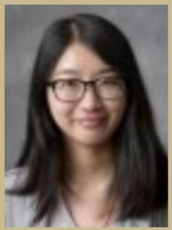
NEURAL NETWORKs AND NUMERICAL PDEs

Z. Cai[1]



Jingshuang Chen[1]

Min Liu[2]

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

Outline

- **Lecture 1**

  Overview and Least-squares formulations for PDEs

- **Lecture 2**

  Least-square neural network (LSNN) method for linear transport problems and nonlinear scalar hyperbolic conservation laws

- **Lecture 3**

  Adaptive Neural Networks (adaptive network enhancement (ANE) method)

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

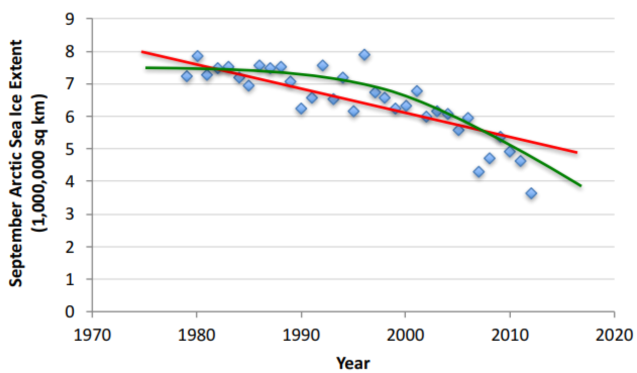
Is ML just a glorified "curve fitting"?

"Learning is any process by which a system improves performance from experience."

- Herbert Simon Definition

E.g. Supervised Learning: Regression

Task: Analyze arctic sea ice extent.
Performance: Mean squared error
Experience: Ice extent in the past 40 years



Data from G. Witt. Journal of Statistics Education, Volume 21, Number 1 (2013)

Given $S = \left\{\left(x_j, y_j = f\left(x_j\right)\right), j \in [n]\right\}$, est./app. f

$$\min_{\theta} R_n(\theta)$$

- **Loss function:**

$$R_n(\theta) = \frac{1}{n}\sum_{j=1}^{n}\left(N\left(x_j;\theta\right) - y_j\right)^2$$

- **Model** N: (piecewise) polynomials, neural nets, ...
- **Objective :**

$$R\left(N(\cdot;\theta)\right) = \begin{cases} \int_\Omega (f(x) - N(x;\theta))^2\, dx & \text{Legendre (1805)} \\ \int_\Omega (f(x) - N(x;\theta))^2\, d\mu & \text{Gauss (1809)} \end{cases}$$

- **Training:** gradient descent (GD), stochastic gradient descent (SGD), ADAM, RMSprop, ...

Neural Networks (NNs): a class of new approximating functions

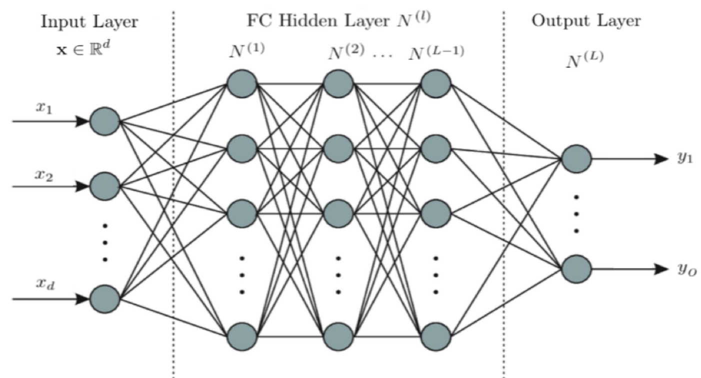
Fully-connected (Multi-Layer Perceptron) NN (Rosenblatt 1958)

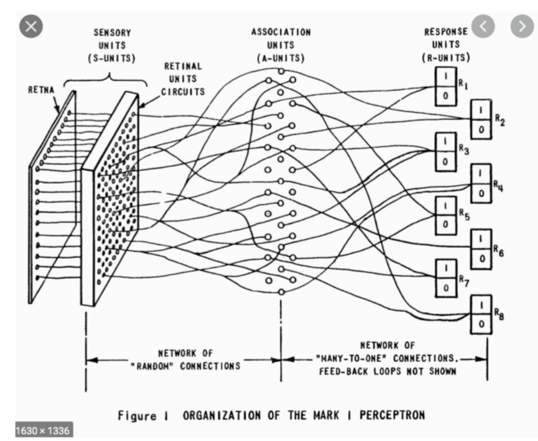
- **DNN function (models)**

Let $\mathbf{x}^{(0)} = \mathbf{x}$ and $\mathbf{x}^{(i)} = \sigma\left(W^{(i)}_{n\times(n+1)}\begin{bmatrix}1\\ \mathbf{x}^{(i-1)}\end{bmatrix}\right)$ for $i = 1, \ldots, l$

$$u(\mathbf{x}; W) = W^{(l+1)}_{1\times(n+1)}\begin{bmatrix}1\\ \mathbf{x}^{(l)}\end{bmatrix}$$

where $W = \left[W^{(1)}_{n\times(d+1)}, W^{(2)}_{n\times(n+1)}, \ldots, W^{(l)}_{n\times(n+1)}, W^{(l+1)}_{1\times(n+1)}\right]$





- **ReLU Activate function**

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

Current States of NNs in Numerical PDEs

- **Conventional Wisdoms**

  competitive: high dimensional PDEs, inverse problems, ...

  not competitive: low dimensional, forward PDEs, ...

- **Features of NNs (a class of new approximating functions)**

  high cost and uncertainty: non-convex optimization

  powerful in approximation: huge expressive power, free knot spline, ...

- **Two-layer NNs**

$$\mathcal{M}_n(\sigma, d) = \left\{ \mathbf{c}_0 + \sum_{i=1}^{n} \mathbf{c}_i \sigma(\boldsymbol{\omega}_i \cdot \mathbf{x} - b_i) \,:\, \mathbf{c}_i \in \mathcal{R}^o,\, b_i \in \mathcal{R},\, \boldsymbol{\omega}_i \in \mathcal{S}^{d-1} \right\},$$

Neural Net as a new class of approximating functions

- **Universal Approximation Theorem** (Cybenko (1989), Hornik-Stinchcombe-White (1989))

  $\mathcal{M}(\sigma, d) = \{v(\mathbf{x}) \in \mathcal{M}_n(\sigma, d) : n \in \mathbb{Z}_+\}$ is dense in $C(K)$ for any compact set $K \in \mathcal{R}^d$, provided that σ is not a polynomial.

- **A Priori Error Estimate** (DeVore-Oskolkov-Petrushev (1997), DeVore-Hanin-Petrova, Yarosky, Shen-Yang-Zhang, E-Wojtowytsch, Siegle-Xu, ……)

  - Why using NN instead of polynomials, finite elements, ...?
  - Why using more than one-hidden layer?
  - How to design NN architecture?
  - ……

PDE and Equivalent Optimization Formulations

- **Partial Differential Equation**
- **Variational Formulation**
- **Equivalent Optimization Formulations**
  - Energy functionals (DeepRitz (E-Yu), ...)
    applicable to a small class of problems,
    Dirichlet boundary conditions (penalization)
  - Various least-squares functionals (PINN (Karniadakis et. al.), LSNN (C.-Chen-Liu), ...)
    applicable to all problems,
    boundary conditions (stabilization),
    what are proper least-squares functionals?

Least-squares Methods for Elliptic Partial Differential Equations

- **Elliptic Partial Differential Equations**

$$\begin{cases} -\text{div}\,(A\nabla u) + \boldsymbol{\beta}\cdot\nabla u + c\,u = f & \text{in } \Omega, \\ u\big|_{\Gamma_D} = g_D, \quad (\mathbf{n}\cdot A\nabla u)\big|_{\Gamma_N} = g_N \end{cases}$$

- **Primitive Least-squares problem** (Bramble-Schatz (1971), …)

$$\text{find } u \in H^2(\Omega) \text{ such that } \quad L(u;\mathbf{f}) = \min_{v\in H^2(\Omega)} L(v;\mathbf{f})$$

where the primitive least-square functional is given by

$$L(v;\mathbf{f}) = \|f + \nabla\cdot(A\nabla v) - Xv\|^2_{0,\Omega} + \|v - g_D\|^2_{3/2\Gamma_D} + \|\mathbf{n}\cdot(A\nabla v) - g_N\|^2_{1/2,\Gamma_N}$$

- **Coercivity and Continuity**

$$\alpha\,\|v\|^2_{2,\Omega} \le L(v;\mathbf{0}) \le C\,\|v\|^2_{2,\Omega}$$

Least-Squares Methods Based on First-Order System

- **First-order system**

$$\left\{\begin{array}{rcll} A^{-1}\boldsymbol{\sigma} + \nabla u & = & \mathbf{0} & \text{in} \quad \Omega, \\ \nabla\cdot\boldsymbol{\sigma} + Xu & = & f & \text{in} \quad \Omega, \\ \nabla\times(A^{-1}\boldsymbol{\sigma}) & = & \mathbf{0} & \text{in} \quad \Omega \end{array}\right.$$

With boundary conditions

$$u\big|_{\Gamma_D} = g_D, \quad (\mathbf{n}\cdot\boldsymbol{\sigma})\big|_{\Gamma_N} = g_N, \quad \text{and } (\mathbf{n}\times\boldsymbol{\sigma})\big|_{\Gamma_D} = -\mathbf{n}\times\nabla g_D$$

- **Least-squares methods**
  - the weighted (inverse) norm method

    (Aziz-Kellogg-Stephens 85, Bramble-Lazarov-Pasiciak 94, …)
  - the div method (Carey-Pehlivanov 94, C.-Lazarov-Manteuffel-McCormick 94, …)
  - the div-curl method (Chang 92, Jiang 93, C.-Manteuffel-McCormick 97, …)

The Div Least-Squares Method

- **Div least-squares problem**

Find $(\boldsymbol{\sigma},\, u) \in \Sigma_N \times U_D \equiv H_N(\text{div};\Omega) \times H^1_D(\Omega)$ such that

$$G(\boldsymbol{\sigma},\, u;\, \mathbf{f}) = \min_{(\boldsymbol{\tau},\, v) \in \Sigma_N \times U_D} G(\boldsymbol{\tau},\, v;\, \mathbf{f}),$$

where the div least-squares functional is given by

$$G(\boldsymbol{\tau}, v; \mathbf{f}) = \|A^{-\frac{1}{2}}(\boldsymbol{\tau} + A\nabla v)\|^2 + \|\nabla \cdot \boldsymbol{\tau} + Xv - f\|^2$$

- **Variational problem** find $(\boldsymbol{\sigma},\, u) \in \Sigma_N \times U_D$ such that

$$b(\boldsymbol{\sigma},\, u;\, \boldsymbol{\tau},\, v) = f(\boldsymbol{\tau},\, v), \quad \forall\, (\boldsymbol{\tau},\, v) \in \Sigma_N \times U_D.$$

- **Continuity and coercivity** (C.-Lazarov-Manteuffel-McCormick 94)

$$\begin{cases} b(\boldsymbol{\sigma},\, u;\, \boldsymbol{\tau},\, v) & \leq & C\, |||(\boldsymbol{\sigma},\, u)|||\; |||(\boldsymbol{\tau},\, v)|||, & \forall\, (\boldsymbol{\sigma},\, u),\, (\boldsymbol{\tau},\, v) \in \Sigma_N \times U_D \\ b(\boldsymbol{\tau},\, v;\, \boldsymbol{\tau},\, v) & \geq & \alpha\, |||(\boldsymbol{\tau},\, v)|||, & \forall\, (\boldsymbol{\tau},\, v) \in \Sigma_N \times U_D \end{cases}$$

LSNN Method

- **Div LS functional with BCs:**

$$\mathcal{G}(\boldsymbol{\tau}, v; \mathbf{f}) = \|A^{-\frac{1}{2}}\boldsymbol{\tau} + A^{\frac{1}{2}}\nabla v\|^2 + \|\nabla\cdot\boldsymbol{\tau} + Xv - f\|^2 + \|v - g_D\|^2_{\frac{1}{2},\Gamma_D} + \|\mathbf{n}\cdot(A\nabla v) - g_N\|^2_{-\frac{1}{2},\Gamma_N}$$

- **LSNN method:** Find $(\boldsymbol{\sigma}_N, u_N) \in \mathcal{M}_N(\sigma, d)^{d+1}$ such that

$$\mathcal{G}(\boldsymbol{\sigma}, u; \mathbf{f}) = \min_{(\boldsymbol{\tau}, v) \in \mathcal{M}_N(\sigma, d)^{d+1}} \mathcal{G}(\boldsymbol{\tau}, v; \mathbf{f})$$

- **Quasi-Optimal Approximation:**

$$|\!|\!|(\boldsymbol{\sigma} - \boldsymbol{\sigma}_N, u - u_N)|\!|\!| \le \left(\frac{M}{\alpha}\right)^{1/2} \inf_{(\boldsymbol{\tau}, v) \in \mathcal{M}_N(\sigma, d)^{d+1}} |\!|\!|(\boldsymbol{\sigma} - \boldsymbol{\tau}, u - v)|\!|\!|.$$

C.-Chen-Liu-Liu, JCP, 420 (2020), 109707

Div LSNN Method

Effect of Numerical Integration

Find $(\boldsymbol{\sigma}_{\mathcal{T}}, u_{\mathcal{T}}) \in \mathcal{M}_N(\sigma, d)^{d+1}$ such that

$$\mathcal{G}_{\mathcal{T}}(\boldsymbol{\sigma}_{\mathcal{T}}, u_{\mathcal{T}}; \mathbf{f}) = \min_{(\boldsymbol{\tau}, v) \in \mathcal{M}_N(\sigma, d)^{d+1}} \mathcal{G}_{\mathcal{T}}(\boldsymbol{\tau}, v; \mathbf{f})$$

$$C \,|\!|\!|(\boldsymbol{\sigma} - \boldsymbol{\sigma}_{\mathcal{T}}, u - u_{\mathcal{T}})|\!|\!|$$

$$\leq \inf_{(\boldsymbol{\tau}, v) \in \mathcal{M}_N(\sigma, L)^{d+1}} \left\{ |\!|\!|(\boldsymbol{\sigma} - \boldsymbol{\tau}, u - v)|\!|\!| + \sup_{\phi \in \mathcal{M}_{2N}(\sigma, L)^{d+1}} \frac{|a(v, \phi) - a_{\mathcal{T}}(v, \phi)|}{\|\phi\|_a} \right\}$$

$$\sup_{\phi \in \mathcal{M}_{2N}(\sigma, L)^{d+1}} \frac{|f(\phi) - f_{\mathcal{T}}(\phi)|}{\|\phi\|_a}$$

Numerical Issues for NN-based Methods

- **Numerical Issues (unlike finite elements)**
  - Numerical Integration (important): adaptive numerical integration
  - Numerical Differentiation (critical): discrete differential operator
  - Algebraic solver (training NN) (critical): methods of gradient descent ???

