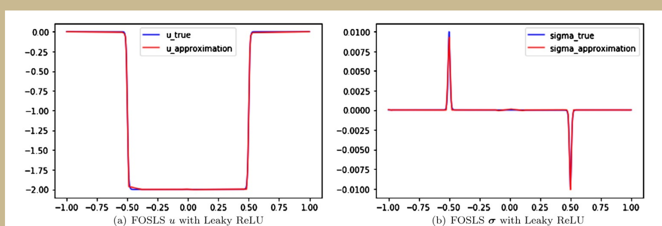
WayneState Nov. 6, 2024

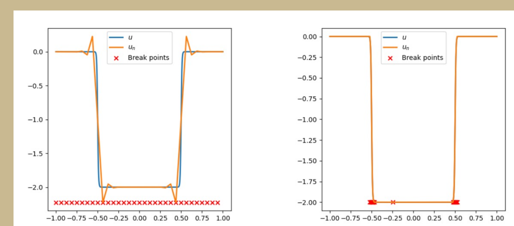
NEURAL NETWORKS IN NUMERICAL PDEs

Zhiqiang Cai[1]

Collaborators: Min Liu, Jingshuang Chen, Junpyo Choi, Brooke Hejnal
A. Doktorova, R. Falgout, C. Herrera, X. Liu, D. Tong, J. Xia



Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs
J. Comput. Phys., 420 (2020), 109707, C.-Chen-Liu-Liu.
1-32-32-24-24-2,   2962 parameters,   about 20 hours



Fast iterative solver for neural network method
C.-Doktorova-Falgout-Herrera
1-32-1,   64 parameters,   a couple of seconds

PURDUE UNIVERSITY
Department of Mathematics[1]
School of Mechanical Engineering[2]

Questions on NN in Numerical PDEs

- **What is Neural Network (NN)?**

  A "new" class of approximating functions

- **Approx. Property? How to choose optimal NN architecture?**

(1) Adaptive Neuron Enhancement Methods (Liu-Chen), (2) Approximation to Discont. Functions (Liu-Choi)

- **Why uses NN instead of FE in numerical PDEs?**

  Nonlinear Hyperbolic Conservation Laws

  (1) Least-squares neural network (LSNN) method (J. Chen, J. Choi, and M. Liu)

  (2) Evolving neural network (ENN) Method (B. Hejnal)

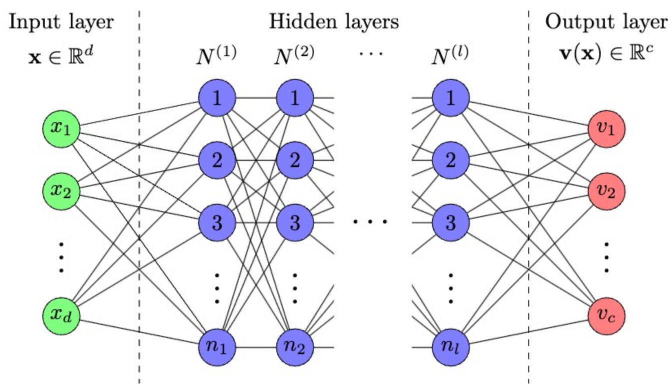
- **How to solve optimization effectively and efficiently?**

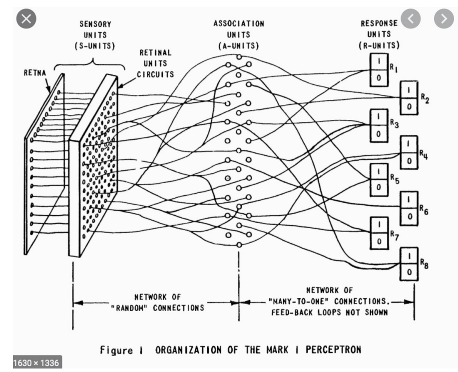
  (1) Structure-guided Gauss-Newton (SgGN) method for ReLU shallow NN (Ding-Liu-Liu-Xia)

  (2) Damped block Newton (dBN) method for 1D Diffusion (Doktorova-Falgout-Herrera)

What is Neural Network (NN)?

- **Fully-connected (Multi-Layer Perceptron) NN (Rosenblatt 1958)**

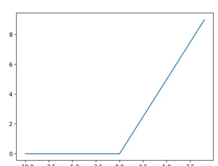




- **A class of approximating functions (ReLU NN)**

$$\sigma(t) = \begin{cases} t, & t > 0, \\ 0, & t \le 0. \end{cases}$$

$$\mathcal{M}_N(d,l) = \left\{ c_0 + \sum_{j=1}^{n_l} c_j x_j^{(l)}(\mathbf{x}) : \mathbf{c} \in \mathcal{R}^{n_l+1}, \boldsymbol{\omega}^{(k)} \in \mathcal{R}^{n_{k-1}\times n_k}, \mathbf{b}^{(k)} \in \mathcal{R}^{n_k} \right\}$$



$$\text{where } \mathbf{x}^{(0)} = \mathbf{x} \text{ and } x_i^{(k)}(\mathbf{x}) = \sigma\left(\boldsymbol{\omega}_i^{(k)} \mathbf{x}^{(k-1)} + b_i^{(k)}\right)$$

The simplest Neural Network: Shallow and 1D

- **Linear finite element on a fixed uniform mesh**

$$\phi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}}, & x \in (x_{i-1}, x_i), \\ \dfrac{x_{i+1} - x}{x_{i+1} - x_i}, & x \in (x_i, x_{i+1}), \\ 0, & \text{otherwise} \end{cases}$$

$$\mathcal{S}_1^0(\Delta) = \text{span}\,\{\phi_i(x)\}_{i=0}^n = \left\{\sum_{i=0}^n c_i\phi_i(x) : c_i \in \mathcal{R}\right\}$$

$$u(x) = x^{0.01},\ x \in [0,1]; \qquad \min_{v\in\mathcal{S}_1^0(\Delta)} \|u - v\| \le Cn^{-0.01}$$

- **Linear free-knot spline in [a,b] (1960s)**

$$\mathcal{S}_1^0(n) = \left\{\sum_{i=0}^n c_i\phi_i(x; x_{i-1}, x_i, x_{i+1}) : c_i \in \mathcal{R},\ x_i \in [a,b]\right\} \quad \min_{v\in\mathcal{S}_1^0(n)} \|u - v\| \le Cn^{-1}$$

$$= \left\{c_0 + c_1(x-a) + \sum_{i=2}^n c_i\sigma(x - x_i) : c_i \in \mathcal{R},\ x_i \in (a,b)\right\}$$

Shallow ReLU NN in R^d

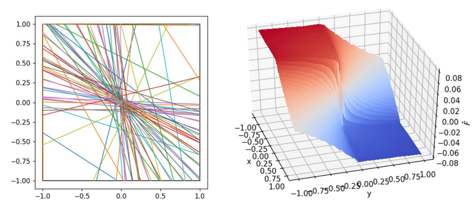
- **Shallow ReLU NN** (C^0 piecewise linear function)

$$\mathcal{M}_n(d) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(\boldsymbol{\omega}_i \mathbf{x} + b_i) \,:\, c_i, b_i \in \mathcal{R},\ \boldsymbol{\omega}_i \in \mathcal{S}^{d-1} \right\}$$

- **Linear Independence**

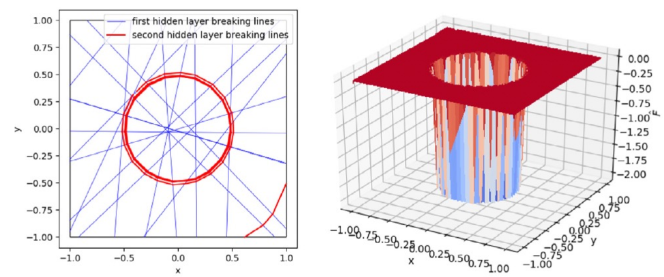
$\{\sigma(\boldsymbol{\omega}_i \mathbf{x} + b_i)\}_{i=1}^{n}$ are linearly independent if $\{\mathcal{P}_i\}_{i=1}^{n}$ are distinct.

