsection, we propose an algorithm for the case of a state-dependent Hamiltonian, inspired by the implicit solution formula (2.3). Consider the state-dependent HJ PDEs defined in a domain $\Omega \subset \mathbb{R}^d$

$$\begin{cases} u_t + H\left(\mathbf{x}, \nabla u\right) = 0 & \text{in } \Omega \times (0, T) \\ u = g & \text{on } \Omega \times \{t = 0\}. \end{cases} \tag{3.5}$$

The system of characteristic ODEs of (3.7) is given by

$$\begin{cases} \dot{\mathbf{x}} = \nabla_{\mathbf{p}} H & \text{(3.6a)} \\ \dot{u} = -H + \mathbf{p}^{\mathrm{T}} \nabla_{\mathbf{p}} H & \text{(3.6b)} \\ \dot{\mathbf{p}} = -\nabla_{\mathbf{x}} H, & \text{(3.6c)} \end{cases}$$

where $\mathbf{p} = \nabla u$. Given that $\mathbf{p}$ is no longer a constant along the characteristic (3.6c), the characteristics are not linear but instead curves. Consequently, computing the integral along these curves becomes highly challenging, making the derivation of an implicit solution formulation difficult.

Piecewise Linear Approximation of Characteristic Curves. We assume that $\mathbf{p}$ remains relatively constant over short time intervals. In other words, we approximate the characteristic curve as linear over short time segments. To this end, we discretize the temporal domain by

$$t_0 = 0 < t_1 = \Delta t < t_2 = 2\Delta t < \cdots < t_N = N\Delta t = T.$$

For each $k = 0, \cdots, N-1$, we can write the solution as follows: for $t \in [t_k, t_k + \Delta t]$,

$$u(\mathbf{x}, t) = u(\mathbf{x}, t_k + \tau) = u^k(\mathbf{x}, \tau)$$

with $t = t_k + \tau$, $\tau \in [0, \Delta t]$. Then u^k can be regarded as the solution of the following HJ PDEs for time $0 \le t \le \Delta t$ with the initial function $u^k(\cdot, 0) = u^{k-1}(\cdot, \Delta t) = u(\cdot, k\Delta t)$:

$$\begin{cases} u_t^k + H\left(\mathbf{x}, \nabla u^k\right) = 0 & \text{in } \Omega \times (0, \Delta t) \\ u^k(\mathbf{x}, 0) = u^{k-1}(\mathbf{x}, \Delta t) & \text{on } \Omega. \end{cases} \tag{3.7}$$

Assuming that $\mathbf{p}$ remains constant within each short time interval $[t_k, t_k + \Delta t]$, similar to the state-independent Hamiltonian discussed in subsection 2.1, we can derive the following *implicit solution formula* for (3.7):

$$u^k(\mathbf{x}, \tau) = -\tau H\left(\mathbf{x}, \nabla u^k(\mathbf{x}, \tau)\right) + \tau \nabla u^k(\mathbf{x}, \tau)^{\mathrm{T}} \nabla_{\mathbf{p}} H\left(\mathbf{x}, \nabla u^k(\mathbf{x}, \tau)\right) \tag{3.8}$$

$$+ u^{k-1}\left(\mathbf{x} - \tau \nabla_{\mathbf{p}} H\left(\mathbf{x}, \nabla u^k\right), \Delta t\right). \tag{3.9}$$

This can be regarded as an implicit Euler discretization of the characteristic ODE (3.6a):

$$\mathbf{x}(\tau) = \mathbf{x}(0) + \tau \nabla_{\mathbf{p}} H\left(\mathbf{x}(\tau), \nabla u(\mathbf{x}(\tau), \tau)\right) + O\left(\tau^2\right)$$

for small $\tau \in [0, \Delta t]$. For notational simplicity, let us denote

$$\begin{aligned} \mathcal{S}\left[u^k, u^{k-1}\right](\mathbf{x}, \tau) =& u^k(\mathbf{x}, \tau) - \tau \nabla u^k(\mathbf{x}, \tau)^{\mathrm{T}} \nabla_{\mathbf{p}} H\left(\mathbf{x}, \nabla u^k(\mathbf{x}, \tau)\right) \\ &+ \tau H\left(\mathbf{x}, \nabla u^k(\mathbf{x}, \tau)\right) - u^{k-1}\left(\mathbf{x} - \tau \nabla_{\mathbf{p}} H\left(\mathbf{x}, \nabla u^k\right), \Delta t\right). \end{aligned}$$

Time Marching Algorithm. Based on these, we propose the following time marching method that solves the HJ PDEs (3.7) with the state dependent H sequentially over short time intervals $[t_k, t_k + \Delta t]$:

1. Set the initial condition $u^0 = g$.

2. For $k = 1, \cdots, N$,

$$u^k = \arg\min_v \int_0^{\Delta t} \int_\Omega \left(\mathcal{S}\left[v, u^{k-1}\right](\mathbf{x}, \tau) \right)^2 \mathrm{d}\mathbf{x}\, \mathrm{d}t. \tag{3.10}$$

For each k, the predicted function u^k approximates the solution u of (3.7) on $t_k \leq t \leq t_{k+1}$, i.e.,

$$u^k(\mathbf{x}, \tau) \approx u(\mathbf{x}, k\Delta t + \tau), \ \forall \tau \in [0, \Delta t], \mathbf{x} \in \Omega.$$

It is important to note that rather than using separate neural networks for each u^k, the model is trained using a single network, ensuring memory efficiency. After training the network for the solution u^{k-1} on the time interval $[t_{k-1}, t_{k-1} + \Delta t]$, the network parameters are saved. These saved parameters are then used as the initial function to train the same network for the subsequent solution u^k by (3.10). As a result, when training u^k, the network is initialized with u^{k-1}, which accelerates the training process. Consequently, although the learning process is divided for time marching, the rapid convergence for each u^k ensures that the overall training time does not increase significantly.

4. Numerical Results. In this section, we evaluate the performance of the proposed deep learning-based method for learning the implicit solution formula through a series of diverse examples and high-dimensional problems. Experiments are conducted on up to 40 dimensions, and both qualitative and quantitative results are presented. Although theoretical verification has not yet been established, extensive experiments on nonconvex Hamiltonians are also included, demonstrating the effectiveness of the proposed method in learning viscosity solutions.

To assess the scalability of the proposed method, we maintain the same experimental configurations across all cases, regardless of dimensionality or domain size. All experiments are conducted using an MLP (3.2) of a depth $L = 5$ and a width $d_\ell = 64$ with the `SoftPlus` activation function $\sigma(x) = \frac{1}{\beta} \log\left(1 + e^{\beta x}\right)$ with $\beta = 100$. Additional experiments on alternative network architectures are presented in Appendix C.1. The network is trained for $N = 200,000$ epochs using Adam optimizer [45] with an initial learning rate of 10^{-3} decayed by a factor of 0.99 whenever the loss decreased. In each epoch, $M = 5,000$ collocation points were uniformly randomly sampled from the domain. When boundary conditions are given, the regularization parameter λ is set to 0.1 and the number of boundary collocation points M_b is set to 200. All experiments are implemented on a single NVIDIA GV100 (TITAN V) GPU. A more detailed description of the experimental configuration is provided in Appendix B.

4.1. Convex Hamiltonians. We begin by measuring the error with respect to the true solution for the theoretically validated convex (or concave) Hamiltonian. Experiments are conducted in up to 40 dimensions. In addition to accuracy, we evaluate computational time and memory consumption to assess the efficiency of the approach in high-dimensional settings. We also investigate the effects of the sampling strategy for collocation points and the network size on performance.

EXAMPLE 4.1 (Quadratic Hamiltonians). *Consider two HJ equations with the quadratic Hamiltonians for which the true viscosity solutions are known analytically:*

- *Quadratic:* $H(\mathbf{p}) = \frac{1}{2}\|\mathbf{p}\|_2^2$ *and initial function* $g(\mathbf{x}) = \|\mathbf{x}\|_1$. *The exact solution is given by* $u^*(\mathbf{x}, t) = \sum_{i:|x_i| \geq t} \left(|x_i| - \frac{t}{2}\right) + \sum_{i:|x_i| < t} \frac{x_i^2}{2t}$.

TABLE 1

Quantitative results for quadratic Hamiltonian Example 4.1 in dimensions $d = 1, 2, 3, 10, 40$. The mean squared errors (MSE) with respect to the exact solution and the memory usage (Mem, in MB) for storing the predicted solutions are reported.

	$d=1$		$d=2$		$d=3$		$d=10$		$d=40$	
Method	MSE	Mem	MSE	Mem	MSE	Mem	MSE	Mem	MSE	Mem
Ours	1.14E-7	0.06	1.91E-7	0.06	3.21E-6	0.06	2.56E-5	0.06	1.30E-3	0.07
PINNs	2.39E-6	0.06	2.14E-5	0.06	1.98E-4	0.06	5.78E-3	0.06	8.00	0.07
WENO (same Mem)	3.84E-5	0.06	1.3E-3	0.06	3.68E-3	0.06	N/A	N/A	N/A	N/A
WENO (same MSE)	1.14E-7	6.17	1.83E-7	51498.41	N/A	N/A	N/A	N/A	N/A	N/A

- *Negative quadratic: $H(\mathbf{p}) = -\frac{1}{2}\|\mathbf{p}\|_2^2$ and initial function $g(\mathbf{x}) = \|\mathbf{x}\|_1$. The exact solution is $u^*(\mathbf{x}, t) = \|\mathbf{x}\|_1 + \frac{dt}{2}$.*

Experiments were conducted on the $1, 2, 3, 10$, and 40 dimensions. We evaluate the performance of the proposed method in comparison with physics-informed neural networks (PINNs) and the classical weighted essentially non-oscillatory (WENO) scheme. The PINNs were trained using the same neural network architecture as our model, while the WENO scheme was implemented on a uniformly discretized grid. Following the evaluation setup of a prior study [9], we considered two WENO configurations to ensure a fair comparison with our deep learning-based approach: one with memory usage comparable to that of the neural networks employed in our method and PINNs for solution storage, and another yielding a similar level of accuracy to our method.

The errors with respect to the exact solutions, along with the corresponding memory usage for storing solutions, are summarized in Tables 1 and 2 for two test cases. The results demonstrate that the proposed method effectively computes solutions to HJ equations even in high-dimensional settings. In contrast, PINNs exhibit significantly higher errors compared to our approach. This discrepancy highlights the difference between the PDE-based loss in PINNs, which lacks information about the viscosity solution, and our implicit formula, which inherently captures the characteristic structure of the solution. For the WENO scheme, mesh resolutions with memory consumption comparable to that of the neural networks are generally insufficient to achieve the same level of accuracy as our proposed method. Achieving comparable accuracy with WENO requires significantly higher resolution, leading to substantially increased memory usage, which becomes computationally infeasible in high-dimensional settings.

It is important to emphasize that the memory usage reported for both our method and PINNs corresponds to the memory required to store a continuous solution function over the entire spatio-temporal domain. In contrast, the WENO scheme computes solutions only at discrete grid points. Therefore, despite using comparable amounts of memory, the proposed method and PINNs offer a significant advantage by providing a global, continuous solution that generalizes across the domain. As discussed in subsection 3.1, the memory requirements of INRs are primarily governed by the network size. While increasing the input dimension enlarges the input layer, the overall network size remains largely unaffected. Consequently, the results presented in Tables 1 and 2 indicate that memory usage is nearly independent of dimensionality. These findings highlight the strong scalability of deep learning approaches with respect to dimension, making them particularly well-suited for solving high-dimensional problems. The computational time and peak memory usage are provided in Appendix C.4.

EXAMPLE 4.2 (Nonsmooth Solutions). *To more closely examine the proposed method, we conduct ablation studies on the distribution of collocation points and the*

TABLE 2

Quantitative results for negative quadratic Hamiltonian in Example 4.1 in dimensions $d = 1, 2, 3, 10, 40$. The mean squared errors (MSE) with respect to the exact solution and the memory usage (Mem, in MB) for storing the predicted solutions are reported.

