Lift-and-Embed Learning Methods for Hyperbolic Conservation Laws

Qi Sun

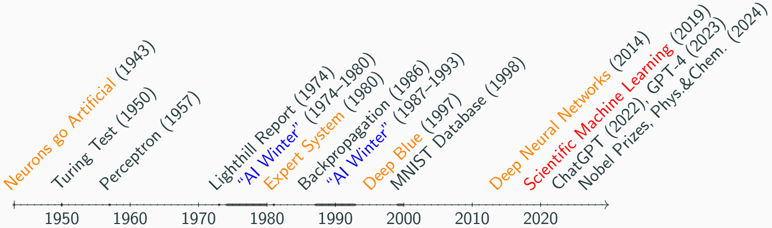
School of Mathematical Sciences, Tongji University

同济大学数学科学学院
School of Mathematical Sciences, Tongji University

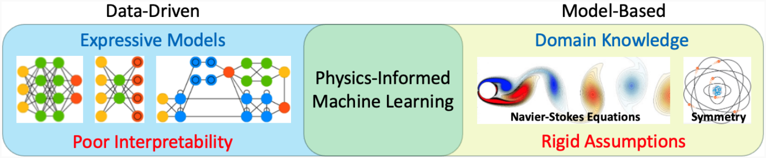
Outline

1. Background

The Evolution of Artificial Intelligence

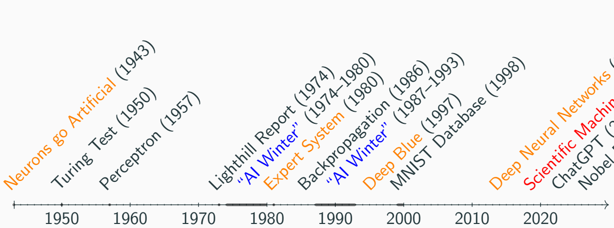


SciML is a new discipline that blends scientific computing and machine learning.

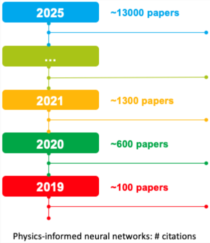


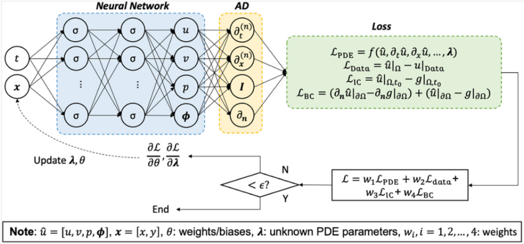
§S. L. Brunton et. al., Machine learning for fluid mechanics, 2020
§G. E. Karniadakis et. al., Physics-informed machine learning, 2021
§S. Cuomo et. al., SciML through PINNs: Where we are and what's next, 2022
§S. L. Brunton et. al., Promising directions of machine learning for PDEs, 2023
§T. D. Ryck et. al., Numerical analysis of PINNs and models in physics-informed machine learning, 2024

The Evolution of Artificial Intelligence



SciML is a new discipline that blends scientific computing and machine learning.

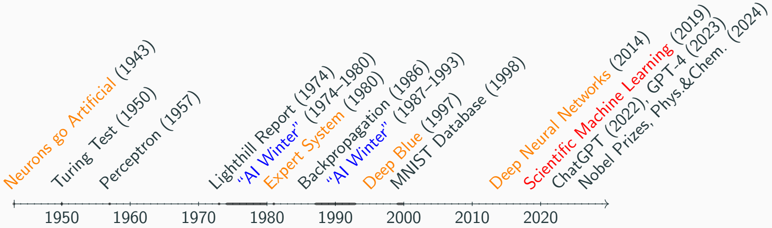




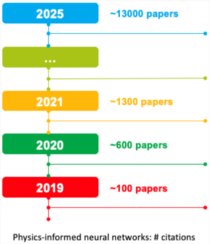
§ I. E. Lagaris et. al., Neural-network methods for boundary value problems with irregular boundaries, 2000

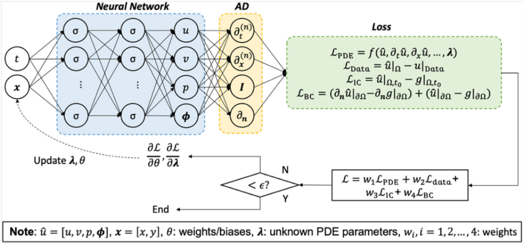
§ M. Raissi et. al., PINNs: DL for solving forward/inverse problems involving nonlinear PDEs, 2019

The Evolution of Artificial Intelligence



SciML is a new discipline that blends scientific computing and machine learning.





Q1: How to embed deeper theoretical insights into scientific machine learning?

§ I. E. Lagaris et. al., Neural-network methods for boundary value problems with irregular boundaries, 2000

§ M. Raissi et. al., PINNs: DL for solving forward/inverse problems involving nonlinear PDEs, 2019

Neural Networks as Universal Approximators

Cybenko's Universal Approximation Theorem

Feed-forward networks with only one hidden layer and non-polynomial activation functions are dense in the space of continuous functions.

§G. Cybenko, Approximation by superpositions of a sigmoidal function, 1989

Neural Networks as Universal Approximators

Cybenko's Universal Approximation Theorem

Feed-forward networks with only one hidden layer and non-polynomial activation functions are dense in the space of continuous functions.

§G. Cybenko, Approximation by superpositions of a sigmoidal function, 1989

Weierstrass Approximation Theorem

Polynomials can uniformly approximate continuous functions over compact sets.

§K. Weierstrass, On the possibility of giving an analytic representation to an arbitrary real function, 1885

Neural Networks as Universal Approximators

Cybenko's Universal Approximation Theorem

Feed-forward networks with only one hidden layer and non-polynomial activation functions are dense in the space of continuous functions.

§G. Cybenko, Approximation by superpositions of a sigmoidal function, 1989

Weierstrass Approximation Theorem

Polynomials can uniformly approximate continuous functions over compact sets.

§K. Weierstrass, On the possibility of giving an analytic representation to an arbitrary real function, 1885

Barron's Approximation Theorem

Feed-forward networks with a single hidden layer of m neurons can approximate a large class of functions with **a dimension-independent rate**.

§A. R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, 1993

- advances in tackling the curse of dimensionality yields practical approaches for solving high-dimensional partial differential equations
  - × high-dimensional Poisson equation, heat equation, ...
  - ✓ Hamilton-Jacobi-Bellman equation, Schrödinger equation, ...

§Z. Hao et. al., PINNacle: A comprehensive benchmark of PINNs for solving PDEs, 2024

§W. Cai et. al., Martingale deep learning for high dimensional PDEs and stochastic optimal controls, 2024

Neural Networks as Universal Approximators

Cybenko's Universal Approximation Theorem

Feed-forward networks with only one hidden layer and non-polynomial activation functions are dense in the space of continuous functions.