- **Physical Partition of NN approximations**



(h) Optimum break lines (69 neurons, 1286 elements)

(i) Optimum NN model of 69 neurons, $\xi = 0.008476$



- **Breaking Lines**

$$\mathcal{P}_i \,:\, \boldsymbol{\omega}_i \mathbf{x} + b_i = 0 \quad \text{for} \ \ i = 1, \ldots, n$$

Physics-Informed Neural Network (PINN): often physics-violated

- **Dissanayake-Phan-Tien (94), Lagaris-Likas-Ftiadis (98), Rasissi-Perdikaris-Karniadakis (19), ...**

PDE: $\mathcal{L}(u(x)) = 0$ at $x \in \Omega \in \mathcal{R}^d$ and $\mathcal{B}(u(x)) = 0$ at $x \in \partial\Omega$

training data: $\{x_i^u\}_{i=1}^{N_u} \subset \Omega$ and $\{x_i^b\}_{i=1}^{N_b} \subset \partial\Omega$

l^2 **residual:**

$$L(u) = \frac{1}{N_u}\sum_{i=1}^{N_u}\left(\mathcal{L}(u(x_i^u))\right)^2 + \frac{1}{N_b}\sum_{i=1}^{N_b}\left(\mathcal{B}(u(x_i^b))\right)^2$$

PINN:

$$u_{\mathcal{N}} = \arg\min_{v \in \mathcal{N}} L(v)$$

- **Characters of the PINN**

(1) a discrete least-squares neural network (LSNN) method

(2) numerical integration: Monte Carlo with random sampling

(3) numerical differentiation: auto-differentiation

(4) unreasonable approximation

How to Design NN Methods from Numerical Analysis Perspective?

- **For a given PDE, start with a proper equivalent formulation**

  natural energy minimization and complementary maximization

  various artificial least-squares minimizations

- **Numerical Issues**
  - Numerical Integration (non-trivial): adaptive numerical integration (L-C-R 23)
  - Numerical Differentiation (critical): physics-preserved numerical differentiation
  - Algebraic solver (critical): iterative/optimization/training algorithms

Why use ReLU Neural Network instead of Finite Element?

- **Scalar Nonlinear Hyperbolic Conservation Laws**

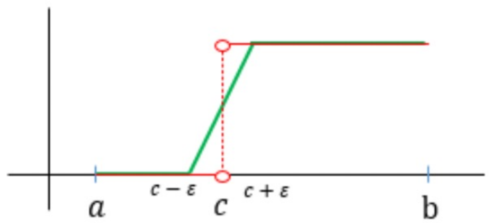
$$\begin{cases} \dfrac{\partial}{\partial t}u(\mathbf{x},t) + \displaystyle\sum_{i=1}^{d} \dfrac{\partial}{\partial x_i} f_i(u(\mathbf{x},t)) &= 0, & \text{in } \mathcal{R}^d \times (0,T), \\ u(\mathbf{x},0) &= u_0(\mathbf{x}), & \text{in } \mathcal{R}^d \end{cases}$$

- **Theoretical and Numerical Difficulties**
    - PDE theory
    - Discontinuous solution with unknown interfaces

Approximation to Unit Step Function with Unknown Interface

- **Unit step function and its CPWL approximation**

$$f_c(x) = \begin{cases} 0, & a < x < c, \\ 1, & c < x < b \end{cases}$$



$$p_c(x) = \begin{cases} 0, & a < x \le c-\varepsilon, \\ \dfrac{x-(c-\varepsilon)}{2\varepsilon}, & c-\varepsilon \le x \le c+\varepsilon, \\ 1, & c+\varepsilon \le x < b \end{cases}$$

$$\|f_c - p_c\|_{L^\infty(I)} = \frac{1}{2} \quad \text{and} \quad \|f_c - p_c\|_{L^r(I)} = \frac{\varepsilon^{1/r}}{2^{1-1/r}(1+r)^{1/r}}$$

- **How to compute or approximate $p_c(x)$ when c is unknown?**

  (1) On fixed quasi-uniform mesh

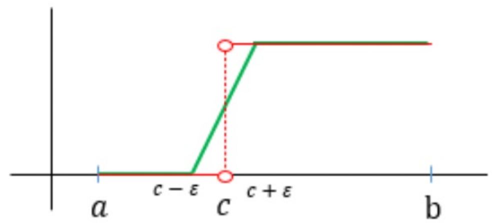
  - very fine mesh-size: $h = \varepsilon$
  - overshooting, oscillation, etc.

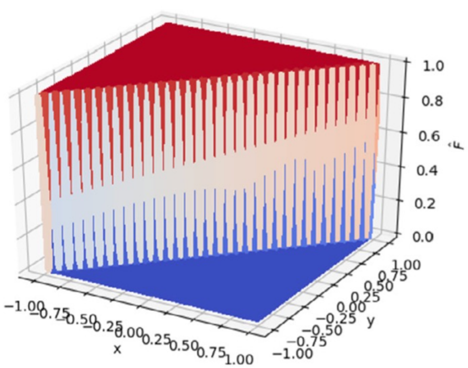
  (2) On moving mesh (neural network)

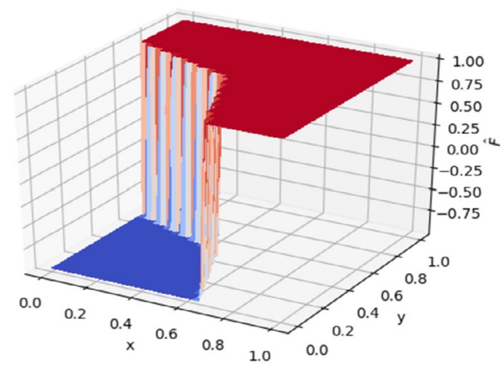
  - two neurons
  - no overshooting or oscillation

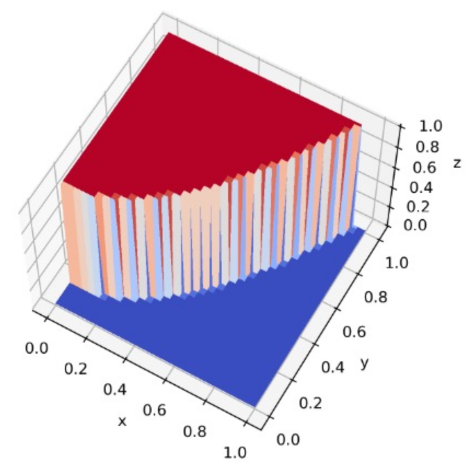
$$p_c(x) = \frac{1}{b_2 - b_1}\left[\sigma(x-b_1) - \sigma(x-b_2)\right], \quad b_1 = c-\varepsilon,\ b_2 = c+\varepsilon$$

Interface of Unit Step Function in R^d









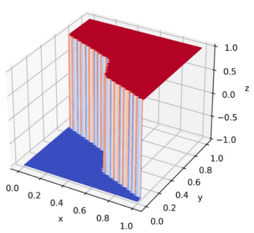
Approximation to Unit Step Function with Unknown Interface in R^d

- **Piecewise Constant function with unknow interface**

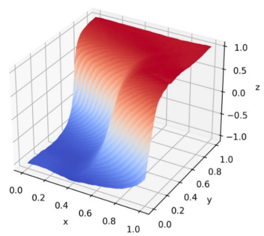
C., J. Choi, and M. Liu (2022) (d=2, 3, l=2; d=4,...,8, l=3)

Let χ(x) be a piecewise constant function with C^0 piecewise smooth interface I, then there exists a CPWL function p(x) generated by a NN with L=$\lceil \log_2(d+1) \rceil$ hidden layers such that for any given $\varepsilon > 0$, we have

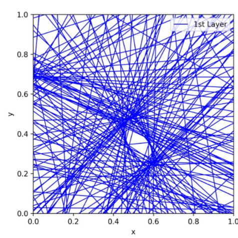
$$\||\chi - p\||_{\boldsymbol{\beta}} \leq \sqrt{2|I|}\,\big|\alpha_1 - \alpha_2\big|\,\sqrt{\varepsilon},$$



(b) Exact solution

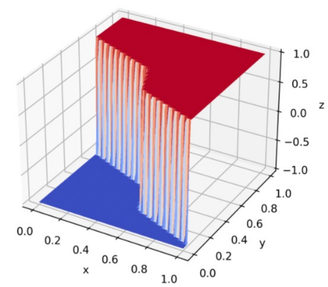


(c) 2 layer NN approximation

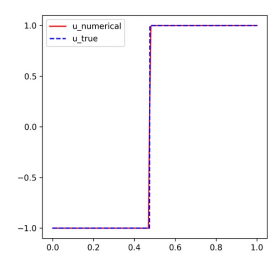


(g) 2 layer NN breaking lines

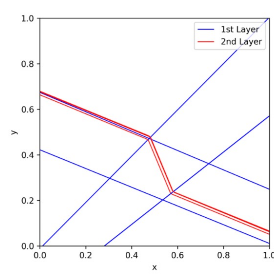
2-300-1



(d) 3 layer NN approximation



(f) 3 layer NN trace on $y = x$



(h) 3 layer NN breaking lines

2-5-5-1

Linear Advection-Reaction Problem and its Least-squares Formulation

- **Linear advection-reaction problem**

$$u_{\boldsymbol{\beta}} + \gamma\, u = f \text{ in } \Omega, \qquad u\big|_{\Gamma_-} = g$$