	$d=1$		$d=2$		$d=3$		$d=10$		$d=40$	
Method	MSE	Mem	MSE	Mem	MSE	Mem	MSE	Mem	MSE	Mem
Ours	8.59E-6	0.06	1.10E-4	0.06	1.15E-4	0.06	1.63E-4	0.06	1.23E-3	0.07
PINNs	1.40E-5	0.06	2.98E-4	0.06	5.53E-4	0.06	2.20E-2	0.06	11.90	0.07
WENO (same Mem)	1.01E-6	0.06	1.30E-3	0.06	1.35E-1	0.06	N/A	N/A	N/A	N/A
WENO (same MSE)	–	–	1.11E-4	3.81	1.53E-4	686.33	N/A	N/A	N/A	N/A

network size and for the following two HJ, both of which admit non-smooth solutions:

- *L^2 Hamiltonian: $H(\mathbf{p}) = \|\mathbf{p}\|_2$ and the initial function is the signed distance function from two $(d-1)$-spheres $g(\mathbf{x}) = \min\{\|\mathbf{x}-\mathbf{c}_1\|_2 - r, \|\mathbf{x}-\mathbf{c}_2\|_2 - r\}$, where $\mathbf{c}_1 = (-0.3, 0, \cdots, 0)$, $\mathbf{c}_2 = (0.3, 0, \cdots, 0)$, and $r = 0.2$. This represents the level set equation [64] that governs the collision of two spheres, initially separated and moving along their respective normal directions, which ultimately results in a collision. The exact solution is given by $u^*(\mathbf{x}, t) = \min\{\|\mathbf{x}-\mathbf{c}_1\|_2 - r - t, \|\mathbf{x}-\mathbf{c}_2\|_2 - r - t\}$.*
- *L^∞ Hamiltonian: $H(\mathbf{p}) = \|\mathbf{p}\|_\infty$ and $g(\mathbf{x}) = \|\mathbf{x}\|_1$, where the exact solution is $u^*(\mathbf{x}, t) = \max\{\|\mathbf{x}\|_1 - t, 0\}$.*

***Effect of Collocation Sampling Strategies**. To investigate the impact of the sampling strategy for collocation points on solving the HJ equation, we compare three different strategies for selecting collocation points $\{(\mathbf{x}_j, t_j)\}_{j=1}^M$ used in computing the loss defined in (3.4):*

1. *Uniform Random: Sampling uniformly at random over the spatio-temporal domain.*
2. *Residual-Adaptive: Adaptive sampling based on regions where the residual of the implicit solution formula (3.3) is large.*
3. *Gradient-Adaptive: Adaptive sampling that emphasizes regions with large gradients of the solution.*

In the limit as the number of collocation points increases, uniform random sampling may under-represent regions where the solution exhibits high curvature or non-smooth behavior. The proposed adaptive strategies aim to counteract this imbalance by concentrating points where the solution is less regular, aligning with ideas from importance sampling or adaptive quadrature. Gradient-adaptive sampling is motivated by the observation that non-smooth features such as kinks are often associated with large solution gradients. By focusing collocation points in such regions, the network is better trained to capture these non-smooth characteristics. Residual-adaptive sampling, on the other hand, aims to minimize the training loss more effectively by allocating more samples where the residual of the implicit solution formula is high, acting as a heuristic form of importance sampling. Notably, residuals are likely to be large in the vicinity of kinks, further reinforcing the coherence between the two adaptive strategies. A visual comparison of the sampled point distributions, provided in Appendix C.5, supports the intended effect of these adaptive sampling strategies. Implementation details for the two adaptive sampling strategies are provided in Appendix B.

Table 3 presents the MSE with respect to the true solution for both L^2 and L^∞ Hamiltonian problems under different sampling strategies. Across all dimensions and sampling methods, our implicit solution formula-based method accurately captures the

TABLE 3

Comparison of sampling strategies for collocation points in terms of MSE for problems in Example 4.2. Three different methods—uniform random sampling, residual-adaptive sampling, and gradient-adaptive sampling—are evaluated across dimensions $d = 1, 10$, and 40.

	L^2 Hamiltonian			L^∞ Hamiltonian		
Sampling Strategy	$d = 1$	$d = 10$	$d = 40$	$d = 1$	$d = 10$	$d = 40$
Uniform random	7.08E-6	5.57E-5	1.13E-3	4.62E-3	1.71E-3	2.89E-3
Residual-adaptive	8.66E-6	1.21E-5	3.87E-4	6.67E-3	5.46E-3	1.27E-3
Gradient-adaptive	5.94E-6	1.91E-6	2.51E-4	4.24E-3	1.63E-3	7.87E-4

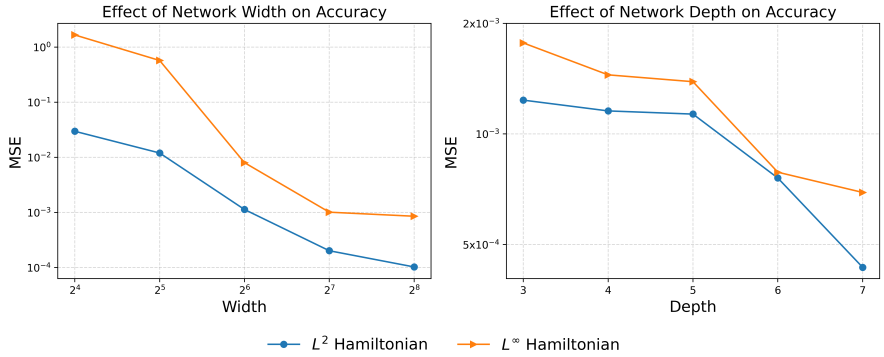
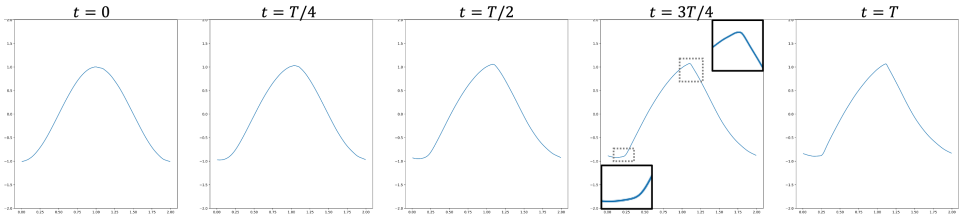


FIG. 1. *Effects of network width (left) and depth (right) on the solution accuracy of 40-dimensional Hamilton–Jacobi equations in Example 4.2. In the left plot, the depth is fixed at 5 while varying the width; in the right plot, the width is fixed at 64 while varying the depth. The plots depict mean squared errors (MSE) on a logarithmic scale for both L^2 and L^∞ Hamiltonians. Results demonstrate that increasing the network size enhances accuracy.*

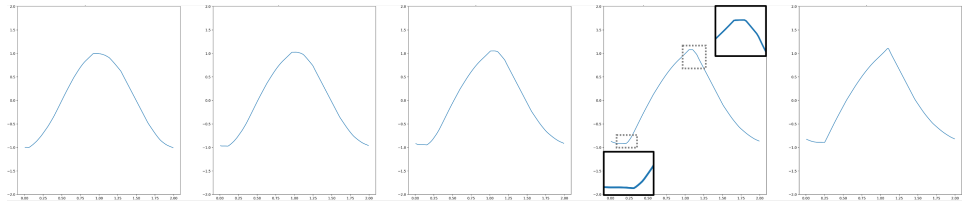
non-smooth viscosity solution, demonstrating its robustness even in high-dimensional settings. While all methods yield accurate solutions, adaptive sampling strategies generally lead to improved performance compared to uniform random sampling. In particular, the gradient-based adaptive sampling, which concentrates points near kink regions, consistently achieves the best accuracy across all dimensions.

***Effect of Network Capacity**. We further investigate the effects of network width and depth on the solution accuracy for the 40-dimensional problems introduced in Example 4.2. Figure 1 illustrates the MSE on a logarithmic scale for both L^2 and L^∞ Hamiltonians as network size varies. All experiments were conducted using our default configuration, which employs a uniform random sampling strategy for the collocation points. The results clearly demonstrate that increasing the network width and depth consistently improves the accuracy of the learned solution. Increasing the network size enhances the representational capacity of the model, enabling it to accurately approximate solutions even in the 40-dimensional setting. Consequently, the error is reduced to the order of 10^{-4}, demonstrating the effectiveness of scaling network size for high-dimensional HJ equations.*

4.2. Nonconvex Hamiltonians. In this subsection, we present experimental results for various Hamiltonians that are neither convex nor concave. While the theoretical proof for the proposed implicit solution formula has not yet been established in the nonconvex case, the experiments show that the proposed method effectively



(a) Ours



(b) PINNs

FIG. 2. *The numerical results for one-dimensional Example* 4.3. *The horizontal axis represents the spatial domain, while the vertical axis corresponds to the solution values. A zoomed-in view at time* $t = \frac{3}{4}T$ *highlights the kink region to better illustrate the differences between the two methods, where the dashed boxes delineate the regions selected for magnification*

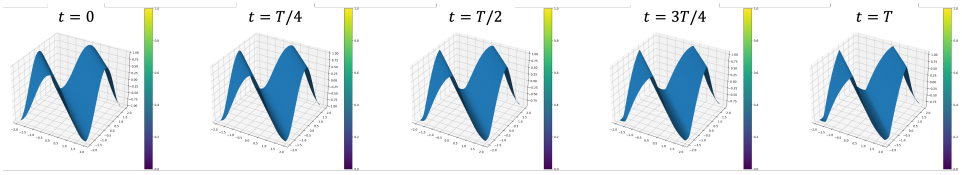
yields viscosity solutions.

EXAMPLE 4.3. *We solve the nonlinear equation with a nonconvex Hamiltonian* $H(\nabla u) = -\cos\left(\sum_{i=1}^{d} u_{x_i} + 1\right)$, *the initial function* $g(\mathbf{x}) = -\cos\left(\frac{\pi}{d}\sum_{i=1}^{d} x_i\right)$, *and periodic boundary conditions presented in [67]. The results for* $d = 1, 2$ *are shown in Figures 2 and 3, respectively, with solutions plotted up to time* $T = 0.2$, *at which point kinks have emerged. Both the proposed method and PINNs capture the solution behavior consistent with the findings in [67]. However, upon closer inspection of the zoomed-in regions, it is evident that the PINN solutions do not accurately capture the kink formation. In contrast, our proposed method accurately captures the formation of kinks in both the one- and two-dimensional cases, demonstrating its effectiveness.*

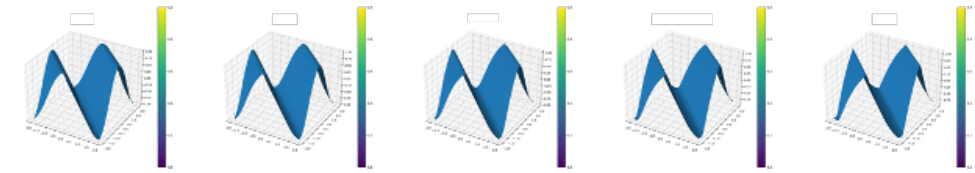
EXAMPLE 4.4. *We solve the two-dimensional Riemann problem [67] with a nonconvex sinusoidal Hamiltonian* $H(\nabla u) = \sin(u_x + u_y)$ *and initial function* $g(\mathbf{x}) = \pi(|y| - |x|)$.

The predicted solution up to $T = 1$ *obtained by the proposed method and PINNs are presented in Figure* 4. *Our method demonstrates behavior consistent with the numerical results in [67], where the initially smooth solution profile gradually sharpens over time. In contrast, although the PINN accurately captures the initial condition, it learns a solution that deviates from the correct viscosity solution. This highlights the strength of the proposed implicit solution formula, particularly in capturing the correct dynamics of kink dynamics where PINNs tend to fail.*

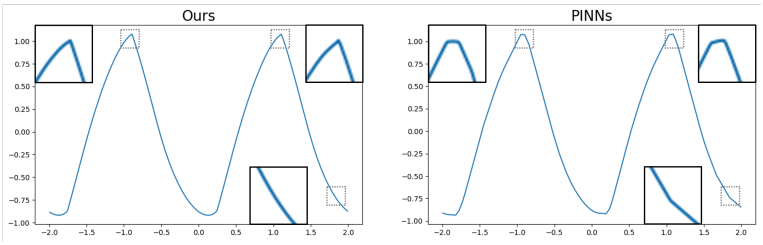
EXAMPLE 4.5. *The above nonconvex examples are actually one-dimensional along the diagonal. To evaluate the performance of the proposed method on fully two-dimensional problems [6], we solve the HJ PDE with* $H(\nabla u) = u_x u_y$ *and* $g(\mathbf{x}) = \sin(x) + \cos(y)$ *with periodic boundary condition and* $T = 1.5$. *The solution is smooth for* $t < 1$ *and exhibits kinks for* $t \geq 1$. *Consistent with the findings in [6], the results of both the proposed method and PINNs shown in Figure* 5 *indicate that the proposed*