§G. Cybenko, Approximation by superpositions of a sigmoidal function, 1989

Weierstrass Approximation Theorem

Polynomials can uniformly approximate continuous functions over compact sets.

§K. Weierstrass, On the possibility of giving an analytic representation to an arbitrary real function, 1885

Barron's Approximation Theorem

Feed-forward networks with a single hidden layer of m neurons can approximate a large class of functions with **a dimension-independent rate**.

§A. R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, 1993

- advances in tackling the curse of dimensionality yields practical approaches for solving high-dimensional partial differential equations
  - × high-dimensional Poisson equation, heat equation, ...
  - ✓ Hamilton-Jacobi-Bellman equation, Schrödinger equation, ...

Q2: When and how are neural networks applied to low-dimensional problems?

§Z. Hao et. al., PINNacle: A comprehensive benchmark of PINNs for solving PDEs, 2024

§W. Cai et. al., Martingale deep learning for high dimensional PDEs and stochastic optimal controls, 2024

2. Hyperbolic Equations with Jump Discontinuities

Scalar Hyperbolic Equations

Consider the scalar hyperbolic conservation law

$$\begin{aligned} \partial_t u + \nabla \cdot f(u) &= 0, \quad (x, t) \in \mathbb{R}^d \times (0, T], \\ u(x, 0) &= u_0(x), \qquad x \in \mathbb{R}^d, \end{aligned}$$

where $u(x, t)$ is a conserved quantity, $f = (f_1, \cdots, f_d)^T$ the outward flux vector. A key issue is the development of discontinuous solutions, even if $u_0(x)$ is smooth.

Scalar Hyperbolic Equations

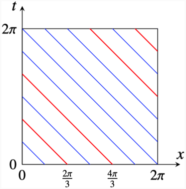
Consider the scalar hyperbolic conservation law

$$\partial_t u + \nabla \cdot f(u) = 0, \quad (x,t) \in \mathbb{R}^d \times (0, T],$$
$$u(x,0) = u_0(x), \qquad x \in \mathbb{R}^d,$$

where $u(x,t)$ is a conserved quantity, $f = (f_1, \cdots, f_d)^T$ the outward flux vector. A key issue is the development of discontinuous solutions, even if $u_0(x)$ is smooth.

Linear Convection Equation: consider $f(u) = -u$ in the one-dimension, namely,

$$\begin{aligned}
&\partial_t u - \partial_x u = 0, && \text{for } (x,t) \in [0, 2\pi] \times (0, 2\pi],\\
&u_0(x) = \begin{cases} 0, & \text{for } x \in [0, \frac{2\pi}{3}),\\ 1, & \text{for } x \in [\frac{2}{3}\pi, \frac{4\pi}{3}),\\ 0, & \text{for } x \in [\frac{4}{3}\pi, 2\pi],\end{cases}\\
&u(0,t) = u(2\pi, t), && \text{for } t \in (0, 2\pi],
\end{aligned}$$



in which $u(x,t)$ is constant along the characteristic line. To be specific,

$$\frac{dx(t)}{dt} = f'(u(x(t),t)) = -1 \;\Rightarrow\; x(t) = -t + x_0 \;\Rightarrow\; \frac{du(x(t),t)}{dt} = 0,$$
$$u(x(t),t) = u(x_0, 0) \;\Rightarrow\; u(x,t) = u_0(x+t).$$

§ J. Hesthaven, Numerical methods for conservation laws: From analysis to algorithms, 2017

Scalar Hyperbolic Equations

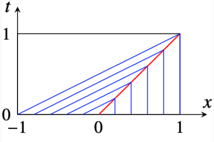
Consider the scalar hyperbolic conservation law

$$\begin{aligned} \partial_t u + \nabla \cdot f(u) = 0, &\quad (x,t) \in \mathbb{R}^d \times (0, T], \\ u(x,0) = u_0(x), &\quad x \in \mathbb{R}^d, \end{aligned}$$

where $u(x,t)$ is a conserved quantity, $f = (f_1, \cdots, f_d)^T$ the outward flux vector. A key issue is the development of discontinuous solutions, even if $u_0(x)$ is smooth.

Inviscid Burgers' Equation: consider $f(u) = \frac{1}{2}u^2$ in the one-dimension, namely,

$$\partial_t u + u\partial_x u = 0, \quad \text{for } (x,t) \in [-1,1] \times (0,1],$$
$$u_0(x) = \begin{cases} 2, & \text{for } x \in [-1, 0), \\ 0, & \text{for } x \in [0, 1], \end{cases}$$



in which $u(x,t)$ is constant along the characteristic line. To be specific,

$$\frac{dx(t)}{dt} = f'(u(x(t),t)) = u(x(t),t) \Rightarrow x(t) = u_0(x_0)t + x_0 \Rightarrow \frac{du(x(t),t)}{dt} = 0,$$

where a shock curve $x = \gamma(t)$ is formed due to characteristic intersection, and

$$\frac{d\gamma(t)}{dt} = s = \frac{[\![f(u)]\!]}{[\![u]\!]} = 1 \;\Rightarrow\; u(x,t) = u_0(x - st).$$

§ J. Hesthaven, Numerical methods for conservation laws: From analysis to algorithms, 2017

Classical numerical approaches using uniform meshes, such as the Lax-Wendroff and upwind schemes, suffer from dispersion or dissipation issues.

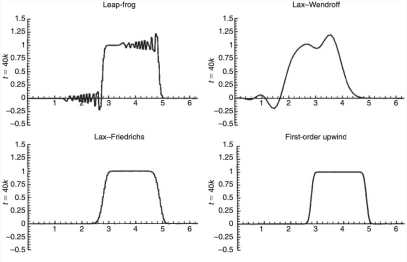


Figure 7.1.1. Solution at $t = 40k$ of $u_t + u_x = 0$ with initial data (7.1.2).

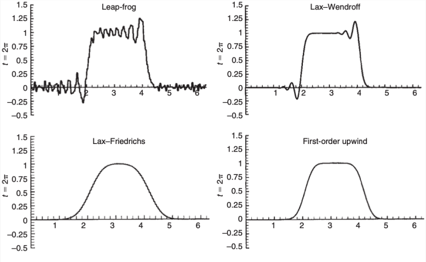


Figure 7.1.2. Solution at $t = 2\pi$ of $u_t + u_x = 0$ with initial data (7.1.2).

§ B. Gustafsson, Time-dependent problems and difference methods, 2013

Mesh-Based Numerical Solvers

Classical numerical approaches using uniform meshes, such as the Lax-Wendroff and upwind schemes, suffer from dispersion or dissipation issues.

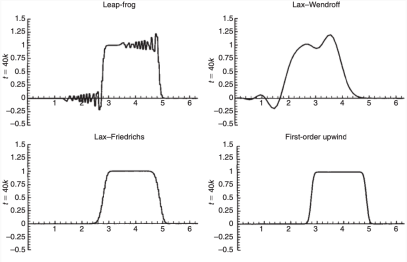


Figure 7.1.1. Solution at $t = 40k$ of $u_t + u_x = 0$ with initial data (7.1.2).