- **Least-squares formulation** Find $u \in V_{\boldsymbol{\beta}}(\Omega) = \{v \in L^2(\Omega) : v_{\boldsymbol{\beta}} \in L^2(\Omega)\}$ such that

$$\mathcal{L}(u; \mathbf{f}) = \min_{v \in V_{\boldsymbol{\beta}}} \mathcal{L}(v; \mathbf{f})$$

where $\mathcal{L}(v; \mathbf{f}) = \|v_{\boldsymbol{\beta}} + \gamma\, v - f\|^2_{0,\Omega} + \|v - g\|^2_{-\boldsymbol{\beta}}$

- **Coercivity and continuity** there exists positive constants α and M such that

$$\alpha\, |||v|||^2_{\boldsymbol{\beta}} \le \mathcal{L}(v; \mathbf{0}) \le M\, |||v|||^2_{\boldsymbol{\beta}}$$

De Sterck-Manteuffel-McCormick-Olson, 2004

Numerical Issues

- **Numerical integration**

$$\int_\Omega \left(v_{\boldsymbol\beta} + \gamma\, v - f\right)^2 (\mathbf{x})\, d\mathbf{x}$$

Adaptive numerical integration (Liu-Ramani-C 2023)

- **Numerical differentiation**

$$u_{\boldsymbol\beta}(\mathbf{x}) = \sum_{i=1}^{d} \beta_i(\mathbf{x}) \frac{\partial u(\mathbf{x})}{\partial x_i}.$$

Is invalid at where the solution is discontinuous.

Physics-preserved numerical differentiation

$$D_{\boldsymbol\beta} v(\mathbf{x}) := \frac{v(\mathbf{x}) - v\big(\mathbf{x} - \rho\bar{\boldsymbol\beta}(\mathbf{x})\big)}{\rho/|\boldsymbol\beta(\mathbf{x})|} \approx v_{\boldsymbol\beta}(\mathbf{x}),$$

- **Algebraic solver (training algorithm)** ???

Least-squares neural network (LSNN) method

- **Discrete LS functional**

$$\mathcal{L}_{\mathcal{T}}\left(v;\mathbf{f}\right) = \sum_{K\in\mathcal{T}} \mathcal{Q}_K\left((D_{\boldsymbol{\beta}}v + \gamma v - f)^2\right).$$

- **LSNN method** find $u_{\mathcal{T}}^N \in \mathcal{M}(L,n)$ such that

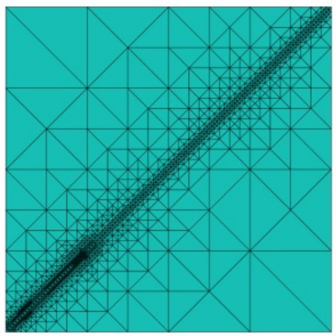
$$\mathcal{L}_{\mathcal{T}}\left(u_{\mathcal{T}}^N;\mathbf{f}\right) = \min_{v\in\mathcal{M}(L,n)} \mathcal{L}_{\mathcal{T}}\left(v;\mathbf{f}\right)$$

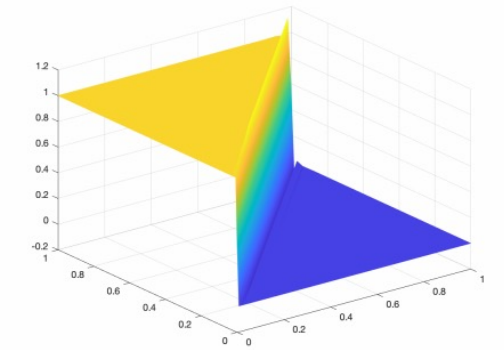
- **A priori error estimate**

$$|||u - u_N|||_{\boldsymbol{\beta}} \le C\left(\left|\alpha_1 - \alpha_2\right|\sqrt{\varepsilon} + \inf_{v\in\mathcal{M}(d,n)} |||\hat{u} + p - v|||_{\boldsymbol{\beta}}\right)$$

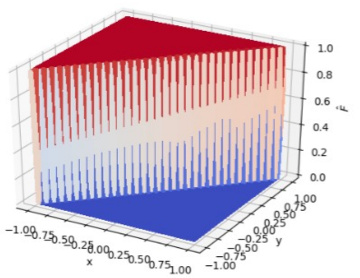
C.-Chen-Liu, JCP, 443 (2021), 110514; C.-Choi-Liu (2022)

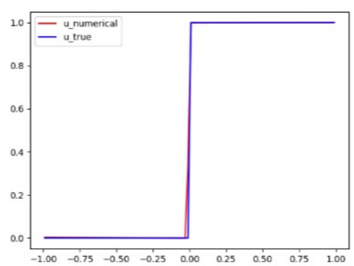
Famous Transport Equation $u_t+u_x=0$



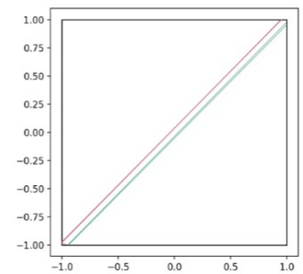


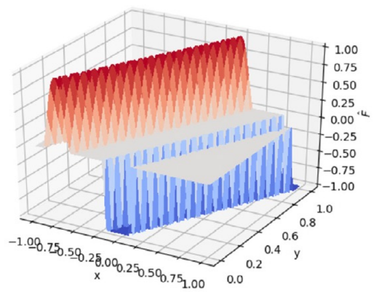
Liu-Zhang, CMAME, 2020

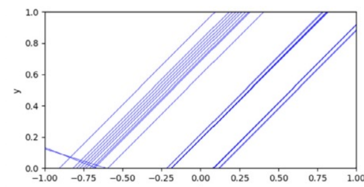




(2-6-1)

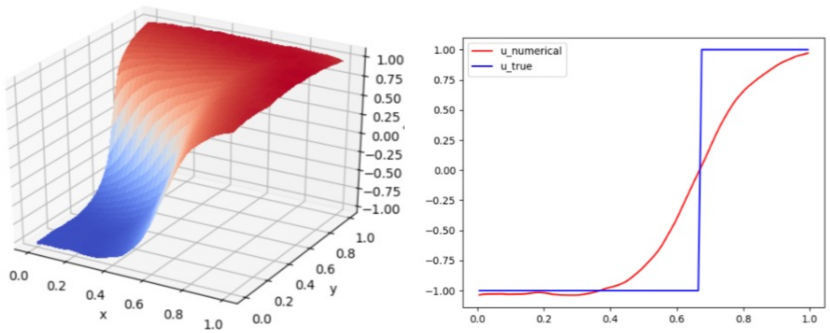




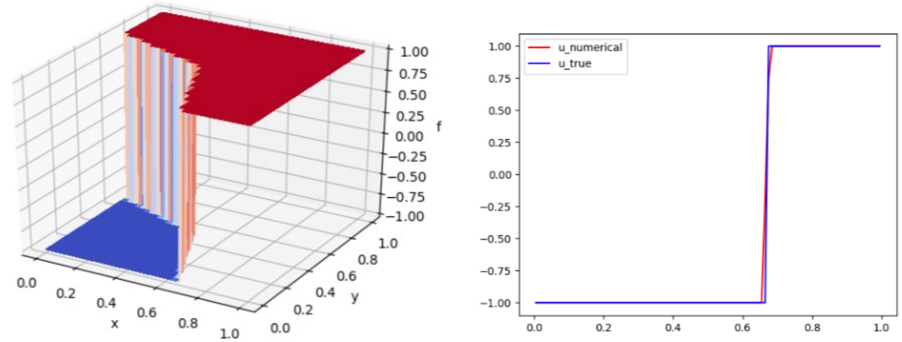


(2-34-1)

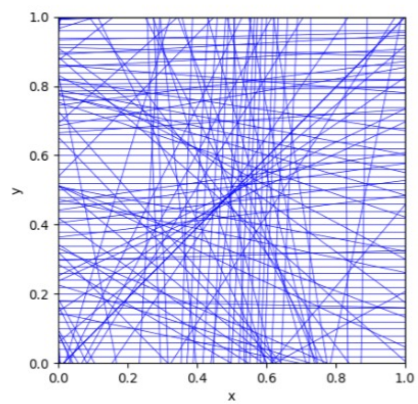
C.-Chen-Liu, JCP, 2021



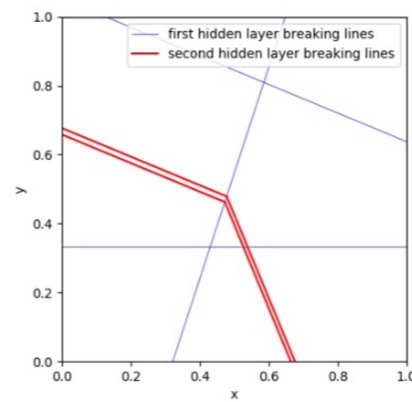
(2-200-1)



(2-5-5-1)



(e) 2-layer NN breaking lines



(f) 3-layer NN breaking lines

C.-Chen-Liu, LSNN method for linear advection-reaction equation, JCP, 443(2021), 110514.