(a) Ours

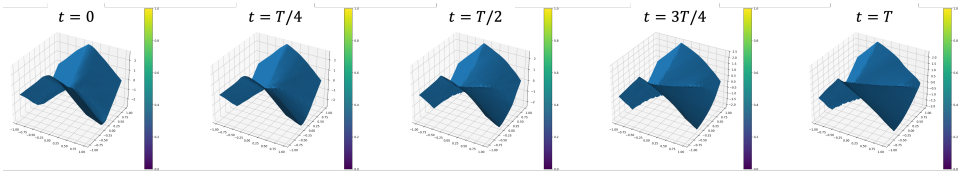


(b) PINNs

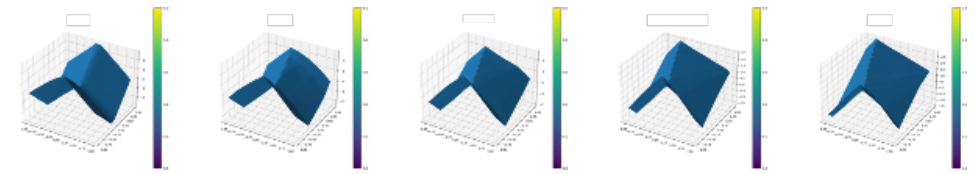


(c) Cut along $x = y$

FIG. 3. *The numerical results for two-dimensional Example* 4.3. *Figures (a) and (b) depict surface plots of the predicted solutions by the proposed method and PINNs, respectively, where the vertical axis represents the solution values over the two-dimensional domain. To examine the differences between the two methods in greater detail, Figure (c) shows cross-sectional plots along the line $x = y$ at the final time. The regions indicated by dashed boxes are zoomed in to provide a clearer visualization of these differences.*

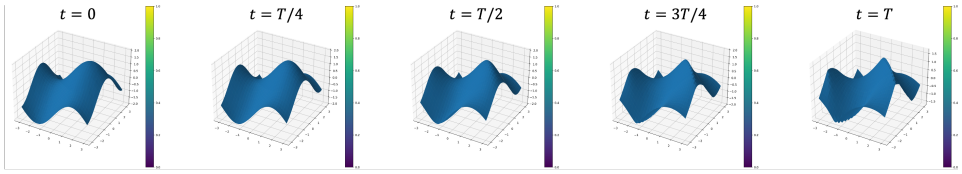


(a) Ours

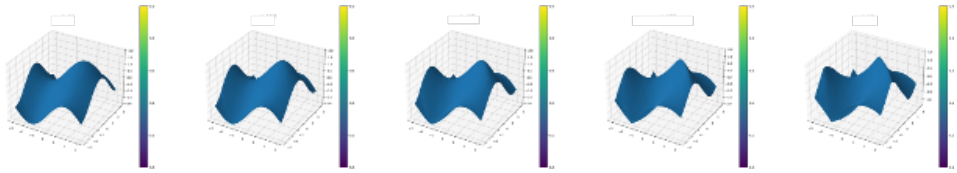


(b) PINNs

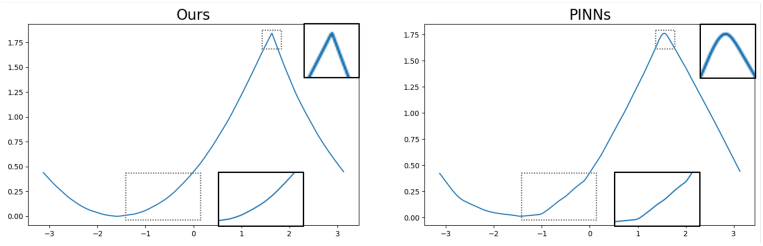
FIG. 4. *Surface plot illustrating the temporal evolution of the numerical solutions obtained by the proposed method and PINNs for Example* 4.4 *at times $t = 0, T/4, T/2, 3T/4, T$, where the vertical axis represents the solution values over the two-dimensional spatial domain.*



(a) Ours



(b) PINNs



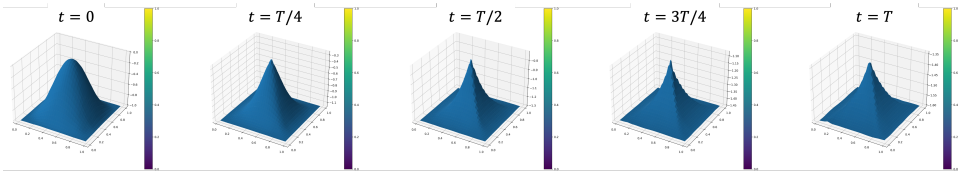
(c) Cut along $y = 0$

FIG. 5. *The numerical results for two-dimensional Example* 4.5. *Figures (a) and (b) depict surface plots of the predicted solutions by the proposed method and PINNs, respectively, where the vertical axis represents the solution values over the two-dimensional domain. To examine the differences between the two methods in greater detail, Figure (c) shows cross-sectional plots along the line* $y = 0$ *at the final time. The regions indicated by dashed boxes are zoomed in to provide a clearer visualization of these differences.*

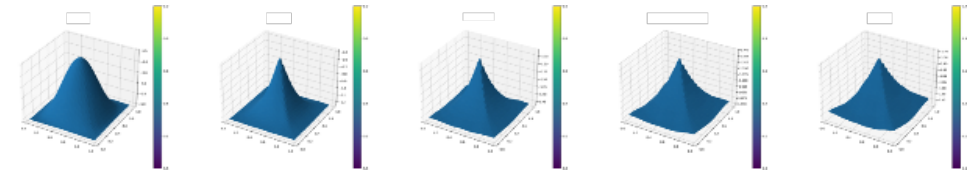
method continues to accurately capture the solution even after the formation of a kink. A closer examination of Figure 5*(c) further reveals that, compared to the PINN, our method more precisely captures the kink and also produces a smoother approximation in the non-singular regions.*

EXAMPLE 4.6 (Eikonal equation). *Consider a two-dimensional nonconvex problem [67] with* $H(\nabla u) = \sqrt{u_x + u_y + 1}$ *and* $g(\mathbf{x}) = \frac{1}{4}(\cos(2\pi x) - 1)(\cos(2\pi y) - 1) - 1$ *that arises in geometric optics. The results up to time* $T = 0.45$ *are shown in Figure* 6*. For the proposed method, the initially smooth function progressively concentrates toward the center of the spatial domain and becomes increasingly sharp over time, in agreement with the numerical findings reported in [67]. In contrast, while the PINN appears to capture the overall shape of the solution, its predictions deteriorate over time—particularly near the flat outer regions of the solution, which become distorted. This behavior highlights a key distinction between the two approaches: PINNs rely solely on minimizing the PDE residual, whereas our method leverages an implicit solution formula grounded in the method of characteristics.*

EXAMPLE 4.7. *(Combustion problem) Consider the combustion problem [48] with* $H(\nabla u) = -\sqrt{u_x + u_y + 1}$ *and* $g(\mathbf{x}) = \cos(2\pi x) - \cos(2\pi y)$*. Results up to time* 0.27 *are given in Figure* 7 *for both our method and PINNs. The proposed method accurately*

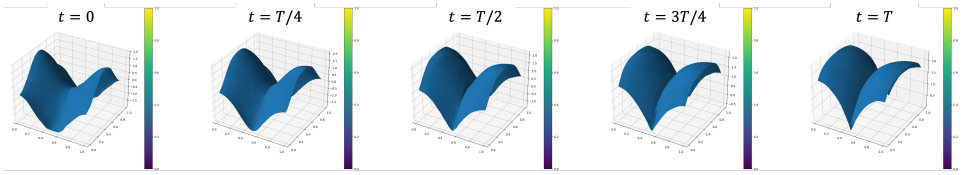


(a) Ours

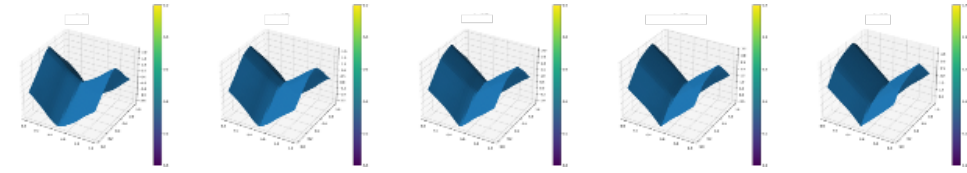


(b) PINNs

FIG. 6. *Surface plot illustrating the evolution of the numerical solutions obtained by the proposed method and PINNs for Example* 4.6 *at times* $t = 0, T/4, T/2, 3T/4, T$*, where the vertical axis represents the solution values over the two-dimensional spatial domain.*



(a) Ours

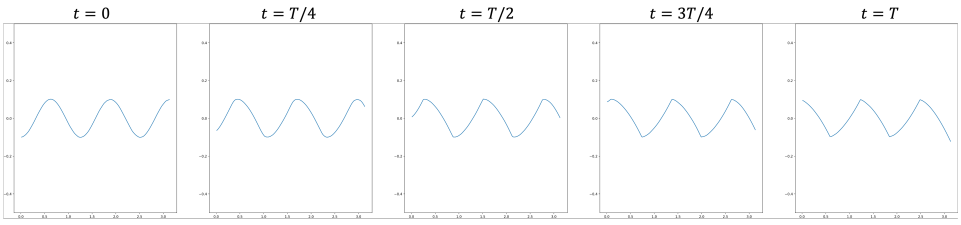


(b) PINNs

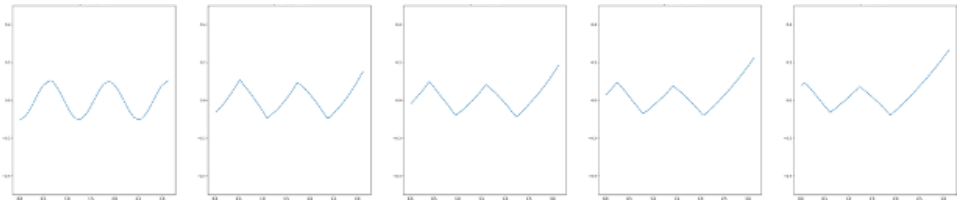
FIG. 7. *Surface plot of the solutions obtained by the proposed method and PINNs for Example* 4.7*, where the vertical axis represents the solution values over the two-dimensional spatial domain.*

captures the initial condition and effectively models the formation of pronounced kinks, consistent with the results reported in [48]. In contrast, despite accurately representing the initial condition, the PINN fails to learn the correct solution dynamics.

EXAMPLE 4.8. *Consider the one-dimensional nonconvex problem with* $H(\nabla u) = u_x^3 - u_x$ *and* $g(\mathbf{x}) = -\frac{1}{10}\cos(5x)$*. The results for the proposed method, in comparison with PINNs, up to time* $T = 0.7$ *are shown in Figure* 8*. For the proposed method, the sinusoidal wave is observed to sharpen progressively over time. This behavior is consistent with the well-established theoretical result that viscosity solutions to HJ equations with polynomial Hamiltonians and sinusoidal initial data evolve into sharp, sawtooth-like profiles while preserving periodicity [50, 26]. In contrast, although the PINN solution also exhibits sharpening, it fails to maintain periodicity, indicating that it learns an incorrect solution structure.*



(a) Ours



(b) PINNs

FIG. 8. *The numerical solution for Example* 4.8 *obtained by the proposed method and PINNs, where horizontal axis represents the one-dimensional spatial domain, while the vertical axis corresponds to the solution values.*

4.3. State-dependent Hamiltonians. This subsection provides experimental validation of the methodology proposed for the state-dependent Hamiltonian in subsection 3.3. It includes error analyses with respect to Δt and addresses various state-dependent Hamiltonians, including a 10-dimensional optimal control problem.

EXAMPLE 4.9. *We evaluate the numerical error and the numerical error and investigate the convergence behavior as the time step* Δt *varies in the following two problems:*

- *Example 4.9.1: One-dimensional variable coefficient linear equation [80] with* $H(x, u_x) = \sin(x)\, u_x$ *and* $g(x) = \sin(x)$ *with periodic boundary condition. The exact solution is expressed by*

$$u^\star(x,t) = \sin\left(2\arctan\left(e^{-t}\tan\left(\frac{x}{2}\right)\right)\right).$$

- *Example 4.9.2: Two-dimensional problem which describes a solid body rotation around the origin [11], where* $H(\mathbf{x}, \nabla u) = -yu_x + xu_y$ *with the periodic boundary condition, and the initial condition is given by*

$$g(x,y) = \begin{cases} 0 & 0.3 \le r, \\ 0.3 - r & 0.1 < r < 0.3 \\ 0.2 & r \le 0.1, \end{cases}$$

where $r = \sqrt{(x-0.4)^2 + (y-0.4)^2}$ *The exact solution is*

$$u^\star(x,y,t) = g(x\cos t + y\sin t, -x\sin t + y\cos t).$$

For both problems, we compute the solution up to $T = 1$*. Since the characteristic curve for the state-dependent Hamiltonian is approximated linearly, the accuracy of the algorithm is influenced by the size of* Δt*. To verify this, we conducted experiments*

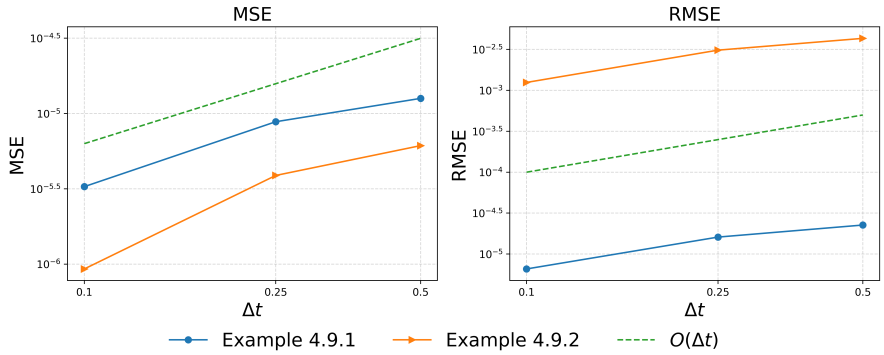
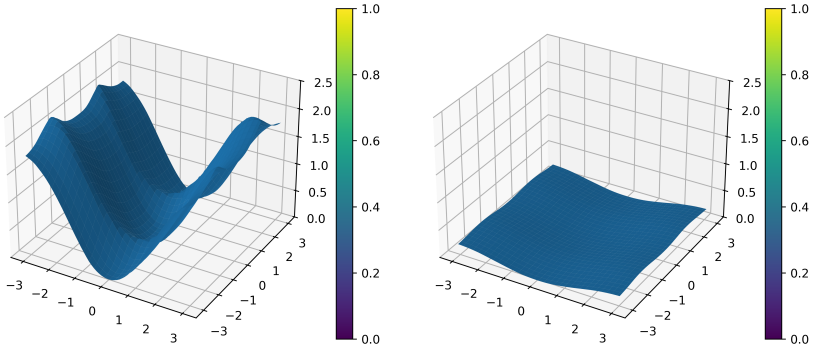


FIG. 9. *Mean squared errors (MSE) and relative mean squared errors (RMSE) between the predicted and exact solutions for Examples 4.9.1 and 4.9.2, plotted as functions of the time step size* Δt.