Lecture II. Linear Advection-Reaction Problem

- **Linear advection-reaction problem**

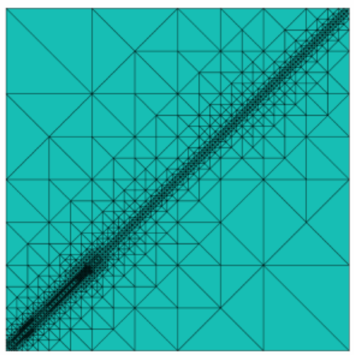
$$\begin{cases} u_{\boldsymbol{\beta}} + \gamma\, u &= f, \quad \text{in } \Omega, \\ u &= g, \quad \text{on } \Gamma_-, \end{cases}$$

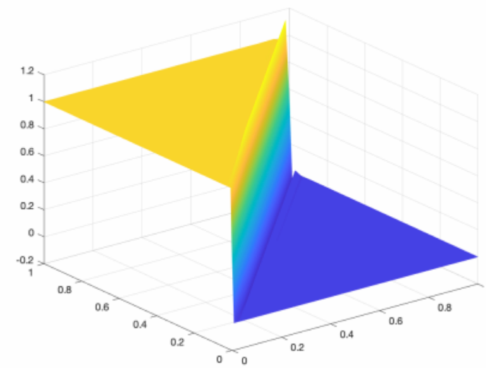
where $u_{\boldsymbol{\beta}}$ is the directional derivative of *u* along advection velocity field

- **Computational difficulty**

The solution is discontinuous, the known interface should not be used!

Model transport problem: $u_t + u_x = 0$



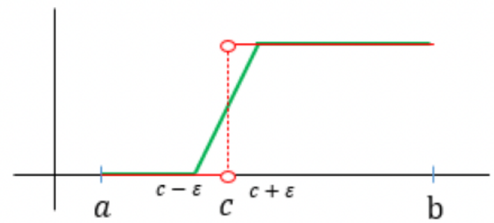


Liu-Zhang, CMAME, 2020

Approximation to Unit Step Function with known interface

- **Unit step function with know interface c**

$$f(x) = \begin{cases} 0, & x \in (a, c), \\ 1, & x \in [c, b). \end{cases}$$



- **Continuous piece-wise linear (CPWL) approximation**

$$p(x) = \begin{cases} 0, & x \in (a, c-\varepsilon), \\ \frac{x-(c-\epsilon)}{2\varepsilon}, & x \in [c-\varepsilon, c+\varepsilon], \\ 1, & x \in (c+\varepsilon, b). \end{cases}$$

- **Error estimate**

$$\|f-p\|_{L^\infty(I)} = \frac{1}{2} \quad \text{and} \quad \|f-p\|_{L^p(I)} = \frac{\varepsilon^{1/p}}{2^{1-1/p}(1+p)^{1/p}}.$$

Approximation to Unit Step Function with Unknown Interface

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

$$f(x) = \begin{cases} 0, & x \in (a, c), \\ 1, & x \in [c, b). \end{cases} \qquad p(x) = \begin{cases} 0, & x \in (a, c-\varepsilon), \\ \frac{x-(c-\epsilon)}{2\varepsilon}, & x \in [c-\varepsilon, c+\varepsilon], \\ 1, & x \in (c+\varepsilon, b). \end{cases}$$

- **Error estimate**

$$\|f-p\|_{L^\infty(I)} = \frac{1}{2} \quad \text{and} \quad \|f-p\|_{L^p(I)} = \frac{\varepsilon^{1/p}}{2^{1-1/p}(1+p)^{1/p}}.$$

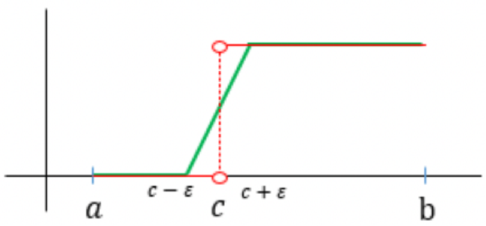
- **CPWL approximations on fixed quasi-uniform mesh**
  - **very fine mesh-size h= ε**
  - **overshooting and oscillation**
- **Is there a better class of approximating functions?**

YES, free knot splines.

Approximation to Unit Step Function with Unknown Interface

- **Unit step function with unknow interface c**

$$f(x) = \begin{cases} 0, & x \in (a, c), \\ 1, & x \in [c, b). \end{cases}$$



- **Neural network approximation**

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

!!! One-hidden layer with two neurons !!!

- **Error estimate (1/p order is irrelevant)**

$$\|f - p\|_{L^\infty(I)} = \frac{1}{2} \quad \text{and} \quad \|f - p\|_{L^p(I)} = \frac{\varepsilon^{1/p}}{2^{1-1/p}(1+p)^{1/p}}.$$

Neural Networks (NNs): a class of new approximating functions

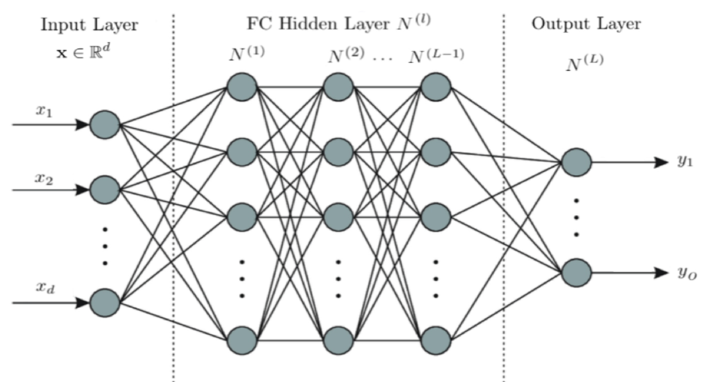
Fully-connected (Multi-Layer Perceptron) NN (Rosenblatt 1958)

- **DNN function (models)**

Let $\mathbf{x}^{(0)} = \mathbf{x}$ and $\mathbf{x}^{(i)} = \sigma\left(W^{(i)}_{n\times(n+1)}\begin{bmatrix}1\\ \mathbf{x}^{(i-1)}\end{bmatrix}\right)$ for $i = 1,\ldots,l$

$$u(\mathbf{x}; W) = W^{(l+1)}_{1\times(n+1)}\begin{bmatrix}1\\ \mathbf{x}^{(l)}\end{bmatrix}$$

where $W = \left[W^{(1)}_{n\times(d+1)}, W^{(2)}_{n\times(n+1)}, \ldots, W^{(l)}_{n\times(n+1)}, W^{(l+1)}_{1\times(n+1)}\right]$



- **One –Hidden layer**

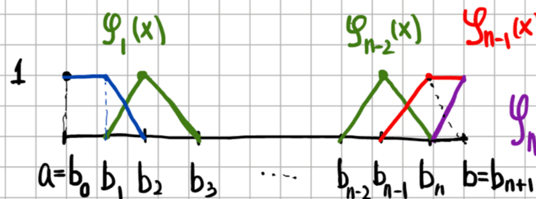
$$\mathcal{M}_n(\sigma, d) = \left\{\mathbf{c}_0 + \sum_{i=1}^{n}\mathbf{c}_i\sigma(\boldsymbol{\omega}_i\cdot\mathbf{x} - b_i)\ :\ \mathbf{c}_i \in \mathcal{R}^o,\ b_i \in \mathcal{R},\ \boldsymbol{\omega}_i \in \mathcal{S}^{d-1}\right\},$$

1D: One-hidden layer NN = Free Knot Spline

Local Basis Functions for ReLU NN

Let $\rho_0(x) = 1$, $\rho_i = (x - b_i)_+ = \begin{cases} 0, & x \le b_i \\ x - b_i, & x > b_i \end{cases}$ for $i = 1, \cdots, n$.

assume that $a = b_0 \le b_1 < b_2 < \cdots < b_n < b_{n+1} = b$ and $h_i = b_{i+1} - b_i$



$\varphi_{n-1}(x) = h_{n-1}^{-1}\left(\rho_{n-1}(x) - \rho_n(x)\right)$

$\varphi_n(x) = h_n^{-1}\rho_n(x)$

$\varphi_0(x) = \rho_0(x) - \sum_{i=1}^{n-1} \varphi_i(x)$

$=$

$$\varphi_1(x) = h_1^{-1}\rho_1(x) - \left(h_1^{-1} + h_2^{-1}\right)\rho_2(x) + h_2^{-1}\rho_3(x) = h_1^{-1}\left(\rho_1(x) - \rho_2(x)\right) - h_2^{-1}\left(\rho_2(x) - \rho_3(x)\right)$$

$$\varphi_i(x) = h_i^{-1}\rho_i(x) - \left(h_i^{-1} + h_{i+1}^{-1}\right)\rho_{i+1}(x) + h_{i+1}^{-1}\rho_{i+2}(x) = h_i^{-1}\left(\rho_i(x) - \rho_{i+1}(x)\right) - h_{i+1}^{-1}\left(\rho_{i+1}(x) - \rho_{i+2}(x)\right)$$

for $i = 1, 2, \cdots, n-2$

$$\varphi_{n-1}(x) = h_{n-1}^{-1}\left(\rho_{n-1}(x) - \rho_n(x)\right), \quad \varphi_n(x) = h_n^{-1}\rho_n(x)$$

$$\varphi_0(x) = \rho_0(x) - h_1^{-1}\left(\rho_1(x) - \rho_2(x)\right) = \rho_0(x) - h_1^{-1}\rho_1(x) + h_1^{-1}\rho_2(x)$$

One-hidden Layer NN in R^d

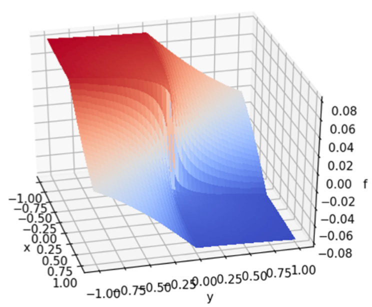
- **One-hidden Layer NN**

$$\mathcal{M}_n(\sigma, d) = \left\{ \mathbf{c}_0 + \sum_{i=1}^{n} \mathbf{c}_i \sigma(\boldsymbol{\omega}_i \cdot \mathbf{x} - b_i) \, : \, \mathbf{c}_i \in \mathcal{R}^o, \, b_i \in \mathcal{R}, \, \boldsymbol{\omega}_i \in \mathcal{S}^{d-1} \right\},$$

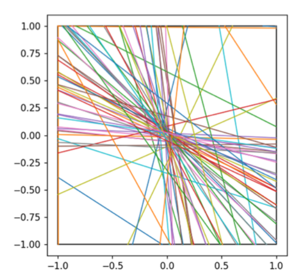
- **Linearly Independence**

Lemma 2.1. *Assume that hyper-planes* $\{\boldsymbol{\omega}_i \cdot \mathbf{x} = b_i\}_{i=1}^n$ *are distinct. Then* $\{\varphi_i(\mathbf{x}; \boldsymbol{\omega}_i, b_i)\}_{i=0}^n$ *are linearly independent.*

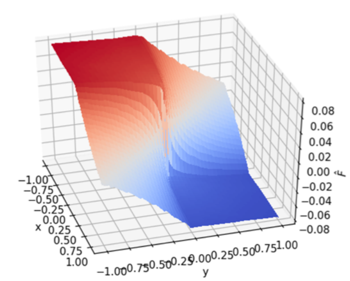
- **Breaking Hyper-Planes** $\mathcal{P}_i \, : \, \boldsymbol{\omega}_i \cdot \mathbf{x} - b_i = 0 \quad \text{for } i = 1, \ldots, n.$
- **Physical Partition of a given NN function**



(a) Target function f(x,y)



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



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

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

- **Piecewise Constant function with unknow interface**

P. Petersen and F. Voigtlaender (2018) (For C^1 and d=2, L=36)

Theorem 3.5. *For $r \in \mathbb{N}$, $d \in \mathbb{N}_{\geq 2}$, and $p, \beta, B > 0$, there are constants $c = c(d, r, p, \beta, B) > 0$ and $s = s(d, r, p, \beta, B) \in \mathbb{N}$, such that for any $K \in \mathcal{K}_{r,\beta,d,B}$ and any $\varepsilon \in (0, 1/2)$, there is a neural network Φ_ε^K with at most $(3 + \lceil \log_2 \beta \rceil) \cdot (11 + 2\beta/d)$ layers, and at most $c \cdot \varepsilon^{-p(d-1)/\beta}$ nonzero, (s, ε)-quantized weights such that*

$$\|\mathrm{R}_\varrho(\Phi_\varepsilon^K) - \chi_K\|_{L^p([-1/2,1/2]^d)} < \varepsilon \quad \text{and} \quad \|\mathrm{R}_\varrho(\Phi_\varepsilon^K)\|_{\sup} \leq 1.$$

Remark 3.6. *Theorem 3.5 establishes approximation rates for piecewise constant functions. It should be noted that the number of required layers is fixed and only depends on the dimension d and the regularity parameter β; in particular, it does* not *depend on the approximation accuracy ε.*

C., J. Choi, and M. Liu (2022) (For C^1 and d=2, L=2)

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 DNN with L=$\lceil \log_2(d+1) \rceil$ hidden layers such that for any given $\varepsilon > 0$, we have

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

LS formulation for linear advection-reaction problem

- **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

$$\alpha\, |||v|||^2_{\boldsymbol{\beta}} \leq \mathcal{L}(v; \mathbf{0}) \leq M\, |||v|||^2_{\boldsymbol{\beta}}$$ where $|||v|||_{\boldsymbol{\beta}} = \left(\|v\|^2_{0,\Omega} + \|v_{\boldsymbol{\beta}}\|^2_{0,\Omega}\right)^{1/2}$

