Journal of Computational and Applied Mathematics 433 (2023) 115298

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Least-squares neural network (LSNN) method for scalar nonlinear hyperbolic conservation laws: Discrete divergence operator[☆]

Zhiqiang Cai [a,*], Jingshuang Chen [a], Min Liu [b]

[a] Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907-2067, United States of America
[b] School of Mechanical Engineering, Purdue University, 585 Purdue Mall, West Lafayette, IN 47907-2088, United States of America

A R T I C L E   I N F O

Article history:
Received 14 February 2023
Received in revised form 20 April 2023

Keywords:
Discrete divergence operator
Least-squares method
ReLU neural network
Scalar nonlinear hyperbolic conservation law

A B S T R A C T

A least-squares neural network (LSNN) method was introduced for solving scalar linear and nonlinear hyperbolic conservation laws (HCLs) in Cai et al. (2021, 2022). This method is based on an equivalent least-squares (LS) formulation and uses ReLU neural network as approximating functions, making it ideal for approximating discontinuous functions with unknown interface location. In the design of the LSNN method for HCLs, the numerical approximation of differential operators is a critical factor, and standard numerical or automatic differentiation along coordinate directions can often lead to a failed NN-based method. To overcome this challenge, this paper rewrites HCLs in their divergence form of space and time and introduces a new discrete divergence operator. As a result, the proposed LSNN method is free of penalization of artificial viscosity.

Theoretically, the accuracy of the discrete divergence operator is estimated even for discontinuous solutions. Numerically, the LSNN method with the new discrete divergence operator was tested for several benchmark problems with both convex and non-convex fluxes, and was able to compute the correct physical solution for problems with rarefaction, shock or compound waves. The method is capable of capturing the shock of the underlying problem without oscillation or smearing, even without any penalization of the entropy condition, total variation, and/or artificial viscosity.

1. Introduction

Numerically approximating solutions of nonlinear hyperbolic conservation laws (HCLs) is a computationally challenging task. This is partly due to the discontinuous nature of HCL solutions at unknown locations, which makes approximation using fixed, quasi-uniform meshes very difficult. Over the past five decades, many advanced numerical methods have been developed to address this issue, including higher order finite volume/difference methods using limiters, filters, ENO/WENO, etc.(e.g., [1–7]) and discontinuous and/or adaptive finite element methods (e.g., [8–14]).

Neural networks (NNs) as a new class of approximating functions have been used recently for solving partial differential equations (see, e.g., [15–17]) due to their versatile expressive power. One of the unique features of NNs is their ability to generate moving meshes implicitly by neurons that can automatically adapt to the target function and the solution of a PDE, which helps overcome the limitations of traditional approximation methods that use fixed meshes. For example,

[☆] This work was supported in part by the National Science Foundation, USA under grant DMS-2110571.
[*] Corresponding author.
E-mail addresses: caiz@purdue.edu (Z. Cai), chen2042@purdue.edu (J. Chen), liu66@purdue.edu (M. Liu).

https://doi.org/10.1016/j.cam.2023.115298
0377-0427/© 2023 Elsevier B.V. All rights reserved.

a ReLU NN generates continuous piece-wise linear functions with irregular and free/moving meshes. This property of ReLU NNs was used in [18] for solving linear advection–reaction problem with discontinuous solution, without requiring information about the location of discontinuous interfaces. Specifically, the least-squares NN method studied in [18] is based on the least-squares formulation in [19,20], and it uses ReLU NNs as the approximating functions while approximating the differential operator by directional numerical differentiation. Compared to various adaptive mesh refinement (AMR) methods that locate discontinuous interfaces through an adaptive mesh refinement process, the LSNN method is significant more efficient in terms of the number of degrees of freedom (DoF) used.

Solutions to nonlinear hyperbolic conservation laws are often discontinuous due to shock formation. It is well-known that the differential form of a HCL is not valid at shock waves, where the solution is discontinuous. As a result, the directional numerical differentiation of the differential operator based on the differential form used in [18] cannot be applied to nonlinear HCLs. To overcome this challenge, the integral form of HCLs (as seen in [7]) must be used, which is valid for problems with discontinuous solutions, particularly at the discontinuous interfaces. This is why the integral form forms the basis of many conservative methods such as Roe's scheme [21], WENO [2,3], etc.

Approximating the divergence operator by making use of the Roe and ENO fluxes, in [22] we tested the resulting LSNN method for scalar nonlinear HCLs. Numerical results for the inviscid Burgers equation showed that the LSNN method with conservative numerical differentiation is capable of capturing the shock without smearing and oscillation. Additionally, the LSNN method has fewer DoF than traditional mesh-based methods. Despite the promising results in [22], limitations were observed with the LSNN method when using conservative numerical differentiation of the Roe and second-order ENO fluxes. For example, the resulting LSNN method is not accurate for complicated initial condition, and has problems with rarefaction waves and non-convex spatial fluxes. To improve accuracy, using "higher order" conservative methods such as ENO or WENO could be considered. However, these conservative schemes are designed for traditional mesh-based methods and the "higher order" here is measured at where solutions are smooth.

In this paper, a new discrete divergence operator is proposed to accurately approximate the divergence of a vector field even in the presence of discontinuities. This operator is defined based on its physical meaning: the rate of net outward flux per unit volume, and is approximated through surface integrals by the *composite* mid-point/trapezoidal numerical integration. Theoretically, the accuracy of the discrete divergence operator can be improved by increasing the number of surface integration points (as shown in Lemma 4.3 and Remark 4.4). The LSNN method, being a "mesh/point-free" space–time method, allows the use of all points on the boundary surfaces of a control volume for numerical integration.

Theoretically, we show that the residual of the LSNN approximation using the newly developed discrete divergence operator is bounded by the best approximation of the class of NN functions in some measure as stated in Lemma 3.1 plus the approximation error from numerical integration and differentiation (Lemma 3.3). Numerically, our results show that the LSNN method with the new discrete divergence operator can accurately solve the inviscid Burgers equation with various initial conditions, compute the viscosity vanishing solution, capture shock without oscillation or smearing, and is much more accurate than the LSNN method in [22]. Note that the LSNN method does not use flux limiters. Moreover, the LSNN method using new discrete divergence operator works well for problems with non-convex flux and accurately simulates compound waves.

Recently, several NN-based numerical methods have been introduced for solving scalar nonlinear hyperbolic conservation laws by various researchers [16,18,22–26]. Those methods can be categorized as the physics informed neural network (PINN) [16,23,25,26] and the least-squares neural network (LSNN) [15,18,22,24] methods. First, both methods are based on the least-squares principle, but the PINN uses the discrete l^2 norm and the LSNN uses the continuous Sobolev norm depending on the underlying problem. Second, the differential operator of the underlying problem is approximated by either automatic differentiation or standard finite difference quotient for the PINN and by specially designed discrete differential operator for the LSNN. For example, the LSNN uses discrete directional differential operator in [18] for linear advection–reaction problems, and various traditional conservative schemes in [22] or discrete divergence operator in this paper (see [24] for its first version) for nonlinear scalar hyperbolic conservation laws.

The original PINN has limitations that have been addressed in several studies (see, e.g., [25,26]). For nonlinear scalar hyperbolic conservation laws, [25] found that the PINN fails to provide reasonable approximate solution of the PDE and modified the loss function by penalizing the artificial viscosity term. [26] applied the discrete l^2 norm to the boundary integral equations over control volumes instead of the differential equations over points and modified the loss function by penalizing the entropy, total variation, and/or artificial viscosity. Even though the least-squares principle permits freedom of various penalizations, choosing proper penalization constants can be challenging in practice and it affects the accuracy, efficiency, and stability of the method. In contrast, the LSNN does not require any penalization constants.

The paper is organized as follows. Section 2 describes the hyperbolic conservation law, its least-squares formulation, and preliminaries. The space–time LSNN method and its block version are presented in Section 3. The discrete divergence operator and its error bound is introduced and analyzed in Section 4. Finally, numerical results for various benchmark test problems are given in Section 5.

2. Problem formulation

Let $\tilde{\Omega}$ be a bounded domain in $\mathbb{R}^d$ ($d = 1, 2$, or 3) with Lipschitz boundary, and $I = (0, T)$ be the temporal interval. Consider the scalar nonlinear hyperbolic conservation law

$$\left\{\begin{array}{rcll} u_t(\mathbf{x},t) + \nabla_{\mathbf{x}}\cdot \tilde{\mathbf{f}}(u) & = & 0, & \text{in } \tilde{\Omega}\times I, \\ u & = & \tilde{g}, & \text{on } \tilde{\Gamma}_-, \\ u(\mathbf{x},0) & = & u_0(\mathbf{x}), & \text{in } \tilde{\Omega}, \end{array}\right. \tag{2.1}$$

where u_t is the partial derivative of u with respect to the temporal variable t; $\nabla_{\mathbf{x}}\cdot$ is a divergence operator with respect to the spatial variable $\mathbf{x}$; $\tilde{\mathbf{f}}(u) = (f_1(u), \dots, f_d(u))$ is the spatial flux vector field; $\tilde{\Gamma}_-$ is the part of the boundary $\partial\tilde{\Omega}\times I$ where the characteristic curves enter the domain $\tilde{\Omega}\times I$; and the boundary data $\tilde{g}$ and the initial data u_0 are given scalar-valued functions. Without loss of generality, assume that $f_i(u)$ is twice differentiable for $i = 1, \dots, d$.

Problem (2.1) is a hyperbolic partial differential equation defined on a space–time domain $\Omega = \tilde{\Omega}\times I$ in $\mathbb{R}^{d+1}$. Denote the inflow boundary of the domain Ω and the inflow boundary condition by

$$\Gamma_- = \left\{\begin{array}{ll} \tilde{\Gamma}_-, & t\in(0,T), \\ \Omega, & t = 0 \end{array}\right. \quad \text{and} \quad g = \left\{\begin{array}{ll} \tilde{g}, & \text{on } \tilde{\Gamma}_-, \\ u_0(\mathbf{x}), & \text{on } \Omega, \end{array}\right.$$

respectively. Then (2.1) may be rewritten as the following compact form

$$\left\{\begin{array}{rcll} \mathbf{div}\,\mathbf{f}(u) & = & 0, & \text{in } \Omega\in\mathbb{R}^{d+1}, \\ u & = & g, & \text{on } \Gamma_-, \end{array}\right. \tag{2.2}$$

where $\mathbf{div} = (\partial_{x_1}, \dots, \partial_{x_d}, \partial_t)$ is a divergence operator with respect to both spatial and temporal variables $\mathbf{z} = (\mathbf{x}, t)$, and $\mathbf{f}(u) = (f_1(u), \dots, f_d(u), u) = (\tilde{\mathbf{f}}(u), u)$ is the spatial and temporal flux vector field. Assume that $u \in L^\infty(\Omega)$. Then u is called a weak solution of (2.2) if and only if

$$-(\mathbf{f}(u), \nabla\varphi)_{0,\Omega} + (\mathbf{n}\cdot\mathbf{f}(u), \varphi)_{0,\Gamma_-} = 0, \quad \forall\, \varphi\in C^1_{\Gamma_+}(\bar{\omega}), \tag{2.3}$$

where $\Gamma_+ = \partial\Omega\setminus\Gamma_-$ is the outflow boundary and $C^1_{\Gamma_+}(\bar{\omega}) = \{\varphi\in C^1(\bar{\omega}) : \varphi = 0 \text{ on } \Gamma_+\}$.