(a) Ours (b) PINNs

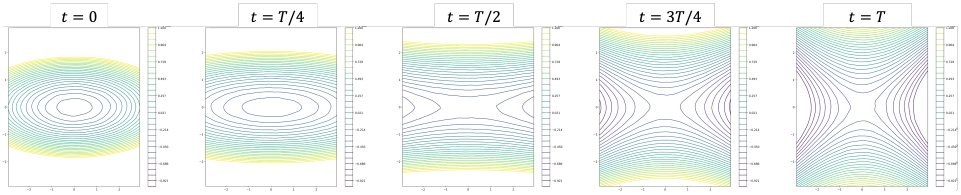
FIG. 10. *Surface plot of the numerical solutions obtained by the proposed method and PINNs at terminal time for Example* 4.10*, where the vertical axis represents the solution values over the two-dimensional spatial domain.*

for various values of $\Delta t = 0.5, 0.25, 0.1$*, and the results are summarized in Figure* 9*. The results show that the linear approximation of the proposed algorithm yields first-order accuracy with respect to* Δt*blue, which is consistent with the use of a first-order temporal discretization scheme.*

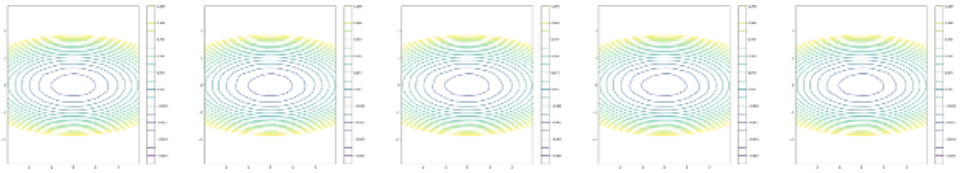
EXAMPLE 4.10. *We solve an optimal control problem related to cost determination [67] with* $H(\mathbf{x}, \nabla u) = u_x \sin y + (\sin y + sign(u_y))\, u_y - \frac{1}{2}\sin^2 y - (1 - \cos x)$ *and* $g(\mathbf{x}) = 0$ *with periodic boundary conditions. The result at* $T = 1$ *is presented in Figure* 10*, along with the PINNs result for comparison. As shown, the proposed method successfully captures the kink in the solution, which is qualitatively consistent with the findings reported in [67]. In contrast, the PINN fails to recover the correct solution behavior.*

EXAMPLE 4.11. *We solve the problem associated with the state-dependent Hamiltonian well-known as the harmonic oscillator:*

$$H^{\pm}(\mathbf{x}, \mathbf{p}) = \pm \frac{1}{2}\left(\|\mathbf{x}\|_2^2 + \|\mathbf{p}\|_2^2\right).$$



(a) Ours



(b) PINNs

FIG. 11. *Evolution of the numerical solutions for Example* 4.11 *with* H^+. *Contour plots over the two-dimensional spatial domain illustrate level sets of the solutions at times* $t = 0, T/4, T/2, 3T/4, T$. *The horizontal and vertical axes correspond to the* x- *and* y-*coordinates, respectively.*

We consider the two-dimensional problem where the initial function is the level set function of an ellipsoid

$$g(x,y) = \frac{1}{2}\left(\frac{x^2}{2.5^2} + y^2 - 1\right). \tag{4.1}$$

The results for H^+ *and* H^- *up to* $T = 0.4$ *are depicted in Figures 11 and 12, respectively. For Hamiltonians with both positive and negative signs, the proposed method successfully captures the corresponding solution behaviors, exhibiting characteristic dispersive and contracting dynamics. In contrast, the PINN fails to learn appropriate solutions in either case, further underscoring the effectiveness of our approach in modeling such dynamic behaviors.*

EXAMPLE 4.12. *We consider a state-dependent nonconvex Hamiltonian of the following form given in [13]:*

$$H(\mathbf{x},\mathbf{p}) = -c(\mathbf{x})\,p_1 + 2\,|p_2| + \|\mathbf{p}\|_2 - 1,$$

where $\mathbf{p} = (p_1, p_2)$ *and*

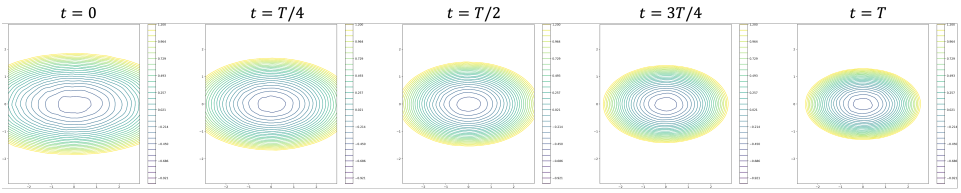
$$c(\mathbf{x}) = 2\left(1 + 3\exp\left(-4\,\|\mathbf{x} - (1,1)\|_2^2\right)\right). \tag{4.2}$$

We employ the initial function as presented in Example 4.11. *The results up to* $T = 1$ *are presented in Figure* 13. *For this problem, both the proposed method and PINN learn similar solutions, which are consistent with those reported in [13].*

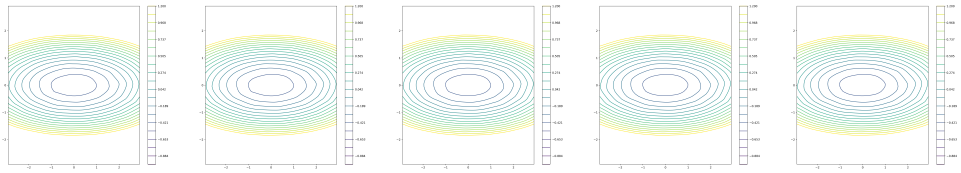
EXAMPLE 4.13. *We test the proposed method for a state-dependent nonconvex Hamiltonian of the following form given in [13]:*

$$H(\mathbf{x},p) = -c(\mathbf{x})\,|p_1| - c(-\mathbf{x})\,|p_2|,$$

where we write $\mathbf{p} = (p_1, p_2)$ *and* $c(\mathbf{x})$ *is a coefficient function as given in* (4.2). *The initial function* g (4.1) *presented in Example* 4.11 *is employed in this instance. The*

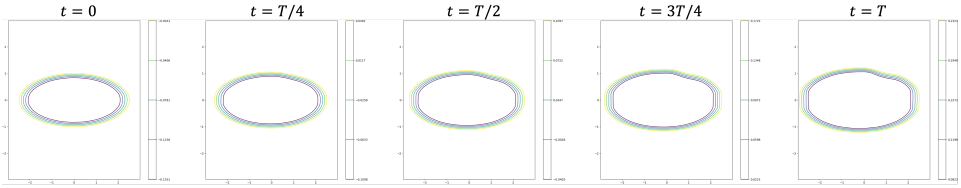


(a) Ours

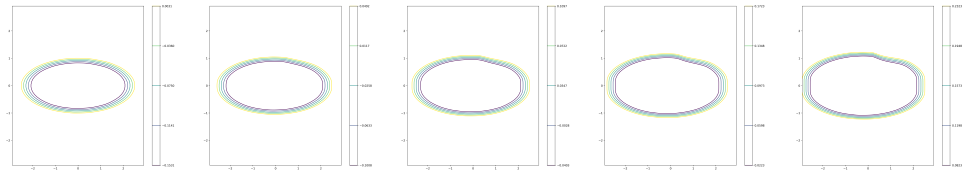


(b) PINNs

FIG. 12. *Evolution of the numerical solutions for Example* 4.11 *with* H^-. *Contour plots over the two-dimensional spatial domain illustrate level sets of the solutions at times* $t = 0, T/4, T/2, 3T/4, T$. *The horizontal and vertical axes correspond to the* x- *and* y-*coordinates, respectively.*



(a) Ours



(b) PINNs

FIG. 13. *Contour plot of the numerical solution obtained by the proposed method and PINNs for Example* 4.12 *over the two-dimensional spatial domain at times* $t = 0, T/4, T/2, 3T/4, T$, *illustrating level sets of the solution. The horizontal and vertical axes represent the* x- *and* y-*coordinates, respectively.*

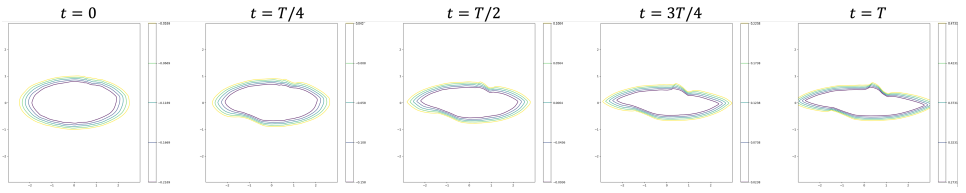
results up to $T = 0.3$ *are presented in Figure* 14, *alongside those obtained by PINNs. The solution attained by the proposed methods is in agreement with those reported in [13], whereas the PINN yields a qualitatively different solution.*

EXAMPLE 4.14. *We solve the following optimal control problem:*

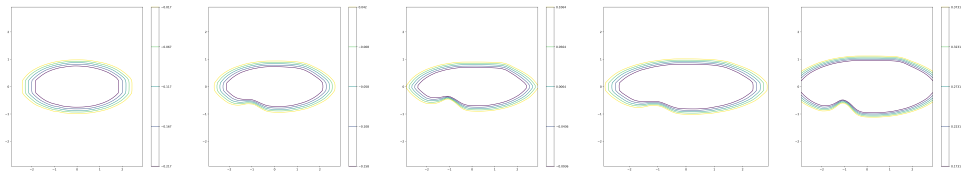
$$u(\mathbf{x}, t) = \inf\Big\{ g(\mathbf{x}(0)) ; \dot{\mathbf{x}}(t) = f(\mathbf{x}(t))\,\mathbf{a}(t),\ \mathbf{x}(t) = \mathbf{x},\ \|\mathbf{a}(t)\|_2 \le 1 \Big\},$$

where g *is defined by*

$$g(\mathbf{x}) = \frac{1}{2}\left(\mathbf{x}^T A \mathbf{x} - 1\right)$$



(a) Ours



(b) PINNs

FIG. 14. *Contour plot of the numerical solution obtained by the proposed method and PINNs for Example* 4.13 *over the two-dimensional spatial domain at times* $t = 0, T/4, T/2, 3T/4, T$*, illustrating level sets of the solution. The horizontal and vertical axes correspond to the* x*- and* y*-coordinates, respectively.*

with $A = diag(0.25, 1)$ *and* f *is given by*

$$f(\mathbf{x}) = 1 + 3\exp\left(-4\left\|\mathbf{x} - (1,1)\right\|_2^2\right).$$

This corresponds to the HJ PDE given in [13]:

$$\begin{cases} u_t + f(\mathbf{x})\left\|\nabla u\right\|_2 = 0 \\ u(\mathbf{x}, 0) = g(\mathbf{x}). \end{cases}$$

When solving the maximization problem $\sup\ g(\mathbf{x}(T))$ *with the same constraints, we obtain the following HJ PDE:*