Least-Squares Neural Network (LSNN) method

- **Scalar nonlinear hyperbolic conservation laws**

$$\frac{\partial}{\partial t}u(\mathbf{x},t)+\sum_{i=1}^{d}\frac{\partial}{\partial x_i}f_i(u(\mathbf{x},t))=0,\ \text{in } \mathcal{R}^d\times(0,T),\qquad u(\mathbf{x},0)=u_0(\mathbf{x}),\ \text{in } \mathcal{R}^d$$

- **Physics Form of HCLs for total flux** $\mathbf{F}(u)=(\mathbf{f}(u),u)$

$$\textbf{div}\,\mathbf{F}(u)=0 \qquad \text{in } \ \Omega\times I\in\mathbb{R}^{d+1}$$

- **Equivalent Least-squares formulation**

$$\text{Find } u\in\mathcal{V}_{\mathbf{F}}=\left\{v\in L^2(\Omega\times I)\,|\,\mathbf{F}(v)\in H(\text{div};\Omega\times I)\right\} \text{ such that}$$

$$u=\underset{v\in\mathcal{V}_{\mathbf{F}}}{\arg\min}\,\mathcal{L}(v;\mathbf{f}),\quad \text{where } \ \mathcal{L}(v;\mathbf{f})=\|\textbf{div}\,\mathbf{F}(v)\|_{0,\Omega\times I}^2+\|v-u_0\|_{0,\Omega\times\{0\}}^2$$

Discrete Divergence Operator

- **Discrete divergence operator**
  + **based on conservative numerical schemes (C.-Chen-Liu, ANM(2022))**
  + **new discrete divergence operator (C.-Chen-Liu, J Comput Appl Math (2023))**

Let $\mathcal{T}$ be a partition of the domain $\Omega \subset \mathbb{R}^{d+1}$.

For any $K \in \mathcal{T}$, let $\mathbf{z}_K$ be the centroid of K.

$$\mathbf{div}_{\mathcal{T}} \mathbf{F}\big(u(\mathbf{z}_K)\big) \approx \text{avg}_K \mathbf{div}\, \mathbf{f}(u) = \frac{1}{|K|} \int_{\partial K} \mathbf{F}(u) \cdot \mathbf{n}\, dS$$

- **Least-squares neural network (LSNN) method**

Find $u_n \in \mathcal{M}(l,n) \subset V_{\mathbf{f}}$ such that $\quad u_n = \underset{v \in \mathcal{M}(l,n)}{\arg\min}\, \mathcal{L}_{\mathcal{T}}(v; \mathbf{f})$,

where $\mathcal{L}_{\mathcal{T}}(v; \mathbf{f})$ is a discrete LS functional based on $\mathcal{L}(v; \mathbf{f})$.

Discrete Divergence Operator in 1D

- **Primitive form over K_{ij}**

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

$$\approx \frac{1}{\delta}Q(\sigma(x_i, x_{i+1}; t); t_j, t_{j+1}, \hat{n}) + \frac{1}{h}Q(u(x; t_j, t_{j+1}); x_i, x_{i+1}, \hat{m}) = \mathrm{div}_{\mathcal{T}}\mathbf{F}\big(u_{ij}\big)$$

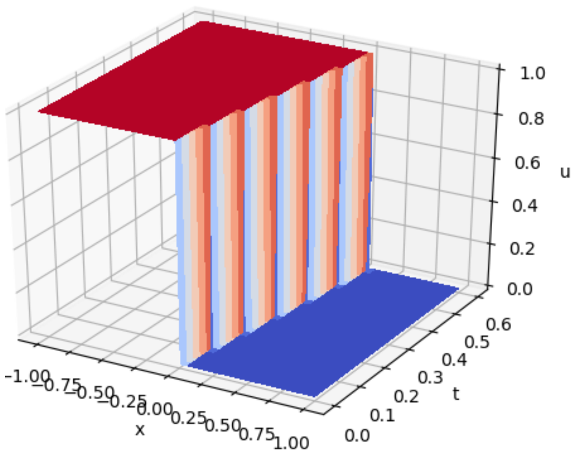
- **Error estimate**

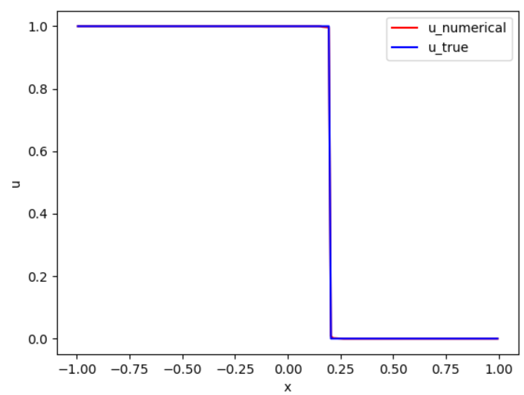
LEMMA 4.3. *Assume that u is a C^2 function of t and a piece-wise C^2 function of x on two vertical and two horizontal edges of K_{ij}, respectively. Moreover, u has only one discontinuous point on each horizontal edge. Then there exists a constant $C > 0$ such that*

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

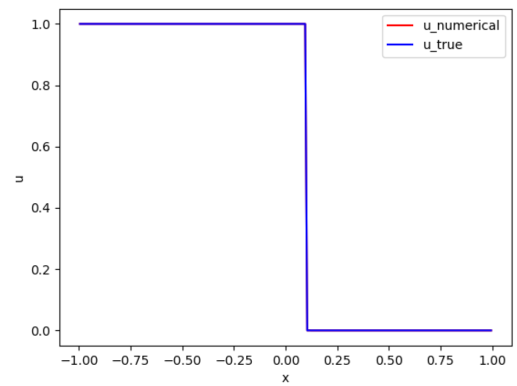
Inviscid Burger Equation $f(u) = \frac{1}{2}u^2$

Riemann Problem Shock formation: exact solution

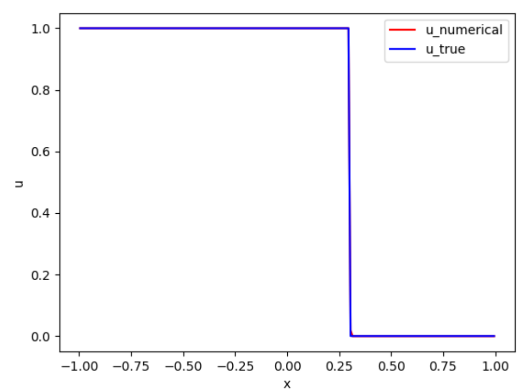




t=0.2



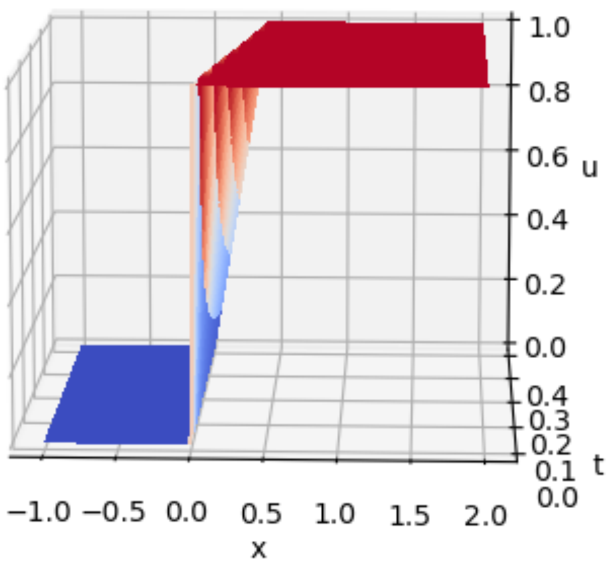
t=0.4

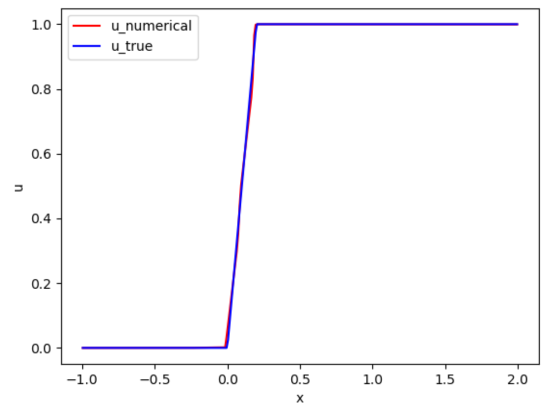


t=0.6 (2-10-10-1)