Denote the collection of square integrable vector fields whose divergence is also square integrable by

$$H(\text{div};\Omega) = \left\{\boldsymbol{\tau}\in L^2(\Omega)^{d+1} \,|\, \mathbf{div}\,\boldsymbol{\tau}\in L^2(\Omega)\right\}.$$

It is then easy to see that solutions of (2.2) are in the following subset of $L^2(\Omega)$

$$\mathcal{V}_{\mathbf{f}} = \left\{v\in L^2(\Omega) \,|\, \mathbf{f}(v)\in H(\text{div};\Omega)\right\}. \tag{2.4}$$

Define the least-squares (LS) functional

$$\mathcal{L}(v; g) = \|\mathbf{div}\,\mathbf{f}(v)\|^2_{0,\Omega} + \|v - g\|^2_{0,\Gamma_-}, \tag{2.5}$$

where $\|\cdot\|_{0,S}$ denotes the standard $L^2(S)$ norm for $S = \Omega$ and Γ_-. Now, the corresponding least-squares formulation is to seek $u\in V_{\mathbf{f}}$ such that

$$\mathcal{L}(u; g) = \min_{v\in V_{\mathbf{f}}} \mathcal{L}(v; g). \tag{2.6}$$

Proposition 2.1. *Assume that $u\in L^\infty(\Omega)$ is a piece-wise C^1 function. Then u is a weak solution of (2.2) if and only if u is a solution of the minimization problem in (2.6).*

Proof. The proposition is a direct consequence of Theorem 2.5 in [27]. □

3. Least-squares neural network method

Based on the least-squares formulation in (2.6), in this section we first describe the least-squares neural network (LSNN) method for the scalar nonlinear hyperbolic conservation law and then estimate upper bound of the LSNN approximation.

To this end, denote a scalar-valued function generated by a l-layer fully connected neural network by

$$\mathcal{N}(\mathbf{z}) = \boldsymbol{\omega}^{(l)}\left(N^{(l-1)}\circ\cdots\circ N^{(2)}\circ N^{(1)}(\mathbf{z})\right) - b^{(l)} : \ \mathbf{z} = (\mathbf{x}, t)\in\mathbb{R}^{d+1} \longrightarrow \mathbb{R}, \tag{3.1}$$

where $\boldsymbol{\omega}^{(l)}\in\mathbb{R}^{n_{l-1}}$, $b^{(l)}\in\mathbb{R}$, and the symbol $\circ$ denotes the composition of functions. For $k = 1, \dots, l-1$, the $N^{(k)} : \mathbb{R}^{n_{k-1}}\to\mathbb{R}^{n_k}$ is called the kth hidden layer of the network defined as follows:

$$N^{(k)}(\mathbf{z}^{(k-1)}) = \tau(\boldsymbol{\omega}^{(k)}\mathbf{z}^{(k-1)} - \mathbf{b}^{(k)}) \quad \text{for } \mathbf{z}^{(k-1)}\in\mathbb{R}^{n_{k-1}}, \tag{3.2}$$

where $\boldsymbol{\omega}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$, $\mathbf{b}^{(k)} \in \mathbb{R}^{n_k}$, $\mathbf{z}^{(0)} = \mathbf{z}$, and $\tau(s)$ is the activation function whose application to a vector is defined component-wisely. In this paper, we will use the rectified linear unit (ReLU) activation function given by

$$\tau(s) = \max\{0, s\} = \begin{cases} 0, & \text{if } s \le 0, \\ s, & \text{if } s > 0. \end{cases} \tag{3.3}$$

As shown in [18], the ReLU is a desired activation function for approximating discontinuous solution.

Denote the set of neural network functions by

$$\mathcal{M}_N = \mathcal{M}_N(l) = \left\{\mathcal{N}(\mathbf{z}) \text{ defined in (3.1)}: \ \boldsymbol{\omega}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}, \ \mathbf{b}^{(k)} \in \mathbb{R}^{n_k} \text{ for } k = 1, \ldots, l\right\},$$

where the subscript N denotes the total number of parameters $\boldsymbol{\theta} = \left\{\boldsymbol{\omega}^{(k)}, \mathbf{b}^{(k)}\right\}$ given by

$$N = M_d(l) = \sum_{k=1}^{l} n_k \times (n_{k-1} + 1).$$

Obviously, the continuity of the activation function $\tau(s)$ implies that $\mathcal{M}_N$ is a subset of $C^0(\Omega)$. Together with the smoothness assumption on spatial flux $\tilde{\mathbf{f}}(u)$, it is easy to see that $\mathcal{M}_N$ is also a subset of $\mathcal{V}_{\mathbf{f}}$ defined in (2.4).

Since $\mathcal{M}_N$ is not a linear subspace, it is then natural to discretize the HCL using a least-squares minimization formulation. Before defining the computationally feasible least-squares neural network (LSNN) method, let us first consider an intermediate least-squares neural network approximation: finding $u^{N}(\mathbf{z}; \boldsymbol{\theta}^*) \in \mathcal{M}_N$ such that

$$\mathcal{L}\big(u^{N}(\cdot; \boldsymbol{\theta}^*); g\big) = \min_{v \in \mathcal{M}_N} \mathcal{L}\big(v(\cdot; \boldsymbol{\theta}); g\big) = \min_{\boldsymbol{\theta} \in \mathbb{R}^N} \mathcal{L}\big(v(\cdot; \boldsymbol{\theta}); g\big). \tag{3.4}$$

Lemma 3.1. *Let u be the solution of (2.2), and let $u^{N} \in \mathcal{M}_N$ be a solution of (3.4). Assume that $\mathbf{f}$ is twice differentiable, then there exists a positive constant C such that*

$$\begin{aligned} \mathcal{L}\big(u^{N}; g\big) &= \inf_{v \in \mathcal{M}_N} \left(\|v - u\|_{0,\Gamma_-}^2 + \left\|\mathbf{div}\ [\mathbf{f}(v) - \mathbf{f}(u)]\ \right\|_{0,\Omega}^2\right) \\ &\le C \inf_{v \in \mathcal{M}_N} \left(\|v - u\|_{0,\Gamma_-}^2 + \left\|\mathbf{div}\ \left[\mathbf{f}'(u)(v - u)\right]\ \right\|_{0,\Omega}^2\right) + h.o.t., \end{aligned} \tag{3.5}$$

where h.o.t. means a higher order term comparing to the first term.

Proof. For any $v \in \mathcal{M}_N$, (2.2) and (3.4) imply that

$$\mathcal{L}\big(u^{N}; g\big) \le \mathcal{L}\big(v; g\big) = \|v - u\|_{0,\Gamma_-}^2 + \left\|\mathbf{div}\ [\mathbf{f}(v) - \mathbf{f}(u)]\ \right\|_{0,\Omega}^2,$$

which proves the validity of the equality in (3.5). By the Taylor expansion, there exists $\{w_i\}_{i=1}^d$ between u and v such that

$$\mathbf{f}(v) - \mathbf{f}(u) = \mathbf{f}'(u)(v - u) + \frac{1}{2}\mathbf{f}''(w)(v - u)^2,$$

where $\mathbf{f}'(u) = (f_1'(u), \ldots, f_d'(u), 1)^t$ and $\mathbf{f}''(w) = (f_1''(w_1), \ldots, f_d''(w_d), 0)^t$. Together with the triangle inequality we have

$$\left\|\mathbf{div}\ [\mathbf{f}(v) - \mathbf{f}(u)]\ \right\|_{0,\Omega} \le \left\|\mathbf{div}\ \left[\mathbf{f}'(u)(v - u)\right]\ \right\|_{0,\Omega} + \frac{1}{2}\ \left\|\mathbf{div}\ \left[\mathbf{f}''(w)(v - u)^2\right]\right\|_{0,\Omega}. \tag{3.6}$$

Notice that the second term in the right-hand side of (3.6) is a higher order term comparing to the first term. Now, the inequality in (3.5) is a direct consequence of the equality in (3.5) and (3.6). This completes the proof of the lemma. □

Remark 3.2. When u is sufficiently smooth, the second term

$$\mathbf{div}\ \left[\mathbf{f}'(u)(v - u)\right] = (v - u)\,\mathbf{div}\,\mathbf{f}'(u) + \mathbf{f}'(u)\cdot\nabla(v - u)$$

may be bounded by the sum of the L^2 norms of $v - u$ and the directional derivative of $v - u$ along the direction $\mathbf{f}'(u)$.

Evaluation of the least-squares functional $\mathcal{L}\big(v;\ g\big)$ defined in (2.5) requires integration and differentiation over the computational domain and the inflow boundary. As in [15], we evaluate the integral of the least-squares functional by numerical integration. To do so, let

$$\mathcal{T} = \{K : K \text{ is an open subdomain of } \Omega\} \quad \text{and} \quad \mathcal{E}_- = \{E = \partial K \cap \Gamma_- : \ K \in \mathcal{T}\}$$

be partitions of the domain Ω and the inflow boundary Γ_-, respectively. For each $K \in \mathcal{T}$ and $E \in \mathcal{E}_-$, let $\mathcal{Q}_K$ and $\mathcal{Q}_E$ be Newton–Cotes quadrature of integrals over K and E, respectively. The corresponding discrete least-squares functional is defined by

$$\mathcal{L}_{\mathcal{T}}\big(v; g\big) = \sum_{K \in \mathcal{T}} \mathcal{Q}_K^2\big(\mathbf{div}_{\mathcal{T}}\mathbf{f}(v)\big) + \sum_{E \in \mathcal{E}_-} \mathcal{Q}_E^2\big(v - g\big), \tag{3.7}$$

where $\mathbf{div}_{\mathcal{T}}$ denotes a discrete divergence operator. The discrete divergence operators of the Roe and ENO type were studied in [22]. In the subsequent section, we will introduce new discrete divergence operators tailor to the LSNN method that are accurate approximations to the divergence operator when applying to discontinuous solution.

With the discrete least-squares functional $\mathcal{L}_{\mathcal{T}}\big(v;\, g\big)$, the least-squares neural network (LSNN) method is to find $u^N_{\mathcal{T}}(\mathbf{z}, \boldsymbol{\theta}^*) \in \mathcal{M}_N$ such that

$$\mathcal{L}_{\mathcal{T}}\big(u^N_{\mathcal{T}}(\cdot, \boldsymbol{\theta}^*);\, g\big) = \min_{v\in\mathcal{M}_N} \mathcal{L}_{\mathcal{T}}\big(v(\cdot;\boldsymbol{\theta});\; g\big) = \min_{\boldsymbol{\theta}\in\mathbb{R}^N} \mathcal{L}_{\mathcal{T}}\big(v(\cdot;\boldsymbol{\theta});\, g\big). \tag{3.8}$$

Lemma 3.3. *Let u, u^N, and $u^N_{\mathcal{T}}$ be the solutions of problems (2.5), (3.4), and (3.8), respectively. Then we have*

$$\mathcal{L}\big(u^N_{\mathcal{T}};g\big) \le \Big|\big(\mathcal{L}-\mathcal{L}_{\mathcal{T}}\big)\big(u^N_{\mathcal{T}};g\big)\Big| + \Big|\big(\mathcal{L}-\mathcal{L}_{\mathcal{T}}\big)\big(u^N;g\big)\Big| + \Big|\mathcal{L}\big(u^N;g\big)\Big|. \tag{3.9}$$

Proof. By the fact that $\mathcal{L}_{\mathcal{T}}(u^N_{\mathcal{T}};\mathbf{f}) \le \mathcal{L}_{\mathcal{T}}(u^N;\mathbf{f})$, we have

$$\begin{aligned}\mathcal{L}\big(u^N_{\mathcal{T}};g\big) &= \big(\mathcal{L}-\mathcal{L}_{\mathcal{T}}\big)\big(u^N_{\mathcal{T}};g\big) + \mathcal{L}_{\mathcal{T}}\big(u^N_{\mathcal{T}};g\big) \le \big(\mathcal{L}-\mathcal{L}_{\mathcal{T}}\big)\big(u^N_{\mathcal{T}};g\big) + \mathcal{L}_{\mathcal{T}}\big(u^N;g\big)\\ &= \big(\mathcal{L}-\mathcal{L}_{\mathcal{T}}\big)\big(u^N_{\mathcal{T}};g\big) + \big(\mathcal{L}_{\mathcal{T}}-\mathcal{L}\big)\big(u^N;g\big) + \mathcal{L}\big(u^N;g\big),\end{aligned} \tag{3.10}$$

which, together with the triangle inequality, implies (3.9). □

This lemma indicates that the minimum of the discrete least-squares functional $\mathcal{L}_{\mathcal{T}}$ over $\mathcal{M}_N$ is bounded by the minimum of the least-squares functional $\mathcal{L}$ over $\mathcal{M}_N$ plus the approximation error of numerical integration and differentiation in $\mathcal{M}_N$.