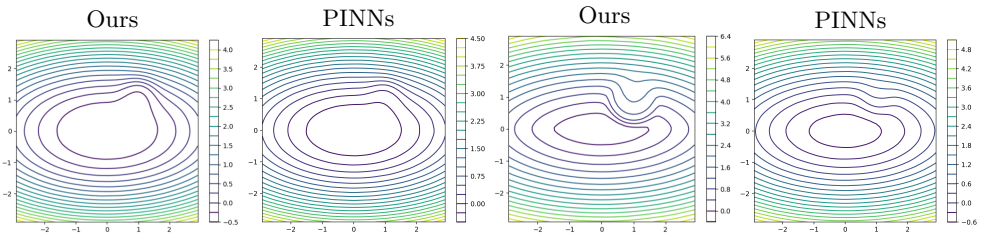
$$\begin{cases} u_t - f(\mathbf{x})\left\|\nabla u\right\|_2 = 0 \\ u(\mathbf{x}, 0) = g(\mathbf{x}). \end{cases}$$

The results for both the minimization (at $T = 0.2$*) and maximization (at* $T = 0.5$*) problems are presented in Figure* 15*, alongside the corresponding results from PINNs. For both problems, the solutions produced by the proposed method are consistent with those reported in [13]. In contrast, while the PINN captures the general trend distinguishing the minimization and maximization cases, it yields overly flattened solutions that fail to accurately represent the correct structure.*

EXAMPLE 4.15. *Consider the following 10-dimensional quadratic optimal control problem presented in [10]:*

$$u(\mathbf{x}, t) = \inf\left\{\int_0^t \left\|\dot{\mathbf{x}}(s)\right\|^2 - \psi(\mathbf{x}(s))\,\mathrm{d}s + g(\mathbf{x}(0))\,;\mathbf{x}(t) = \mathbf{x}\right\},$$

where the potential function $\psi : \mathbb{R}^d \to (-\infty, 0]$ *is* $\psi(\mathbf{x}) = \sum_{i=1}^d \psi_i(\mathbf{x}_i)$*, where each*



(a) Minimization (b) Maximization

FIG. 15. *Numerical solutions at the terminal time computed by the proposed method and PINNs for Example* 4.14 *over the two-dimensional spatial domain. Contour plots illustrate level sets of the solution at the terminal time. The horizontal and vertical axes correspond to the x- and y-coordinates, respectively.*

function $\psi_i : \mathbb{R} \to (-\infty, 0]$ *is a positively 1-homogeneous concave function given by*

$$\psi_i(x) = \begin{cases} -a_i x & x \geq 0, \\ b_i x & x < 0, \end{cases}$$

with parameters $(a_1, \cdots, a_d) = (4, 6, 5, \cdots, 5)$ *and* $(b_1, \cdots, b_d) = (3, 9, 6, \cdots, 6)$*. The corresponding HJ PDE reads:*

$$\begin{cases} u_t + \frac{1}{2} \|\nabla u\|^2 + \psi(\mathbf{x}) = 0 \\ u(\mathbf{x}, 0) = g(\mathbf{x}). \end{cases}$$

We conduct experiments for the two initial cost functions:

- *A quadratic initial function* $g_1(\mathbf{x}) = \frac{1}{2} \|\mathbf{x} - \mathbf{1}\|_2^2$*, where* $\mathbf{1}$ *denotes the d-dimensional vector whose elements are all one.*

- *A nonconvex initial function*

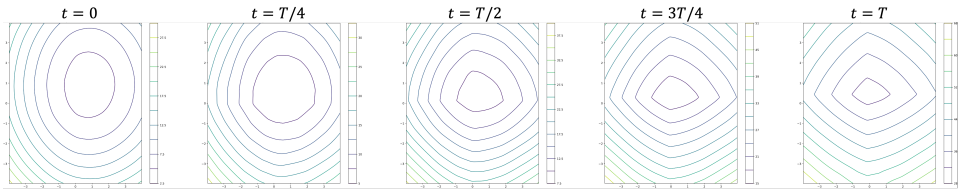
$$g_2(\mathbf{x}) = \min_{j \in \{1,2,3\}} g_j(\mathbf{x}) = \min_{j \in \{1,2,3\}} \frac{1}{2} \|\mathbf{x} - \mathbf{y}_j\|_2^2 - \alpha_j,$$

  where $\mathbf{y}_1 = (-2, 0, \cdots, 0)$*,* $\mathbf{y}_2 = (2, -2, -1, 0, \cdots, 0)$*,* $\mathbf{y}_3 = (0, 2, 0, \cdots, 0)$*,* $\alpha_1 = -0.5$*,* $\alpha_2 = 0$*, and* $\alpha_3 = -1$*.*

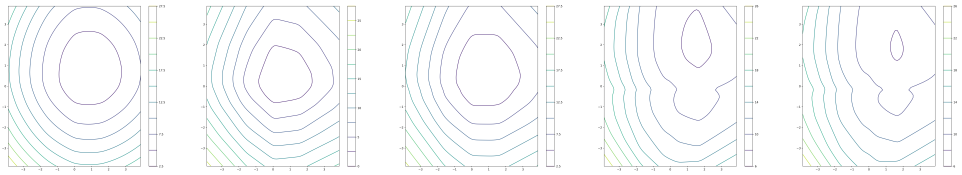
Figures 16 and 17 present two-dimensional slices of the solutions in the xy plane for both cases up to time $T = 0.5$*.*

For both cases, the results obtained by the proposed method are consistent with the experimental findings presented in [10], which demonstrate that the solution exhibits non-trivial dynamics, as evidenced by the nonlinear evolution of the level sets and the formation of multiple kinks over time. In contrast, PINNs fail to recover the correct solution, highlighting the advantage of the proposed method over PINNs in solving high-dimensional optimal control problems.

5. Conclusion. We have introduced a novel implicit solution method for HJ PDEs derived from the characteristics of the PDE. This formula aligns with the Hopf-Lax formula for convex Hamiltonians but simplifies it by removing the need for Legendre transforms, thereby enhancing computational efficiency and broadening its practical applicability. The proposed formula not only bridges the method of characteristics, the Hopf-Lax formula, and Bellman's principle from control theory

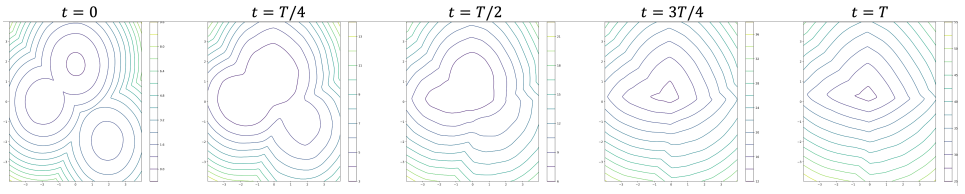


(a) Ours

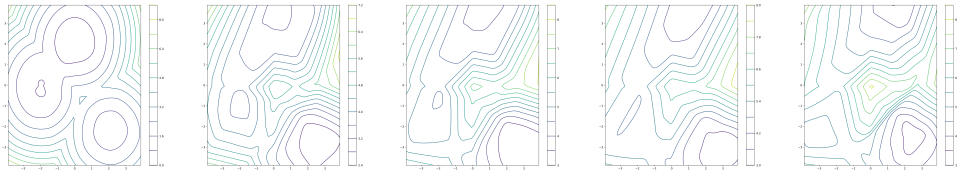


(b) PINNs

FIG. 16. *Evolution of the numerical solutions of the proposed method and PINNs for the 10-dimensional optimal control problem in Example* 4.15 *with the quadratic initial function* g_1. *Contour plots of two-dimensional slices of the solutions in the xy-plane at times* $t = 0, T/4, T/2, 3T/4, T$ *are presented. The horizontal and vertical axes represent the* x- *and* y-*coordinates, respectively.*



(a) Ours



(b) PINNs

FIG. 17. *Evolution of the numerical numerical solutions of the proposed method and PINNs for the 10-dimensional optimal control problem in Example* 4.15 *with the nonconvex initial function* g_2. *Contour plots of two-dimensional slices of the solutions in the xy-plane at times* $t = 0, T/4, T/2, 3T/4, T$ *are presented. The horizontal and vertical axes represent the* x- *and* y-*coordinates, respectively.*

but also offers a simple and effective numerical approach for solving HJ PDEs. By integrating deep learning, the formula provides a scalable method that effectively mitigates the curse of dimensionality. Experimental results demonstrate its robustness and effectiveness across various high-dimensional and nonconvex problems without tuning the configuration of the deep learning model. These findings validate the method as a versatile and computationally efficient tool for solving high-dimensional, nonconvex dynamic systems and optimal control problems governed by HJ PDEs.

An important direction for future work includes a rigorous analysis of the proposed implicit solution formula. While experimental results demonstrate the method's

effectiveness on various nonconvex problems, a comprehensive analysis is needed to confirm whether the proposed formula describes the viscosity solution of HJ PDEs in nonconvex problems. Since the formula involves the first derivatives and is a composite of multiple terms, the proposed minimization problem (3.1) is nonconvex, making the convergence of gradient descent non-trivial. Consequently, a convergence analysis would be an important future endeavor.

Regarding the deep learning approach, we approximate the expectation loss (3.1) using Monte Carlo integration (3.4), which introduces a discrepancy between the empirical and expectation losses. A valuable research direction could involve investigating whether the stochastic gradient descent process, with its random collocation points at each epoch, converges to the global minimum of the expectation loss in the context of stochastic approximation. Although we focused on scalability by maintaining a fixed model configuration across experiments, future research should explore the optimal selection of collocation points and network size for different problem dimensions. Furthermore, the investigation of using automatic differentiation to compute exact derivatives of the network, rather than finite differences such as ENO/WENO, presents an intriguing avenue for future research, particularly in its ability to capture kinks. For state-dependent Hamiltonians, the development of higher-order methods beyond the proposed first-order linear approximation of the characteristic curve would be a promising direction. Finally, the simplicity and efficiency of the proposed method open up avenues for its application to a wide range of problems, including level set evolutions, optimal transport, mean field games, and inverse problems, which would constitute valuable extensions of this work.

Acknowledgments. We would like to express our sincere gratitude to Tingwei Meng for the valuable discussions on the implicit solution formula during the early stages of our manuscript.

Appendix A. Proofs of Theorems.

A.1. Proof of Theorem 2.1.

Proof. The differentiability of the Hamiltonian, assumed in both theorems, is required for the implicit solution formula (2.3).

The remaining assumptions on the Hamiltonian and the initial condition, as stated in Theorems 2.1 and 2.2, are required to ensure the validity of the Hopf–Lax (2.4) and Hopf (A.12) formulas, respectively. First, we can observe that the implicit solution formula (2.3) exactly satisfies the initial condition $u = g$ of (2.1) at the initial time $t = 0$.

Under the assumptions (2.6) on H and the l.s.c. condition on g, the viscosity solution of the HJ PDE is described by the Hopf-Lax formula (2.4)

$$u(\mathbf{x}, t) = tH^*\left(\frac{\mathbf{x} - \mathbf{y}^\star}{t}\right) + g(\mathbf{y}^\star), \tag{A.1}$$

where

$$\mathbf{y}^\star = \operatorname{argmin}_{\mathbf{y}}\left\{tH^*\left(\frac{\mathbf{x} - \mathbf{y}}{t}\right) + g(\mathbf{y})\right\}. \tag{A.2}$$

Given that H is differentiable, differentiating both sides of (A.1) with respect to $\mathbf{x}$ at the point leads to the Euler-Lagrange equation corresponding to the Hopf-Lax formula:

$$\nabla u(\mathbf{x}, t) = \nabla H^*\left(\frac{\mathbf{x} - \mathbf{y}^\star}{t}\right) + \frac{\partial}{\partial \mathbf{y}}\left(H^*\left(\frac{\mathbf{x} - \mathbf{y}}{t}\right) + g(\mathbf{y})\right) \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}\Big|_{\mathbf{y}=\mathbf{y}^\star} \tag{A.3}$$

$$= \nabla H^*\left(\frac{\mathbf{x} - \mathbf{y}^\star}{t}\right), \tag{A.4}$$

where the last equality follows from the fact that $\mathbf{y}^\star$ is the optimizer as in (A.2). Since H is strictly convex (or concave), it satisfies $(\nabla H^*)^{-1} = \nabla H$. Therefore, it follows that

$$(\nabla H^*)^{-1}(\nabla u(\mathbf{x}, t)) = \frac{\mathbf{x} - \mathbf{y}^\star}{t} \tag{A.5}$$

$$\iff \mathbf{y}^\star = \mathbf{x} - t(\nabla H^*)^{-1}(\nabla u(\mathbf{x}, t)) \tag{A.6}$$

$$= \mathbf{x} - t\nabla H(\nabla u(\mathbf{x}, t)) \tag{A.7}$$

By expanding the Legendre transform in the Hopf-Lax formula (2.1), the viscosity solution is expressed as follows

$$u(\mathbf{x}, t) = \inf_{\mathbf{y}} \sup_{\mathbf{z}}\left\{t\left(\mathbf{z}^{\mathrm{T}}\left(\frac{\mathbf{x} - \mathbf{y}}{t}\right) - H(\mathbf{z})\right) + g(\mathbf{y})\right\} \tag{A.8}$$

$$= \inf_{\mathbf{y}} \sup_{\mathbf{z}}\left\{\mathbf{z}^{\mathrm{T}}(\mathbf{x} - \mathbf{y}) - tH(\mathbf{z}) + g(\mathbf{y})\right\}. \tag{A.9}$$

Differentiating (A.9) with respect to $\mathbf{z}$ provides that the optimal $\mathbf{z}^\star$ satisfies

$$\mathbf{x} - \mathbf{y}^\star - t\nabla H(\mathbf{z}^\star) = 0.$$

Together with (A.7), we have

$$\mathbf{z}^\star = \nabla u. \tag{A.10}$$

Substituting these (A.7) and (A.10) into (A.9) results in the implicit solution formula (2.3). ∎

A.2. Proof of Theorem 2.2.