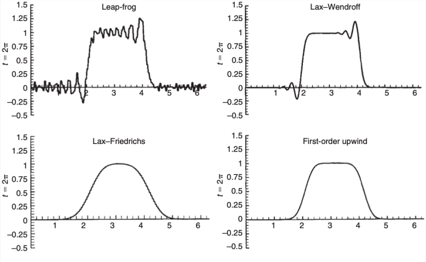


Figure 7.1.2. Solution at $t = 2\pi$ of $u_t + u_x = 0$ with initial data (7.1.2).

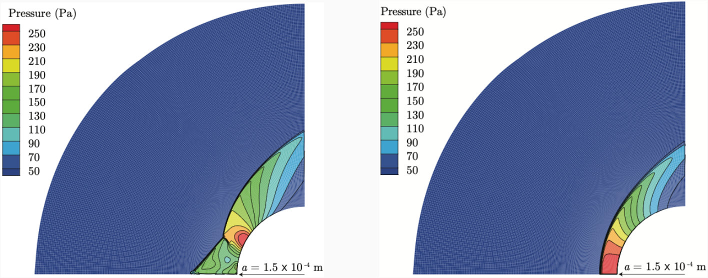
§ B. Gustafsson, Time-dependent problems and difference methods, 2013

Various refinements are developed to capture sharp solution transitions, e.g.,

- ENO/WENO-based finite volume schemes;
- discontinuous/adaptive finite element methods.

Mesh-Based Numerical Solvers

Classical numerical approaches using uniform meshes, such as the Lax-Wendroff and upwind schemes, suffer from dispersion or dissipation issues.



§ J. Bruns, Physical diffusion cures the Carbuncle problem, 2015

Various refinements are developed to capture sharp solution transitions, e.g.,

- ENO/WENO-based finite volume schemes;
- discontinuous/adaptive finite element methods.

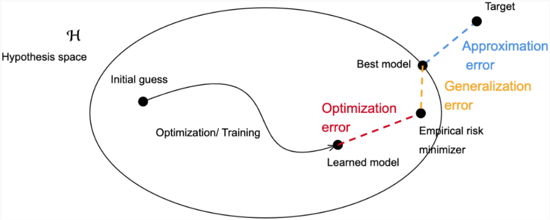
However, anomalous solutions are reported for complex problems, and challenges remain in balancing algorithmic efficiency, accuracy, and robustness.

§ J. Abbasi et. al., Challenges and advancements in modeling shock fronts with PINNs, 2025

Overview of Deep Learning-Based Solvers

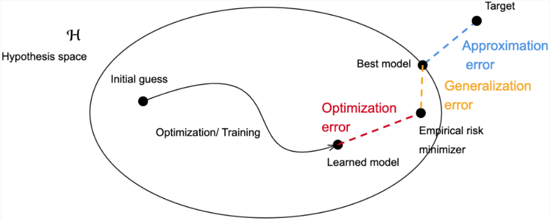
Overview of Current Research

Unlike mesh-based approximations of differential operators, deep learning methods use automatic differentiation to avoid dispersion and dissipation issues



Overview of Current Research

Unlike mesh-based approximations of differential operators, deep learning methods use automatic differentiation to avoid dispersion and dissipation issues, but vanilla learning approaches exhibit poor performance in handling discontinuities.



"Construction of Loss Function"

Strong Form: including a proper diffusion term to reduce hyperbolicity

$$\partial_t u + \nabla \cdot f(u) = 0 \;\Rightarrow\; \partial_t u + \nabla \cdot f(u) = \epsilon \Delta u$$

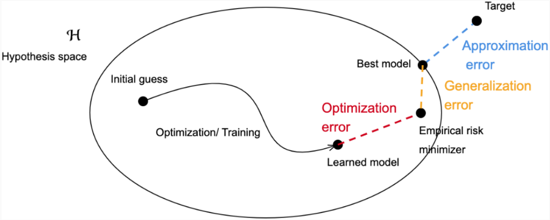
followed by the minimization of least square residual in a pointwise manner

$$\min_\theta \|\partial_t \hat{u} + \nabla \cdot f(\hat{u}) - \epsilon \Delta \hat{u}\|^2_{L^2(\Omega_T)} + \beta_{\mathrm{I}} \|\hat{u} - u_0\|^2_{L^2(\Omega_0)} + \beta_{\mathrm{B}} \|f(\hat{u}) \cdot \boldsymbol{n} - g\|^2_{L^2(\Gamma_{in})}$$

where $u(x,t)$ is parametrized using a neural network $\hat{u}(x,t;\theta)$ with parameter θ.

Overview of Current Research

Unlike mesh-based approximations of differential operators, deep learning methods use automatic differentiation to avoid dispersion and dissipation issues, but vanilla learning approaches exhibit poor performance in handling discontinuities.



"Construction of Loss Function"

Strong Form: including a proper diffusion term to reduce hyperbolicity

$$\partial_t u + \nabla \cdot f(u) = 0 \ \Rightarrow \ \partial_t u + \nabla \cdot f(u) = \epsilon \Delta u$$

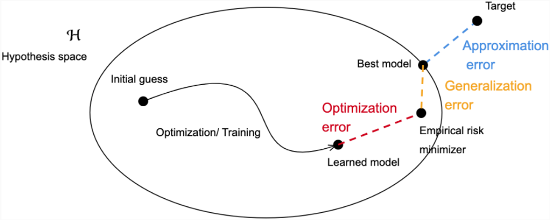
followed by the minimization of least square residual in a pointwise manner

$$\min_\theta \|\partial_t \hat{u} + \nabla \cdot f(\hat{u}) - \epsilon \Delta \hat{u}\|^2_{L^2(\Omega_T)} + \beta_\mathrm{I} \|\hat{u} - u_0\|^2_{L^2(\Omega_0)} + \beta_\mathrm{B} \|f(\hat{u}) \cdot \boldsymbol{n} - g\|^2_{L^2(\Gamma_{in})}$$

where $u(x,t)$ is parametrized using a neural network $\hat{u}(x,t;\theta)$ with parameter θ. Unfortunately, it introduces modelling error and slows down the training process.

Overview of Current Research

Unlike mesh-based approximations of differential operators, deep learning methods use automatic differentiation to avoid dispersion and dissipation issues, but vanilla learning approaches exhibit poor performance in handling discontinuities.