In the remainder of this section, we describe the block space–time LSNN method introduced in [22] for dealing with the training difficulty over a relative large computational domain Ω. The method is based on a partition $\{\Omega_{k-1,k}\}_{k=1}^{n_b}$ of the computational domain Ω. To define $\Omega_{k-1,k}$, let $\{\Omega_k\}_{k=1}^{n_b}$ be subdomains of Ω satisfying the following inclusion relation

$$\emptyset = \Omega_0 \subset \Omega_1 \subset \cdots \subset \Omega_{n_b} = \Omega.$$

Then set $\Omega_{k-1,k} = \Omega_k \setminus \Omega_{k-1}$ for $k = 1, \ldots, n_b$. Assume that $\Omega_{k-1,k}$ is in the range of influence of

$$\Gamma_{k-1,k} = \partial\Omega_{k-1,k} \cap \partial\Omega_{k-1} \quad \text{and} \quad \Gamma^k_- = \partial\Omega_{k-1,k} \cap \Gamma_-.$$

Denote by $u^k = u|_{\Omega_{k-1,k}}$ the restriction of the solution u of (2.2) on $\Omega_{k-1,k}$, then u^k is the solution of the following problem:

$$\left\{\begin{array}{rcll} \mathbf{div}_{\mathcal{T}}\mathbf{f}(u^k) &=& 0, & \text{in } \ \Omega_{k-1,k} \in \mathbb{R}^{d+1},\\ u^k &=& u^{k-1}, & \text{on } \ \Gamma_{k-1,k},\\ u^k &=& g, & \text{on } \ \Gamma^k_-. \end{array}\right. \tag{3.11}$$

Let

$$\mathcal{L}^k\big(v;\, u^{k-1}, g\big) = \|\mathbf{div}\,\mathbf{f}(v)\|^2_{0,\Omega_{k-1,k}} + \|v-u^{k-1}\|^2_{0,\Gamma_{k-1,k}} + \|v-g\|^2_{0,\Gamma^k_-},$$

and define the corresponding discrete least-squares functional $\mathcal{L}^k_{\mathcal{T}}\big(v;\, u^{k-1}, g\big)$ over the subdomain $\Omega_{k-1,k}$ in a similar fashion as in (3.7). Now, the block space–time LSNN method is to find $u^k_{\mathcal{T}}(\mathbf{z}, \boldsymbol{\theta}^*_k) \in \mathcal{M}_N$ such that

$$\mathcal{L}^k_{\mathcal{T}}\big(u^k_{\mathcal{T}}(\cdot, \boldsymbol{\theta}^*_k);\, u^{k-1}, g\big) = \min_{v\in\mathcal{M}_N} \mathcal{L}^k_{\mathcal{T}}\big(v(\cdot;\boldsymbol{\theta});\; u^{k-1}, g\big) = \min_{\boldsymbol{\theta}\in\mathbb{R}^N} \mathcal{L}^k_{\mathcal{T}}\big(v(\cdot;\boldsymbol{\theta});\, u^{k-1}, g\big) \tag{3.12}$$

for $k = 1, \ldots, n_b$.

4. Discrete divergence operator

As seen in [18,22], numerical approximation of the differential operator is critical for the success of the LSNN method. Standard numerical or automatic differentiation along coordinate directions generally results in an inaccurate LSNN method, even for linear problems when solutions are discontinuous. This is because the differential form of the HCL is invalid at discontinuous interface. To overcome this difficulty, we used the discrete directional differentiation for linear problems in [18] and the discrete divergence operator of the Roe and ENO type for nonlinear problems in [22].

In this section, we introduce a new discrete divergence operator based on the definition of the divergence operator. Specifically, for each $K \in \mathcal{T}$, let $\mathbf{z}^i_{_K} = (\mathbf{x}^i_{_K}, t^i_{_K})$ and ω_i for $i \in J$ be the quadrature points and weights for the quadrature $\mathcal{Q}_K$, where J is the index set. Hence, the discrete least-squares functional becomes

$$\mathcal{L}_{\mathcal{T}}\big(v;g\big) = \sum_{K\in\mathcal{T}} \left(\sum_{i\in J} \omega_i\, \mathbf{div}_{\mathcal{T}}\mathbf{f}\big(v(\mathbf{z}^i_{_K})\big)\right)^2 + \sum_{E\in\mathcal{E}_-} \mathcal{Q}^2_E\big(v-g\big).$$

To define the discrete divergence operator $\mathbf{div}_{\mathcal{T}}$, we first construct a set of control volumes

$$\mathcal{V} = \{V : V \text{ is an open subdomain of } \Omega\}$$

such that $\mathcal{V}$ is a partition of the domain Ω and that each quadrature point is the centroid of a control volume $V \in \mathcal{V}$. Denote by V_K^i the control volume corresponding to the quadrature point $\mathbf{z}_K^i$, by the definition of the divergence operator, we have

$$\mathbf{div}\,\mathbf{f}\big(u(\mathbf{z}_K^i)\big) \approx \text{avg}_{V_K^i}\,\mathbf{div}\,\mathbf{f}(u) = \frac{1}{|V_K^i|}\int_{\partial V_K^i} \mathbf{f}(u)\cdot\mathbf{n}\,dS, \tag{4.1}$$

where the average of a function φ over V_K^i is defined by

$$\text{avg}_{V_K^i}\,\varphi = \frac{1}{|V_K^i|}\int_{V_K^i} \varphi(\mathbf{z})\,d\mathbf{z}.$$

The average of φ with respect to the partition $\mathcal{V}$ is denoted by $\text{avg}_{\mathcal{V}}\varphi$ and defined as a piece-wise constant function through its restriction on each $V \in \mathcal{V}$ by

$$\text{avg}_{\mathcal{V}}\varphi\big|_V = \text{avg}_V\varphi.$$

Now we may design a discrete divergence operator $\mathbf{div}_{\mathcal{T}}$ acting on the total flux $\mathbf{f}(u)$ by approximating the surface integral on the right-hand side of (4.1).

All existing conservative schemes of various order such as Roe, ENO, WENO, etc. may be viewed as approximations of the surface integral using values of $\mathbf{f}(u)$ at some *mesh points*, where most of them are outside of $\bar{V}$. These conservative schemes are nonlinear methods because the procedure determining proper mesh points to be used for approximating the average of the spatial flux is a nonlinear process due to possible discontinuity.

Because the LSNN method is a "mesh/point-less" space–time method, all points on $\partial V \in \mathbb{R}^{d+1}$ are at our disposal for approximating the surface integral. Hence, the surface integral can be approximated as accurately as desired by using only points on ∂V. When u and hence $f_i(u)$ are discontinuous on ∂V, the best linear approximation strategy is to use piece-wise constant/linear functions on a sufficiently fine partition of each face of ∂V, instead of higher order polynomials on each face. This suggests that a composite lower-order numerical integration such as the composite mid-point/trapezoidal quadrature would provide accurate approximation to the surface integral in (4.1), and hence the resulting discrete divergence operator would be accurate approximation to the divergence operator, even if the solution is discontinuous.

4.1. One dimension

For clarity of presentation, the discrete divergence operator described above will be first introduced in this section in one dimension. To this end, to approximate single integral $I(\varphi) = \int_c^d \varphi(s)\,ds$, we will use the composite midpoint/trapezoidal rule:

$$Q(\varphi(s); c, d, p) = \begin{cases} \dfrac{d-c}{p}\displaystyle\sum_{i=0}^{p-1}\varphi\big(s_{i+1/2}\big), & \text{midpoint,} \\ \dfrac{d-c}{2p}\left(\varphi(c)+\varphi(d)+2\displaystyle\sum_{i=1}^{p-1}\varphi\big(s_i\big)\right), & \text{trapezoidal,} \end{cases} \tag{4.2}$$

where $\{s_i\}_{i=0}^p$ uniformly partitions the interval $[c, d]$ into p sub-intervals.

Let $\Omega = (a, b)\times(0, T)$. For simplicity, assume that the integration partition $\mathcal{T}$ introduced in Section 3 is a uniform partition of the domain Ω; i.e.,

$$\mathcal{T} = \{K = K_{ij} : i = 0, 1, \ldots, m-1;\ j = 0, 1, \ldots, n-1\} \text{ with } K_{ij} = (x_i, x_{i+1})\times(t_j, t_{j+1}),$$

where $x_i = a + ih$ and $t_j = j\tau$ with $h = (b-a)/m$ and $\delta = T/n$. For integration subdomain K_{ij}, the set of quadrature points is

$$\begin{aligned} &M_{ij} = \{\mathbf{z}_{i+\frac{1}{2},j+\frac{1}{2}}\} && \text{for the midpoint rule,} \\ &T_{ij} = \{\mathbf{z}_{i,j}, \mathbf{z}_{i+1,j}, \mathbf{z}_{i,j+1}, \mathbf{z}_{i+1,j+1}\} && \text{for the trapezoidal rule,} \\ \text{and } &S_{ij} = M_{ij}\cup T_{ij}\cup\{\mathbf{z}_{i+\frac{1}{2},j}, \mathbf{z}_{i,j+\frac{1}{2}}, \mathbf{z}_{i+1,j+\frac{1}{2}}, \mathbf{z}_{i+\frac{1}{2},j+1}\} && \text{for the Simpson rule,} \end{aligned}$$

where $\mathbf{z}_{i+k,j+l} = \big(x_i + kh, t_j + l\delta\big)$ for $k, l = 0$, $1/2$, or 1. Based on those quadrature points, the sets of control volumes may be defined accordingly. For example, the control volume $\mathcal{V}_m$ for the midpoint rule is $\mathcal{T}$; the control volume $\mathcal{V}_t$ for the trapezoidal rule is obtained by shifting control volumes in $\mathcal{V}_m$ by $\dfrac{1}{2}(h, \delta)$ plus half-size control volumes along the boundary; and the control volume $\mathcal{V}_s$ for the Simpson rule is obtained in a similar fashion as $\mathcal{V}_t$ on the element size of $h/2$ and $\delta/2$ for space and time, respectively.

For simplicity of presentation, we define the discrete divergence operator only for the midpoint rule for it can be defined in a similar fashion for other quadrature. Since $\mathcal{V}_m = \mathcal{T}$, i.e., the control volume of $\mathcal{V}_m$ is the same as the element of $\mathcal{T}$, for each control volume $V = K_{ij}$, denote its centroid by

$$\mathbf{z}_V = \mathbf{z}_{ij} = (x_i + h/2, t_j + \delta/2).$$

Denote by $\sigma = f(u)$ the spatial flux, then the total flux is the two-dimensional vector field $\mathbf{f}(u) = (\sigma, u)$. Denote the first-order finite difference quotients by

$$\sigma(x_i, x_{i+1}; t) = \frac{\sigma(x_{i+1}, t) - \sigma(x_i, t)}{x_{i+1} - x_i} \quad \text{and} \quad u(x; t_j, t_{j+1}) = \frac{u(x, t_{j+1}) - u(x, t_j)}{t_{j+1} - t_j}.$$

Then the surface integral in (4.1) becomes

$$\frac{1}{|K_{ij}|} \int_{\partial K_{ij}} \mathbf{f}(u) \cdot \mathbf{n}\, dS = \delta^{-1} \int_{t_j}^{t_{j+1}} \sigma(x_i, x_{i+1}; t)\, dt + h^{-1} \int_{x_i}^{x_{i+1}} u(x; t_j, t_{j+1})\, dx. \tag{4.3}$$

Approximating single integrals by the composite midpoint/trapezoidal rule, we obtain the following discrete divergence operator

$$\mathbf{div}_{\mathcal{T}} \mathbf{f}\big(u(\mathbf{z}_{ij})\big) = \delta^{-1} Q(\sigma(x_i, x_{i+1}; t); t_j, t_{j+1}, \hat{n}) + h^{-1} Q(u(x; t_j, t_{j+1}); x_i, x_{i+1}, \hat{m}). \tag{4.4}$$

Remark 4.1. Denote by $u_{i,j}$ as approximation to $u(x_i, t_j)$. (4.4) with $\hat{m} = \hat{n} = 1$ using the trapezoidal rule leads to the following implicit conservative scheme for the one-dimensional scalar nonlinear HCL:

$$\frac{u_{i+1,j+1} + u_{i,j+1}}{\delta} + \frac{f\left(u_{i+1,j+1}\right) - f\left(u_{i,j+1}\right)}{h} = \frac{u_{i+1,j} + u_{i,j}}{\delta} - \frac{f\left(u_{i+1,j}\right) - f\left(u_{i,j}\right)}{h} \tag{4.5}$$

for $i = 0, 1, \ldots, m-1$ and $j = 0, 1, \ldots, n-1$.