Proof. First, we can observe that the implicit solution formula (2.3) exactly satisfies the initial condition $u = g$ of (2.1) at the initial time $t = 0$.

Under the assumptions on H and g stated in the theorem, the viscosity solution of the HJ PDE is described by the Hopf formula (A.12)

$$u(\mathbf{x}, t) = -\left\{g^*(\mathbf{z}^\star) + tH(\mathbf{z}^\star) - \mathbf{x}^{\mathrm{T}}\mathbf{z}^\star\right\}, \tag{A.11}$$

where

$$\mathbf{z}^\star = \operatorname{argmin}_{\mathbf{z}}\left\{g^*(\mathbf{z}) + tH(\mathbf{z}) - \mathbf{x}^{\mathrm{T}}\mathbf{z}\right\}. \tag{A.12}$$

By expanding the Legendre transform of g in (A.11), we have

$$\begin{aligned} u(\mathbf{x}, t) &= \inf_{\mathbf{y}}\left\{\mathbf{z}^{\star\mathrm{T}}(\mathbf{x} - \mathbf{y}) - tH(\mathbf{z}^\star) + g(\mathbf{y})\right\} \\ &= \mathbf{z}^{\star\mathrm{T}}(\mathbf{x} - \mathbf{y}^\star) - tH(\mathbf{z}^\star) + g(\mathbf{y}^\star). \end{aligned}$$

Differentiating the both side of (A.11) with respect to $\mathbf{x}$ induces

$$\nabla u = -\frac{\partial}{\partial \mathbf{z}}\left\{g^*(\mathbf{z}^\star) + tH(\mathbf{z}^\star)\right\} \cdot \frac{\partial \mathbf{z}^\star}{\partial \mathbf{x}} + \mathbf{z}^\star = \mathbf{z}^\star,$$

where the last equality follows from the optimality of $\mathbf{z}^\star$. Consequently, we have $\mathbf{z}^\star = \nabla u$. Furthermore, differentiating (4) with respect to $\mathbf{z}$ provides that the optimal $\mathbf{y}^\star$ satisfies

$$\mathbf{x} - \mathbf{y}^\star - t\nabla H(\mathbf{z}^\star) = 0,$$

that is,

$$\mathbf{y}^\star = \mathbf{x} - t\nabla H(\mathbf{z}^\star) = \mathbf{x} - t\nabla H(\nabla u),$$

which concludes the proof. ∎

Appendix B. Implementation Details. All derivatives required for the loss computation are evaluated using PyTorch's automatic differentiation. Collocation points are sampled uniformly at random from the spatio-temporal domain using PyTorch's built-in random number generation utilities. For clarity and reproducibility, we include a representative pseudocode in Python illustrating the training loop based on the implicit solution formulation below.