De Sterck-Manteuffel-McCormick-Olson, 2004, Bochev-Gunzburger, 2016

Least-squares neural network (LSNN) method

- **LSNN method** find $u_N \in \mathcal{M}(d,n)$ such that

$$\mathcal{L}(u_N, \mathbf{f}) = \min_{v \in \mathcal{M}(d,n)} \mathcal{L}(v, \mathbf{f})$$

where $\mathcal{M}(d, 1, \lceil \log_2(d+1) \rceil + 1, n) = \mathcal{M}(d,n)$

- **Quasi-optimal approximation**

$$|||u - u_N|||_{\boldsymbol{\beta}} \le \left(\frac{M}{\alpha}\right)^{1/2} \inf_{v \in \mathcal{M}(d,n)} |||u - v|||_{\boldsymbol{\beta}}.$$

- **A priori error estimate**

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

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

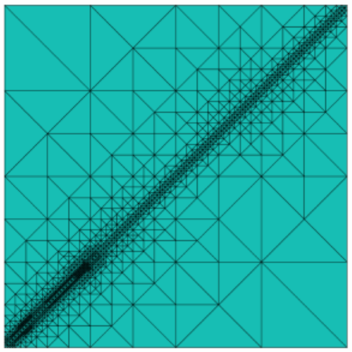
Numerical Issues for NN-based Methods

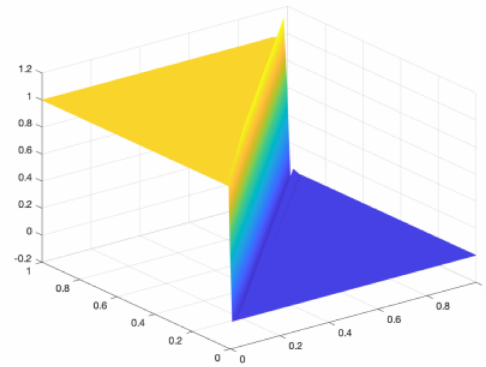
- **LSNN method** find $u_N \in \mathcal{M}(d,n) = \mathcal{M}(d,1,\lceil \log_2(d+1) \rceil + 1, n)$ such that

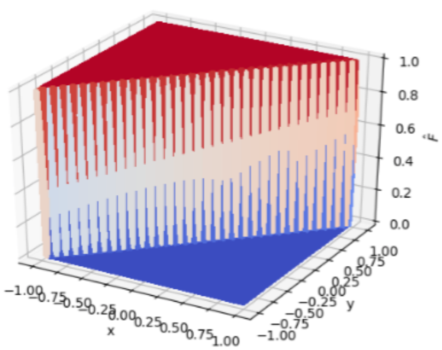
$$\mathcal{L}(u_N, \mathbf{f}) = \min_{v \in \mathcal{M}(d,n)} \mathcal{L}(v, \mathbf{f})$$

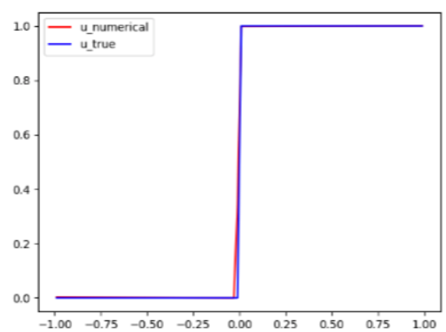
- **Numerical Issues (unlike finite elements)**
  - Numerical Integration (important): adaptive numerical integration
  - Numerical Differentiation (critical): discrete directional derivative
  - Algebraic solver (training NN) (critical): methods of gradient descent ???

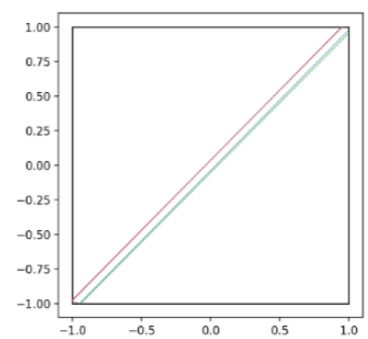
Famous Transport Equation $u_t+u_x=0$







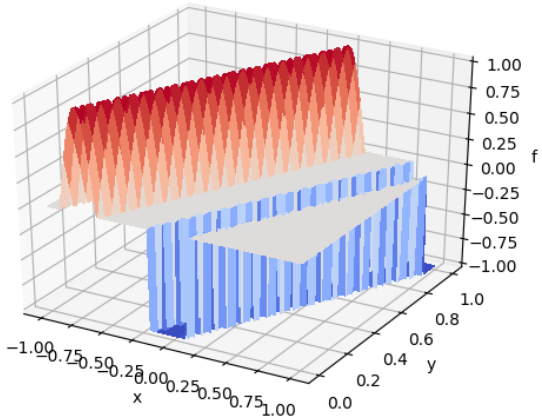




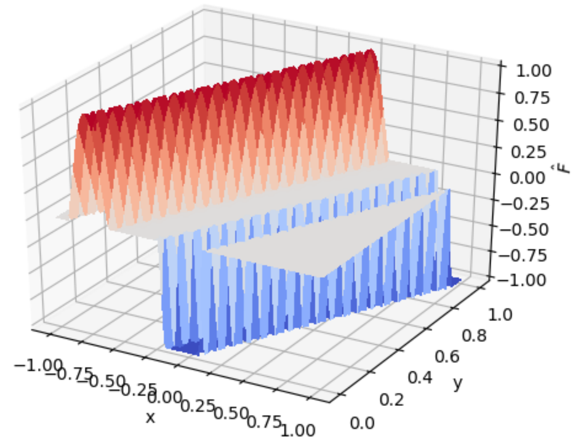
(2-6-1)

C.-Chen-Liu, JCP, 2021

Two discontinuous interfaces

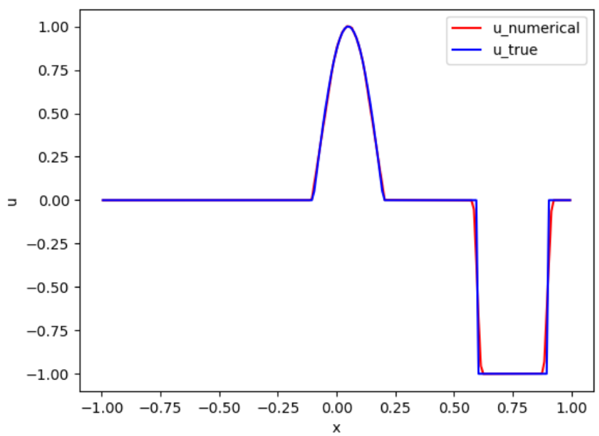


(a) Exact solution u

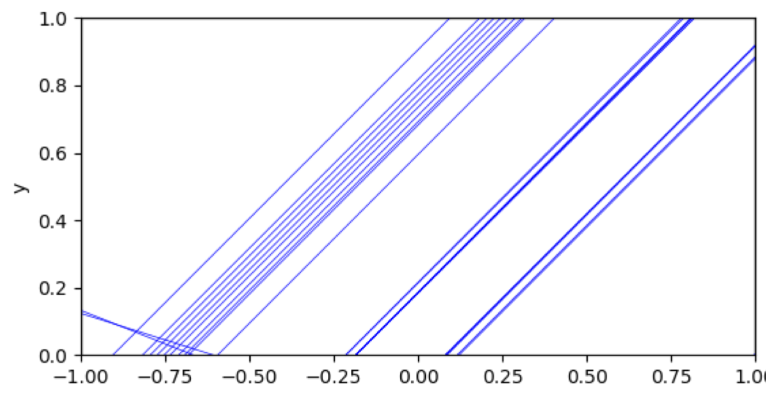


(b) Network approximation $\bar{u}_{\mathcal{T}}^{N}$

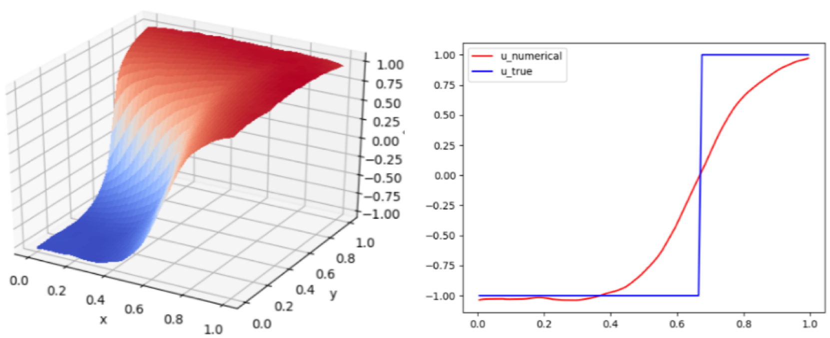
(2-31-1)



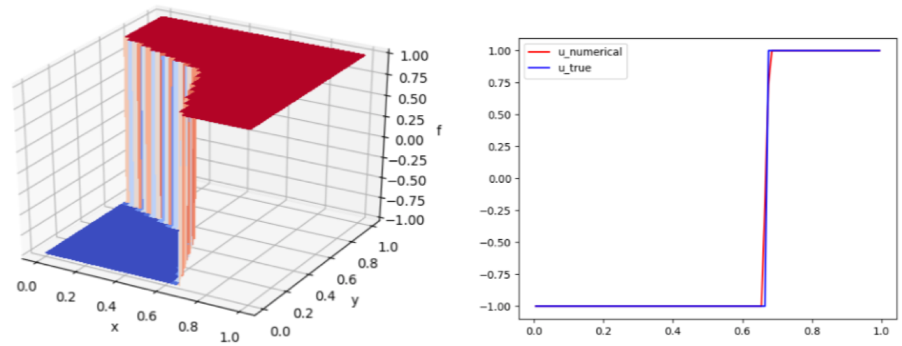
(c) Traces of the exact solution and approximation $\bar{u}_{\mathcal{T}}^{N}$ on the plane $y = 0.8$



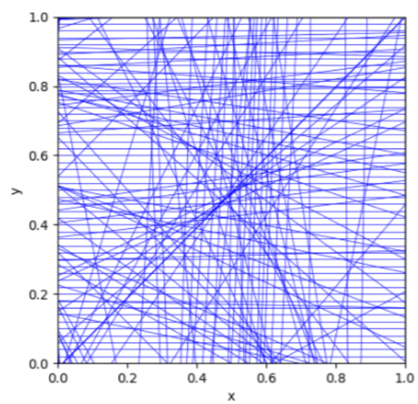
(d) Network breaking lines



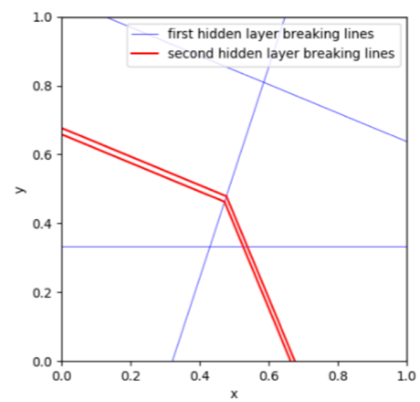
(2-200-1)



(2-5-5-1)

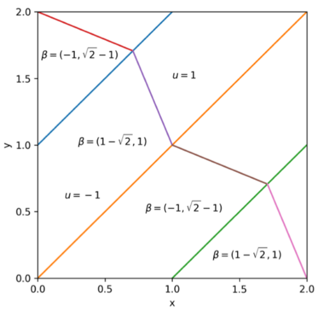


(e) 2-layer NN breaking lines

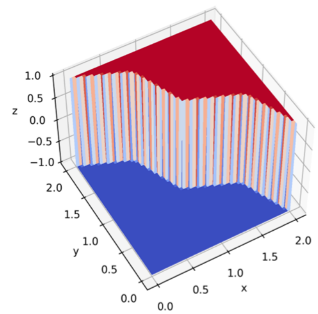


(f) 3-layer NN breaking lines

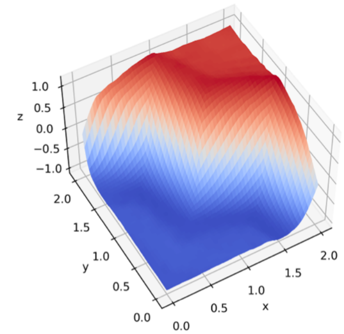
C.-Chen-Liu, LSNN method for linear advection-reaction equation, JCP, 443(2021), 110514.



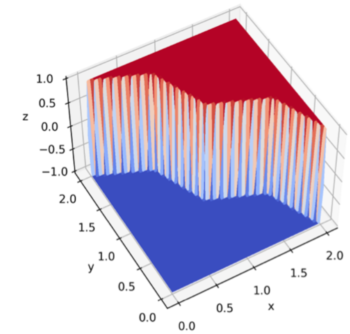
Velocity field



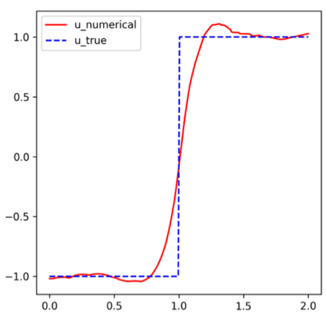
Exact solution

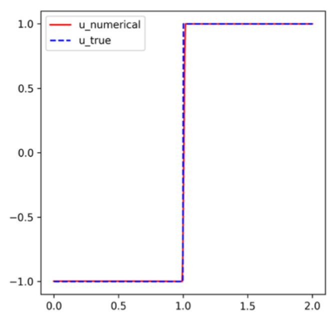


(2-300-1)

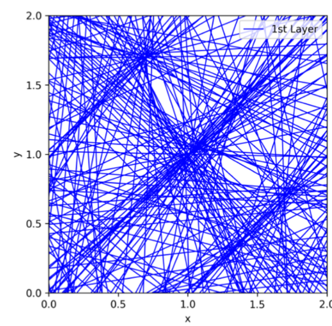


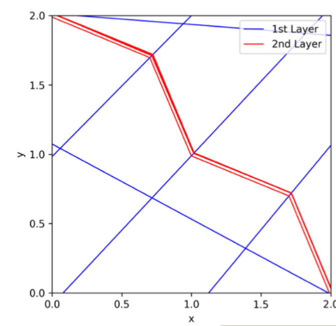
(2-6-6-1)





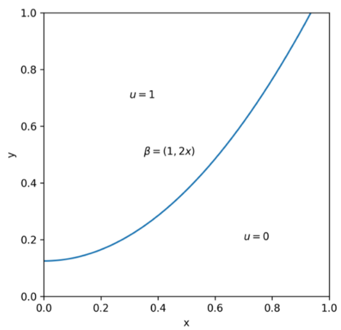
Trace on y=x



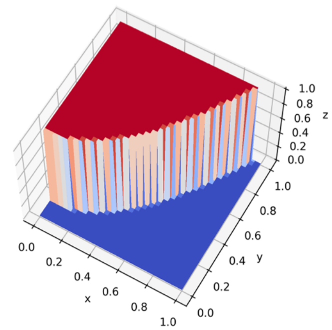


Physical partitions

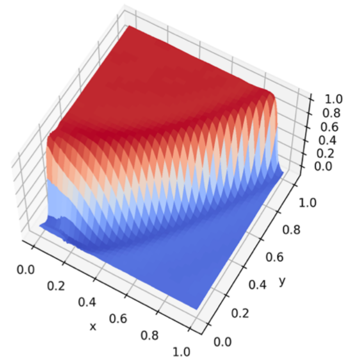
C.-Choi-Liu, LSNN method for linear advection-reaction equation: general discontinuous interface.