Below, we state error estimates of the discrete divergence operator defined in (4.4) and postpone their proof to Appendix.

Lemma 4.2. *For any $K_{ij} \in \mathcal{T}$, assume that u is a C^2 function on every edge of the rectangle ∂K_{ij}. Then there exists a constant $C > 0$ such that*

$$\begin{aligned} &\|\mathbf{div}_{\mathcal{T}} \mathbf{f}(u) - avg_{\mathcal{T}} \mathbf{div}\, \mathbf{f}(u)\|_{L^p(K_{ij})} \\ &\le C \left(\frac{h^{1/p}\delta^2}{\hat{n}^2} \|\sigma_{tt}(x_{i+1}, x_i; \cdot)\|_{L^p(t_j, t_{j+1})} + \frac{h^2 \delta^{1/p}}{\hat{m}^2} \|u_{xx}(\cdot; t_{j+1}, t_j)\|_{L^p(x_i, x_{i+1})} \right). \end{aligned} \tag{4.6}$$

This lemma indicates that $\hat{m} = 1$ and $\hat{n} = 1$ are sufficient if the solution is smooth on ∂K_{ij}. In this case, we may use higher order numerical integration, e.g., the Gauss quadrature, to approximate the surface integral in (4.3) for constructing a higher order discrete divergence operator.

When u is discontinuous on ∂K_{ij}, error estimate on the discrete divergence operator becomes more involved. To this end, first we consider the case that the discontinuous interface Γ_{ij} (a straight line) intersects two horizontal boundary edges of K_{ij}. Denote by $u_{ij} = u|_{K_{ij}}$ the restriction of u in K_{ij} and by $[\![u_{ij}]\!]_{t_l}$ the jump of u_{ij} on the horizontal boundary edge $t = t_l$ of K_{ij}, where $l = j$ and $l = j + 1$.

Lemma 4.3. *Assume that u is a C^2 function of t and a piece-wise C^2 function of x on two vertical and two horizontal edges of K_{ij}, respectively. Moreover, u has only one discontinuous point on each horizontal edge. Then there exists a constant $C > 0$ such that*

$$\begin{aligned} &\|\mathbf{div}_{\mathcal{T}} \mathbf{f}(u) - avg_{\mathcal{T}} \mathbf{div}\, \mathbf{f}(u)\|_{L^p(K_{ij})} \\ &\le C \left(\frac{h^{1/p}\delta^2}{\hat{n}^2} + \frac{h^2 \delta^{1/p}}{\hat{m}^2} + \frac{h \delta^{1/p}}{\hat{m}^{1+1/q}} \right) + \frac{(h\delta)^{1/p}}{\hat{m}} \sum_{l=j}^{j+1} [\![u_{ij}]\!]_{t_l}. \end{aligned} \tag{4.7}$$

Remark 4.4. Lemma 4.3 implies that the choice of the number of sub-intervals of (x_i, x_{i+1}) on the composite numerical integration depends on the size of the jump of the solution and that large $\hat{m}$ would guarantee accuracy of the discrete divergence operator when u is discontinuous on ∂K_{ij}.

Remark 4.5. Error bounds similar to (4.7) hold for the other cases: Γ_{ij} intercepts *(i)* two vertical edges or *(ii)* one horizontal and one vertical edges of K_{ij}. Specifically, we have

$$\|\mathbf{div}_{\mathcal{T}}\mathbf{f}(u) - \mathrm{avg}_{\mathcal{T}}\mathbf{div}\,\mathbf{f}(u)\|_{L^p(K_{ij})} \le C\left(\frac{h^{1/p}\delta^2}{\hat{n}^2} + \frac{h^2\delta^{1/p}}{\hat{m}^2} + \frac{h^{1/p}\delta}{\hat{n}^{1+1/q}}\right) + \frac{(h\delta)^{1/p}}{\hat{n}}\sum_{l=i}^{i+1}[\![\sigma_{ij}]\!]_{x_l}$$

for the case *(i)* and

$$\|\mathbf{div}_{\mathcal{T}}\mathbf{f}(u) - \mathrm{avg}_{\mathcal{T}}\mathbf{div}\,\mathbf{f}(u)\|_{L^p(K_{ij})} \le C\left(\frac{h^{1/p}\delta^2}{\hat{n}^2} + \frac{h^2\delta^{1/p}}{\hat{m}^2} + \frac{h\delta^{1/p}}{\hat{m}^{1+1/q}} + \frac{h^{1/p}\delta}{\hat{n}^{1+1/q}}\right) + E_{ij}$$

for the case *(ii)*, where $E_{ij} = (h\delta)^{1/p}\left(\frac{1}{\hat{m}}[\![u_{ij}]\!]_{t_l} + \frac{1}{\hat{n}}[\![\sigma_{ij}]\!]_{x_l}\right)$ with $x_l = x_i$ or x_{i+1} and $t_l = t_j$ or t_{j+1}.

4.2. Two dimensions

This section describes the discrete divergence operator in two dimensions. As in one dimension, the discrete divergence operator is defined as an approximation to the average of the divergence operator through the composite mid-point/trapezoidal quadrature to approximate the surface integral (4.1). Extension to three dimensions is straightforward.

To this end, we first describe the composite mid-point/trapezoidal numerical integration for approximating a double integral over a rectangle region $T = (c_1, d_1) \times (c_2, d_2)$

$$\begin{aligned} I(\varphi) &= \int_T \varphi(s_1, s_2)\, ds_1 ds_2 \\ &\approx Q\big(\varphi(s_1, s_2); c_1, d_1, p_1; c_2, d_2, p_2\big) \equiv Q\Big(Q\big(\varphi(s_1, \cdot); c_1, d_1, p_1\big)(s_2); c_2, d_2, p_2\Big), \end{aligned}$$

where $Q\big(\varphi(s_1, \cdot); c_1, d_1, p_1\big)$ is the composite quadrature defined in (4.2).

For simplicity, let $\Omega = \tilde{\Omega} \times I = (a_1, b_1) \times (a_2, b_2) \times (0, T)$, and assume that the integration partition $\mathcal{T}$ introduced in Section 3 is a uniform partition of the domain Ω; i.e.,

$$\mathcal{T} = \{K = K_{ijk} : \ i = 0, 1, \ldots, m_1 - 1; \ j = 0, 1, \ldots, m_2 - 1; \ k = 0, 1, \ldots, n - 1\}$$

with $K_{ijk} = (x_i, x_{i+1}) \times (y_j, y_{j+1}) \times (t_k, t_{k+1})$, where

$$x_i = a_1 + ih_1, \quad y_j = a_2 + jh_2, \quad \text{and} \quad t_k = k\delta,$$

and $h_l = (b_l - a_l)/m_l$ for $l = 1, 2$ and $\delta = T/n$ are the respective spatial and temporal sizes of the integration mesh. Again, we define the discrete divergence operator only corresponding to the midpoint rule. Denote the mid-point of K_{ijk} by

$$\mathbf{z}_{ijk} = (x_i + \frac{h_1}{2}, y_j + \frac{h_2}{2}, t_k + \frac{\delta}{2}).$$

Let $\boldsymbol{\sigma} = (\sigma_1, \sigma_2) = (f_1(u), f_2(u))$, then the space–time flux is the three-dimensional vector field: $\mathbf{f}(u) = (\boldsymbol{\sigma}, u) = (\sigma_1, \sigma_2, u)$. Denote the first-order finite difference quotients by

$$\begin{aligned} &\sigma_1(y, t; x_i, x_{i+1}) = \frac{\sigma_1(x_{i+1}, y, t) - \sigma_1(x_i, y, t)}{x_{i+1} - x_i}, \quad \sigma_2(x, t; y_j, y_{j+1}) = \frac{\sigma_2(x, y_{j+1}, t) - \sigma_1(x,y_j, t)}{y_{j+1} - y_j}, \\ &\text{and} \ \ u(x, y; t_k, t_{k+1}) = \frac{u(x, y, t_{k+1}) - u(x, y, t_k)}{t_{k+1} - t_k}. \end{aligned}$$

Denote three faces of ∂K_{ijk} by

$$K_{ij}^{xy} = (x_i, x_{i+1}) \times (y_j, y_{j+1}), \ \ K_{ik}^{xt} = (x_i, x_{i+1}) \times (t_k, t_{k+1}), \ \ \text{and} \ \ K_{jk}^{yt} = (y_j, y_{j+1}) \times (t_k, t_{k+1}).$$

Then the surface integral in (4.1) becomes

$$\begin{aligned} \frac{1}{|K_{ijk}|}\int_{\partial K_{ijk}} \mathbf{f}(u)\cdot\mathbf{n}\, dS &= (h_2\delta)^{-1}\int_{K_{jk}^{yt}} \sigma_1(y, t; x_{i+1}, x_i)\, dydt \\ &+ (h_1\delta)^{-1}\int_{K_{ik}^{xt}} \sigma_2(x, t; y_{j+1}, y_j)\, dxdt + (h_1h_2)^{-1}\int_{K_{ij}^{xy}} u(x, y; t_{k+1}, t_k)\, dxdy. \end{aligned} \tag{4.8}$$

Approximating double integrals by the composite midpoint/trapezoidal rule, we obtain the following discrete divergence operator

$$\begin{aligned}\mathbf{div}_{\mathcal{T}}\mathbf{f}\big(u(\mathbf{z}_{ijk})\big) &= (h_2\delta)^{-1}Q\left(\sigma_1(y,t;x_{i+1},x_i);y_j,y_{j+1},\hat{m}_2;t_k,t_{k+1},\hat{n}\right)\\ &\quad+(h_1\delta)^{-1}Q\left(\sigma_2(x,t;y_{j+1},y_j);x_i,x_{i+1},\hat{m}_1;t_k,t_{k+1},\hat{n}\right)\\ &\quad+(h_1h_2)^{-1}Q\left(u(x,y;t_{k+1},t_k);x_i,x_{i+1},\hat{m}_1;y_j,y_{j+1},\hat{m}_2\right).\end{aligned} \tag{4.9}$$

4.3. Integration mesh size

The discrete divergence operator defined in (4.4) and (4.9) for the respective one- and two-dimension is based on the composite midpoint/trapezoidal rule. As shown in Lemmas 4.2 and 4.3 and Remark 4.5, the discrete divergence operator can be as accurate as desired for the discontinuous solution provided that the size of integration mesh is sufficiently small.

To reduce computational cost, note that the discontinuous interfaces of the solution u lie on d-dimensional hyper-planes. Hence, they only intersect with a small portion of control volumes in $\mathcal{T}$. This observation suggests that sufficiently fine meshes are only needed for control volumes at where the solution is possibly discontinuous. To realize this idea, we divide the set of control volumes into two subsets:

$$\mathcal{T}=\mathcal{T}_c\cup\mathcal{T}_d,$$

where the solution u is continuous in each control volume of $\mathcal{K}_c^l$ and possibly discontinuous at some control volumes of $\mathcal{T}_d$; i.e.,

$$\mathcal{T}_c=\{K\in\mathcal{T}:\ u\in C(K)\}\quad\text{and}\quad \mathcal{T}_d=\mathcal{T}\setminus\mathcal{T}_c.$$

Next, we describe how to determine the set of control volumes $\mathcal{T}_d$ in one dimension by the range of influence. It is well-known that characteristic curves are straight lines before their interception and are given by

$$x=x(T_l)+(t-T_l)f'\big(u\,(x(T_l),T_l)\big). \tag{4.10}$$

For $i=0,1,\ldots,m$, let

$$\hat{x}_i=x_i+(T_{l+1}-T_l)f'\big(u_{_N}^l\,(x_i,T_l)\big),$$

where $u_{_N}^l\,(x_i,T_l)$ is the neural network approximation from the previous time block

$$\Omega\times I_{l-1}=(a,b)\times(T_{l-1},T_l).$$

Clearly, the solution u is discontinuous in a control volume $V_i\times I_l^k$ if either (1) $u(x,T_l)$ is discontinuous at the interval V_i or (2) there are two characteristic lines intercepting in $V_i\times I_l^k$. In the first case, $V_i\times I_l^k$ is in $\mathcal{K}_d^l$ if $u_{_N}^l(x,T_l)$ has a sharp change in the interval V_i; moreover, either $V_{i-1}\times I_l^k\in\mathcal{K}_d^l$ if $\hat{x}_i<x_i$ or $V_{i+1}\times I_l^k\in\mathcal{K}_d^l$ if $\hat{x}_{i+1}>x_{i+1}$. In the second case, assume that $\hat{x}_i>\hat{x}_{i+1}$, then $V_i\times I_l^k\in\mathcal{K}_d^l$ if $\hat{x}_i<x_{i+1}$.