"Construction of Loss Function"

Weak Form: shifts smoothness requirements to the test function

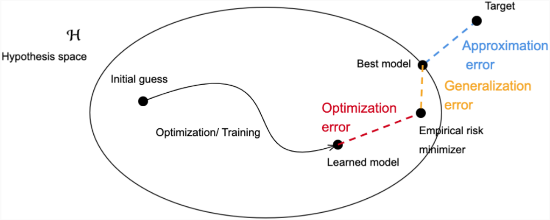
$$\int_0^T \int_\Omega (u\varphi_t + f(u)\varphi_x)\, dxdt + \int_\Omega u_0(x)\varphi(x,0)dx = 0, \quad \forall\ \varphi \in C_c^1(\Omega_T)$$

requiring the inclusion of entropy-admissible pair (η, q) to establish uniqueness

$$\min_\theta \max_\varphi \|\eta(\hat{u})\varphi_t + q(\hat{u})\varphi_x\|_{L^2(\Omega\times(0,T])} + \beta_\mathrm{I} \left\|\hat{u} - u_0\right\|^2_{L^2(\Omega_0)} + \beta_\mathrm{B} \left\|f(\hat{u}) \cdot \boldsymbol{n} - g\right\|^2_{L^2(\Gamma_{in})}$$

Overview of Current Research

Unlike mesh-based approximations of differential operators, deep learning methods use automatic differentiation to avoid dispersion and dissipation issues, but vanilla learning approaches exhibit poor performance in handling discontinuities.



"Construction of Loss Function"

Weak Form: shifts smoothness requirements to the test function

$$\int_0^T \int_\Omega \left(u\varphi_t + f(u)\varphi_x\right) dxdt + \int_\Omega u_0(x)\varphi(x,0)dx = 0, \quad \forall\ \varphi \in C_c^1(\Omega_T)$$

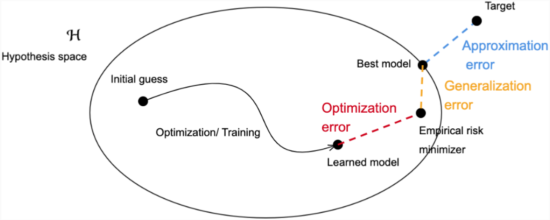
requiring the inclusion of entropy-admissible pair (η, q) to establish uniqueness

$$\min_\theta \max_\varphi \|\eta(\hat{u})\varphi_t + q(\hat{u})\varphi_x\|_{L^2(\Omega\times(0,T])} + \beta_{\mathrm{I}}\,\|\hat{u} - u_0\|^2_{L^2(\Omega_0)} + \beta_{\mathrm{B}}\,\|f(\hat{u})\cdot \boldsymbol{n} - g\|^2_{L^2(\Gamma_{in})}$$

However, difficulty in recovering discontinuities through neural networks persists.

Overview of Current Research

Unlike mesh-based approximations of differential operators, deep learning methods use automatic differentiation to avoid dispersion and dissipation issues, but vanilla learning approaches exhibit poor performance in handling discontinuities.



Adaptive Sampling Strategy: generate points on regions with higher errors

§Z. Mao et. al., Physics-informed neural networks for high-speed flows, 2020

Architecture Design: choose proper activation functions and network structures

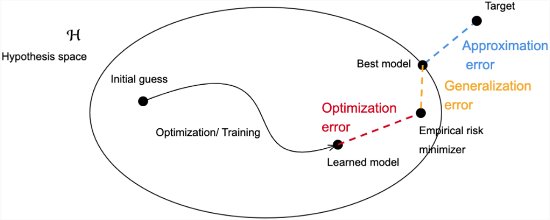
§D. Santa et. al., Discontinuous neural networks and discontinuity learning, 2023

Integration with Classical Methods: employing neural networks to learn, either partially or entirely, the finite volume or discontinuous Galerkin schemes

§Y. Bar-Sinai et. al., learning data-driven discretizations for partial differential equations, 2019

Overview of Current Research

Unlike mesh-based approximations of differential operators, deep learning methods use automatic differentiation to avoid dispersion and dissipation issues, but vanilla learning approaches exhibit poor performance in handling discontinuities.



Adaptive Sampling Strategy: generate points on regions with higher errors

[§] Z. Mao et. al., Physics-informed neural networks for high-speed flows, 2020

Architecture Design: choose proper activation functions and network structures

[§] D. Santa et. al., Discontinuous neural networks and discontinuity learning, 2023

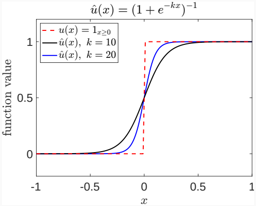
Integration with Classical Methods: employing neural networks to learn, either partially or entirely, the finite volume or discontinuous Galerkin schemes

[§] Y. Bar-Sinai et. al., learning data-driven discretizations for partial differential equations, 2019

Q3: Where is the breakthrough in next-generation scientific machine learning?

3. Lift-and-Embed Learning Methods

Toy Example: Approximation of Heaviside Function

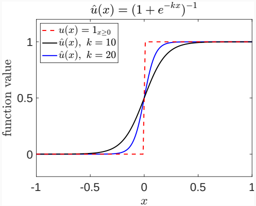


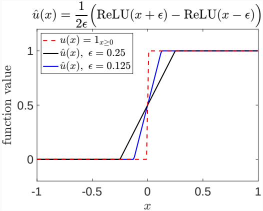
Kernel Smoothing: convolution with a suitable kernel function, e.g.,

$$u(x) = \mathbf{1}_{x\geq 0} \approx \hat{u}(x) = (u * G)(x) = \int_{-\infty}^{+\infty} u(\tau)G(x-\tau)\,d\tau = \frac{1}{1+e^{-kx}}$$

where $G(y) = ke^{-ky}(1+e^{-ky})^{-2}$ is Sigmoid kernel function and discontinuity is smoothed out in a manner analogous to the method of vanishing viscosity.

Toy Example: Approximation of Heaviside Function





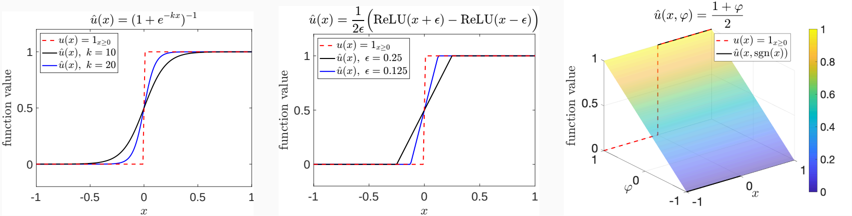
Finite Element Interpolation: approximation with piecewise linear elements on the partition $-1 = x_0 < x_1 < x_2 < x_3 = 1$ with $x_2 = -x_1 = \epsilon$

$$u(x) \approx \hat{u}(x) = \sum_{j=0}^{3} u(x_j)\phi_j(x) = \frac{1}{2\epsilon}\Big(\mathrm{ReLU}(x+\epsilon) - \mathrm{ReLU}(x-\epsilon)\Big)$$

where basis functions are rewritten in terms of $\mathrm{ReLU}(x) = \max(0, x)$. The latter is closely related to a single-hidden-layer feedforward neural network.