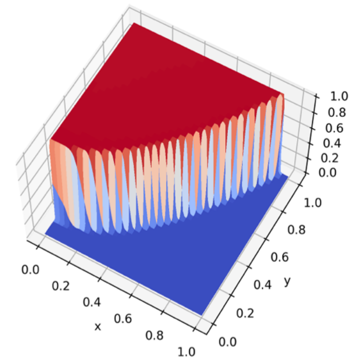
Curve interface



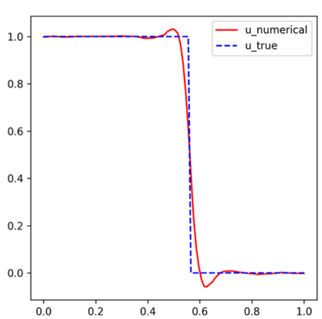
Exact solution

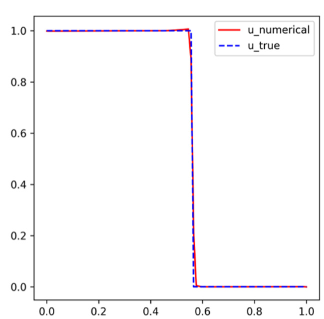


(2-3000-1)

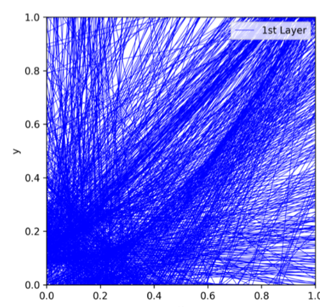


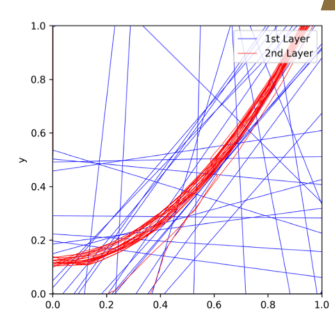
(2-60-60-1)





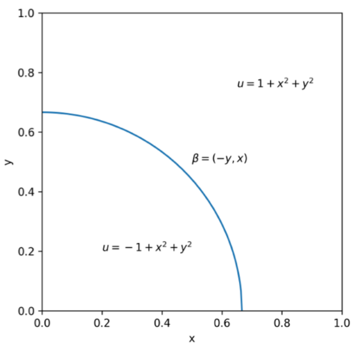
Trace on y=x



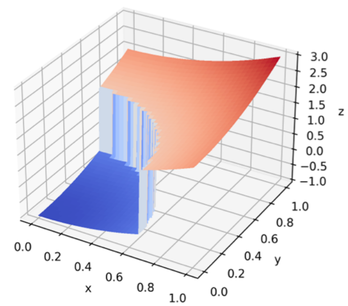


Physical partitions

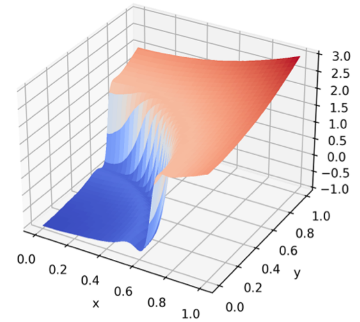
C.-Choi-Liu, LSNN method for linear advection-reaction equation: general discontinuous interface.



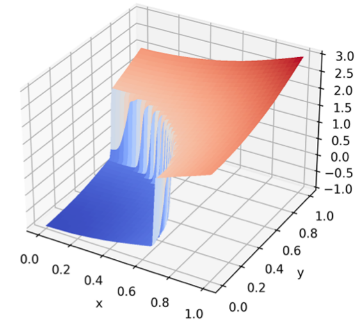
Curve interface



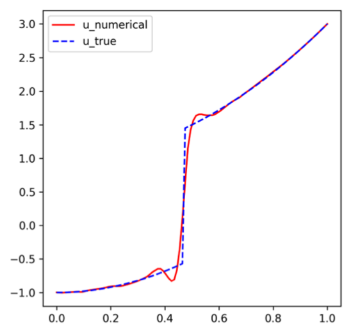
Exact solution

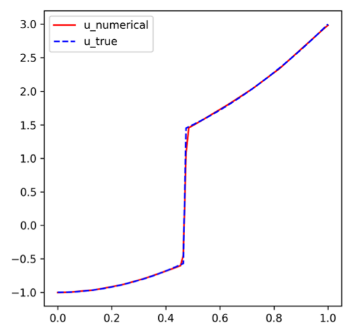


(2-4000-1)

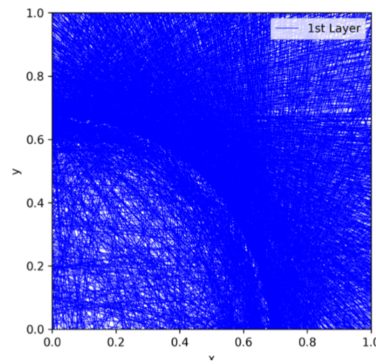


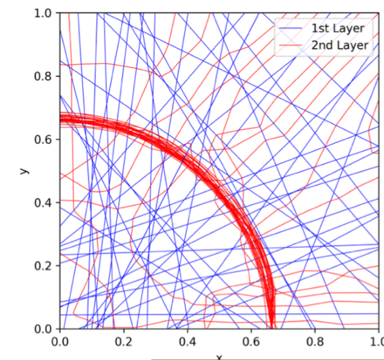
(2-65-65-1)





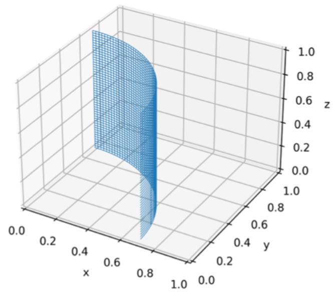
Trace on y=x



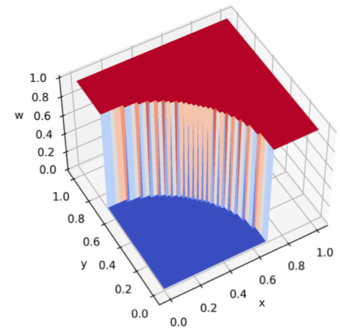


Physical partitions

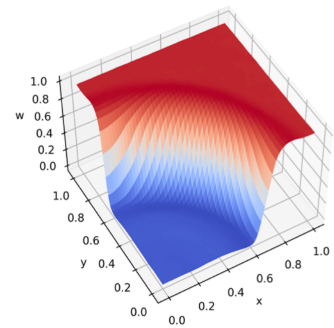
C.-Choi-Liu, LSNN method for linear advection-reaction equation: general discontinuous interface.



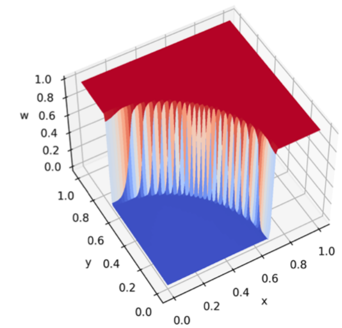
Surface interface



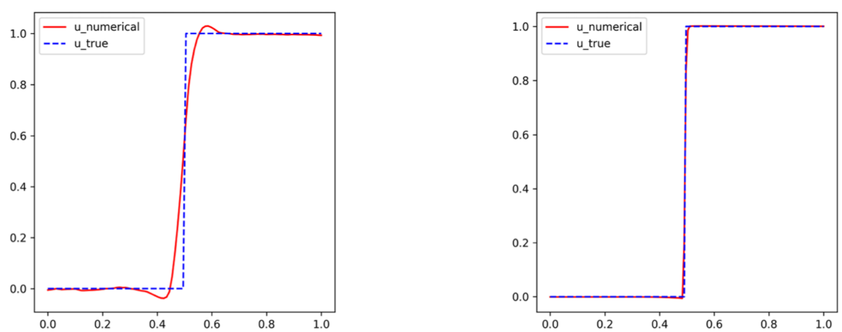
Exact solution at z=0.5



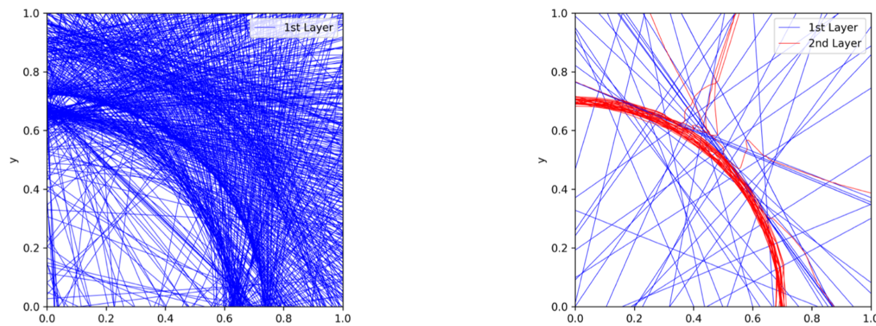
(2-1500-1)



(2-50-50-1)



Trace on y=x



Physical partitions

C.-Choi-Liu, LSNN method for linear advection-reaction equation: general discontinuous interface.

Lecture III. Scalar Hyperbolic Conservation Laws

- **Scalar Nonlinear Hyperbolic Conservation Laws**

$$\begin{cases} u_t(\mathbf{x},t) + \nabla_{\mathbf{x}} \cdot \mathbf{f}(u) &= 0, & \text{in } \Omega \times I, \\ u &= g, & \text{on } \Gamma_-, \\ u(\mathbf{x},0) &= u_0(\mathbf{x}), & \text{in } \Omega, \end{cases}$$

- **Numerical Difficulties**
  - Issues on mathematical theory of PDE
  - Solutions are discontinuous without a priori knowledge of locations

LSNN method for scalar nonlinear HCLs

- **Scalar nonlinear hyperbolic conservation laws**

$$u_t(\mathbf{x},t) + \nabla_{\mathbf{x}} \cdot \mathbf{f}(u) = 0, \text{ in } \Omega \times I, \quad u\big|_{\Gamma_-} = g, \quad u(\mathbf{x},0)\big|_{\Omega} = u_0(\mathbf{x})$$

- **Least-squares formulation**

Find $u \in V_{\mathbf{f}} = \big\{v \in L^2(\Omega \times I) | \, (\mathbf{f}(v), v) \in H(\text{div}; \Omega \times I)\big\}$ such that

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

where $\mathcal{L}(v; \mathbf{g}) = \|v_t + \nabla_{\mathbf{x}} \cdot \mathbf{f}(v)\|^2_{0,\Omega \times I} + \|v - g\|^2_{0,\Gamma_-} + \|v(\mathbf{x},0) - u_0(\mathbf{x})\|^2_{0,\Omega}$

- **Well-posedness???**

LSNN method for scalar nonlinear HCLs

- **Least-squares formulation**

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

$$\mathcal{L}\big(u;\, \mathbf{g}\big) = \min_{v \in V_{\mathbf{f}}} \mathcal{L}\big(v;\, \mathbf{g}\big)$$

where $\mathcal{L}(v; \mathbf{g}) = \|v_t + \nabla_{\mathbf{x}} \cdot \mathbf{f}(v)\|^2_{0,\Omega\times I} + \|v - g\|^2_{0,\Gamma_-} + \|v(\mathbf{x}, 0) - u_0(\mathbf{x})\|^2_{0,\Omega}$

- **LSNN method** finding $u^N(\mathbf{z}; \boldsymbol{\theta}^*) \in \mathcal{M}_N$ such that

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

- **Numerical Issues:** integration, differentiation, …

C.-Chen-Liu, ANM (2022) and arXiv: 2110.10895v2 [math.NA]

Discrete Divergence Operator

- **Divergence operator**

$$0 = u_t + \nabla_{\mathbf{x}} \cdot \mathbf{f}(u) = \mathbf{div}\,\big(u, \mathbf{f}(u)\big) = \mathbf{div}\,\mathbf{F}(u)$$

- **Discrete divergence operator**
  - + **based on conservative numerical schemes (C.-Chen-Liu, ANM(2022))**
  - + **new discrete divergence operator (C.-Chen-Liu** *arXiv:2110.10895v2[math.NA]*)

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.