5. Numerical experiments

This section presents numerical results of the block space–time LSNN method for one and two dimensional problems. Let $\Omega=\tilde{\Omega}\times(0,T)$. The kth space–time block is defined as

$$\Omega_{k-1,k}=\Omega_k\setminus\Omega_{k-1}=\tilde{\Omega}\times\left(\frac{(k-1)T}{n_b},\frac{kT}{n_b}\right)\quad\text{for}\ \ k=1,\ldots,n_b,$$

where $\Omega_k=\tilde{\Omega}\times(0,kT/n_b)$. For efficient training, the least-squares functional is modified as follows:

$$\mathcal{L}^k\big(v;u^{k-1},g\big)=\|\mathbf{div}\,\mathbf{f}(v)\|_{0,\Omega_{k-1,k}}^2+\alpha(\|v-u^{k-1}\|_{0,\Gamma_{k-1,k}}^2+\|v-g\|_{0,\Gamma_-^k}^2), \tag{5.1}$$

where α is a weight to be chosen empirically.

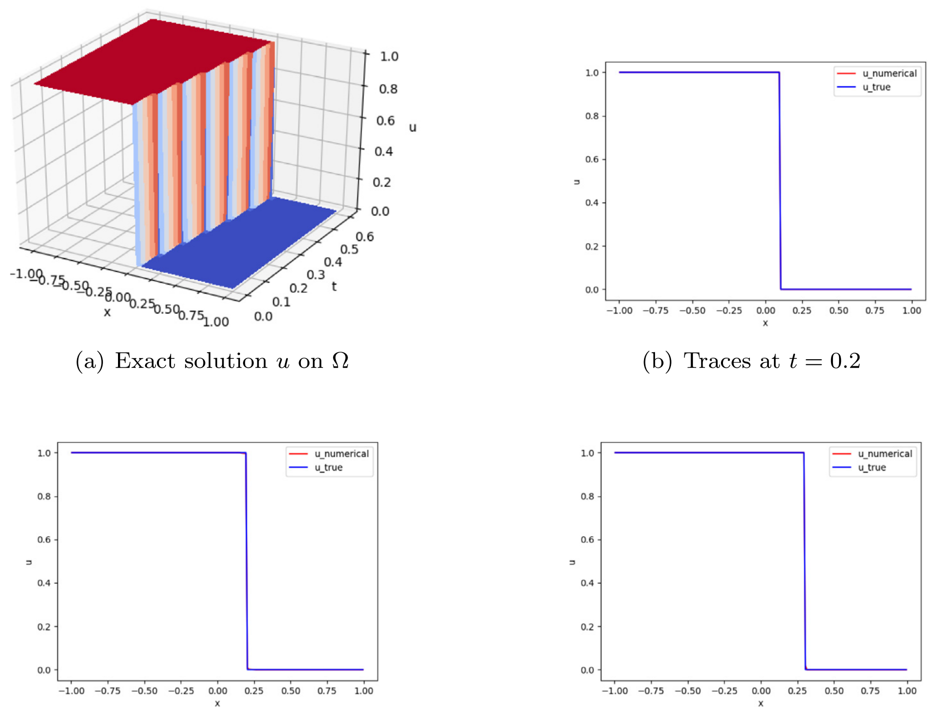
Unless otherwise stated, the integration mesh $\mathcal{T}_k$ is a uniform partition of $\Omega_{k-1,k}$ with $h=\delta=0.01$, and the discrete divergence operator defined in (4.4) is based on the composite trapezoidal rule with $\hat{m}=\hat{n}=2$. Three-layer or four-layer neural network are employed for all test problems and are denoted by $d_in-n_1-n_2(-n_3)$-1 with n_1, n_2 and n_3 neurons in the respective first, second and third (for a four-layer NN)layers. The same network structure is used for all time blocks.

The network is trained by using the ADAM [28] (a variant of the method of gradient descent) with either a fixed or an adaptive learning rate to iteratively solve the minimization problem in (3.12). Parameters of the first block is initialized by an approach introduced in [29], and those for the current block is initialized by using the NN approximation of the previous block (see Remark 4.1 of [22]).

The solution of the problem in (3.11) and its corresponding NN approximation are denoted by u^k and $u_{\mathcal{T}}^k$, respectively. Their traces are depicted on a plane of given time and exhibit capability of the numerical approximation in capturing shock/rarefaction.

Table 1
Relative L^2 errors of Riemann problem (shock) for Burgers' equation.

Network structure	Block	$\frac{\|u^k-u^k_{\mathcal{T}}\|_0}{\|u^k\|_0}$
	$\Omega_{0,1}$	0.048774
2-10-10-1	$\Omega_{1,2}$	0.046521
	$\Omega_{2,3}$	0.044616



(a) Exact solution u on Ω (b) Traces at $t = 0.2$

(c) Traces at $t = 0.4$ (d) Traces at $t = 0.6$

Fig. 1. Approximation results of Riemann problem (shock) for Burgers' equation.

5.1. Inviscid Burgers' equation

This section reports numerical results of the block space–time LSNN method for the one dimensional inviscid Burgers equation, where the spatial flux is $\tilde{\mathbf{f}}(u) = f(u) = \frac{1}{2}u^2$.

The first two test problems are the Riemann problem with the initial condition: $u_0(x) = u(x, 0) = u_L$ if $x \le 0$ or u_R if $x \ge 0$.

Shock formation. When $u_L = 1 > 0 = u_R$, a shock is formed immediately with the shock speed $s = \left(u_L + u_R\right)/2$. The first test problem is defined on a computational domain $\Omega = (-1, 1) \times (0, 0.6)$ with inflow boundary

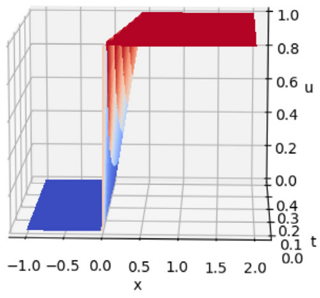
$$\Gamma_- = \Gamma_-^L \cup \Gamma_-^R \equiv \{(-1, t) : t \in [0, 0.6]\} \cup \{(1, t) : t \in [0, 0.6]\}$$

and boundary conditions: $g = u_L = 1$ on Γ_-^L and $g = u_R = 0$ on Γ_-^R. With $n_b = 3$ blocks, weight $\alpha = 20$, a fixed learning rate 0.003, and 30 000 iterations for each block, the relative errors in the L^2 norm are reported in Table 1. Traces of the exact solution and numerical approximation on the planes $t = kT/n_b$ for $k = 1, 2, 3$ are depicted in Fig. 1(b)–(d), which clearly indicate that the LSNN method is capable of capturing the shock formation and its speed. Moreover, it approximates the solution well without oscillations.

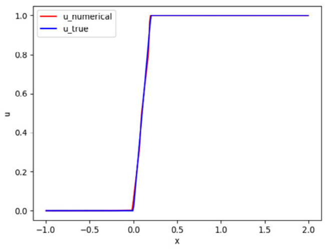
Rarefaction waves. When $u_L = 0 < 1 = u_R$, the range of influence of all points in $\mathbb{R}$ is a proper subset of $\mathbb{R} \times [0, \infty)$. Hence, the weak solution of the scalar hyperbolic conservation law is not unique. The second test problem is defined on a computational domain $\Omega = (-1, 2) \times (0, 0.4)$ with inflow boundary condition $g = 0$ on $\Gamma_- = \{(-1, t) : t \in [0, 0.4]\}$. As shown in Section 5.1.2 of [22], the LSNN method using Roe's scheme has a limitation to resolve the rarefaction. Numerical results of the LSNN method using the discrete divergence operator ($n_b = 2$, $\alpha = 10$, a fixed learning rate 0.003, and 40 000

Table 2
Relative L^2 errors of Riemann problem (rarefaction) for Burgers' equation.

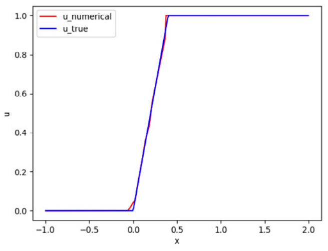
Network structure	Block	$\frac{\|u^k - u^k_{\mathcal{T}}\|_0}{\|u^k\|_0}$
2-10-10-1	$\Omega_{0,1}$	0.013387
	$\Omega_{1,2}$	0.010079



(a) Exact solution u on Ω



(b) Traces at $t = 0.2$



(c) Traces at $t = 0.4$

Fig. 2. Approximation results of Riemann problem (rarefaction) for Burgers' equation.

iterations) are reported in Table 2. Traces of the exact solution and numerical approximation on the planes $t = 0.2$ and $t = 0.4$ are depicted in Fig. 2. This test problem shows that the LSNN method using the $\mathbf{div}_{\mathcal{T}}$ is able to compute the physically relevant vanishing viscosity solution (see, e.g., [7,30]) without special treatment. This is possibly due to the fact that the LSNN approximation is continuous.

Sinusoidal initial condition. The third test problem has smooth initial condition $u_0(x) = 0.5 + \sin(\pi x)$ and is defined on the computational domain $\Omega = (0, 2) \times (0, 0.8)$ with inflow boundary

$$\Gamma_- = \Gamma_-^L \cup \Gamma_-^R \equiv \{(0, t) : t \in [0, 0.8]\} \cup \{(2, t) : t \in [0, 0.8]\}.$$

The shock of the problem appears at $t = 1/\pi \approx 0.318$. This is the same test problem as in Section 5.2 of [22] (see also [31,32]). The goal of this experiment is to compare numerical performances of the LSNN methods using the $\mathbf{div}_{\mathcal{T}}$ introduced in this paper and the ENO scheme in [22].

Since the solution of this problem is implicitly given, to accurately measure the quality of NN approximations, a benchmark reference solution $\hat{u}$ is generated using the traditional mesh-based method. In particular, the third-order accurate WENO scheme [3] and the fourth-order Runge–Kutta method are employed for the respective spatial and temporal discretizations with a fine mesh ($\Delta x = 0.001$ and $\Delta t = 0.0002$) on the computational domain Ω.

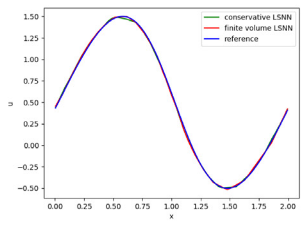
The LSNN using $\mathbf{div}_{\mathcal{T}}$ is implemented with the same set of hyper parameters as in Section 5.2 of [22], i.e., training weight $\alpha = 5$ and an adaptive learning rate which starts with 0.005 and reduces by half for every 25 000 iterations. Setting $n_b = 16$ and on each time block, the total number of iterations is set as 50 000 and the size of the NN model is 2-30-30-1. Although we observe some error accumulation when the block evolves for both the LSNN methods, the one using $\mathbf{div}_{\mathcal{T}}$ performs better than that using ENO (see Table 3 for the relative L^2 norm error and Fig. 3(a)–(h) for graphs near the left side of the interface).

5.2. Riemann problem with $f(u) = \frac{1}{4}u^4$

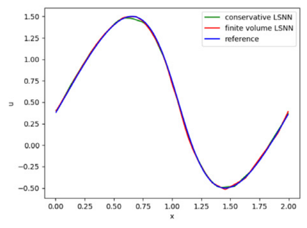
The goals of this set of numerical experiments are twofold. First, we compare the performance of the LSNN method using the composite trapezoidal/mid-point rule in (4.2). Second, we investigate the impact of the number of sub-intervals of the composite quadrature rule on the accuracy of the LSNN method.