...

```
# Problem setup
dim = 2
T = 1
domain_min = [-1, -1]
domain_max = [1, 1]

# Network initialization
network = MLP(input_dim=dim + 1, layers=[64, 64, 64, 64, 64], activation
    ='softplus')
optimizer = Adam(network.parameters(), lr=1e-3)

for epoch in range(200000):
```

```
    # Sample collocation points uniformly in space-time domain
    pnts = random_sampler(M, domain_min, domain_max, T)  # (M, dim+1)
        with time in pnts[:,0]

    # Forward pass: predict solution at collocation points
    pred = network(pnts)  # (M, 1)

    # Compute spatial gradients of predictions
    grad_pred = gradient(pnts, pred)[:, 1:]  # exclude time dimension

    # Evaluate Hamiltonian and its gradient
    Hamiltonian_pred = H(grad_pred)
    H_diff_grad = H_diff(grad_pred)

    # Compute implicit solution formula loss
    loss = ((pred
             - pnts[:, [0]] * torch.sum(H_diff_grad * grad_pred, dim=1,
                keepdim=True)
             + pnts[:, [0]] * Hamiltonian_pred
             - u0(pnts[:, 1:] - pnts[:, [0]] * H_diff_grad)) ** 2).mean
                ()

    # Backpropagation and optimization step
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

LISTING 1
Training procedure using implicit solution formula for HJ equations.

If boundary conditions are specified, 200 points are uniformly sampled from the boundary of the domain. The corresponding boundary loss is computed using the formulations given in (3.2) and (3.2). This boundary loss is scaled by a factor of 0.1 and added to the loss from the implicit solution formula to form the total loss used for training.

The PINN baseline are trained under the same network architecture and experimental configuration as the proposed method. For the PDE loss, we used 5,000 collocation points, consistent with those used to compute the loss from the implicit representation formula. An additional 200 points are sampled to compute the loss associated with the initial condition. All collocation points are randomly sampled from a uniform distribution at each training epoch. The PDE loss and the initial condition loss are combined with equal weights, both having a regularization coefficient of 1. We observed that both loss terms decreased consistently during training, indicating stable convergence.

For WENO, the spatial and temporal domains are discretized using uniform grids. A fifth-order WENO reconstruction is employed to approximate spatial derivatives, and the Lax-Friedrichs numerical flux is adopted for Hamiltonian evaluation. For time integration, a third-order strong stability-preserving Runge-Kutta (SSP-RK3) method with a CFL number of 0.2 is utilized, effectively balancing accuracy and stability in the presence of nonlinearities.

Adaptive Sampling. Residual-based adaptive sampling selects half of the total M collocation points ($M/2$) uniformly at random from the computational domain, while the remaining $M/2$ points are chosen based on large residual values. Specifically, $5M$ candidate points are first sampled uniformly at random, and from these, the top $M/2$ points with the highest residual values—computed using the implicit solution formula—are selected.

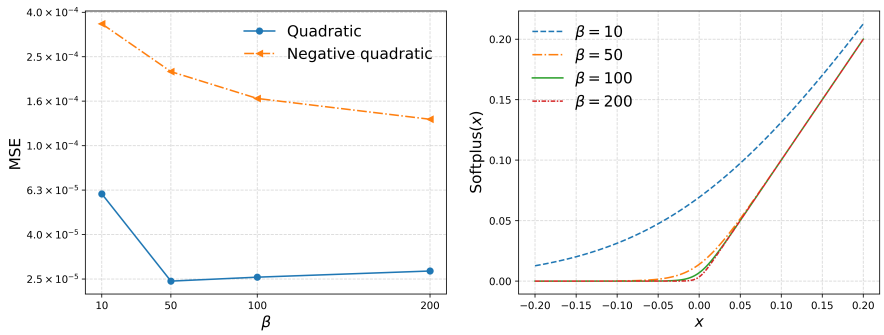


FIG. 18. *Effect of the smoothness parameter β on the SoftPlus activation function. (Left) Experimental results for the 10-dimensional cases of two examples from Example 4.1, demonstrating the impact of different β values. (Right) The SoftPlus activation function plotted for various β values.*

Gradient-based adaptive sampling follows a similar approach: $M/2$ points are drawn uniformly at random, and the remaining $M/2$ are selected based on the magnitude of the solution gradient. To identify high-gradient regions, $5M$ candidate points are uniformly sampled, and the gradient of the solution is evaluated at each point. The $M/2$ points with the largest gradient magnitudes are then selected for training.

Appendix C. Additional Results.

C.1. Influence of Network Architecture. Throughout our experiments, we employed an MLP with the `SoftPlus` activation function to approximate the solution. While the ReLU activation function is widely adopted due to its practical advantages—including the ability to approximate non-smooth functions—it is not suitable for our setting because the loss function involves computing derivatives of the network with respect to its input. To address this, we use the `SoftPlus` activation, a smooth approximation of ReLU that has been widely adopted in the context of implicit neural representations (INRs) [34, 56, 30].

Impact of SoftPlus Smoothness Parameter β. The smoothness of the SoftPlus activation function is controlled by the parameter β, as illustrated in the right subfigure of Figure 18. To examine its effect, we conducted experiments on 10-dimensional instances of two test cases from Example 4.1 using various β values. The results, presented in Figure 18, show that the quadratic example yields a smooth solution, while the negative quadratic case produces a non-smooth solution characterized by a kink. Notably, for the non-smooth problem, increasing β sharpens the activation function, resulting in a more accurate approximation of the solution. This suggests that sharper activations are beneficial for capturing kinks in the solution. In contrast, the smooth quadratic case exhibits minimal sensitivity to changes in β, with only a slight increase in error observed as β increases.

Evaluation of Alternative Network Architectures. To examine the impact of network architecture on performance, we conducted additional experiments using alternative configurations. Specifically, we evaluated an MLP with the `tanh` activation function, as well as the SIREN architecture [76], which incorporates sinusoidal activations. For both models, we preserved the same network structure as in our baseline architecture of width 64 and depth 5, modifying only the activation function. For SIREN, we adopted the initialization scheme proposed in the original paper [76]

TABLE 4

Comparison of network architecture in terms of MSE for problems in Example 4.1. Three different methods—MLP with `SoftPlus` *activation function, MLP with* `tanh` *activation function, SIREN, and Transformer—are evaluated across dimensions* $d = 1, 10$, *and* 40.

	Quadratic Hamiltonian			Negative quadratic Hamiltonian		
Sampling Method	$d = 1$	$d = 10$	$d = 40$	$d = 1$	$d = 10$	$d = 40$
MLP (`SoftPlus`)	1.14E-7	2.56E-5	1.30E-3	8.59E-6	1.63E-4	1.23E-3
MLP (`tanh`)	8.11E-7	1.16E-4	8.82E-2	2.35E-5	9.95E-4	9.99
SIREN	1.53E-5	1.25E-4	3.44E-2	2.89E-5	1.98E-3	11.44
Transformer	9.57E-5	1.43E-4	3.88	1.51E-3	2.89E-3	33.87

to ensure stable training.

Additionally, we implemented a Transformer-based architecture inspired by [85]. This model consists of a single encoder-decoder block with two-headed multi-head self-attention and sinusoidal activation functions. Input coordinates are embedded into a latent space of dimension 32 via a linear layer, followed by encoder and decoder stages. The final output is generated through a feedforward network comprising two hidden layers of width 512, with sinusoidal activations applied throughout.

These architectures are evaluated on the two HJ equations presented in Example 4.1 across spatial dimensions $d = 1, 10$, and 40. The MSE with respect to the exact solutions are summarized in Table 4. As the input dimensionality increases—particularly in the 40-dimensional case—we observe significant performance degradation in MLPs with `tanh` activations, SIREN, and even Transformer architectures. The degradation in the performance of `tanh`-based MLPs and SIREN can be attributed to inherent limitations of their activation functions in high-dimensional settings. The `tanh` activation saturates for large input magnitudes, leading to vanishing gradients and inefficient optimization. Likewise, SIREN's sinusoidal activations become increasingly oscillatory in higher dimensions, resulting in unstable gradient behavior and poor convergence.

In contrast, MLPs with `SoftPlus` activations exhibit improved scalability in high dimensions, due to the smooth and non-saturating nature of the activation, which ensures stable gradient propagation. This can be attributed to the smooth, non-saturating nature of the activation function, which facilitates stable gradient propagation throughout the network. Since the derivative of `SoftPlus` is a sigmoid function that approaches unity for large positive inputs, it helps prevent vanishing gradients during backpropagation. This property is particularly advantageous when optimizing loss functions that involve nested derivatives, as in our case.

Although Transformer architecture offers greater expressivity through its self-attention mechanism, it still suffers in high-dimensional regimes. The self-attention module incurs quadratic computational and memory costs with respect to input size, making it less efficient as dimensionality increases. Furthermore, standard attention may struggle to capture meaningful spatial dependencies in high-dimensional Euclidean spaces without additional structural priors or specialized encodings. Together, these limitations hinder both the expressivity and scalability of the architecture in high-dimensional regimes.

C.2. L^∞ Errors. To further assess the performance of the proposed method, we additionally report the L^∞ norm errors of both the solution and its gradient for the examples in Examples 4.1 and 4.2. These results are summarized in Table 5.

TABLE 5
L^∞ norm errors for problems in Examples 4.1 and 4.2.

Sampling Strategy	Solution Error			Gradient Error		
	$d=1$	$d=10$	$d=40$	$d=1$	$d=10$	$d=40$
Quadratic Hamiltonian	1.47E-3	2.19E-2	6.99E-1	2.02E-2	5.31E-2	4.99E-1
Negative quadratic Hamiltonian	4.28E-2	5.28E-2	1.02E-1	5.00E-1	1.29E0	1.37E0
L^2 Hamiltonian	5.66E-3	6.15E-3	5.13E-2	1.99E-1	1.96E-1	1.27E-1
L^∞ Hamiltonian	2.15E-1	9.88E-2	1.13E-1	1.14E0	9.05E-1	1.13E0

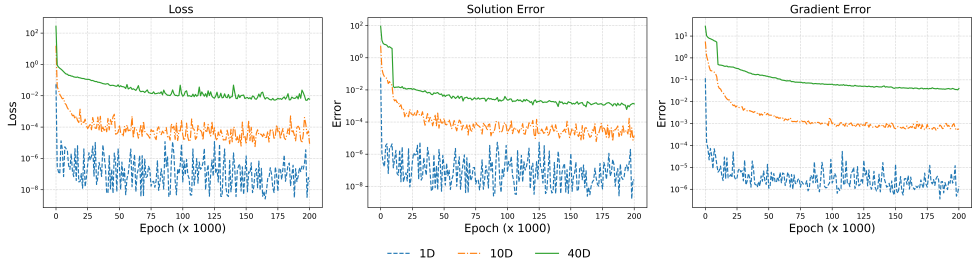
While L^2 norm errors provide a global measure of accuracy, the L^∞ norm captures the worst-case error across the domain, offering a more stringent criterion. This is particularly relevant in the context of HJ equations, where accurately capturing sharp features such as kinks or discontinuities in the gradient is critical.

C.3. Training Dynamics. Figure 19 illustrates the training behavior of the proposed model for the HJ equations presented in Example 4.1, under both quadratic and negative quadratic Hamiltonians. The training loss and the mean squared error (MSE) with respect to the exact solution are plotted on a logarithmic scale over training epochs for three representative dimensions: $d = 1$, $d = 10$, and $d = 40$. The HJ equation is closely related to optimal control problems, where it is important not only to approximate the value function u, but also to accurately recover the optimal control. To this end, we additionally evaluate the MSE of the gradient of the solution ∇u, which determines the control. In the case of the quadratic-type Hamiltonians considered in Example 4.1, where $\nabla H(\nabla u) = \pm \nabla u$, the gradient error directly corresponds to the control error.

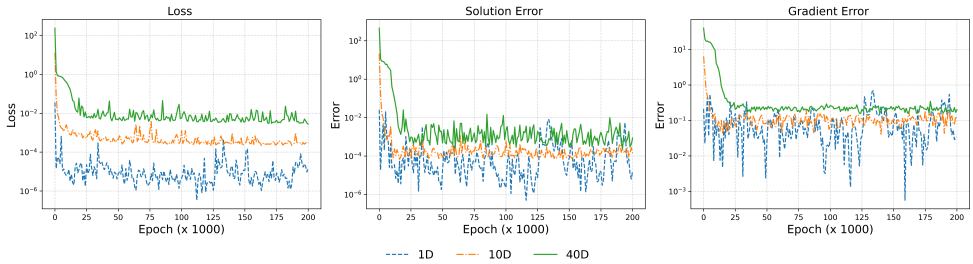
We observe that the training loss consistently decreases over epochs in all cases, indicating stable optimization behavior regardless of the problem dimension. The MSE of the MSE of both the solution and its gradient exhibits a similar convergence trend and closely follows the training loss throughout. Interestingly, it can be observed that the variance in both loss and MSE curves is highest in the 1D case, while it noticeably decreases as the dimensionality increases. This reduction in variance in higher dimensions suggests that training becomes more stable in high-dimensional settings. These results support the conclusion that our approach generalizes well across dimensions and remains robust for solving both convex and concave HJ equations.

C.4. Computational Complexity. To assess the scalability of each method with respect to dimensionality, we measure both the total computational time and peak memory usage required by the proposed method, PINNs, and the classical WENO scheme to compute the solution. For deep learning–based methods, the memory consumption is recorded as the maximum memory usage observed during a single training epoch, which typically corresponds to the backpropagation step. For the WENO scheme, the reported memory usage corresponds to the maximum memory required at each time step iteration. The results for the problems in Example 4.1 are summarized in Table 6.

It can be observed that for INR-based methods—our method and PINNs—neither the computational time nor the memory usage increases drastically with dimension. While both methods demonstrate similar trends, PINNs generally incur slightly higher computational and memory costs. This is primarily due to the need to differentiate the neural network with respect to both spatial and temporal inputs, as well as the necessity to compute separate loss terms corresponding to the initial condition and the



(a) Quadratic Hamiltonian



(b) Negative quadratic Hamiltonian

FIG. 19. *Log-scaled loss and MSE values for both the solution and its gradient are plotted over training epochs for problems in Example 4.1 across dimensions $d = 1, 10, 40$. The results demonstrate convergence of both the training loss and error, highlighting the stability and accuracy of the method across dimensions.*

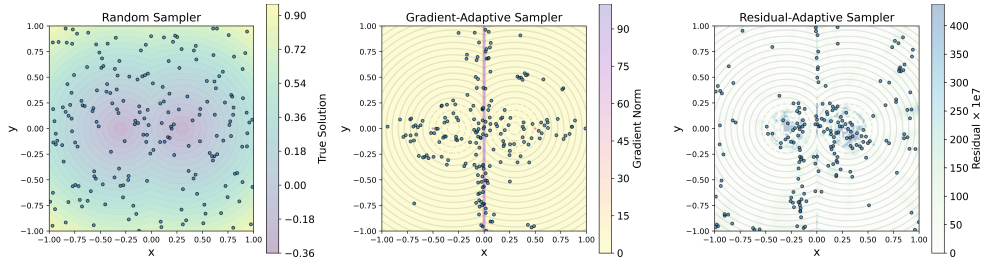
TABLE 6

Computational time (s) and peak memory usage (MB) for solving the quadratic and negative quadratic Hamiltonians in Example 4.1 across different problem dimensions $d = 1, 2, 3, 10, 40$.

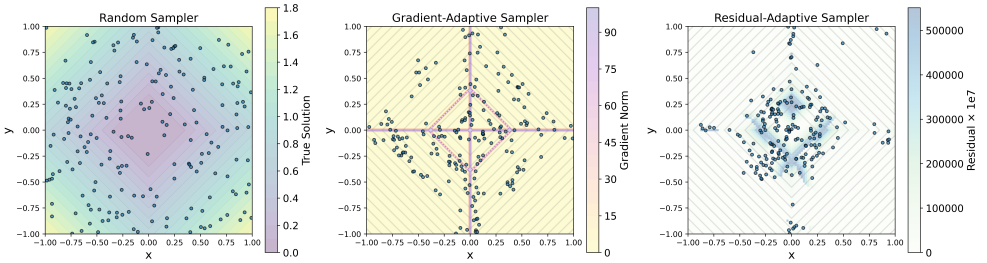
	$d=1$		$d=2$		$d=3$		$d=10$		$d=40$	
Sampling Method	Mem	Time	Mem	Time	Mem	Time	Mem	Time	Mem	Time
Ours	52.86	1162.38	52.92	1231.55	52.98	1238.52	53.39	1365.75	55.13	1431.31
PINNs	52.91	1294.81	52.96	1328.55	52.99	1354.11	53.28	1464.04	54.48	1654.49
WENO (same Mem)	0.09	0.06	0.19	0.07	0.62	0.06	N/A	N/A	N/A	N/A
WENO (same Err):										
Quadratic HJ	6.41	1.02	52017.08	2015.01	N/A	N/A	N/A	N/A	N/A	N/A
Negative Quadratic HJ	–	–	4.94	0.32	813.8234	34.84	N/A	N/A	N/A	N/A

HJ PDE. Due to the need for training, both neural network–based methods generally incur higher computational costs compared to WENO. However, it is important to emphasize that the reported computational times for these methods correspond to learning a continuous solution representation over the entire spatio-temporal domain, rather than computing solutions only at a fixed set of discretized points. Furthermore, in the two-dimensional quadratic HJ case, achieving a level of accuracy comparable to that of the proposed method requires WENO to use significantly more memory and computational time. As discussed in Section 3.1, the memory requirements of INRs are primarily determined by the network size. Consequently, as shown in Table 6, memory usage remains largely independent of dimensionality.

In contrast, the WENO scheme—based on discretizing the spatio-temporal domain on a grid—exhibits rapidly increasing computational cost and memory usage



(a) L^2 Hamiltonian



(b) L^∞ Hamiltonian

FIG. 20. *Comparison of collocation point distributions generated by three sampling strategies for two examples in Example 4.2. (Left) Uniform random sampling distributes points evenly across the domain, shown overlaid on the true solution surface. (Center) Gradient-adaptive sampling concentrates points in regions with large solution gradients (background heatmap), highlighting sharp features such as kinks. (Right) Residual-adaptive sampling focuses points in areas with high residuals of the implicit solution formula (background heatmap), corresponding to regions where the network has not yet fully learned the solution.*

as the dimension grows. In high-dimensional settings (e.g., $d = 10, 40$), the method becomes computationally infeasible. These findings collectively highlight the strong scalability and efficiency of deep learning–based approaches, making them particularly suitable for solving high-dimensional PDEs.