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

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 4.2. *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*

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

Discrete Divergence Operator in 1D

- Primitive form over K_{ij}

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

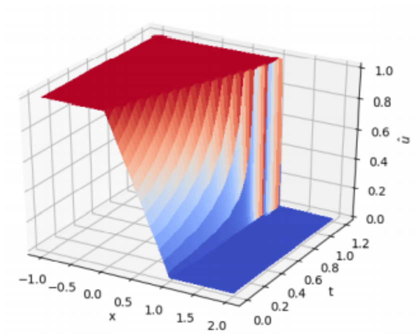
$$\approx \frac{1}{\delta}Q(\sigma(x_i, x_{i+1}; t); t_j, t_{j+1}, \hat{n}) + \frac{1}{h}Q(u(x; t_j, t_{j+1}); x_i, x_{i+1}, \hat{m}) = \mathrm{div}_{\mathcal{T}}\mathbf{F}\big(u_{ij}\big)$$

- Error estimate

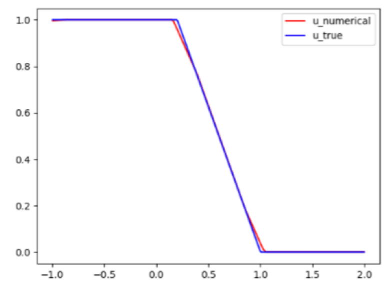
LEMMA 4.3. *Assume that u is a C^2 function of t and a piece-wise C^2 function of x on two vertical and two horizontal edges of K_{ij}, respectively. Moreover, u has only one discontinuous point on each horizontal edge. Then there exists a constant $C > 0$ such that*

$$\|\mathbf{div}_{\mathcal{T}}\mathbf{f}(u) - \mathrm{avg}_{\mathcal{T}}\mathbf{div}\,\mathbf{f}(u)\|_{L^p(K_{ij})}$$

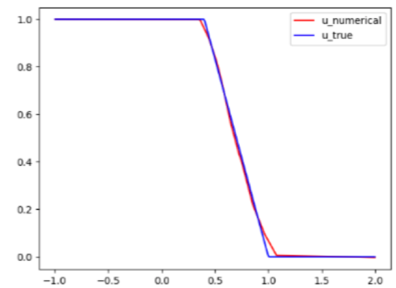
$$\leq C\left(\frac{h^{1/p}\delta^2}{\hat{n}^2} + \frac{h^2\delta^{1/p}}{\hat{m}^2} + \frac{h\delta^{1/p}}{\hat{m}^{1+1/q}}\right) + \frac{(h\delta)^{1/p}}{\hat{m}}\sum_{l=j}^{j+1}[\![u_{ij}]\!]_{t_l}. \quad (4.7)$$



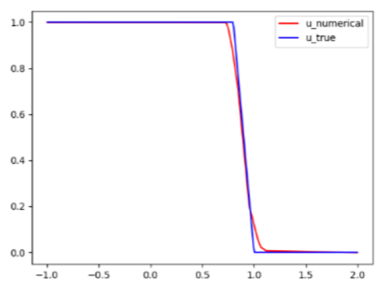
(a) Exact solution u on $\Omega \times I$



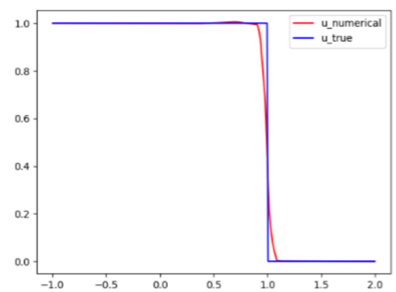
(a) Traces of exact solution and approximation $u_{1,\mathcal{T}}$ on the plane $t = 0.2$



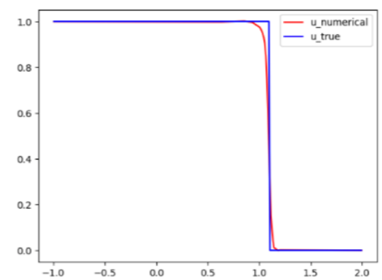
(c) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ on the plane $t = 0.4$



(d) Traces of exact solution and approximation $u_{4,\mathcal{T}}$ on the plane $t = 0.8$



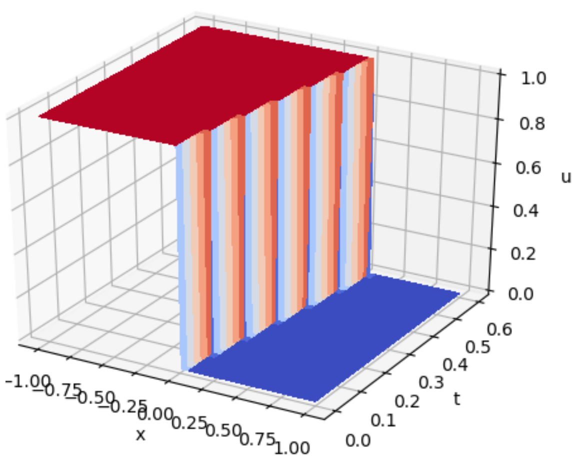
(e) Traces of exact solution and approximation $u_{5,\mathcal{T}}$ on the plane $t = 1.0$

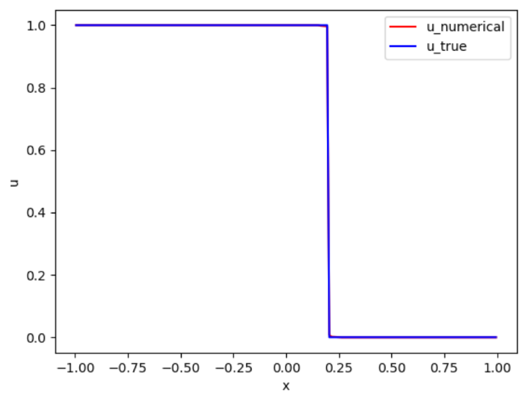


(f) Traces of exact solution and approximation $u_{6,\mathcal{T}}$ on the plane $t = 1.2$

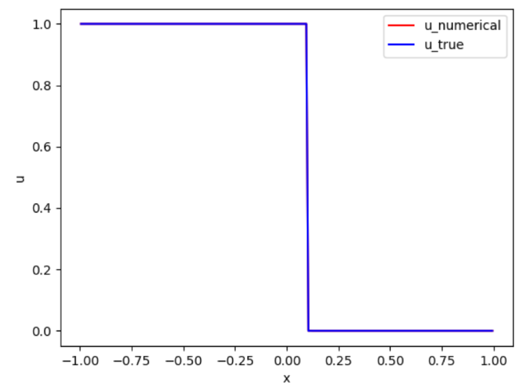
Inviscid Burger Equation $f(u) = \frac{1}{2}u^2$

Riemann Problem Shock formation: exact solution

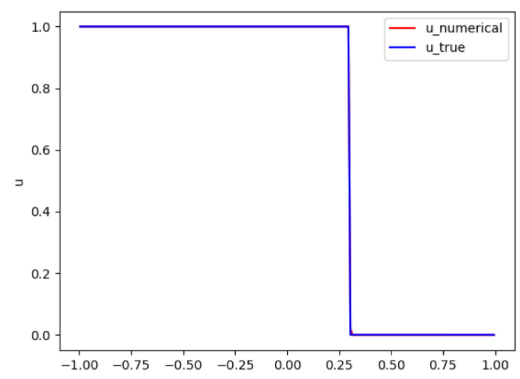




t=0.2



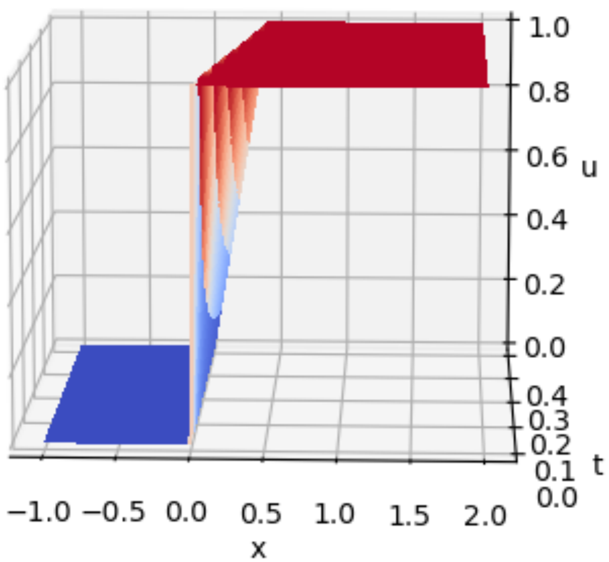
t=0.4

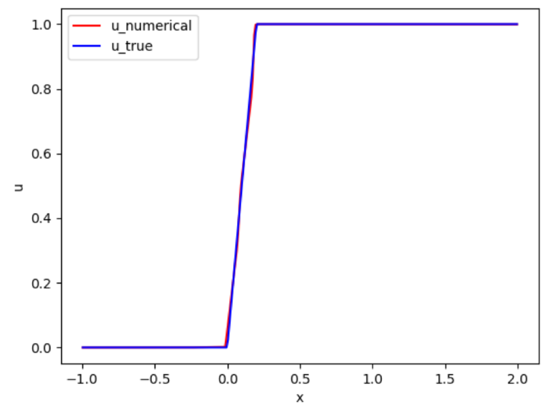


t=0.6 (2-10-10-1)

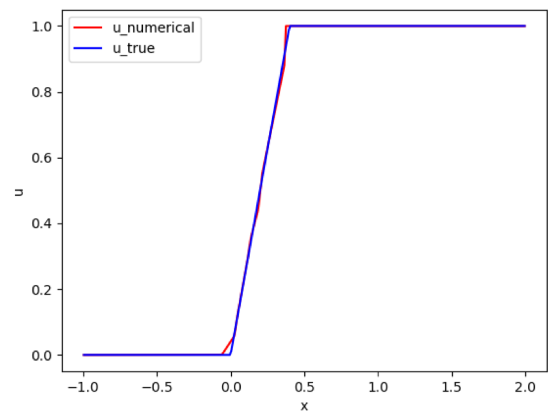
C.-Chen-Liu, arXiv: 2110.10895 [math.NA]

Riemann Problem
Rarefaction wave: exact solution





t=0.2

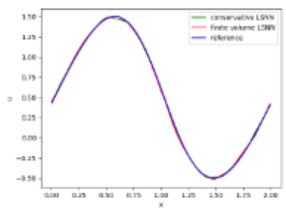


t=0.4

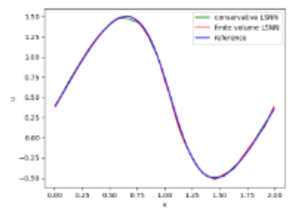
(2-10-10-1)

Inviscid Burgers equation with smooth initial

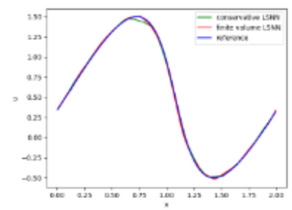
$$u_0(x) = 0.5 + \sin(\pi x).$$



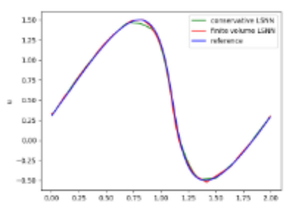
(a) Traces of reference and numerical solutions $u_{1,\mathcal{T}}$ on the plane $t = 0.05$



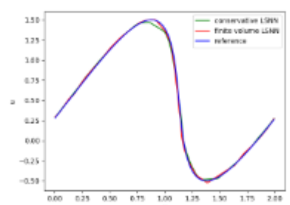
(b) Traces of reference and numerical solutions $u_{2,\mathcal{T}}$ on the plane $t = 0.1$



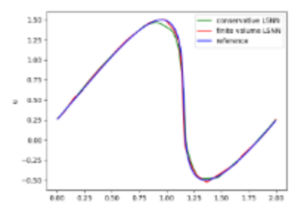
(c) Traces of reference and numerical solutions $u_{3,\mathcal{T}}$ on the plane $t = 0.15$



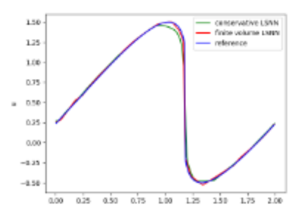
(d) Traces of reference and numerical solutions $u_{4,\mathcal{T}}$ on the plane $t = 0.2$



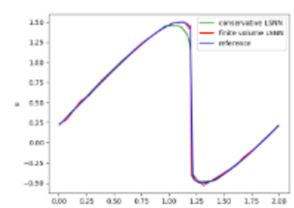
(e) Traces of reference and numerical solutions $u_{5,\mathcal{T}}$ on the plane $t = 0.25$



(f) Traces of reference and numerical solutions $u_{6,\mathcal{T}}$ on the plane $t = 0.3$



(g) Traces of reference and numerical solutions $u_{7,\mathcal{T}}$ on the plane $t = 0.35$

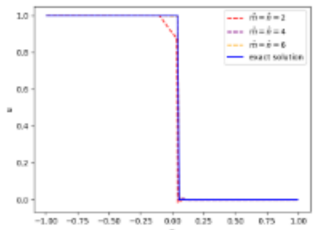


(h) Traces of reference and numerical solutions $u_{8,\mathcal{T}}$ on the plane $t = 0.4$

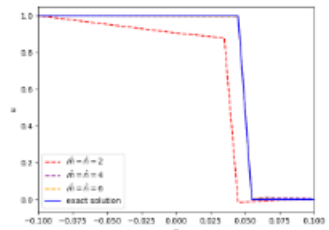
(2-30-30-1)

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

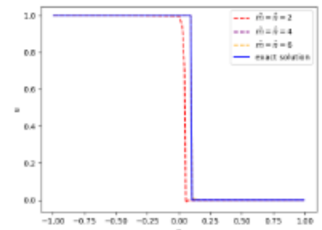
Riemann Problem with Higher order flux $f(u) = \frac{1}{4}u^4$



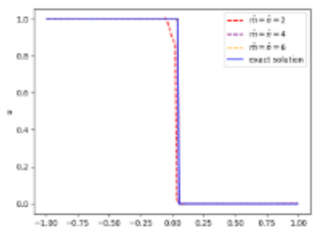
(a) Traces of exact and numerical solutions $u_{1,\mathcal{T}}$ using the trapezoidal rule on the plane $t = 0.2$



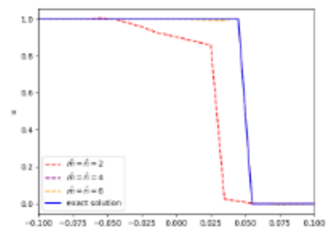
(b) Zoom-in plot near the discontinuous interface of sub-figure (a)



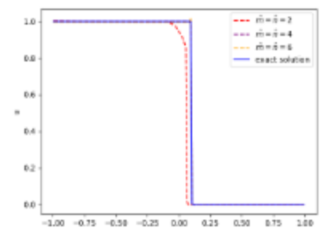
(c) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ using the trapezoidal rule on the plane $t = 0.4$



(d) Traces of exact and numerical solutions $u_{1,\mathcal{T}}$ using the mid-point rule on the plane $t = 0.2$