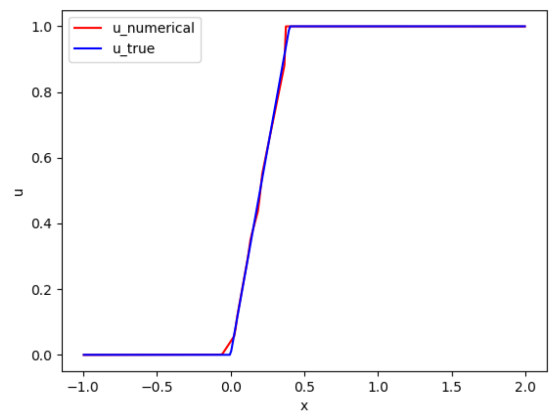
C.-Chen-Liu, arXiv: 2110.10895 [math.NA]

Riemann Problem
Rarefaction wave: exact solution





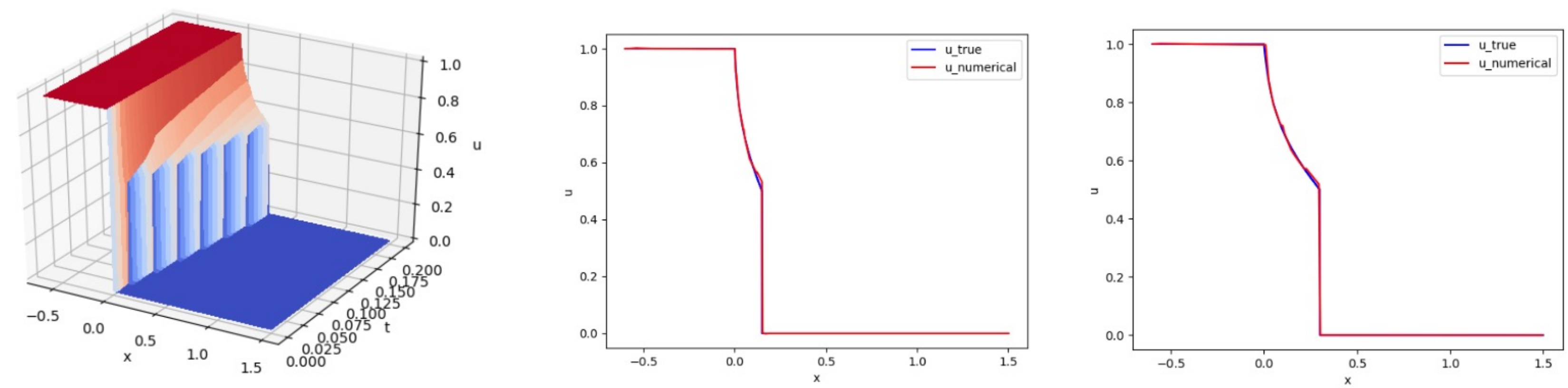
t=0.2



t=0.4

(2-10-10-1)

Buckley-Leverett Problem $f(u) = u(1-u)/[u^2 + a(1-u)^2]$

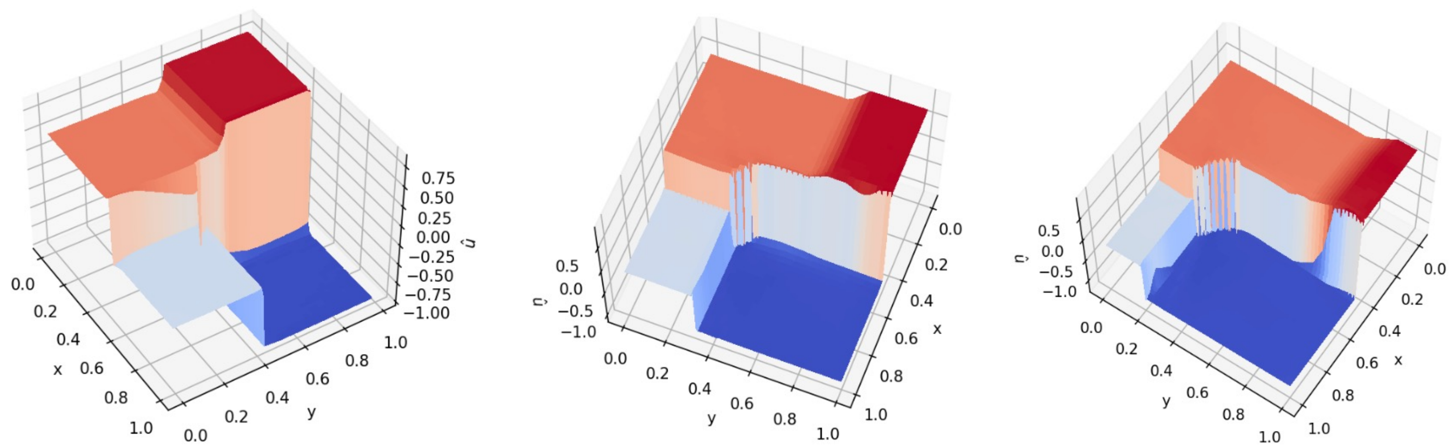


(a) Numerical solution u_N on Ω (b) Traces at $t = 0.1$ (c) Traces at $t = 0.2$

FIG. 6. *Numerical results of Buckley-Leverett Riemann problem*

2D Inviscid Burger Equation $f(u) = \frac{1}{2}(u^2, u^2)$

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



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

Evolving Neural Network (ENN) Method (C. and B. Hejnal)

- **One-Dimensional Scalar Nonlinear Hyperbolic Conservation Laws**

$$
\begin{cases}
\dfrac{\partial}{\partial t}u(x,t) + \dfrac{\partial}{\partial x}f\big(u(x,t)\big) & = & 0, & \text{in } \; \Omega \times (0,T), \\
u(x,t) & = & g(t), & \text{on } \; \Gamma_-, \\
u(x,0) & = & u_0(x), & \text{in } \; \Omega
\end{cases}
$$

- **Two features of HCLs (characteristic line and shock formation)**

$$
\begin{cases}
\dfrac{d}{dt}x(t) & = & f'\big(u(x(t),t)\big) \\
x(t_0) & = & x_0
\end{cases}
$$

$$x(t) = x_0 + (t - t_0)\, f'\big(u(x_0, t_0)\big)$$

$$\frac{d}{dt}u(x(t), t) = 0$$

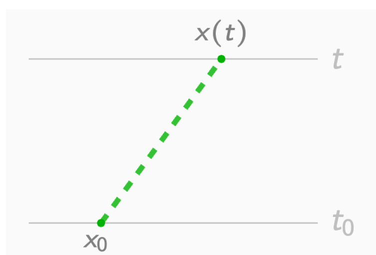
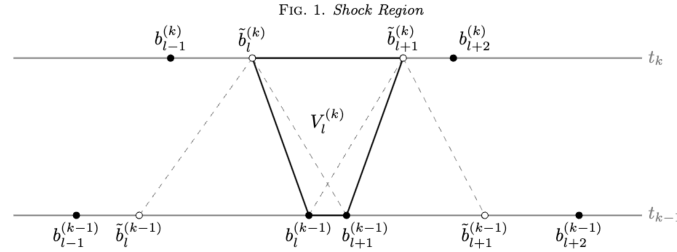