§Z. Cai et. al., Least-squares neural network method for linear advection-reaction equation, 2024
§Z. Cai et. al., Evolving neural network method for 1D scalar hyperbolic conservation laws, 2023

Toy Example: Approximation of Heaviside Function



Lift-and-Embed Approach: embedding non-smooth functions within a higher-dimensional space to achieve smoothness (not unique), for instance,

$$u(x) = \hat{u}(x, \mathrm{sgn}(x)) \quad \text{with} \quad \hat{u}(x, \varphi) = \frac{1+\varphi}{2}$$

as ensured by Tietze extension theorem: any real-valued, continuous function on a closed subset of a normal space can be extended to the entire space.

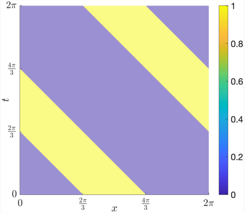
§J. R. Munkres, Topology, 2020

§W. Hu et. al., A discontinuity-capturing neural network with categorical embedding and applications, 2025

§Q. Sun et. al., Lift-and-Embed learning method for solving scalar hyperbolic equations, 2025

Identified Discontinuity Locations

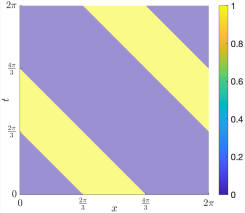
Linear Convection Equation Revisited

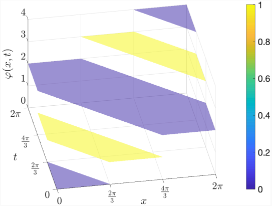


Recall the linear problem with 2π-periodic boundary condition, namely,

$$\begin{aligned} &\partial_t u(x,t) - \partial_x u(x,t) = 0, && \text{for } (x,t) \in \Omega = (0,2\pi) \times (0,2\pi], \\ &u_0(x) = H(x - \tfrac{2\pi}{3}) - H(x - \tfrac{4\pi}{3}), && \text{for } x \in (0,2\pi). \end{aligned}$$

Linear Convection Equation Revisited





Recall the linear problem with 2π-periodic boundary condition, namely,

$$\begin{aligned} &\partial_t u(x,t) - \partial_x u(x,t) = 0, && \text{for } (x,t) \in \Omega = (0,2\pi) \times (0,2\pi], \\ &u_0(x) = H(x - \tfrac{2\pi}{3}) - H(x - \tfrac{4\pi}{3}), && \text{for } x \in (0,2\pi). \end{aligned}$$

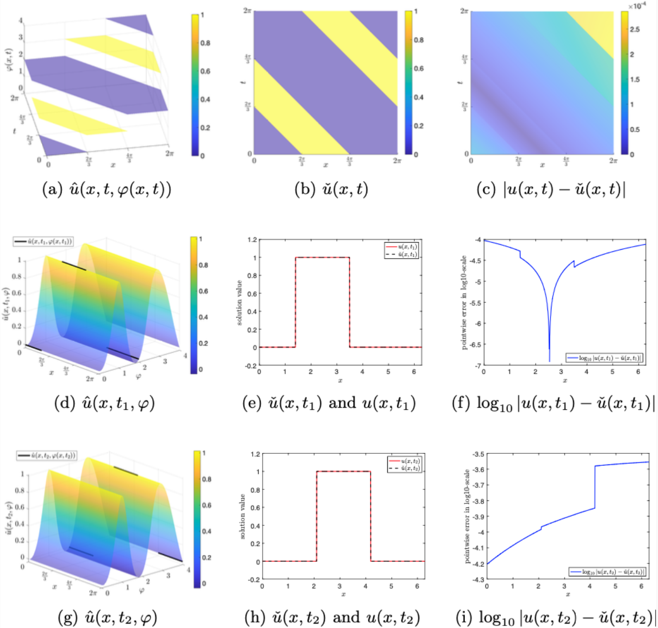
By introducing an augmented variable into our solution ansatz

$$u(x,t) = \hat{u}(x,t,\varphi(x,t)) \quad \text{with} \quad \varphi(x,t) = \sum_{i=1}^{2} \sum_{k=0}^{1} H(x - st - x_i - 2k\pi),$$

the original problem is embedded into a higher-dimensional space, that is,

$$\begin{aligned} &\partial_t \hat{u}(x,t,\varphi(x,t)) - \partial_x \hat{u}(x,t,\varphi(x,t)) = 0, && \text{for } (x,t) \in \Omega \setminus \Gamma, \\ &\hat{u}(x,t,\varphi^+(x,t)) - \hat{u}(x,t,\varphi^-(x,t)) = u_0^+(x_i) - u_0^-(x_i), && \text{for } (x,t) \in \Gamma, \\ &\hat{u}(x,0,\varphi(x,0)) = u_0(x), && \text{for } x \in (0,2\pi). \end{aligned}$$

Linear Convection Equation Revisited

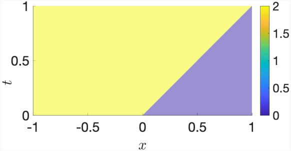


(a) $\hat{u}(x,t,\varphi(x,t))$ (b) $\check{u}(x,t)$ (c) $|u(x,t)-\check{u}(x,t)|$

(d) $\hat{u}(x,t_1,\varphi)$ (e) $\check{u}(x,t_1)$ and $u(x,t_1)$ (f) $\log_{10}|u(x,t_1)-\check{u}(x,t_1)|$

(g) $\hat{u}(x,t_2,\varphi)$ (h) $\check{u}(x,t_2)$ and $u(x,t_2)$ (i) $\log_{10}|u(x,t_2)-\check{u}(x,t_2)|$

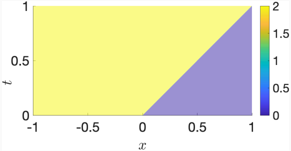
Inviscid Burgers' Equation Revisited

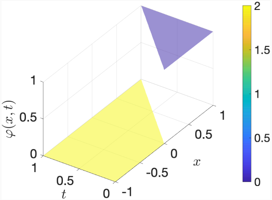


Recall the nonlinear problem with inflow boundary condition being omitted, i.e.,

$$\begin{aligned} &\partial_t u(x,t) + u(x,t)\partial_x u(x,t) = 0, \quad && \text{for } (x,t) \in \Omega = (-1,1) \times (0,1], \\ &u_0(x) = 2H(-x), && \text{for } x \in (-1,1), \end{aligned}$$

Inviscid Burgers' Equation Revisited





Recall the nonlinear problem with inflow boundary condition being omitted, i.e.,

$$\begin{aligned} &\partial_t u(x,t) + u(x,t)\partial_x u(x,t) = 0, && \text{for } (x,t) \in \Omega = (-1,1)\times(0,1], \\ &u_0(x) = 2H(-x), && \text{for } x \in (-1,1), \end{aligned}$$