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

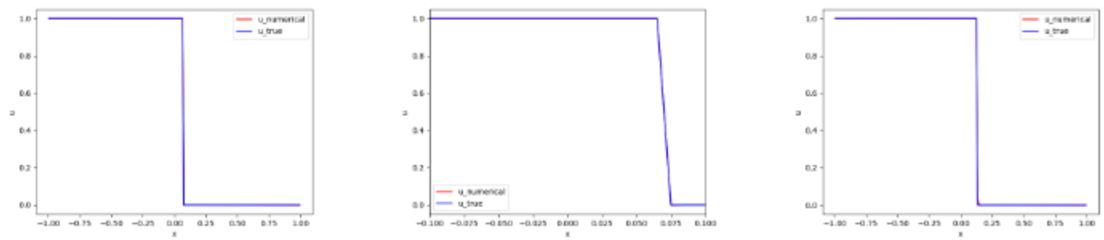


(f) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ using the mid-point rule on the plane $t = 0.4$

(2-10-10-1)

FIG. 5. *Numerical results of the problem with* $f(u) = \frac{1}{4}u^4$ *using the composite trapezoidal and mid-point rules*

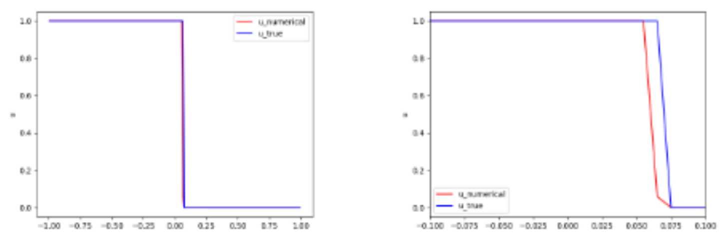
Riemann Problem with Non-convex flux $f(u) = \frac{1}{3}u^3$



(a) Traces of exact and numerical solutions $u_{1,\mathcal{T}}$ on the plane $t = 0.2$

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

(c) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ on the plane $t = 0.4$



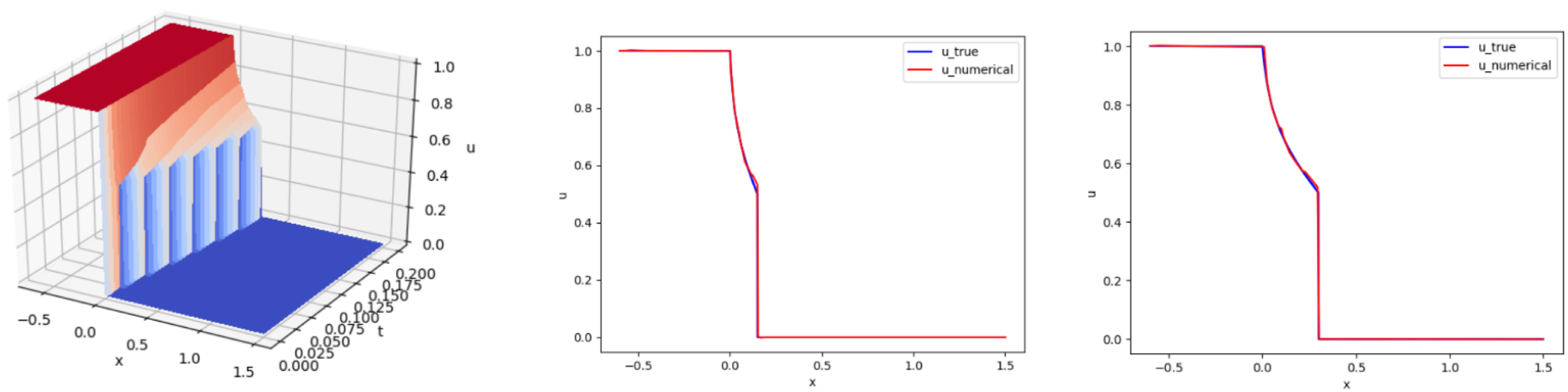
(d) Traces of exact and numerical solutions $u_{1,\mathcal{T}}$ on the plane $t = 0.2$

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

(2-10-10-1)

FIG. 6. *Numerical results of Riemann problem with a non-convex flux* $f(u) = \frac{1}{3}u^3$

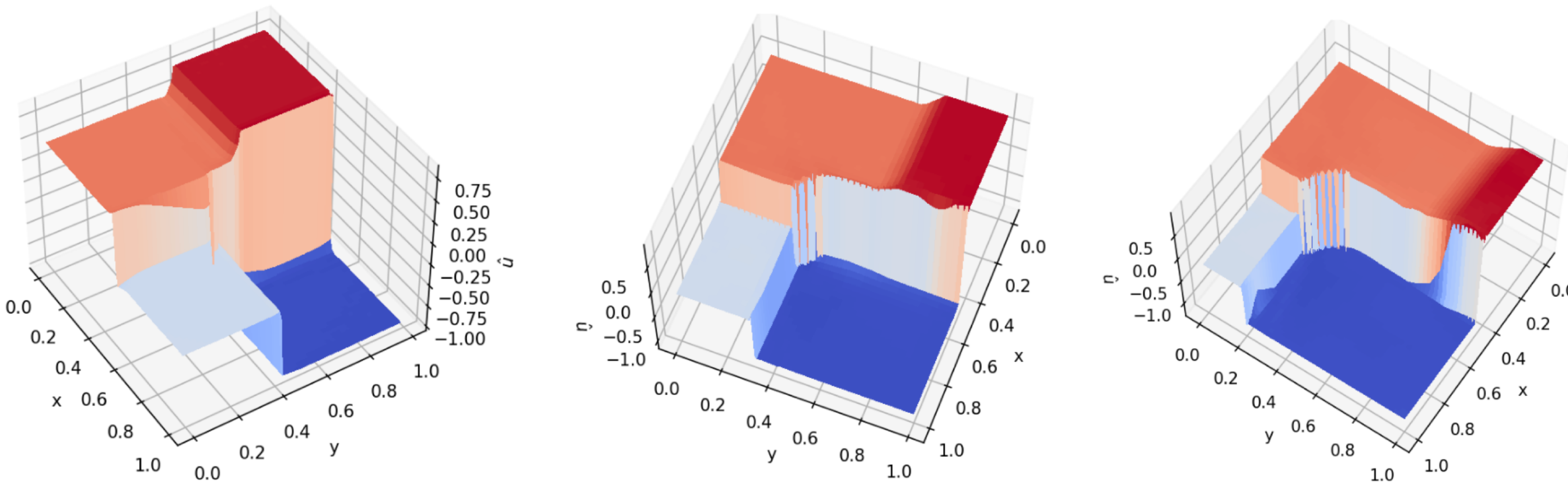
Buckley-Leverett Problem $f(u) = u(1-u)/[u^2 + a(1-u)^2]$



(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 Burgers' equation



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

FIG. 6. *Numerical results of 2D Burgers' equation.*

Lecture III. Adaptive Neural Network

- **NN Approximation**

find $u_N \in \mathcal{M}_N(\sigma, L)$ such that

$$\mathcal{L}\big(u_N(\cdot;\boldsymbol{\theta}^*);\, \mathbf{g}\big) = \min_{v\in\mathcal{M}_N(\sigma,L)} \mathcal{L}\big(v(\cdot;\boldsymbol{\theta});\, \mathbf{g}\big)$$

- **A Fundamental Question in Scientific Computing**

for a given $\epsilon > 0$, how to design an *optimal* NN $\mathcal{M}_N(\sigma, L)$ such that

$$|||u - u_N||| \le \epsilon\, |||u|||?$$

AutoML and Neural Architecture Search in AI does not address this question!!!

Adaptive Network Enhancement (ANE) method

ANE method (similar to Adaptive Mesh Refinement (AMR))

$$\text{train} \;\rightarrow\; \text{estimate} \;\rightarrow\; \text{enhance}.$$

Key question:

How to enhance NN when the current NN approximation is not within the prescribed accuracy?

Liu-C.-Chen, CAMWA, 113 (2022), 34-44; 103-116; JCP, 455 (2022), 111021

Network Enhancement Strategy (NES)

how many neurons will be added?

- **Global NES**

$$n_k = \min\left\{2n_{k-1}, \left\lceil \left(\hat{\xi}^{(k-1)}/\epsilon\right)^{1/\alpha_k} n_{k-1} \right\rceil\right\}$$

where $\alpha_k = \ln\left(\hat{\xi}^{(k-2)}/\hat{\xi}^{(k-1)}\right) \Big/ \ln\left(n_{k-1}/n_{k-2}\right)$, $\hat{\xi}^{(i)}$ is the estimator.

- **Local NES based on physical partition**

$$n_k = n_{k-1} + \#\hat{\mathcal{K}}_{k-1}$$

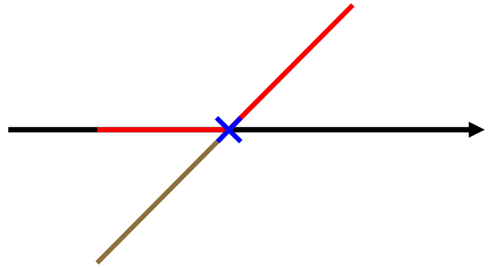
where $\hat{\mathcal{K}}_{k-1}$ is the set of marked physical subdomains

Physical Partition

$\mathcal{K}_n = \{K\}$ The partition formed by the hyper-break planes and the boundary of the domain

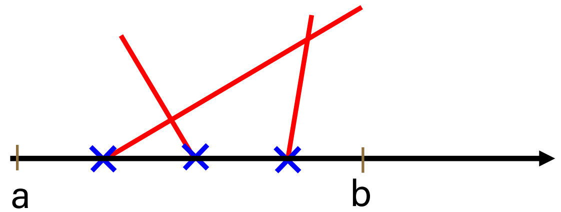
- Breaking Hyper-planes (Linear part of neurons)

$$N^{(l)}(\mathbf{x}^{(l-1)}) = \sigma(\boldsymbol{\omega}^{(l)}\mathbf{x}^{(l-1)} - \mathbf{b}^{(l)})$$



- Breaking Hyper planes (assuming ReLU^k)

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



When using ReLU^k as the activation function, NN functions are piece-wise defined on the physical partition

Local Enhancement Strategy

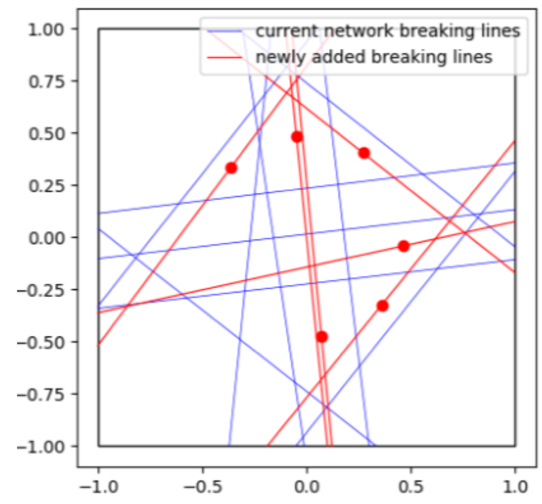
$$n_k = n_{k-1} + \#\hat{\mathcal{K}}_{k-1}$$

- **Marking Strategy**
  - The average marking strategy

$$\hat{\mathcal{K}}_n = \left\{ K \in \mathcal{K}_n \,:\, \xi_K \geq \frac{1}{\#\mathcal{K}_n} \sum_{K \in \mathcal{K}_n} \xi_K \right\}$$

  - The bulk marking strategy: finding a minimal subset such that

$$\sum_{K \in \hat{\mathcal{K}}_n} \xi_K^2 \geq \gamma_1 \sum_{K \in \mathcal{K}_n} \xi_K^2 \quad \text{for} \quad \gamma_1 \in (0, 1).$$



- **Newly added neuron initialization**

  Breaking hyper-plane is through the centroid of a marked sub-domain, and orient along the principal direction

Adaptive Network Enhancement (ANE) Method

ANE Algorithm (two-layer) Given a tolerance $\epsilon > 0$, starting with a two-layer ReLU NN with a small number of neurons,

(1) "solve" the optimization problem;

(2) estimate *a posteriori* error estimator $\xi = \left(\sum_{K \in \mathcal{K}} \xi_K^2 \right)^{1/2}$;

(3) if $\xi < \epsilon$, then stop; otherwise, go to Step (4);

(4) add new neurons to the network by using the network enhancement strategy, then go to Step (1).

Liu-C., CAMWA, 113 (2022), 34-44; 103-116.

Initialization in training non-convex optimization

- **Non-convex optimization**

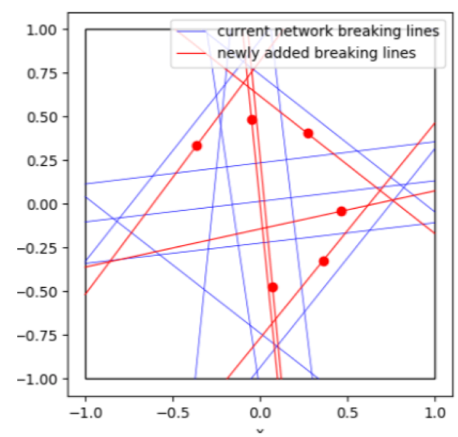
  many local and global optimizers $\Longrightarrow$ high cost and **uncertainty**

- **Initialization**

  - The method of continuation

    ANE is a good continuation process with respect to the number of neurons

  - Initialization of newly added neurons

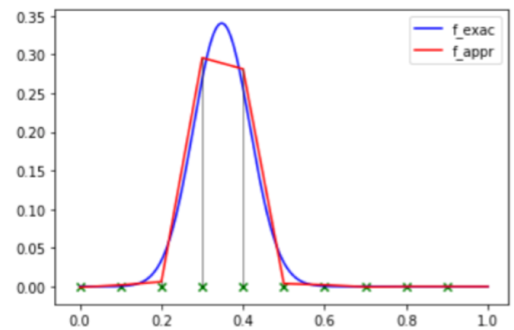


Adaptive 2-Layer NN

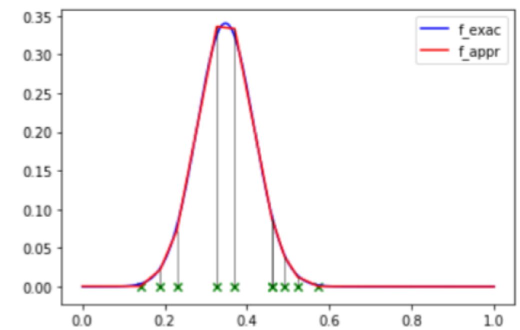
$$f(x) = x\left(e^{-(x-\frac{1}{3})^2/k} - e^{-\frac{4}{9}/k}\right)$$