The test problem is the Riemann problem with a convex flux $\mathbf{f}(u) = (f(u), u) = (\frac{1}{4}u^4, u)$ and the initial condition $u_L = 1 > 0 = u_R$. The computational domain is chosen to be $\Omega = (-1, 1) \times (0, 0.4)$. Relative L^2 errors of the LSNN method using the $\mathbf{div}_{\mathcal{T}}$ (2-10-10-1 NN model, $n_b = 2$, $\alpha = 20$, a fixed learning rate 0.003 for the first 30 000 iterations and 0.001 for the remaining) are reported in Tables 4 and 5; and traces of the exact and numerical solutions are depicted in Fig. 4.

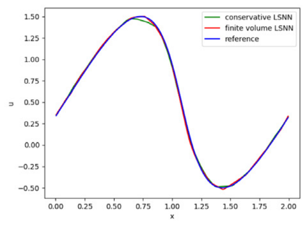
Clearly, Tables 4 and 5 indicate that the accuracy of the LSNN method depends on the number of sub-intervals ($\hat{m}$ and $\hat{n}$) for the composite quadrature rule; i.e., the larger $\hat{m}$ and $\hat{n}$ are, the more accurate the LSNN method is. Moreover, the accuracy using the composite trapezoidal and mid-point rules in the LSNN method is comparable.



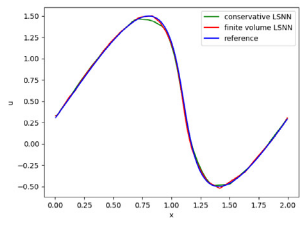
(a) Traces at $t = 0.05$



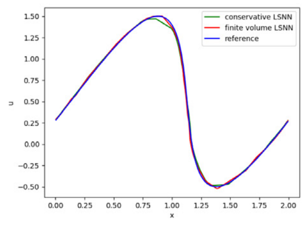
(b) Traces at $t = 0.1$



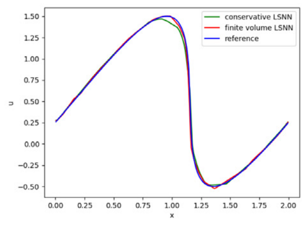
(c) Traces at $t = 0.15$



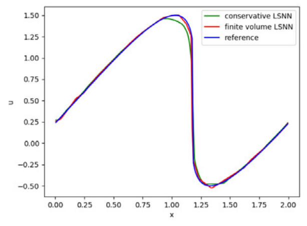
(d) Traces at $t = 0.2$



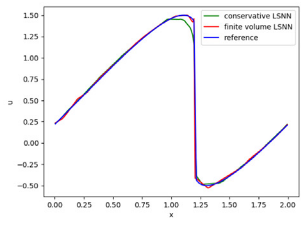
(e) Traces at $t = 0.25$



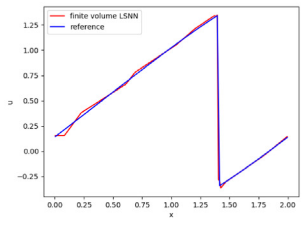
(f) Traces at $t = 0.3$



(g) Traces at $t = 0.35$

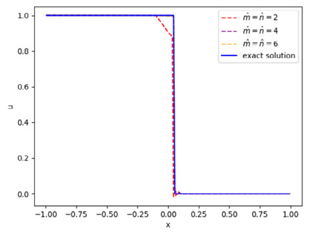


(h) Traces at $t = 0.4$

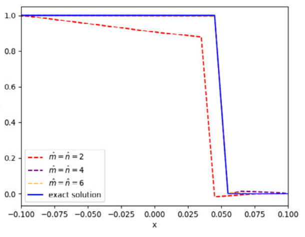


(i) Traces at $t = 0.8$

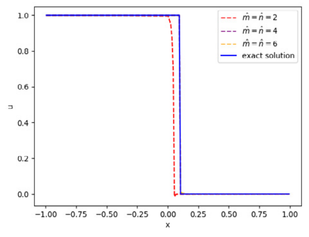
Fig. 3. Approximation results of Burgers' equation with a sinusoidal initial condition.



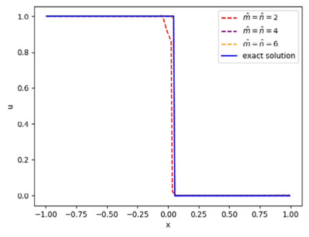
(a) Traces at $t = 0.2$ (trapezoidal)



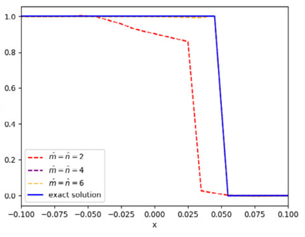
(b) Zoom-in plot near the discontinuous interface of sub-figure (a)



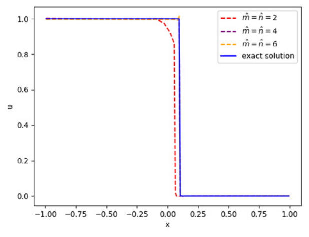
(c) Traces at $t = 0.4$ (trapezoidal)



(d) Traces at $t = 0.2$ (mid-point)



(e) Zoom-in plot near the discontinuous interface of sub-figure (d)



(f) Traces at $t = 0.4$ (mid-point)

Fig. 4. Numerical results of the problem with $f(u) = \frac{1}{4}u^4$ using the composite trapezoidal and mid-point rules.

Table 3
Relative L^2 errors of Burgers' equation with a sinusoidal initial condition.

Network structure	Block	LSNN using $\mathbf{div}_{\mathcal{T}}$ $\frac{\|u^k-u^k_{\mathcal{T}}\|_0}{\|u^k\|_0}$	LSNN using ENO [22] $\frac{\|u^k-u^k_{\mathcal{T}}\|_0}{\|u^k\|_0}$
2-30-30-1	$\Omega_{0,1}$	0.010641	0.010461
	$\Omega_{1,2}$	0.011385	0.012517
	$\Omega_{2,3}$	0.012541	0.019772
	$\Omega_{3,4}$	0.014351	0.022574
	$\Omega_{4,5}$	0.016446	0.029011
	$\Omega_{5,6}$	0.018634	0.038852
	$\Omega_{6,7}$	0.031103	0.075888
	$\Omega_{7,8}$	0.053114	0.078581
	$\Omega_{8,9}$	0.053562	–
	$\Omega_{9,10}$	0.064933	–
	$\Omega_{10,11}$	0.061354	–
	$\Omega_{11,12}$	0.077982	–
	$\Omega_{12,13}$	0.061145	–
	$\Omega_{13,14}$	0.070554	–
	$\Omega_{14,15}$	0.068539	–
	$\Omega_{15,16}$	0.065816	–

Table 4
Relative L^2 errors of the problem with $f(u) = \frac{1}{4}u^4$ using the composite trapezoidal rule (4.2).

Time block	Number of sub-intervals		
	$\hat{m} = \hat{n} = 2$	$\hat{m} = \hat{n} = 4$	$\hat{m} = \hat{n} = 6$
$\Omega_{0,1}$	0.067712	0.010446	0.004543
$\Omega_{1,2}$	0.108611	0.008275	0.009613

Table 5
Relative L^2 errors of the problem with $f(u) = \frac{1}{4}u^4$ using the composite mid-point rule (4.2).

Time block	Number of sub-intervals		
	$\hat{m} = \hat{n} = 2$	$\hat{m} = \hat{n} = 4$	$\hat{m} = \hat{n} = 6$
$\Omega_{0,1}$	0.096238	0.007917	0.003381
$\Omega_{1,2}$	0.159651	0.007169	0.005028

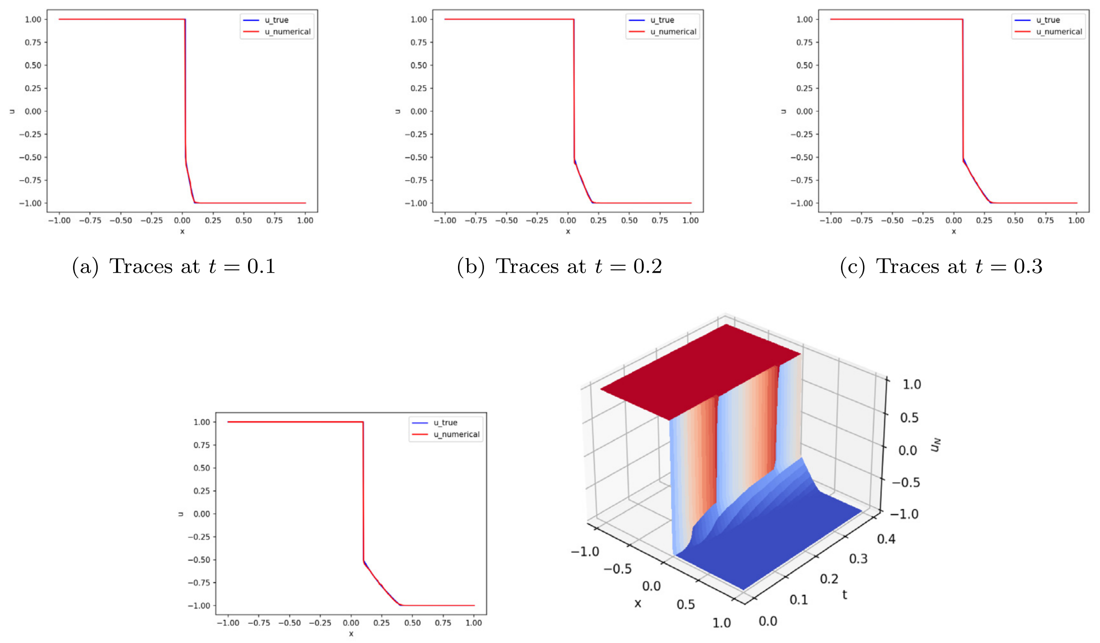
Table 6
Relative L^2 errors of Riemann problem with a non-convex flux $f(u) = \frac{1}{3}u^3$.

Network structure	Block	$\frac{\|u^k-u^k_{\mathcal{T}}\|_0}{\|u^k\|_0}$
2-64-64-64-1	$\Omega_{0,1}$	0.03277
	$\Omega_{1,2}$	0.03370
	$\Omega_{2,3}$	0.03450
	$\Omega_{3,4}$	0.03578

5.3. Riemann problem with non-convex fluxes

The test problem for a non-convex flux is a modification of the test problem in Section 5.2 by replacing the flux with $f(u) = \frac{1}{3}u^3$ and the initial condition with $u_L = 1 > -1 = u_R$. The Riemann solution consists partly of a rarefaction wave together with a shock wave which brings a new level of challenge with a compound wave. The exact solution is obtained through Osher's formulation [33] which has a shock speed s=0.25 and a shock jump from 1 to -0.5 when $t > 0$.

The block space–time LSNN method using the $\mathbf{div}_{\mathcal{T}}$ with $\hat{m} = \hat{n} = 4$ is utilized for this problem. Four time blocks are computed on the temporal domain (0, 0.4) and a relative larger network structure (2-64-64-64-1) is tested with a smaller integration mesh size $h = \delta = 0.005$ to compute the compound wave more precisely. We tune the hyper parameter $\alpha = 200$, and all time blocks are computed with a total of 60 000 iterations (learning rate starts with 1e-3 and decay to 20% every 20 000 iterations). Due to the random initial guess for the second hidden layer parameters, the experiment is replicated several times. Similar results are obtained as the best result reported in Table 6 and Fig. 5(a)–(e). These experiments demonstrate that the LSNN method can capture the compound wave for non-convex flux problems as well.



(a) Traces at $t = 0.1$ (b) Traces at $t = 0.2$ (c) Traces at $t = 0.3$

(d) Traces at $t = 0.4$ (e) Numerical Solution u_N on Ω

Fig. 5. Numerical results of Riemann problem with a non-convex flux $f(u) = \frac{1}{3}u^3$.