By introducing an augmented variable into our solution ansatz

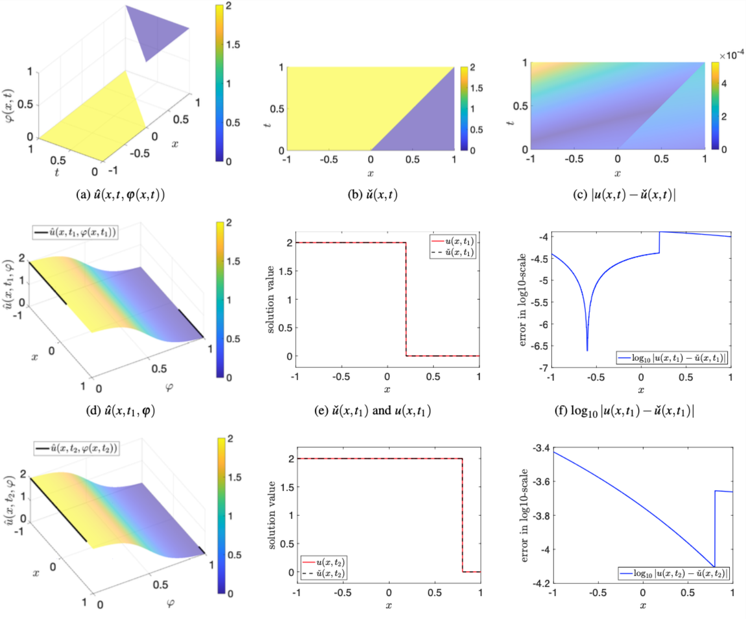
$$u(x,t) = \hat{u}(x,t,\varphi(x,t)) \quad \text{with} \quad \varphi(x,t) = H(x - st) = \mathbf{1}_{x \geq st},$$

in which $s = 1$, the governing equation can then be reformulated as

$$\begin{aligned} &\partial_t \hat{u}(x,t,\varphi(x,t)) - \hat{u}(x,t,\varphi(x,t))\partial_x \hat{u}(x,t,\varphi(x,t)) = 0, && \text{for } (x,t) \in \Omega \setminus \Gamma, \\ &\frac{[\![f(\hat{u})]\!]}{[\![\hat{u}]\!]} = \frac{1}{2}\left(\hat{u}(x,t,\varphi^+(x,t)) + \hat{u}(x,t,\varphi^-(x,t))\right) = s, && \text{on } \Gamma, \\ &\hat{u}(x,0,\varphi(x,0)) = u_0(x), && \text{for } x \in (-1,1). \end{aligned}$$

§Q. Sun et. al., Lift-and-Embed learning method for solving scalar hyperbolic equations, 2025

Inviscid Burgers' Equation Revisited



(a) $\hat{u}(x,t,\varphi(x,t))$ (b) $\check{u}(x,t)$ (c) $|u(x,t)-\check{u}(x,t)|$

(d) $\hat{u}(x,t_1,\varphi)$ (e) $\check{u}(x,t_1)$ and $u(x,t_1)$ (f) $\log_{10}|u(x,t_1)-\check{u}(x,t_1)|$

(g) $\hat{u}(x,t_2,\varphi)$ (h) $\check{u}(x,t_2)$ and $u(x,t_2)$ (i) $\log_{10}|u(x,t_2)-\check{u}(x,t_2)|$

Learning with Identified Discontinuity Locations

% *Preparation*

– generate collocation points X_{Intrr}, X_{Shock}, X_{Bndry}, and X_{Initl};

– calculate $\varphi(x,t)$ with *a-priori* knowledge of discontinuities;

% *Training Process*

– construct and initialize the network model $\hat{u}(x,t,\varphi;\theta)$;

while maximum number of epochs is not reached **do**

– network training on the shuffled dataset with a suitable learning rate, i.e.,

$$\theta^* = \arg\min_{\theta} L_{\mathsf{Intrr}}(\hat{u}) + \beta_{\mathsf{S}} L_{\mathsf{Shock}}(\hat{u}) + \beta_{\mathsf{B}} L_{\mathsf{Bndry}}(\hat{u}) + \beta_{\mathsf{I}} L_{\mathsf{Initl}}(\hat{u});$$

end while

% *Testing Process*

– forward pass of the trained model on the testing dataset, i.e.,

$$\check{u}(x,t) = \hat{u}(x,t,\varphi(x,t);\theta^*).$$

$$L_{\mathsf{Intrr}}(\hat{u}) = \int_0^T \int_{\Omega\backslash\Gamma} |\partial_t \hat{u} - \nabla \cdot f(\hat{u})|^2 dx dt, \quad L_{\mathsf{Initl}}(\hat{u}) = \int_{\Omega} |\hat{u} - u_0|^2 dx,$$

$$L_{\mathsf{Shock}}(\hat{u}) = \int_{\Gamma} |-s[\![\hat{u}]\!] + [\![f(\hat{u})]\!]|^2 ds, \quad L_{\mathsf{Bndry}}(\hat{u}) = \int_0^T \int_{\partial\Omega} |\hat{u} - g|^2 dx dt.$$

Unknown Discontinuity Locations

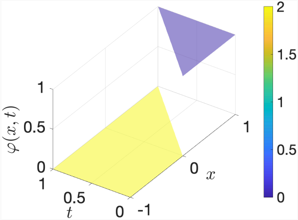
Inviscid Burgers' Equation

Consider the inviscid Burgers' equation with inflow boundary condition, namely,

$$\begin{aligned}
&\partial_t u(x,t) + u(x,t)\partial_x u(x,t) = 0, && \text{for } (x,t) \in \Omega = (-1,1) \times (0,1], \\
&u_0(x) = 2H(-x), && \text{for } x \in (-1,1),
\end{aligned}$$

where shock speed s is unknown.

Inviscid Burgers' Equation



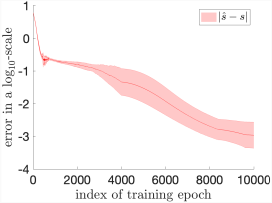
Consider the inviscid Burgers' equation with inflow boundary condition, namely,

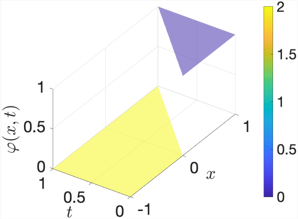
$$\begin{aligned} &\partial_t u(x,t) + u(x,t)\partial_x u(x,t) = 0, \quad && \text{for } (x,t) \in \Omega = (-1,1) \times (0,1], \\ &u_0(x) = 2H(-x), && \text{for } x \in (-1,1), \end{aligned}$$

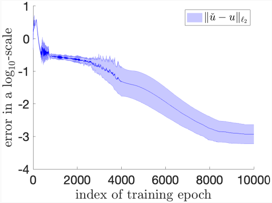
where shock speed s is unknown. With augmented variable being constructed in a similar fashion as before, namely,

$$u(x,t) = \hat{u}(x,t,\varphi(x,t)) \quad \text{with} \quad \varphi(x,t) = H(x - \hat{s}t),$$