Fig. 1. *Shock Region*



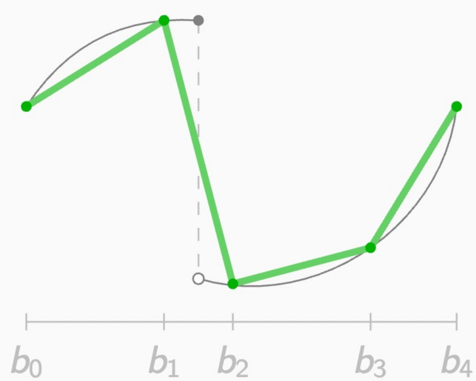
Representation of Initial Data

Set of Neural Network Functions:

$$M_n = \left\{ N(x) = c_{-1} + \sum_{i=0}^{n} c_i \sigma(\omega_i x - b_i) \; : \; b_i,\, c_i,\, \omega_i \in \mathbb{R} \right\}$$

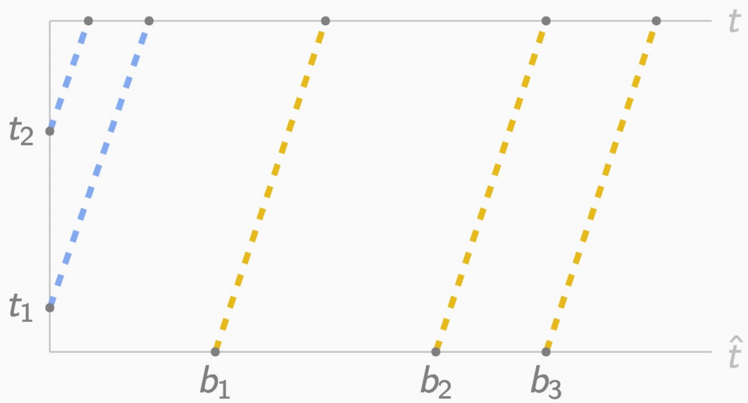
Least-Squares Approximation: Find $u_N^{(0)} \in M_n$ such that

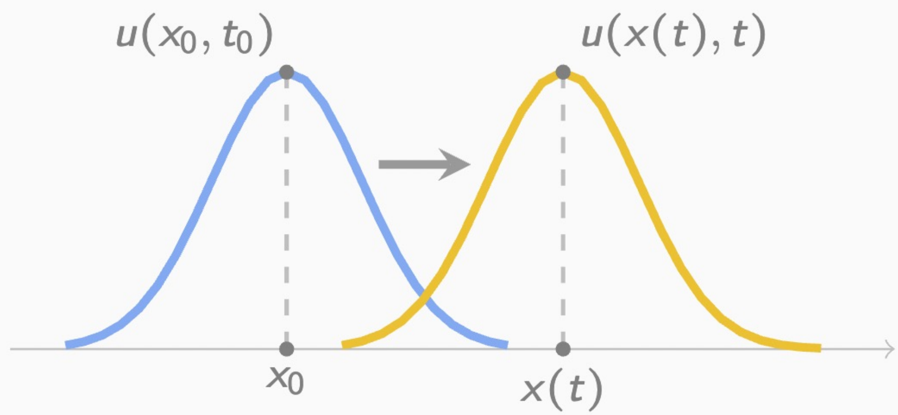
$$\left\| u_N^{(0)} - u_0 \right\|_{L^2(\Omega)} = \min_{v \in M_n} \left\| v - u_0 \right\|_{L^2(\Omega)}$$



Characteristic Scheme

- Propagate breaking points of the initial and boundary data

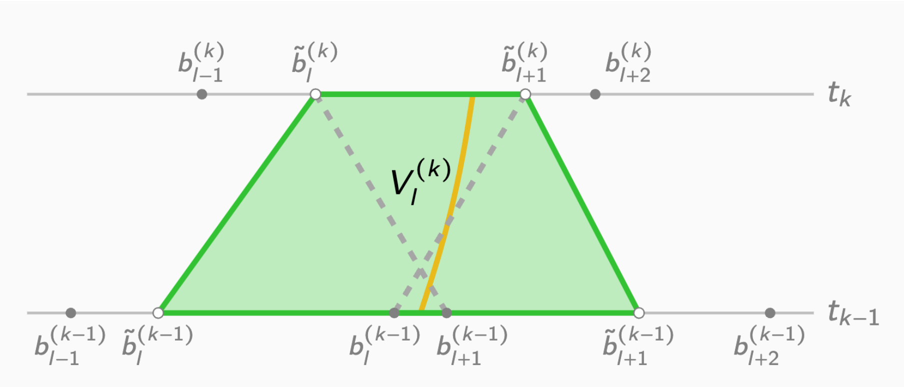




- Error Estimate

$$\left\| u(\cdot, t_k) - u_N^{(k)} \right\|_{L^p(\Omega)} \leq \left(\left\| u_0 - u_N^{(0)} \right\|_{L^p(\Omega)}^p + \| g - g_N \|_{L^p(I)}^p \right)^{1/p}$$

Finite Volume Characteristic Scheme



$$\int_{\partial V_l^{(k)}} \begin{pmatrix} \boldsymbol{f(u)} \\ \boldsymbol{u} \end{pmatrix} \cdot \hat{\boldsymbol{n}}\, ds = 0$$

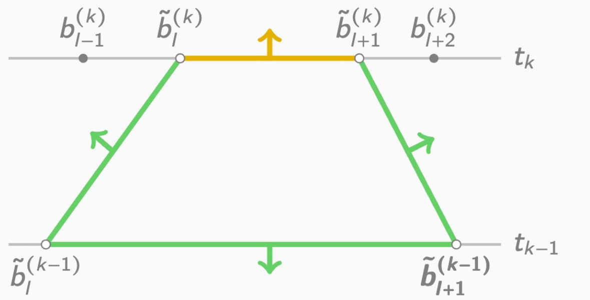
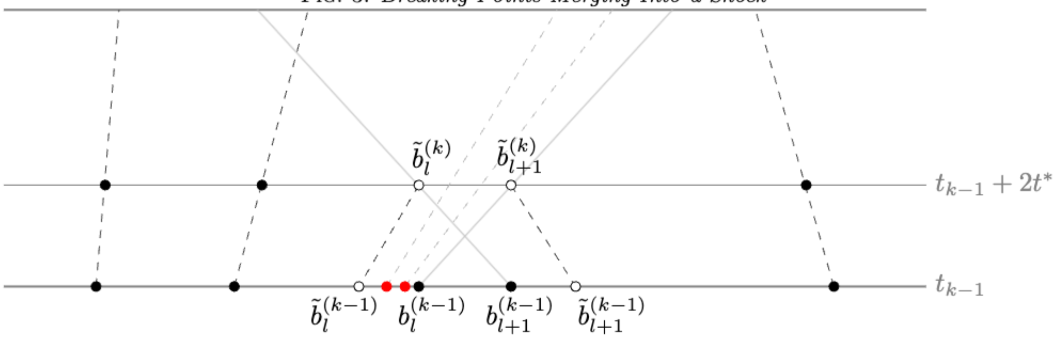
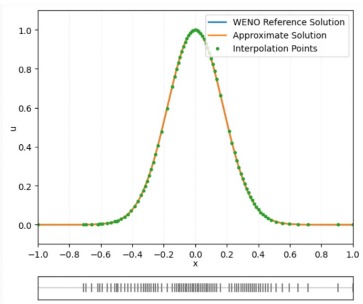


Fig. 3. *Breaking Points Merging Into a Shock*

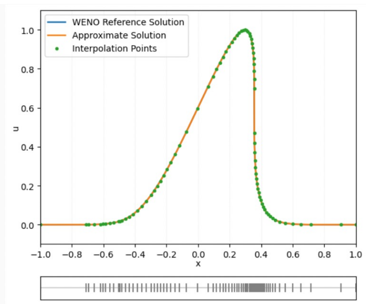


Shock Formation (exponential initial profile)

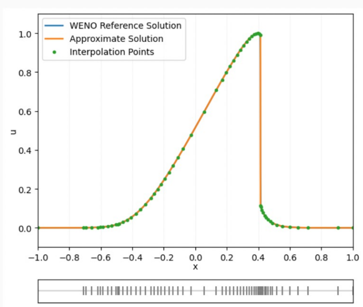
Inviscid Burgers' Equation



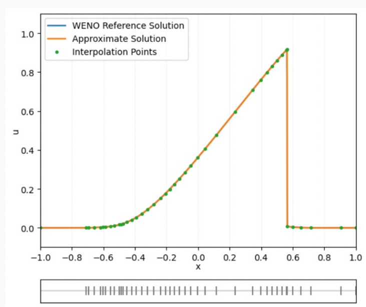
Initial data



$t = 0.3$



$t = 0.4$



$t = 0.7$

Time	$\frac{\lVert \tilde{u}(\cdot, t_k) - u_N^{(k)} \rVert_{L^2(\Omega)}}{\lVert \tilde{u}(\cdot, t_k) \rVert_{L^2(\Omega)}}$	n_k
0.0	6.6207×10^{-4}	83
0.2	7.2902×10^{-4}	83
0.4	1.2718×10^{-2}	61
0.6	2.1803×10^{-2}	47
0.8	2.0423×10^{-2}	40
1.0	1.4822×10^{-2}	37