5.4. *Two-dimensional problem*

Consider a two-dimensional inviscid Burgers equation, where the spatial flux vector field is $\tilde{\mathbf{f}}(u) = \frac{1}{2}(u^2, u^2)$. Given a piece-wise constant initial data

$$u_0(x, y) = \begin{cases} -0.2, & \text{if } x < 0.5 \text{ and } y > 0.5, \\ -1.0, & \text{if } x > 0.5 \text{ and } y > 0.5, \\ 0.5, & \text{if } x < 0.5 \text{ and } y < 0.5, \\ 0.8, & \text{if } x > 0.5 \text{ and } y < 0.5, \end{cases} \tag{5.2}$$

this problem has an exact solution given in [34].

The test problem is set on computational domain $\Omega = (0, 1)^2 \times (0, 0.5)$ with inflow boundary conditions prescribed by using the exact solution. Our numerical result using a 4-layer LSNN (3-48-48-48-1) with 3D $\mathbf{div}_{\mathcal{T}}$ $(\hat{m} = \hat{n} = \hat{k} = 2)$ are reported in Table 7. The corresponding hyper parameters setting is as follows: $n_b = 5$, $\alpha = 20$, the first time block is trained with 30 000 iteration where the first 10 000 iterations are using learning rate 0.003 and the rest iterations are trained using learning rate of 0.001; all remaining time blocks are trained with 20 000 iterations using fixed learning rate of 0.001. Fig. 6 presents the graphical results at time $t = 0.1$, 0.3, and 0.5. This experiment shows that the proposed LSNN method can be extended to two dimensional problems and can capture the shock and rarefaction waves in two dimensions.

6. Discussion and conclusion

The ReLU neural network provides a new class of approximating functions that is ideal for approximating discontinuous functions with unknown interface location [18]. Making use of this unique feature of neural networks, this paper studied the least-squares ReLU neural network (LSNN) method for solving scalar nonlinear hyperbolic conservation laws.

In the design of the LSNN method for HCLs, the numerical approximation of differential operators is a critical factor, and standard numerical or automatic differentiation along coordinate directions can often lead to a failed NN-based method. To overcome this challenge, this paper introduced a new discrete divergence operator $\mathbf{div}_{\mathcal{T}}$ based on its physical meaning.

Numerical results for several test problems show that the LSNN method using the $\mathbf{div}_{\mathcal{T}}$ does overcome limitations of the LSNN method with conservative flux in [22]. Moreover, for the one dimensional test problems with fluxes $f(u) = \frac{1}{4}u^4$ and $\frac{1}{3}u^3$, the accuracy of the method may be improved greatly by using enough number of sub-intervals in the composite trapezoidal/mid-point quadrature.

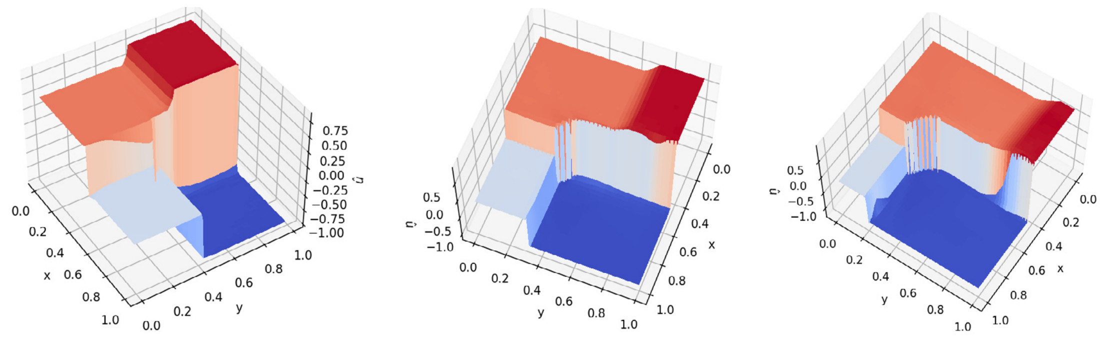

(a) $t = 0.1$ (b) $t = 0.3$ (c) $t = 0.5$

Fig. 6. Numerical results of 2D Burgers' equation.

Table 7
Relative L^2 errors of Riemann problem (shock) for 2D Burgers' equation.

Network structure	Block	$\frac{\|u^k-u^k_{\mathcal{T}}\|_0}{\|u^k\|_0}$
3-48-48-48-1	$\Omega_{0,1}$	0.093679
	$\Omega_{1,2}$	0.121375
	$\Omega_{2,3}$	0.163755
	$\Omega_{3,4}$	0.190460
	$\Omega_{4,5}$	0.213013

Compared to other NN-based methods like the PINN and its variants, the LSNN method introduced in this paper free of any penalization such as the entropy, total variation, and/or artificial viscosity, etc. Usually, choosing proper penalization constants can be challenging in practice and it affects the accuracy, efficiency, and stability of the method.

Even though the number of degrees of freedom for the LSNN method is several order of magnitude less than those of traditional mesh-based numerical methods, training NN is computationally intensive and complicated. For a network with more than one hidden layer, random initialization of the parameters in layers beyond the first hidden layer would cause some uncertainty in training NN (iteratively solving the resulting non-convex optimization) as observed in Section 5.2. This issue plus designation of a proper architecture of NN would be addressed in a forthcoming paper using the adaptive network enhancement (ANE) method developed in [29,35,36].

Data availability

No data was used for the research described in the article.

Appendix

In the appendix, we provide the proofs of Lemmas 4.2 and 4.3. First, denote the integral and the mid-point/trapezoidal rule of a function φ over an interval $[0, \rho]$ by

$$I(\varphi) = \int_0^\rho \varphi(s)\,ds \quad \text{and} \quad Q(\varphi; 0, \rho, 1) = \begin{cases} \rho\,\varphi(\rho/2), & \text{midpoint}, \\ \dfrac{\rho}{2}\big(\varphi(0) + \varphi(\rho)\big), & \text{trapezoidal}, \end{cases}$$

respectively. Let $p,\ q \in (1, \infty]$ such that $1/p + 1/q = 1$. It is easy to show the following error bounds:

$$\big|I(\varphi) - Q(\varphi; 0, \rho, 1)\big| \le \begin{cases} C\rho^{2+1/q}\|\varphi''\|_{L^p(0,\rho)}, & \text{if } \varphi \in C^2(0, \rho), \\ C\rho^{1+1/q}\|\varphi'\|_{L^p(0,\rho)}, & \text{if } \varphi \in C^1(0, \rho). \end{cases} \tag{A.1}$$

Proof of Lemma 4.2. We prove Lemma 4.2 only for the mid-point rule because it may be proved in a similar fashion for the trapezoidal rule. To this end, denote uniform partitions of the intervals $[x_i, x_{i+1}]$ and $[t_j, t_{j+1}]$ by

$$x_i = x_i^0 < x_i^1 < \cdots < x_i^{\hat{m}} = x_{i+1}, \text{ and } \quad t_j = t_j^0 < t_j^1 < \cdots < t_j^{\hat{n}} = t_{j+1},$$

respectively, where $x_i^k = x_i + k\hat{h}$ and $t_j^k = t_j + k\hat{\delta}$; and $\hat{h} = h/\hat{m}$ and $\hat{\delta} = \delta/\hat{n}$ are the numerical integration mesh sizes. By (A.1), we have

$$\left|\int_{t_j^k}^{t_j^{k+1}} \sigma(x_i, x_{i+1}; t)\,dt - \hat{\delta}\sigma(x_i, x_{i+1}; t_j^{k+1/2})\right| \le C\,\hat{\delta}^{2+1/q}\|\sigma_{tt}(x_i, x_{i+1}; \cdot)\|_{L^p(t_j^k, t_j^{k+1})},$$

$$\text{and}\quad \left|\int_{x_i^k}^{x_i^{k+1}} u(x; t_j, t_{j+1})\,dx - \hat{h}u(x_i^{k+1/2}; t_j, t_{j+1})\right| \le C\,\hat{h}^{2+1/q}\|u_{xx}(\cdot; t_j, t_{j+1})\|_{L^p(x_i^k, x_i^{k+1})},$$

which, together with (4.3), (4.4), and the triangle and the Hölder inequalities, implies

$$\begin{aligned}
&|K_{ij}|^{1/q}\left\|\mathbf{div}_{\mathcal{T}}\,\mathbf{f}(u) - \mathrm{avg}_{\mathcal{T}}\,\mathbf{div}\,\mathbf{f}(u)\right\|_{L^p(K_{ij})} = |K_{ij}|\left|\mathrm{avg}_{K_{ij}}\mathbf{div}\,\mathbf{f}(u) - \mathbf{div}_{\mathcal{T}}\,\mathbf{f}\big(u(\mathbf{m}_{ij})\big)\right| \\
&\le C\left\{h\hat{\delta}^{2+1/q}\sum_{k=0}^{\hat{n}-1}\|\sigma_{tt}(x_i, x_{i+1}; \cdot)\|_{L^p(t_j^k, t_j^{k+1})} + \delta\hat{h}^{2+1/q}\sum_{k=0}^{\hat{m}-1}\|u_{xx}(\cdot; t_j, t_{j+1})\|_{L^p(x_i^k, x_i^{k+1})}\right\} \\
&\le C\left\{h\hat{\delta}^{2+1/q}\hat{n}^{1/q}\|\sigma_{tt}(x_i, x_{i+1}; \cdot)\|_{L^p(t_j, t_{j+1})} + \delta\hat{h}^{2+1/q}\hat{m}^{1/q}\|u_{xx}(\cdot; t_j, t_{j+1})\|_{L^p(x_i, x_{i+1})}\right\}.
\end{aligned}$$

This completes the proof of Lemma 4.2. □

To prove Lemma 4.3, we need to estimate an error bound of numerical integration for piece-wise smooth and discontinuous integrant over interval $[0, \rho]$.

Lemma A.1. *For any $0 < \hat{\rho} < \rho/2$, assume that $\varphi \in C^1\big((0, \hat{\rho})\big) \cap C^1\big((\hat{\rho}, \rho)\big)$ is a piece-wise C^1 function. Denote by $j_\varphi = |\varphi(\hat{\rho}^+) - \varphi(\hat{\rho}^-)|$ the jump of $\varphi(s)$ at $s = \hat{\rho}$. Then there exists a positive constant C such that*

$$\begin{aligned}
\big|I(\varphi) - Q(\varphi; 0, \rho, 1)\big| &\le C\rho^{1+1/q}\|\varphi'\|_{L^p\left((0,\rho)\setminus\{\hat{\rho}\}\right)} + \begin{cases} \hat{\rho}\, j_\varphi, & \textit{mid-point}, \\ \left|\dfrac{\rho}{2} - \hat{\rho}\right| j_\varphi, & \textit{trapezoidal} \end{cases} \\
&\le C\rho^{1+1/q}\|\varphi'\|_{L^p\left((0,\rho)\setminus\{\hat{\rho}\}\right)} + \frac{\rho}{2} j_\varphi.
\end{aligned} \tag{A.2}$$

Proof. Denote the linear interpolant of φ on the interval $[0, \rho]$ by $\varphi_1(s) = \varphi(0)\dfrac{\rho - s}{\rho} + \varphi(\rho)\dfrac{s}{\rho}$. For any $s \in (0, \hat{\rho})$, by the fact that $\varphi(0) - \varphi_1(0) = 0$, a standard argument on the error bound of interpolant yields that there exists a $\xi_- \in (0, \hat{\rho})$ such that

$$\varphi(s) - \varphi_1(s) = \varphi'(\xi_-)s - \frac{s}{\rho}(\varphi(\rho) - \varphi(0)),$$

which implies

$$\int_0^{\hat{\rho}} (\varphi(s) - \varphi_1(s))\,ds = \int_0^{\hat{\rho}} \varphi'(\xi_-)s\,ds - \frac{\hat{\rho}^2}{2\rho}(\varphi(\rho) - \varphi(0)).$$

In a similar fashion, there exists a $\xi_- \in (\hat{\rho}, \rho)$ such that

$$\int_{\hat{\rho}}^{\rho} (\varphi(s) - \varphi_1(s))\,ds = \int_{\hat{\rho}}^{\rho} \varphi'(\xi_+)(s - \rho)\,ds + \frac{(\rho - \hat{\rho})^2}{2\rho}(\varphi(\rho) - \varphi(0)).$$

Combining the above inequalities and using the triangle and the Hölder inequalities give

$$\begin{aligned}
\big|I(\varphi) - Q_t(\varphi)\big| &= \left|\int_0^{\hat{\rho}}\varphi'(\xi_-)s ds + \int_{\hat{\rho}}^{\rho}\varphi'(\xi_+)(s - \rho)ds + \frac{\rho - 2\hat{\rho}}{2}(\varphi(\rho) - \varphi(0))\right| \\
&\le \frac{1}{(1+q)^{1/q}}\rho^{1+1/q}\left(\|\varphi'\|_{L^p(0,\hat{\rho})} + \|\varphi'\|_{L^p(\hat{\rho},\rho)}\right) + \left|\frac{\rho}{2} - \hat{\rho}\right| |\varphi(\rho) - \varphi(0)| \\
&\le \frac{2^{1/q}}{(1+q)^{1/q}}\rho^{1+1/q}\|\varphi'\|_{L^p\left((0,\rho)\setminus\{\hat{\rho}\}\right)} + \left|\frac{\rho}{2} - \hat{\rho}\right| |\varphi(\rho) - \varphi(0)|.
\end{aligned}$$

It follows from the triangle and the Hölder inequalities that

$$\begin{aligned}
|\varphi(\rho) - \varphi(0)| &\le \left|\int_{\hat{\rho}}^{\rho}\varphi'(s)\,ds\right| + \left|\int_0^{\hat{\rho}}\varphi'(s)\,ds\right| + j_\varphi \\
&\le \rho^{1/q}\left(\|\varphi'\|_{L^p(0,\hat{\rho})} + \|\varphi'\|_{L^p(\hat{\rho},\rho)}\right) + j_\varphi \le (2\rho)^{1/q}\|\varphi'\|_{L^p((0,\rho)\setminus\{\hat{\rho}\})} + j_\varphi.
\end{aligned}$$

Now, the above two inequalities and the fact that $\left|\dfrac{\rho}{2} - \hat{\rho}\right| \le \dfrac{\rho}{2}$ imply (A.2) for the trapezoidal rule.