C.5. Visualization of Sampling Distributions. Figure 20 illustrates the spatial distributions of collocation points generated by three different sampling strategies applied to the two examples described in Example 4.2. In each subplot, collocation points are overlaid on a background heatmap representing a characteristic quantity related to the sampling criterion: the true solution for uniform sampling, the gradient norm for gradient-adaptive sampling, and the residual of the implicit solution formula for residual-adaptive sampling.

The figures clearly demonstrate that both adaptive sampling methods concentrate points near kinks and regions where the solution is less regular or the model error remains significant, in contrast to the uniform sampler which distributes points homogeneously. This targeted placement of collocation points effectively enhances the network's ability to resolve challenging features and improves approximation quality in these critical areas.

REFERENCES

[1] M. AKIAN, S. GAUBERT, AND A. LAKHOUA, *The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis*, SIAM

Journal on Control and Optimization, 47 (2008), pp. 817–848.
[2] M. Ansari, A. Khajepour, and E. Esmailzadeh, *Application of level set method to optimal vibration control of plate structures*, Journal of Sound and Vibration, 332 (2013), pp. 687–700.
[3] S. Bansal, A. Bajcsy, E. Ratner, A. D. Dragan, and C. J. Tomlin, *A hamilton-jacobi reachability-based framework for predicting and analyzing human motion for safe planning*, in 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2020, pp. 7149–7155.
[4] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, *Hamilton-jacobi reachability: A brief overview and recent advances*, in 2017 IEEE 56th Annual Conference on Decision and Control (CDC), IEEE, 2017, pp. 2242–2253.
[5] M. Bardi and L. C. Evans, *On hopf's formulas for solutions of hamilton-jacobi equations*, Nonlinear Analysis: Theory, Methods & Applications, 8 (1984), pp. 1373–1381.
[6] S. Bryson and D. Levy, *High-order central weno schemes for multidimensional hamilton-jacobi equations*, SIAM Journal on Numerical Analysis, 41 (2003), pp. 1339–1369.
[7] Y. Cao and N. Wan, *Optimal proportional reinsurance and investment based on hamilton–jacobi–bellman equation*, Insurance: Mathematics and Economics, 45 (2009), pp. 157–162.
[8] V. Caselles, R. Kimmel, and G. Sapiro, *Geodesic active contours*, International journal of computer vision, 22 (1997), pp. 61–79.
[9] H. Chen, R. Wu, E. Grinspun, C. Zheng, and P. Y. Chen, *Implicit neural spatial representations for time-dependent pdes*, in International Conference on Machine Learning, PMLR, 2023, pp. 5162–5177.
[10] P. Chen, J. Darbon, and T. Meng, *Hopf-type representation formulas and efficient algorithms for certain high-dimensional optimal control problems*, Computers & Mathematics with Applications, 161 (2024), pp. 90–120.
[11] Y. Cheng and C.-W. Shu, *A discontinuous galerkin finite element method for directly solving the hamilton–jacobi equations*, Journal of Computational Physics, 223 (2007), pp. 398–415.
[12] Y. T. Chow, J. Darbon, S. Osher, and W. Yin, *Algorithm for overcoming the curse of dimensionality for time-dependent non-convex hamilton–jacobi equations arising from optimal control and differential games problems*, Journal of Scientific Computing, 73 (2017), pp. 617–643.
[13] Y. T. Chow, J. Darbon, S. Osher, and W. Yin, *Algorithm for overcoming the curse of dimensionality for state-dependent hamilton-jacobi equations*, Journal of Computational Physics, 387 (2019), pp. 376–409.
[14] M. G. Crandall and P.-L. Lions, *Viscosity solutions of hamilton-jacobi equations*, Transactions of the American mathematical society, 277 (1983), pp. 1–42.
[15] E. Cristiani and M. Falcone, *Fast semi-lagrangian schemes for the eikonal equation and applications*, SIAM Journal on Numerical Analysis, 45 (2007), pp. 1979–2011.
[16] J. Cui, S. Liu, and H. Zhou, *A supervised learning scheme for computing hamilton-jacobi equation via density coupling*, arXiv preprint arXiv:2401.15954, (2024).
[17] J. Darbon, P. M. Dower, and T. Meng, *Neural network architectures using min-plus algebra for solving certain high-dimensional optimal control problems and hamilton–jacobi pdes*, Mathematics of Control, Signals, and Systems, 35 (2023), pp. 1–44.
[18] J. Darbon, G. P. Langlois, and T. Meng, *Overcoming the curse of dimensionality for some hamilton–jacobi partial differential equations via neural network architectures*, Research in the Mathematical Sciences, 7 (2020), p. 20.
[19] J. Darbon and T. Meng, *On some neural network architectures that can represent viscosity solutions of certain high dimensional hamilton–jacobi partial differential equations*, Journal of Computational Physics, 425 (2021), p. 109907.
[20] J. Darbon and S. Osher, *Algorithms for overcoming the curse of dimensionality for certain hamilton–jacobi equations arising in control theory and elsewhere*, Research in the Mathematical Sciences, 3 (2016), p. 19.
[21] M. de León, J. C. Marrero, and D. M. de Diego, *Linear almost poisson structures and hamilton-jacobi equation. applications to nonholonomic mechanics*, arXiv preprint arXiv:0801.4358, (2008).
[22] D. Delahaye, S. Puechmorel, P. Tsiotras, and E. Féron, *Mathematical models for aircraft trajectory design: A survey*, in Air Traffic Management and Systems: Selected Papers of the 3rd ENRI International Workshop on ATM/CNS (EIWAC2013), Springer, 2014, pp. 205–247.
[23] H. H. Denman and L. H. Buch, *Solution of the hamilton-jacobi equation for certain dissipative classical mechanical systems*, Journal of Mathematical Physics, 14 (1973), pp. 326–329.
[24] S. Dolgov, D. Kalise, and K. K. Kunisch, *Tensor decomposition methods for high-*

dimensional hamilton–jacobi–bellman equations, SIAM Journal on Scientific Computing, 43 (2021), pp. A1625–A1650.
[25] S. Dolgov, D. Kalise, and L. Saluzzi, *Data-driven tensor train gradient cross approximation for hamilton–jacobi–bellman equations*, SIAM Journal on Scientific Computing, 45 (2023), pp. A2153–A2184.
[26] L. C. Evans, *Partial differential equations*, vol. 19, American Mathematical Society, 2022.
[27] L. C. Evans and P. E. Souganidis, *Differential games and representation formulas for solutions of hamilton-jacobi-isaacs equations*, Indiana University mathematics journal, 33 (1984), pp. 773–797.
[28] M. Falcone and R. Ferretti, *Semi-lagrangian schemes for hamilton–jacobi equations, discrete representation formulae and godunov methods*, Journal of computational physics, 175 (2002), pp. 559–575.
[29] M. Falcone and R. Ferretti, *Semi-Lagrangian approximation schemes for linear and Hamilton—Jacobi equations*, SIAM, 2013.
[30] A. Ferrer-Sánchez, J. D. Martín-Guerrero, R. R. de Austri-Bazan, A. Torres-Forné, and J. A. Font, *Gradient-annihilated pinns for solving riemann problems: Application to relativistic hydrodynamics*, Computer Methods in Applied Mechanics and Engineering, 424 (2024), p. 116906.
[31] W. H. Fleming and W. M. McEneaney, *A max-plus-based algorithm for a hamilton–jacobi–bellman equation of nonlinear filtering*, SIAM Journal on Control and Optimization, 38 (2000), pp. 683–710.
[32] P. A. Forsyth and G. Labahn, *Numerical methods for controlled hamilton-jacobi-bellman pdes in finance*, Journal of Computational Finance, 11 (2007), p. 1.
[33] G. Gilboa and S. Osher, *Nonlocal operators with applications to image processing*, Multiscale Modeling & Simulation, 7 (2009), pp. 1005–1028.
[34] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, *Implicit geometric regularization for learning shapes*, International Conference on Machine Learning, 2020.
[35] E. Hopf, *Generalized solutions of non-linear equations of first order*, Journal of Mathematics and Mechanics, 14 (1965), pp. 951–973.
[36] C. Huré, H. Pham, and X. Warin, *Deep backward schemes for high-dimensional nonlinear pdes*, Mathematics of Computation, 89 (2020), pp. 1547–1579.
[37] C. Imbert, R. Monneau, and H. Zidani, *A hamilton-jacobi approach to junction problems and application to traffic flows*, ESAIM: Control, Optimisation and Calculus of Variations, 19 (2013), pp. 129–166.
[38] G.-S. Jiang and D. Peng, *Weighted eno schemes for hamilton–jacobi equations*, SIAM Journal on Scientific computing, 21 (2000), pp. 2126–2143.
[39] D. Kalise, S. Kundu, and K. Kunisch, *Robust feedback control of nonlinear pdes by numerical approximation of high-dimensional hamilton–jacobi–isaacs equations*, SIAM Journal on Applied Dynamical Systems, 19 (2020), pp. 1496–1524.
[40] D. Kalise and K. Kunisch, *Polynomial approximation of high-dimensional hamilton–jacobi–bellman equations and applications to feedback control of semilinear parabolic pdes*, SIAM Journal on Scientific Computing, 40 (2018), pp. A629–A652.
[41] W. Kang and L. Wilcox, *A causality free computational method for hjb equations with application to rigid body satellites*, in AIAA Guidance, Navigation, and Control Conference, 2015, p. 2009.
[42] W. Kang and L. C. Wilcox, *Mitigating the curse of dimensionality: sparse grid characteristics method for optimal feedback control and hjb equations*, Computational Optimization and Applications, 68 (2017), pp. 289–315.
[43] R. L. Karandikar and M. Vidyasagar, *Convergence rates for stochastic approximation: Biased noise with unbounded variance, and applications*, Journal of Optimization Theory and Applications, (2024), pp. 1–39.
[44] K. Khanin and A. Sobolevski, *Particle dynamics inside shocks in hamilton–jacobi equations*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 368 (2010), pp. 1579–1593.
[45] D. P. Kingma, *Adam: A method for stochastic optimization*, The International Conference on Learning Representations, (2014).
[46] S. N. Kruzhkov, *The cauchy problem in the large for certain non-linear first order differential equations*, in Doklady Akademii Nauk, vol. 132, Russian Academy of Sciences, 1960, pp. 36–39.
[47] S. N. Kruzhkov, *Generalized solutions of nonlinear first order equations with several independent variables. ii*, Matematicheskii Sbornik, 114 (1967), pp. 108–134.
[48] F. Lafon and S. Osher, *High order two dimensional nonoscillatory methods for solving*

hamilton–jacobi scalar equations, Journal of Computational Physics, 123 (1996), pp. 235–253.

[49] J. A. Laval and L. Leclercq, *The hamilton–jacobi partial differential equation and the three representations of traffic flow*, Transportation Research Part B: Methodological, 52 (2013), pp. 17–30.

[50] P. D. Lax, *Hyperbolic systems of conservation laws and the mathematical theory of shock waves*, SIAM, 1973.

[51] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, *Robot manipulator control: theory and practice*, CRC Press, 2003.

[52] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, *Fourier neural operator for parametric partial differential equations*, arXiv preprint arXiv:2010.08895, (2020).

[53] F. Lin and R. D. Brandt, *An optimal control approach to robust control of robot manipulators*, IEEE Transactions on robotics and automation, 14 (1998), pp. 69–77.

[54] Z. Lin, J. Duan, S. E. Li, H. Ma, J. Li, J. Chen, B. Cheng, and J. Ma, *Policy-iteration-based finite-horizon approximate dynamic programming for continuous-time nonlinear optimal control*, IEEE Transactions on Neural Networks and Learning Systems, 34 (2022), pp. 5255–5267.

[55] P.-L. Lions, *Generalized solutions of hamilton-jacobi equations*, (No Title), (1982).

[56] Y. Lipman, *Phase transitions, distance functions, and implicit neural representations*, in International Conference on Machine Learning, 2021.

[57] L. Lu, P. Jin, and G. E. Karniadakis, *Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators*, arXiv preprint arXiv:1910.03193, (2019).

[58] W. M. McEneaney, *Max-plus methods for nonlinear control and estimation*, vol. 2, Springer Science & Business Media, 2006.

[59] J. C. Miñano, P. Benítez, and A. Santamaría, *Hamilton-jacobi equation in momentum space*, Optics Express, 14 (2006), pp. 9083–9092.

[60] I. Mitchell, A. Bayen, and C. J. Tomlin, *Computing reachable sets for continuous dynamic games using level set methods*, Submitted January, (2004).

[61] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, *Adaptive deep learning for high-dimensional hamilton–jacobi–bellman equations*, SIAM Journal on Scientific Computing, 43 (2021), pp. A1221–A1247.

[62] N. Nüsken and L. Richter, *Solving high-dimensional hamilton–jacobi–bellman pdes using neural networks: perspectives from the theory of controlled diffusions and measures on path space*, Partial differential equations and applications, 2 (2021), p. 48.

[63] S. Osher, *A level set formulation for the solution of the dirichlet problem for hamilton–jacobi equations*, SIAM Journal on Mathematical Analysis, 24 (1993), pp. 1145–1152.

[64] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces*, Appl. Mech. Rev., 57 (2004), pp. B15–B15.

[65] S. Osher and N. Paragios, *Geometric level set methods in imaging, vision, and graphics*, Springer Science & Business Media, 2007.

[66] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations*, Journal of computational physics, 79 (1988), pp. 12–49.

[67] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for hamilton–jacobi equations*, SIAM Journal on numerical analysis, 28 (1991), pp. 907–922.

[68] C. Parzani and S. Puechmorel, *On a hamilton-jacobi-bellman approach for coordinated optimal aircraft trajectories planning*, Optimal Control Applications and Methods, 39 (2018), pp. 933–948.

[69] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic differentiation in pytorch*, (2017).

[70] M. Pereira, Z. Wang, T. Chen, E. Reed, and E. Theodorou, *Feynman-kac neural network architectures for stochastic control using second-order fbsde theory*, in Learning for Dynamics and Control, PMLR, 2020, pp. 728–738.

[71] J. Qiu and C.-W. Shu, *Hermite weno schemes for hamilton–jacobi equations*, Journal of Computational Physics, 204 (2005), pp. 82–99.

[72] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational physics, 378 (2019), pp. 686–707.

[73] S. Rajeev, *A hamilton–jacobi formalism for thermodynamics*, Annals of Physics, 323 (2008), pp. 2265–2285.

[74] A. Sideris and J. E. Bobrow, *An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems*, in Proceedings of the 2005, American Control Conference, 2005., IEEE, 2005, pp. 2275–2280.
[75] J. Sirignano and K. Spiliopoulos, *Dgm: A deep learning algorithm for solving partial differential equations*, Journal of computational physics, 375 (2018), pp. 1339–1364.
[76] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, *Implicit neural representations with periodic activation functions*, Advances in neural information processing systems, 33 (2020), pp. 7462–7473.
[77] A. I. Subbotin, *Generalized solutions of first order PDEs: the dynamical optimization perspective*, Springer Science & Business Media, 2013.
[78] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, *Neural geometric level of detail: Real-time rendering with implicit 3d shapes*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 11358–11367.
[79] S. Wang, X. Yu, and P. Perdikaris, *When and why pinns fail to train: A neural tangent kernel perspective*, Journal of Computational Physics, 449 (2022), p. 110768.
[80] J. Yan and S. Osher, *A local discontinuous galerkin method for directly solving hamilton–jacobi equations*, Journal of Computational Physics, 230 (2011), pp. 232–244.
[81] L. Yang, S. Liu, T. Meng, and S. J. Osher, *In-context operator learning with data prompts for differential equation problems*, Proceedings of the National Academy of Sciences, 120 (2023), p. e2310142120.
[82] I. Yegorov and P. M. Dower, *Perspectives on characteristics based curse-of-dimensionality-free numerical approaches for solving hamilton–jacobi equations*, Applied Mathematics & Optimization, 83 (2021), pp. 1–49.
[83] B. Yu et al., *The deep ritz method: a deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics, 6 (2018), pp. 1–12.
[84] X. Zhang, T. Cheng, and L. Ju, *Implicit form neural network for learning scalar hyperbolic conservation laws*, in Mathematical and Scientific Machine Learning, PMLR, 2022, pp. 1082–1098.
[85] Z. Zhao, X. Ding, and B. A. Prakash, *Pinnsformer: A transformer-based framework for physics-informed neural networks*, The International Conference on Learning Representations, (2024).
[86] M. Zhou, J. Han, and J. Lu, *Actor-critic method for high dimensional static hamilton–jacobi–bellman partial differential equations based on neural networks*, SIAM Journal on Scientific Computing, 43 (2021), pp. A4043–A4066.