ENN

- **83** breaking points
- **418** time steps

WENO

- **2000** mesh points
- **5000** time steps

Shock Formation (sinusoidal initial profile)

Inviscid Burgers' Equation

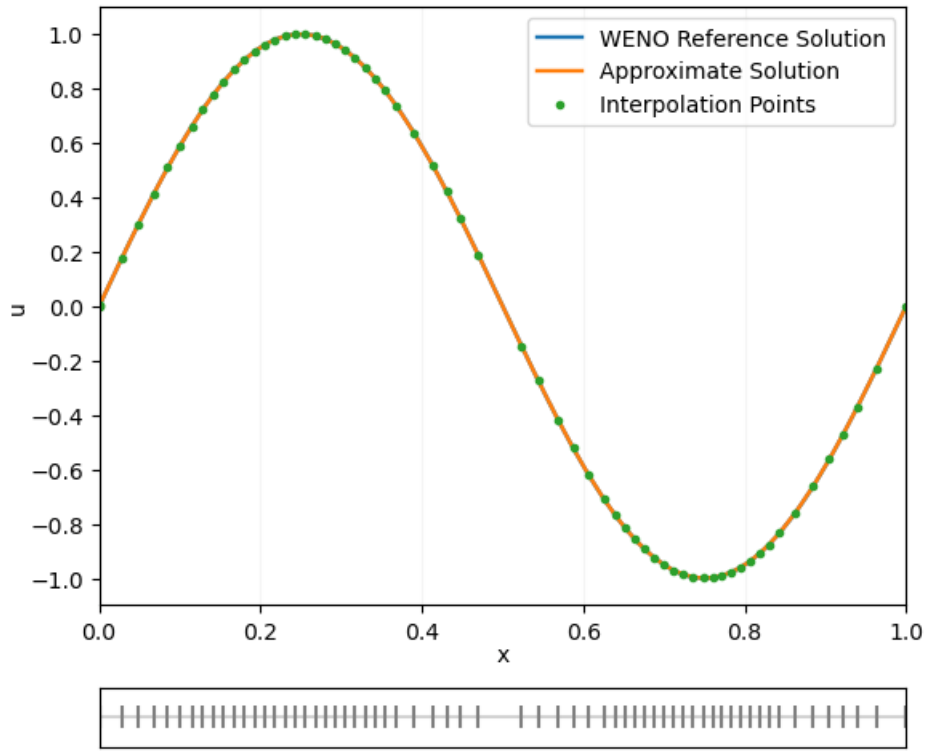
Time	$\frac{\|\tilde{u}(\cdot,t_k)-u_N^{(k)}\|_{L^2(\Omega)}}{\|\tilde{u}(\cdot,t_k)\|_{L^2(\Omega)}}$	n_k
0.0	6.6923×10^{-4}	78
0.1	7.8352×10^{-4}	78
0.2	4.0166×10^{-2}	56
0.3	5.1491×10^{-2}	38
0.4	5.3515×10^{-2}	30
0.5	5.4162×10^{-2}	25

ENN

- **78** breaking points
- **587** time steps

WENO

- **1000** mesh points
- **2500** time steps

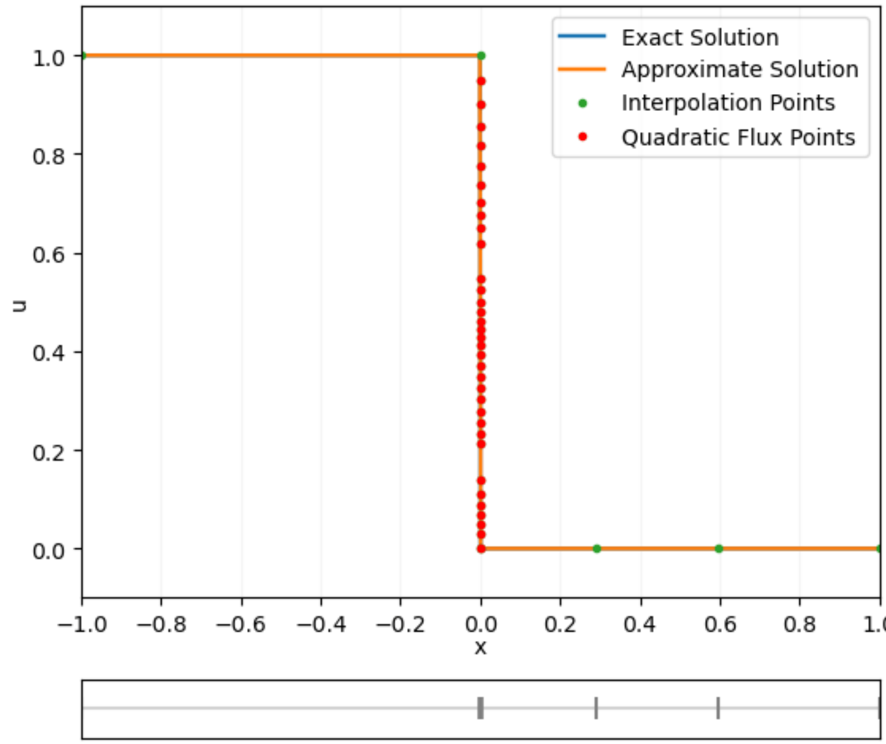


Compound Wave

Buckley-Leverett Equation

$$f(u) = \frac{u^2}{u^2 + \frac{1}{2}(1-u)^2}$$

Time	$\frac{\|u(\cdot,t_k)-u_N^{(k)}\|_{L^2(\Omega)}}{\|u(\cdot,t_k)\|_{L^2(\Omega)}}$	n_k
0.00	1.3732×10^{-2}	40
0.25	6.9084×10^{-3}	16
0.50	5.8335×10^{-3}	15



Summary

- **NN provides a new class of approximating functions**

  "Moving" mesh vs uniform mesh and adaptive mesh

- **Scalar hyperbolic conservation laws**

  LSNN (a space-time approach)

    No numerical artifacts such as overshooting, oscillation, or smearing
    Complicated and expensive iterative solvers

  ENN (a time-marching approach)

    Super accurate and efficient for 1D scalar HCLs comparing with existing methods
    Extension to multi-dimension?

How to solve optimization problem effectively and efficiently?

Non-Convex Least-squares Data Fitting

For a given data set $\left\{\left(\mathbf{x}^i, u^i\right)\right\}_{i=1}^m$ with $\mathbf{x}^i \in \Omega \subset \mathcal{R}^d$ and $u^i \in \mathcal{R}$, let

- $\mathcal{J}(v) = \dfrac{1}{2}\sum_{i=1}^{m} \mu^i \left(v(\mathbf{x}^i;\boldsymbol{\theta}) - u^i\right)^2$ be a discrete LS loss function
- $\mathcal{M}_n(\Omega;\boldsymbol{\theta})$ be a set of neural network functions

finding $u_n(\mathbf{x};\boldsymbol{\theta}) \in \mathcal{M}_n(\Omega;\boldsymbol{\theta})$ such that

$$u_n(\mathbf{x};\boldsymbol{\theta}) = \underset{v\in\mathcal{M}_n(\Omega;\boldsymbol{\theta})}{\arg\min}\ \mathcal{J}(v) = \underset{\boldsymbol{\theta}\in R^N}{\arg\min}\, \mathcal{J}(v(\cdot;\boldsymbol{\theta}))$$

Iterative/Optimization/Training Algorithms

methods of gradient descent (Adam, SGD, ...),
Newton's methods (BFGS, ...),
Gauss-Newton's methods

Optimality Conditions for Critical Points

- **neural network approximation**

$$u_n(\mathbf{x}) = c_0 + \sum_{i=1}^{n} c_i \sigma(\boldsymbol{\omega}_i \cdot \mathbf{x} + b_i) = c_0 + \sum_{i=1}^{n} c_i \sigma(\mathbf{r}_i \cdot \mathbf{y})$$