Inviscid Burgers' Equation







Consider the inviscid Burgers' equation with inflow boundary condition, namely,

$$\begin{aligned} &\partial_t u(x,t) + u(x,t)\partial_x u(x,t) = 0, && \text{for } (x,t) \in \Omega = (-1,1) \times (0,1], \\ &u_0(x) = 2H(-x), && \text{for } x \in (-1,1), \end{aligned}$$

where shock speed s is unknown. With augmented variable being constructed in a similar fashion as before, namely,

$$u(x,t) = \hat{u}(x,t,\varphi(x,t)) \quad \text{with} \quad \varphi(x,t) = H(x - \hat{s}t),$$

shock speed can be inferred concurrently with the training of network solution

$$L_{\text{Shock}}(\hat{u}) \quad \Rightarrow \quad L_{\text{Shock}}^{\text{inv}}(\hat{u},\hat{s}) = \int_\Gamma \left| \hat{s}[\![\hat{u}]\!] - [\![f(\hat{u})]\!] \right|^2 ds$$

§Q. Sun et. al., Lift-and-Embed learning method for solving scalar hyperbolic equations, 2025

Learning with Unknown Discontinuity Locations

% *Preparation*

– generate datasets X_{Intrr}, X_{Bndry}, X_{Initl}, and temporal grid $\{t_i\}_{i=1}^n$;

– initialize $\{\hat{s}_i = \hat{s}(t_i)\}_{i=1}^n$ and treat them as extra trainable parameters;

% *Training Process*

– construct and initialize the network model $\hat{u}(x, t, \varphi; \theta)$;

while maximum number of epochs is not reached **do**

– numerically solve $\hat{\gamma}'(t) = \hat{s}(t)$, then reconstruct $\hat{s}(t)$ and $\hat{\gamma}(t)$;

– construct the augmented variable $\varphi(x, t)$ and resample dataset X_{Shock};

– network training on the shuffled dataset with a suitable learning rate, i.e.,

$$\theta^*, \hat{s}_i^* = \arg\min_{\theta, s_i} L_{\mathsf{Intrr}}(\hat{u}) + \beta_{\mathsf{S}} L_{\mathsf{Shock}}^{\mathsf{inv}}(\hat{u}, \hat{s}_i) + \beta_{\mathsf{B}} L_{\mathsf{Bndry}}(\hat{u}) + \beta_{\mathsf{I}} L_{\mathsf{Initl}}(\hat{u})$$

end while

% *Testing Process*

– forward pass of the trained model on the testing dataset, i.e.,

$$\check{u}(x, t) = \hat{u}(x, t, \varphi(x, t); \theta^*).$$

Learning with Unknown Discontinuity Locations

% *Preparation*

– generate datasets X_{Intrr}, X_{Bndry}, X_{Initl}, and temporal grid $\{t_i\}_{i=1}^n$;

– initialize $\{\hat{s}_i = \hat{s}(t_i)\}_{i=1}^n$ and treat them as extra trainable parameters;

% *Training Process*

– construct and initialize the network model $\hat{u}(x, t, \varphi; \theta)$;

while maximum number of epochs is not reached **do**

– numerically solve $\hat{\gamma}'(t) = \hat{s}(t)$, then reconstruct $\hat{s}(t)$ and $\hat{\gamma}(t)$;

– construct the augmented variable $\varphi(x, t)$ and resample dataset X_{Shock};

– network training on the shuffled dataset with a suitable learning rate, i.e.,

$$\theta^*, \hat{s}_i^* = \arg\min_{\theta, s_i} L_{\text{Intrr}}(\hat{u}) + \beta_{\text{S}} L_{\text{Shock}}^{\text{inv}}(\hat{u}, \hat{s}_i) + \beta_{\text{B}} L_{\text{Bndry}}(\hat{u}) + \beta_{\text{I}} L_{\text{Initl}}(\hat{u})$$

end while

% *Testing Process*

– forward pass of the trained model on the testing dataset, i.e.,

$$\check{u}(x, t) = \hat{u}(x, t, \varphi(x, t); \theta^*).$$

Remark: the total number of collocation points remains unchanged regardless of the increased dimensionality, with computation conducted only on hyperplanes.

4. Numerical Experiments

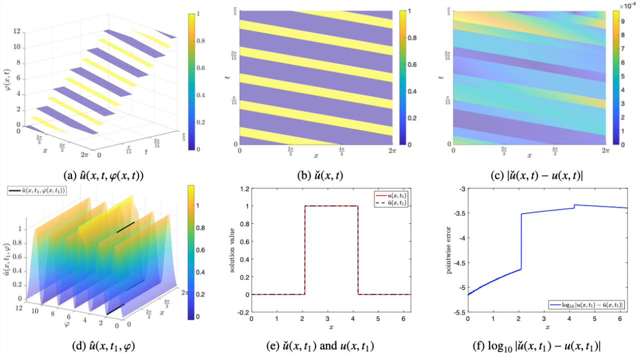
Convection Equation: Large Coefficient

Consider a widely reported failure mode of vanilla learning approach, namely,

$$\begin{aligned}
&\partial_t u(x,t) - 50\partial_x u(x,t) = 0, && \text{for } (x,t) \in (0,2\pi) \times (0, \tfrac{\pi}{5}],\\
&u_0(x) = H(x - \tfrac{2\pi}{3}) - H(x - \tfrac{4\pi}{3}), && \text{for } x \in (0,2\pi),\\
&u(0,t) = u(2\pi,t), && \text{for } t \in (0, \tfrac{\pi}{5}].
\end{aligned}$$

where our solution ansatz is constructed in a similar fashion as before

$$u(x,t) = \hat{u}(x,t,\varphi(x,t)) \ \text{ with } \ \varphi(x,t) = \sum_{i=1}^{2}\sum_{k=0}^{n_i} H(x - 50t - x_i - 2k\pi).$$

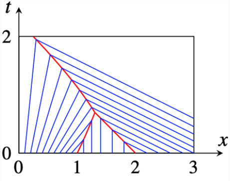


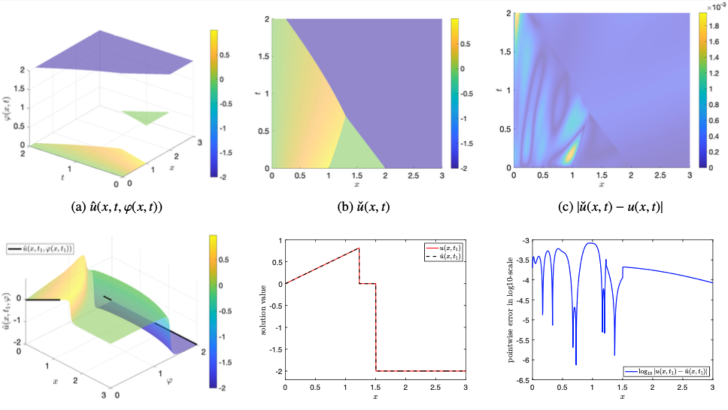
(a) $\hat{u}(x,t,\varphi(x,t))$ (b) $\check{u}(x,t)$ (c) $|\check{u}(x,t) - u(x,t)|$