To prove the validity of (A.2) for the mid-point rule, note that for any $s \in (0, \hat{\rho})$ we have

$$\begin{aligned}\varphi(s) - \varphi(\rho/2) &= \int_{\hat{\rho}}^{s} \varphi'(s)\,ds + \int_{\rho/2}^{\hat{\rho}} \varphi'(s)\,ds + \varphi(\hat{\rho}^-) - \varphi(\hat{\rho}^+) \\ &\le (\hat{\rho} - s)^{1/q}\|\varphi'\|_{L^p(s,\hat{\rho})} + (\rho/2 - \hat{\rho})^{1/q}\|\varphi'\|_{L^p(\hat{\rho},\rho/2)} + \varphi(\hat{\rho}^-) - \varphi(\hat{\rho}^+),\end{aligned}$$

which, together with the triangle inequality, implies

$$\left|\int_0^{\hat{\rho}} \left(\varphi(s) - \varphi(\rho/2)\right) ds\right| \le \left(\frac{\rho}{2}\right)^{1+1/q} \left(\|\varphi'\|_{L^p(0,\hat{\rho})} + \|\varphi'\|_{L^p(\hat{\rho},\rho/2)}\right) + \hat{\rho} j_\varphi.$$

Similarly, we have

$$\left|\int_{\hat{\rho}}^{\rho} \left(\varphi(s) - \varphi(\rho/2)\right) ds\right| \le \frac{2q}{1+q}\left(\frac{\rho}{2}\right)^{1+1/q} \|\varphi'\|_{L^p(\hat{\rho},\rho)}.$$

Now, (A.2) for the mid-point rule follows from the triangle inequality and the above two inequalities. This completes the proof of the lemma. □

Now, we are ready to prove the validity of Lemma 4.3.

Proof of Lemma 4.3. By the assumption, the discontinuous interface Γ_{ij} intercepts two horizontal edges at $(\hat{x}_i^l, t_l)$ for $l = j, j+1$. Without loss of generality, assume that $\hat{x}_i^j \in \left(x_i^{k_j}, x_i^{k_j+1}\right)$ and $\hat{x}_i^{j+1} \in \left(x_i^{k_{j+1}}, x_i^{k_{j+1}+1}\right)$ for some k_j and k_{j+1} in $\{0, 1, \ldots, \hat{m}\}$. Let $\hat{I}_{ij} = \left(x_i^{k_j}, x_i^{k_j+1}\right) \cup \left(x_i^{k_j}, x_i^{k_j+1}\right)$. The same proof of Lemma 4.2 leads to

$$\begin{aligned}&\left\|\mathbf{div}_{\mathcal{T}}\,\mathbf{f}(u) - \mathrm{avg}_{\mathcal{T}}\,\mathbf{div}\,\mathbf{f}(u)\right\|_{L^p(K_{ij})} \\ &\le C\left\{\frac{h^{1/p}\delta^2}{\hat{n}^2}\|\sigma_{tt}(x_i, x_{i+1}; \cdot)\|_{L^p(t_j,t_{j+1})} + \frac{h^2\delta^{1/p}}{\hat{m}^2}\|u_{xx}(\cdot; t_j, t_{j+1})\|_{L^p\left((x_i,x_{i+1})\setminus\hat{I}_{ij}\right)}\right\} \\ &\quad + \frac{\delta}{(h\delta)^{1/q}}\sum_{l=j}^{j+1}\left|\int_{x_i^{k_l}}^{x_i^{k_l+1}} u(x; t_j, t_{j+1})\,dx - \hat{h}u(x_i^{k_l+\frac{1}{2}}; t_j, t_{j+1})\right|,\end{aligned}$$

which, together with Lemma A.1, implies

$$\begin{aligned}&\left\|\mathbf{div}_{\mathcal{T}}\,\mathbf{f}(u) - \mathrm{avg}_{\mathcal{T}}\,\mathbf{div}\,\mathbf{f}(u)\right\|_{L^p(K_{ij})} \\ &\le C\left(\frac{h^{1/p}\delta^2}{\hat{n}^2} + \frac{h^2\delta^{1/p}}{\hat{m}^2}\right) + \frac{\hat{h}\delta}{(h\delta)^{1/q}}\sum_{l=j}^{j+1}\left\{C\hat{h}^{1/q}\|u_x(\cdot; t_j, t_{j+1})\|_{L^p\left((x_i,x_{i+1})\setminus\{\hat{x}_i^l\}\right)} + [\![u(\hat{x}_i^l, t_l)]\!]\right\}.\end{aligned}$$

Now, (4.7) follows from $\hat{h} = h/\hat{m}$. This completes the proof of Lemma 4.3. □

References

[1] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (2) (1981) 357–372.
[2] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (2) (1988) 439–471.
[3] C.-W. Shu, Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws, in: Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Springer, 1998, pp. 325–432.
[4] D. Gottlieb, C.-W. Shu, On the Gibbs phenomenon and its resolution, SIAM Rev. 39 (4) (1997) 644–668.
[5] J.S. Hesthaven, Numerical Methods for Conservation Laws: From Analysis to Algorithms, SIAM, 2017.
[6] J.S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Springer Science & Business Media, 2007.
[7] R.J. LeVeque, Numerical Methods for Conservation Laws, Birkhäuser, Boston, 1992.
[8] B. Cockburn, C.-W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws, Math. Comp. 52 (1989) 411–435.
[9] F. Brezzi, L.D. Marini, E. Süli, Discontinuous Galerkin methods for first-order hyperbolic problems, Math. Models Methods Appl. Sci. 14 (12) (2004) 1893–1903.
[10] W. Dahmen, C. Huang, C. Schwab, G. Welper, Adaptive Petrov–Galerkin methods for first order transport equations, SIAM J. Numer. Anal. 50 (5) (2012) 2420–2445.
[11] L. Demkowicz, J. Gopalakrishnan, A class of discontinuous Petrov–Galerkin methods. Part I: The transport equation, Comput. Methods Appl. Mech. Engrg. 199 (23–24) (2010) 1558–1572.
[12] E. Burman, A posteriori error estimation for interior penalty finite element approximations of the advection-reaction equation, SIAM J. Numer. Anal. 47 (5) (2009) 3584–3607.
[13] P. Houston, J.A. Mackenzie, E. Süli, G. Warnecke, A posteriori error analysis for numerical approximations of friedrichs systems, Numer. Math. 82 (3) (1999) 433–470.
[14] P. Houston, R. Rannacher, E. Süli, A posteriori error analysis for stabilised finite element approximations of transport problems, Comput. Methods Appl. Mech. Engrg. 190 (11–12) (2000) 1483–1508.

[15] Z. Cai, J. Chen, M. Liu, X. Liu, Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs, J. Comput. Phys. 420 (2020) 109707.
[16] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[17] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1139–1364.
[18] Z. Cai, J. Chen, M. Liu, Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation, J. Comput. Phys. 443 (2021) 110514.
[19] P. Bochev, J. Choi, Improved least-squares error estimates for scalar hyperbolic problems, Comput. Methods Appl. Math. 1 (2) (2001) 115–124.
[20] H. De Sterck, T.A. Manteuffel, S.F. McCormick, L. Olson, Least-squares finite element methods and algebraic multigrid solvers for linear hyperbolic PDEs, SIAM J. Sci. Comput. 26 (1) (2004) 31–54.
[21] A. Harten, B. Engquist, S. Osher, S.R. Chakravarthy, Uniformly high order accurate essentially non-oscillatory schemes, III, in: Upwind and High-Resolution Schemes, Springer, 1987, pp. 218–290.
[22] Z. Cai, J. Chen, M. Liu, Least-squares ReLU neural network (LSNN) method for scalar nonlinear hyperbolic conservation law, Appl. Numer. Math. 174 (2022) 163–176.
[23] Y. Bar-Sinai, S. Hoyer, J. Hickey, M.P. Brenner, Learning data-driven discretizations for partial differential equations, PNAS 116 (31) (2019) 15344–15349.
[24] Z. Cai, J. Chen, M. Liu, Finite volume least-squares neural network (FV-LSNN) method for scalar nonlinear hyperbolic conservation laws, 2021, arXiv:2110.10895, [math.NA].
[25] O. Fuks, H. Tchelepi, Limitations of physics informed machine learning for nonlinear two-phase transport porous media, J. Mach. Learn. Model. Comput. 1 (1) (2020) 19–37.
[26] R.G. Patel, I. Manickam, N.A. Trask, M.A. Wood, M. Lee, I. Tomas, E.C. Cyr, Thermodynamically consistent physics-informed neural networks for hyperbolic systems, J. Comput. Phys. 449 (2022).
[27] H. De Sterck, T.A. Manteuffel, S.F. McCormick, L. Olson, Numerical conservation properties of H(div)-conforming least-squares finite element methods for the Burgers equation, SIAM J. Sci. Comput. 26 (5) (2005) 1573–1597.
[28] D.P. Kingma, J. Ba, ADAM: A method for stochastic optimization, in: International Conference on Representation Learning, San Diego, 2015, arXiv preprint arXiv:1412.6980.
[29] M. Liu, Z. Cai, J. Chen, Adaptive two-layer ReLU neural network: I. best least-squares approximation, Comput. Math. Appl. 113 (2022) 34–44.
[30] J.W. Thomas, Numerical Partial Differential Equations: Finite Difference Methods, Vol. 22, Springer Science & Business Media, 2013.
[31] D.I. Ketcheson, R.J. LeVeque, M.J. del Razo, Riemann Problems and Jupyter Solutions, SIAM, Philadelphia, 2020.
[32] Z. Zhao, Y. Chen, J. Qiu, A hybrid Hermite WENO scheme for hyperbolic conservation laws, J. Comput. Phys. 405 (2020) 109175.
[33] S. Osher, Riemann solvers, the entropy condition, and difference approximations, SIAM J. Numer. Anal. 21 (1984) 217–235.
[34] J.-L. Guermond, M. Nazarov, A maximum-principle preserving C^0 finite element method for scalar conservation equations, Comput. Methods Appl. Mech. Engrg. 272 (2014) 198–213.
[35] M. Liu, Z. Cai, Adaptive two-layer ReLU neural network: II. Ritz approximation to elliptic PDEs, Comput. Math. Appl. 113 (2022) 103–116.
[36] Z. Cai, J. Chen, M. Liu, Self-adaptive deep neural network: Numerical approximation to functions and PDEs, J. Comput. Phys. 455 (2022) 111021.