Comparing adaptive neural network with fixed networks for testing problem (

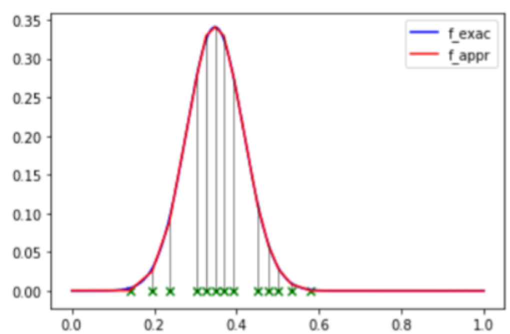
Network (neurons)	# Parameters	$\|f - f_{\mathcal{T}}\|_{\mathcal{T}}/\|f\|$
Fixed (20)	41	0.007644
Fixed (38)	77	0.003762
Adaptive (10→13→20)	41	0.003837



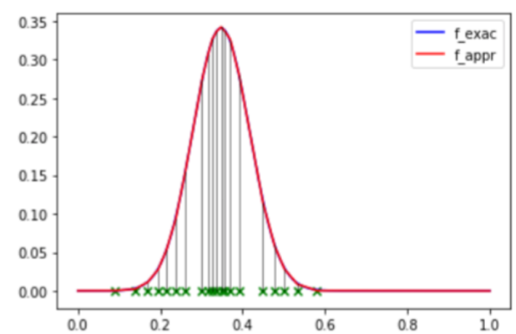
(a) Initial NN model with 10 uniform break points



(b) Optimized NN model with 10 neurons



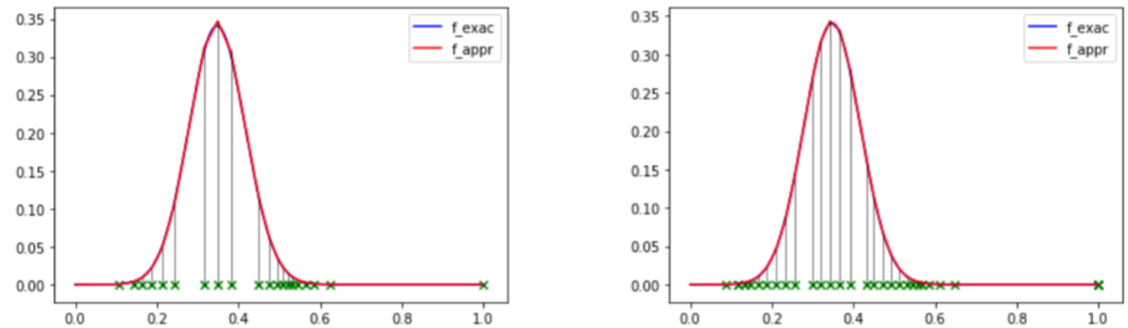
(c) Optimized NN model with 13 neurons using ANE



(d) Optimized NN model with 20 neurons using ANE

Adaptive 2-Layer NN

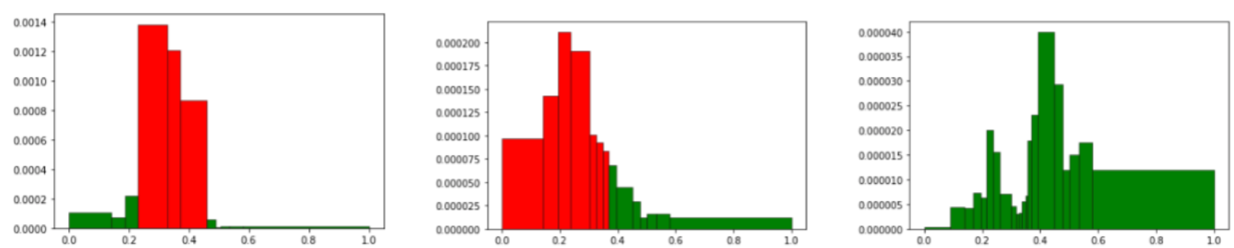
$$f(x) = x\left(e^{-(x-\frac{1}{3})^2/k} - e^{-\frac{4}{9}/k}\right)$$



(e) Fixed NN model with 20 neurons

(f) Fixed NN model with 38 neurons

FIG. 1. *Results of using two-layer ReLU networks for approximating function (7.1)*



(a) 10 neurons (with three marked elements)

(b) 13 neurons (with seven marked elements

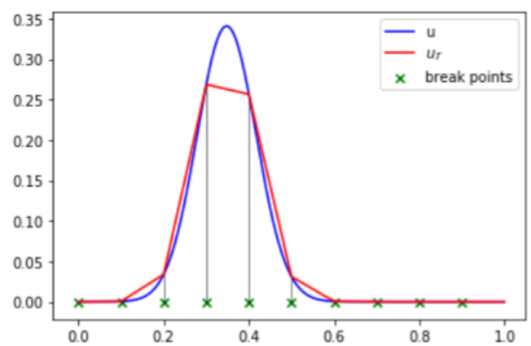
(c) 20 neurons

FIG. 2. *Error distribution on physical partitions generated in the ANE process for the first test problem, where red partitions are the elements to be refined.*

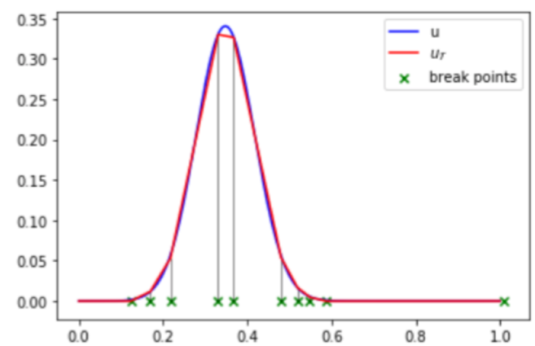
Adaptive 2-Layer NN for One-dimensional Poisson Problem

Poisson equation: comparing adaptive network with fixed networks using Energy functional

NN (hidden layer neurons)	#Parameters	$\frac{\|u-u_\tau\|_0}{\|u\|_0}$	$\frac{\|u'-u'_\tau\|_0}{\|u'\|_0}$	$\xi_{\text{rel}} = \frac{\|\sigma_\mathcal{T}+u'_\mathcal{T}\|_0}{\|\sigma_\mathcal{T}\|_0}$
Fixed 2-layer (25)	51	0.012943	0.149020	0.164645
Fixed 2-layer (50)	101	**0.006108**	0.089470	0.095394
Adaptive 2-layer (25)	51	0.007794	**0.075847**	0.076366
Fixed 4-layer (24-14-14) [2]	623	0.029161	0.160666	-

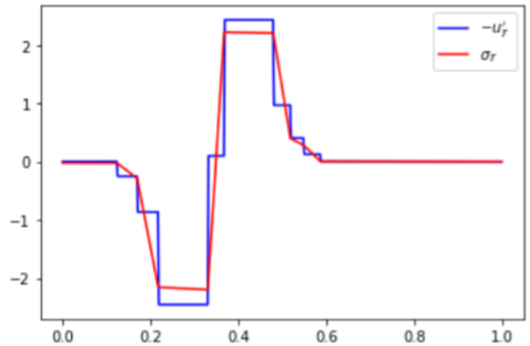


(a) Initial model $u_\mathcal{T}$ with 10 neurons $\frac{\|u'-u'_\mathcal{T}\|_0}{\|u'\|_0}$=0.522380

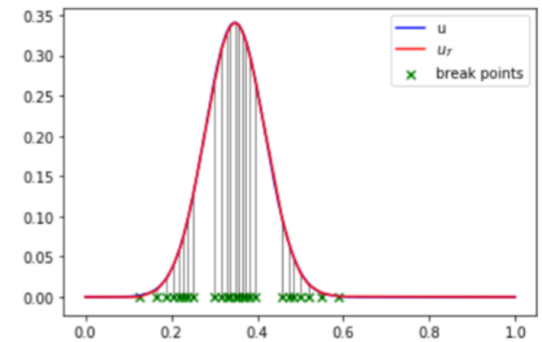


(b) Optimized model $u_\mathcal{T}$ with 10 neurons, $\frac{\|u'-u'_\mathcal{T}\|_0}{\|u'\|_0}$=0.229533

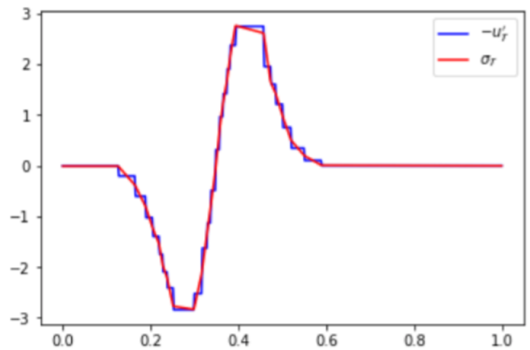
Adaptive 2-Layer NN for One-dimensional Poisson Problem



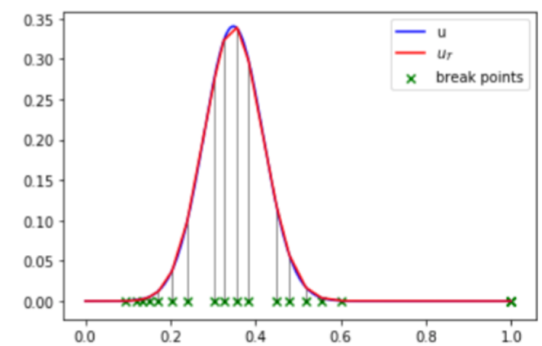
(c) Recovered flux $\sigma_\mathcal{T}$ and the calculated $-u'_\mathcal{T}$ of 10 neurons, $\dfrac{\|\sigma_\mathcal{T} + u'_\mathcal{T}\|_0}{\|\sigma_\mathcal{T}\|_0}$=0.278647



(d) Adaptive model $u_\mathcal{T}$ with 25 neurons, $\dfrac{\|u' - u'_\mathcal{T}\|_0}{\|u'\|_0}$=0.075847

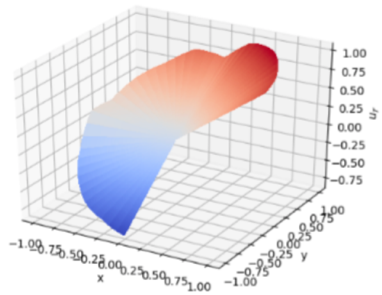


(e) Recovered flux $\sigma_\mathcal{T}$ and the calculated $-u'_\mathcal{T}$ of 25 neurons, $\dfrac{\|\sigma_\mathcal{T} + u'_\mathcal{T}\|_0}{\|\sigma_\mathcal{T}\|_0}$=0.076366

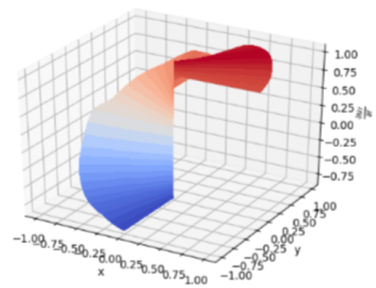


(f) A fixed model $u_\mathcal{T}$ with 25 neurons, $\dfrac{\|u' - u'_\mathcal{T}\|_0}{\|u'\|_0}$=0.151279

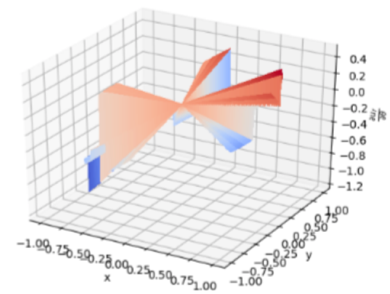
Adaptive 2-Layer NN for Poisson equation in L-shape domain



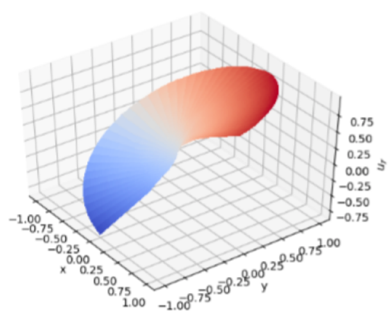
(d) Initial $u_{\mathcal{T}}$ (20 neurons)



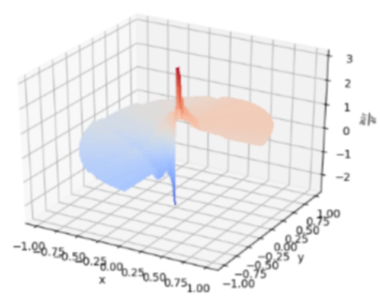
(e) Initial $\partial_r u_{\mathcal{T}}$ (20 neurons)



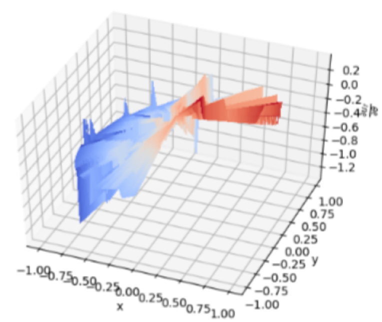
(f) Initial $\partial_\theta u_{\mathcal{T}}$ (20 neurons)



(g) Optimal $u_{\mathcal{T}}$ (20 neurons)

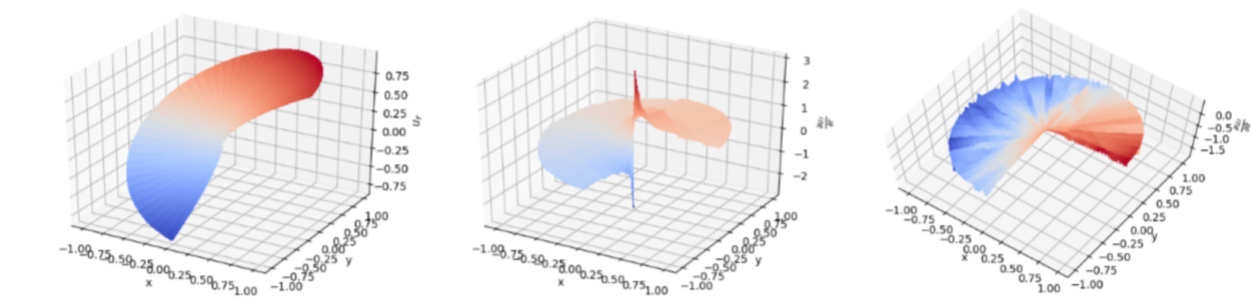


(h) $\partial_r u_{\mathcal{T}}$ (20 neurons)