(d) $\hat{u}(x,t_1,\varphi)$ (e) $\check{u}(x,t_1)$ and $u(x,t_1)$ (f) $\log_{10}|\check{u}(x,t_1) - u(x,t_1)|$

Identified Locations: Shock-Shock Interaction

$$
\begin{aligned}
&\partial_t u(x,t) + u(x,t)\partial_x u(x,t) = 0, && \text{for } (x,t) \in \Omega,\\
&u_0(x) = \begin{cases} x, & \text{for } 0 \le x < 1,\\ 0, & \text{for } 1 \le x < 2,\\ -2, & \text{for } 2 \le x \le 3, \end{cases}\\
&u(0,t) = 0,\ \ u(3,t) = -2, && \text{for } t \in (0,2].
\end{aligned}
$$



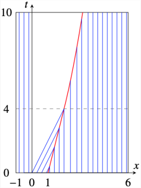


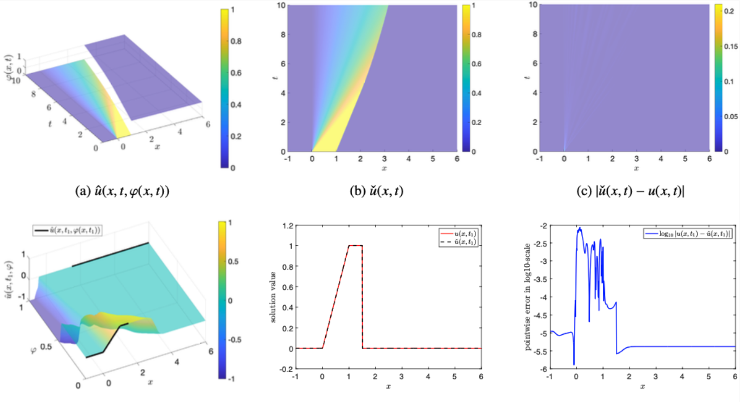
(a) $\hat{u}(x,t,\varphi(x,t))$ (b) $\check{u}(x,t)$ (c) $|\check{u}(x,t) - u(x,t)|$

(d) $\hat{u}(x,t_1,\varphi)$ (e) $\check{u}(x,t_1)$ and $u(x,t_1)$ (f) $\log_{10}|\check{u}(x,t_1) - u(x,t_1)|$

Identified Locations: Rarefaction-Shock Interaction

$$
\begin{aligned}
&\partial_t u(x,t) + \tfrac{1}{2} u(x,t)\partial_x u(x,t) = 0, && \text{for } (x,t) \in \Omega,\\
&u_0(x) = \begin{cases} 0, & \text{for } -1 \le x \le 0,\\ 1, & \text{for } 0 < x < 1,\\ 0, & \text{for } 1 \le x \le 6, \end{cases}\\
&u(-1,t) = u(6,t) = 0, && \text{for } t \in (0,10],
\end{aligned}
$$





(a) $\hat{u}(x,t,\varphi(x,t))$ (b) $\check{u}(x,t)$ (c) $|\check{u}(x,t) - u(x,t)|$

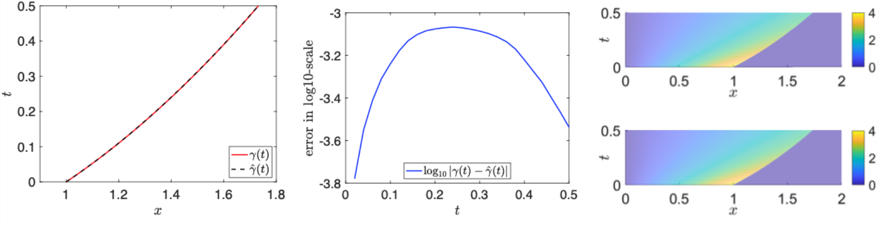
(d) $\hat{u}(x,t_1,\varphi)$ (e) $\check{u}(x,t_1)$ and $u(x,t_1)$ (f) $\log_{10}|\check{u}(x,t_1) - u(x,t_1)|$

Unknown Locations: Curved Shock Trajectory

Consider the Burgers' equation with a curved shock trajectory $\gamma(t) = \sqrt{1+4t}$

$$\begin{aligned} &\partial_t u(x,t) + u(x,t)\partial_x u(x,t) = 0, && \text{for } (x,t) \in (0,2) \times (0,0.5], \\ &u_0(x) = \begin{cases} 4x, & \\ 0, & \end{cases} && \begin{array}{l} \text{for } 0 \le x < 1, \\ \text{for } 1 \le x \le 2, \end{array} \\ &u(0,t) = u(2,t) = 0, && \text{for } t \in (0,0.5], \end{aligned}$$

the recovered shock curve and trained network solution are depicted below.



(a) $\gamma(t)$ and $\hat{\gamma}(t)$ (b) $\log_{10}|\gamma(t)-\hat{\gamma}(t)|$ (c) $u(x,t)$ and $\check{u}(x,t)$

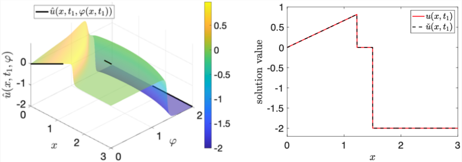
5. Concluding Remarks

Concluding Remarks

Knowledge-Embedded Machine Learning

Q1: How to embed deeper theoretical insights into scientific machine learning?

Our Answer: incorporate domain knowledge as an augmented variable into the solution ansatz, followed by parametrization through smooth neural networks.



Concluding Remarks

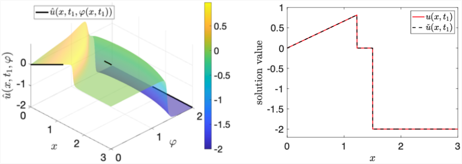
Knowledge-Embedded Machine Learning

Q1: How to embed deeper theoretical insights into scientific machine learning?

Our Answer: incorporate domain knowledge as an augmented variable into the solution ansatz, followed by parametrization through smooth neural networks.

Q2: When and how are neural networks applied to low-dimensional problems?

Our Answer: lift-and-embed the singular problems within a higher-dimensional space, followed by projecting the trained models back onto the original plane.



Concluding Remarks

Knowledge-Embedded Machine Learning

Q1: How to embed deeper theoretical insights into scientific machine learning?

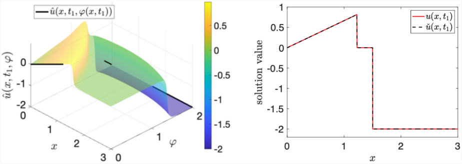
Our Answer: incorporate domain knowledge as an augmented variable into the solution ansatz, followed by parametrization through smooth neural networks.

Q2: When and how are neural networks applied to low-dimensional problems?

Our Answer: lift-and-embed the singular problems within a higher-dimensional space, followed by projecting the trained models back onto the original plane.

Q3: Where is the breakthrough in next-generation scientific machine learning?

Our Answer: $\times$ universal approximator $\checkmark$ rethink problem in higher dimensions



Thank You!