- **linear and nonlinear parameters**

$$\hat{\mathbf{c}} = (c_0, \mathbf{c}) \qquad \text{with } \mathbf{c} = (c_1, \ldots, c_n)$$

$$\text{and } \mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_n) \qquad \text{with } \mathbf{r}_i = (\boldsymbol{\omega}_i, b_i)$$

- **Optimality Conditions for Critical Points**

$$\nabla_{\hat{\mathbf{c}}} \mathcal{J}\left(u_n(\cdot; \hat{\mathbf{c}}, \mathbf{r})\right) = \mathbf{0} \quad \text{and} \quad \nabla_{\mathbf{r}} \mathcal{J}\left(u_n(\cdot; \hat{\mathbf{c}}, \mathbf{r})\right) = \mathbf{0}$$

Mass and Layer Gauss-Newton Matrices

- **Mass Matrix A(r) for linear parameters**

$$\mathbf{0} = \nabla_{\hat{\mathbf{c}}} \mathcal{J}\left(u_n(\cdot; \hat{\mathbf{c}}, \mathbf{r})\right) = \mathbf{A}(\mathbf{r})\, \hat{\mathbf{c}} - \mathbf{f}(\mathbf{r})$$

where $\mathbf{A}(\mathbf{r}) = \sum_{i=1}^{m} \mu^i \left[\mathbf{\Sigma}\mathbf{\Sigma}^T\right](\mathbf{x}^i)$ is SPD

- **Gradient with respect to r**

$$\mathbf{0} = \nabla_{\mathbf{r}} \mathcal{J}\left(u_n(\cdot; \hat{\mathbf{c}}, \mathbf{r})\right) = \left(D(\mathbf{c}) \otimes I_{d+1}\right) \mathbf{G}(\hat{\mathbf{c}}, \mathbf{r})$$

- **Layer Gauss-Newton's matrix for nonlinear parameters**

$$GN = \big(D(\mathbf{c}) \otimes I_{d+1}\big)\mathcal{H}(\mathbf{r})\big(D(\mathbf{c}) \otimes I_{d+1}\big)$$

where $\mathcal{H}(\mathbf{r}) = \sum_{i=1}^{m} \mu^i \left[\mathbf{H}\mathbf{H}^T\right] \otimes \left[\mathbf{y}\mathbf{y}^T\right](\mathbf{x}^i)$ is SPD

The Structure-guided Gauss-Newton (SgGN) Method

Given $\left(\hat{\mathbf{c}}^{(k)}, \mathbf{r}^{(k)}\right) = \left(\mathbf{c}^{(k)}, c_0^{(k)}, \mathbf{r}^{(k)}\right)$, compute $\left(\hat{\mathbf{c}}^{(k+1)}, \mathbf{r}^{(k+1)}\right)$:

(i) Compute $\hat{\mathbf{c}}^{(k+1)}$ by solving the system of linear equations

$$\mathcal{A}\big(\mathbf{r}^{(k)}\big)\,\hat{\mathbf{c}}^{(k+1)} = \mathbf{f}\big(\mathbf{r}^{(k)}\big)$$

(ii) If $c_i^{(k+1)} \neq 0$ for all i, compute the search direction

$$\mathbf{p}^{(k+1)} = \left(D^{-1}\big(\mathbf{c}^{(k+1)}\big) \otimes I_{d+1}\right)\mathbf{s}^{(k+1)},$$

where $\mathbf{s}^{(k+1)} = -\mathcal{H}\big(\mathbf{r}^{(k)}\big)^{-1}\mathbf{G}\big(\mathbf{c}^{(k+1)}, c_0^{(k+1)}, \mathbf{r}^{(k)}\big)$.

(iii) Compute the nonlinear parameter

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \gamma_{k+1}\mathbf{p}^{(k+1)},$$

where the damping parameter γ_{k+1} is computed by

$$\gamma_{k+1} = \arg\min_{\gamma\in\mathcal{R}} \mathcal{J}_\mu\left(u_n\big(\cdot; \mathbf{c}^{(k+1)}, \mathbf{r}^{(k)} + \gamma\mathbf{p}^{(k+1)}\big)\right).$$

Initialization

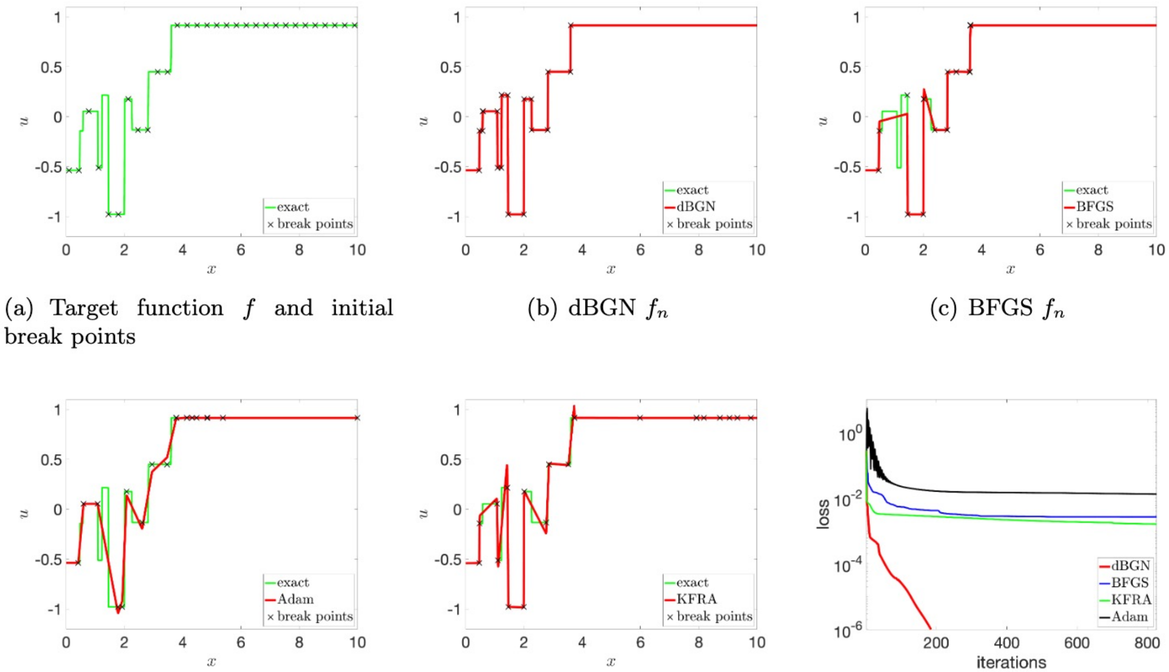
$$u_n(\mathbf{x}) = c_0 + \sum_{i=1}^{n} c_i \sigma(\boldsymbol{\omega}_i \cdot \mathbf{x} + b_i) = c_0 + \sum_{i=1}^{n} c_i \sigma(\mathbf{r}_i \cdot \mathbf{y})$$

- **Breaking Line Initialization**
  - nonlinear parameters $\mathbf{r}_i$: partitioning the domain by uniform breaking lines
  - linear parameters c_i: minimizing the functional with fixed $\mathbf{r}_i$
- **Methods of Continuation**
  - Adaptive Neuron Enhancement (ANE) Method

    Liu-C., CAMWA, 113 (2022), 34-44 and 103-116; C.-Chen-Liu, JCP, 455 (2022), 111021
  - Physical model continuation, …

    C.-Chen-Liu, JCP, 443 (2021)

Step Function



(a) Target function f and initial break points (b) dBGN f_n (c) BFGS f_n

(d) Adam f_n (e) KFRA f_n (f) Loss curves

Figure 2: One-dimensional piece-wise constant function approximation results

Table 1: Comparison for one-dimensional piece-wise constant function.

Method	dBGN		BFGS		KFRA	Adam
Iteration	9	825	207	825	825	$10,000$
$J(f_n)$	8.76E-4	6.56E-9	4.03E-3	2.65E-3	1.61E-3	8.14E-3

Summary

- **The SgGN method**
  - using the NN structure as well as quadratic structure
  - producing accurate approximation than other training algorithms
- **Mass and Layer Gauss-Newton matrices**

  positive definite   no need of shifting

  ill-conditioned: ??? fast solvers ???
  C.-Doktorova-Falgout-Herrera (1D elliptic PDEs)
- **Importance of Initialization**

THANK YOU

Collaborators: Min Liu Jingshuang Chen, Junpyo Choi, and Brooke Hejnal

Tong Ding, Min Liu, Xinyu Liu, and Jianlin Xia

Ana Doktorova, Rob Falgout, and Cesar Herrera

https://www.math.purdue.edu/~caiz/paper.html