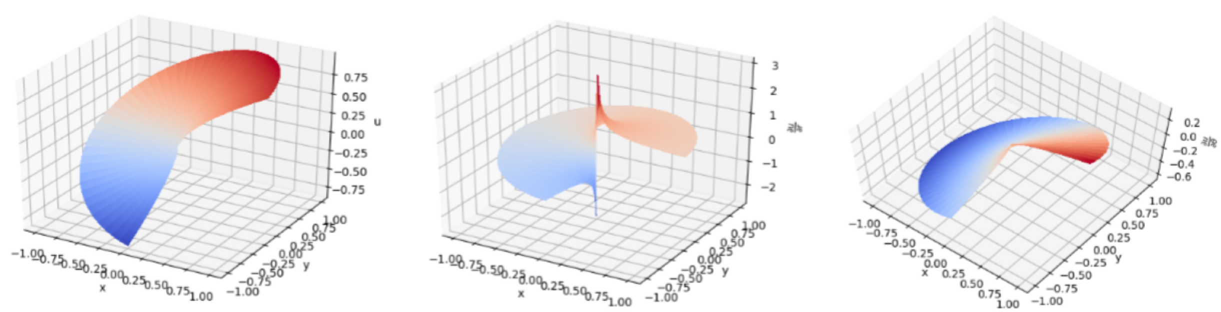


(i) $\partial_\theta u_{\mathcal{T}}$ (20 neurons)

Adaptive 2-Layer NN for Poisson equation in L-shape domain

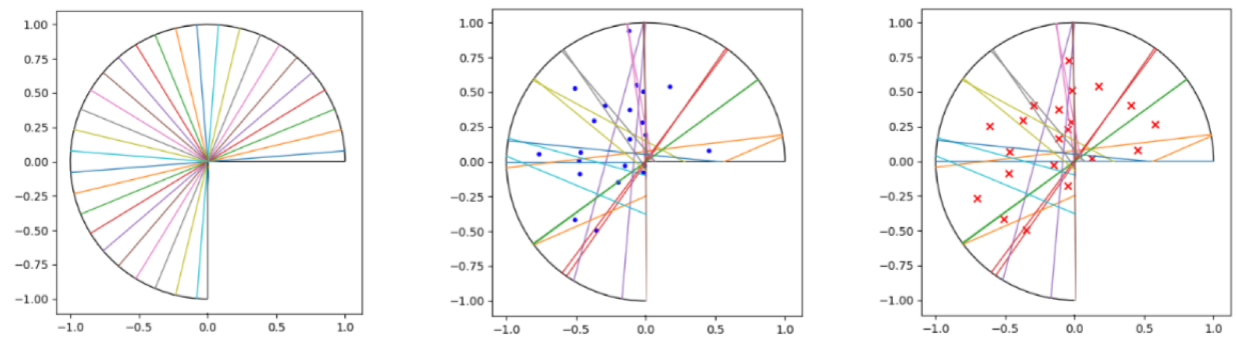


(j) Adaptive NN of $u_{\mathcal{T}}$ (86 neurons) (k) $\partial_r u_{\mathcal{T}}$ (86 neurons) (l) $\partial_\theta u_{\mathcal{T}}$ (86 neurons)



(a) The exact solution u (b) The exact $\partial_r u$ (c) The exact $\partial_\theta u$

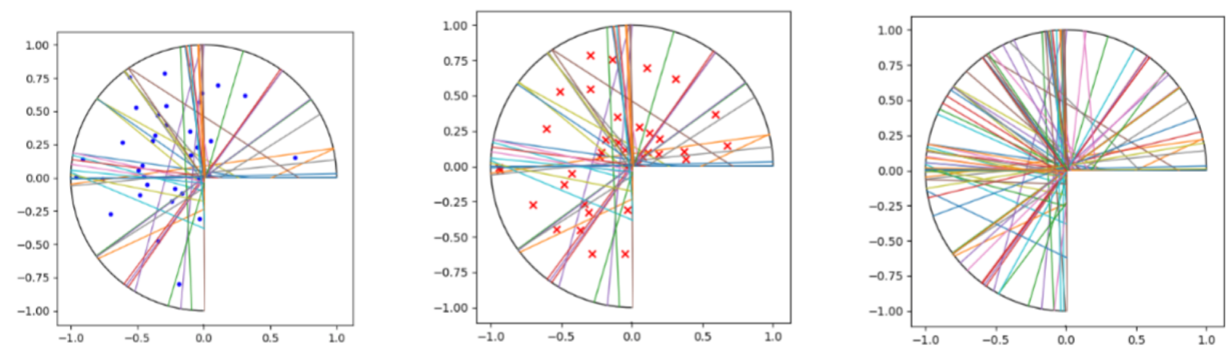
Adaptive 2-Layer NN for Poisson equation in L-shape domain



(a) Initial break lines of 20 neurons

(b) Optimal break lines of 20 neurons with marked elements using (5.2)

(c) Elements marked with the exact local error



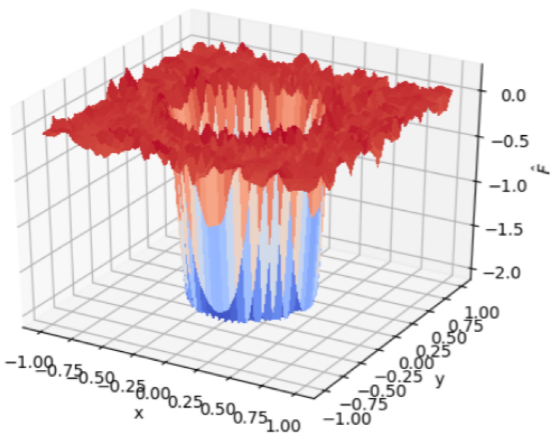
(d) Optimal break lines of 42 neurons with marked elements using (5.2)

(e) Elements marked with the exact local error

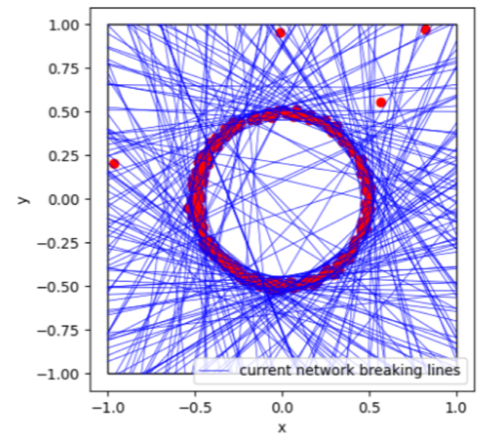
(f) Final break lines of 86 neurons

Adaptive 2-Layer NN

$$f(x,y) = \tanh\left(\frac{1}{\alpha}(x^2+y^2-\frac{1}{4})\right) - \tanh\left(\frac{3}{4\alpha}\right)$$



(a) Approximation using fixed 2-174-1 NN



(b) PP of the approximation by 2-174-1 NN and centers of elements with large errors (red)

Adaptive Multi-Layer Neural Network

- **Improvement Rate**

$$\eta_r = \left(\frac{\xi^{\text{old}} - \xi^{\text{new}}}{\xi^{\text{old}}} \right) \Big/ \left(\frac{(N^{\text{new}})^r - (N^{\text{old}})^r}{(N^{\text{new}})^r} \right)$$

- **Adding a New Layer**

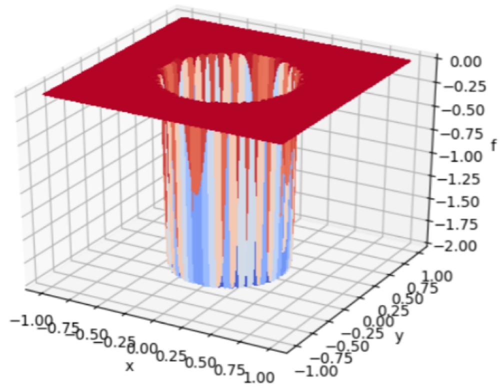
$$\eta_r \leq \delta,$$

where $\delta \in (0, 2)$, is a prescribed expectation rate.

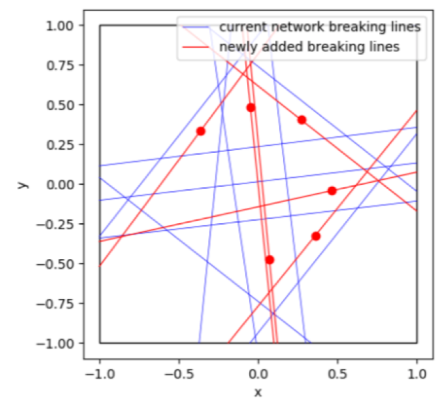
Liu-C.-Chen, JCP, 455 (2022), 111021.

Adaptive 3-Layer NN

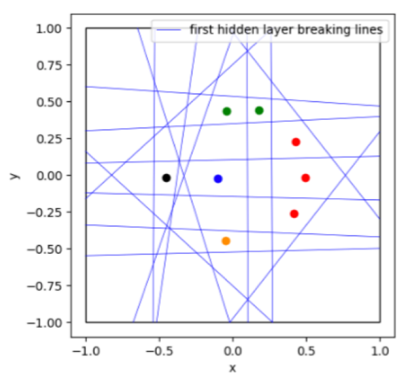
$$f(x,y) = \tanh\left(\frac{1}{\alpha}(x^2+y^2-\frac{1}{4})\right) - \tanh\left(\frac{3}{4\alpha}\right)$$

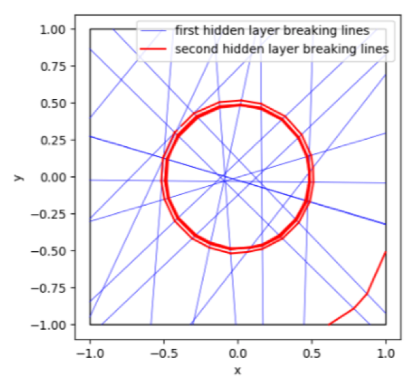


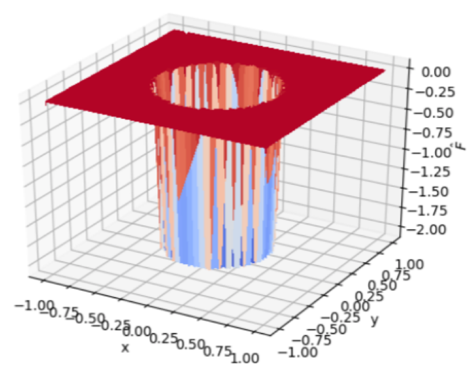
(a) The target function f with a circular transitional layer



(b) PP of the approximation using 2-12-1 NN and centers of the marked elements (red dots)

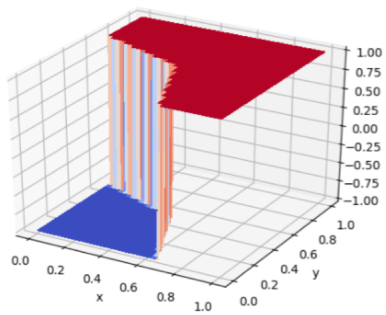




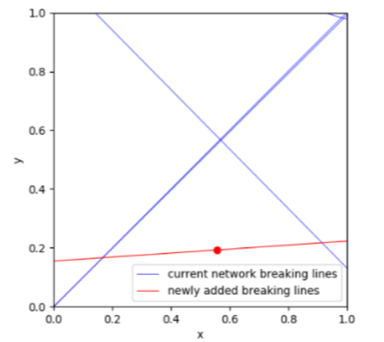


(2-18-5-1)

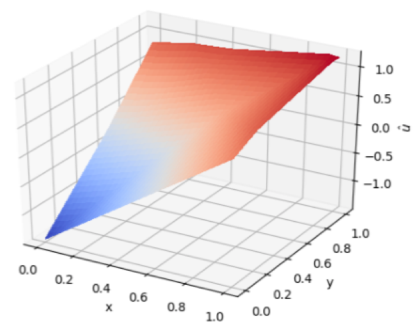
Adaptive 3-Layer NN for Linear Advection-Reaction Problem



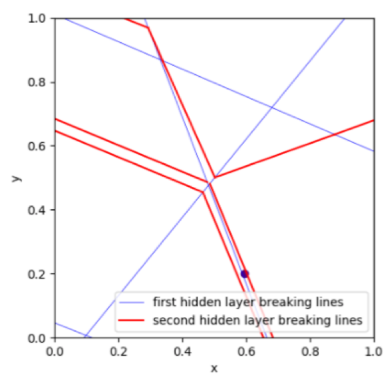
(a) Exact solution u



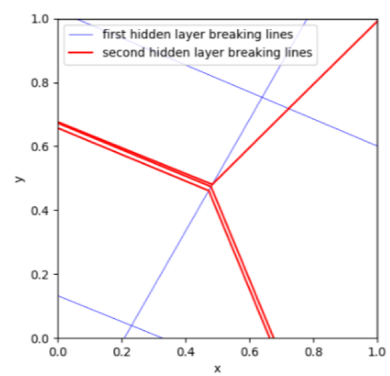
(b) PP by 2-6-1 NN, the marked element (red dot), and new breaking line (red line)



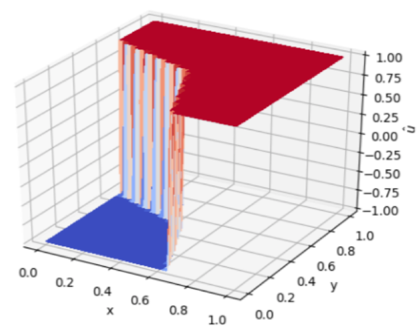
(c) Approximation by 2-7-1 NN



(d) PP by 2-7-3-1 NN and the marked element



(e) PP by adaptive 2-7-4-1 NN



(f) Approximation using adaptive 2-7-4-1 NN

Summary

- **NNs provide a new class of approximating functions**

  Free mesh vs fixed mesh and adaptive mesh

- **Non-convex optimization**

  Bottleneck, the method of continuation, ...

- **Scalar hyperbolic conservation laws**

  Neural Net is the best class of approximating functions for scalar HCLs.

- **Adaptive Neural Network**
  - Automatically design a relatively small NN within the prescribed tolerance
  - A natural continuation process for obtaining a good initial

THANK YOU