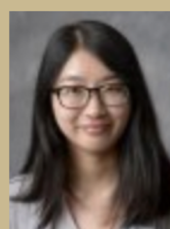
NEURAL NETs AND NUMERICAL PDEs

Zhiqiang Cai[1]



Jingshuang Chen[1]   Min Liu[2]

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

Outline

- Neural Network

  Why using NN?

- How to Design Competitive NN Method for Solving PDEs?

  Least-Squares Neural Network (LSNN) Method

- How to Design Neural Architecture?

  Adaptive Network Enhancement (ANE) Method

- **How to Train NN?**

  the Method of Gradient Descent and the Method of Continuation

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

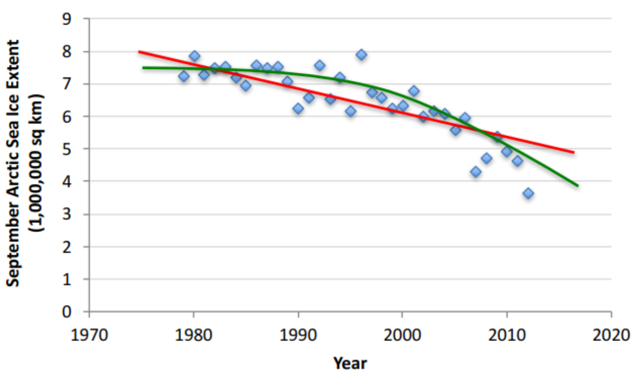
Is ML just a glorified "curve fitting"?

"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, ...

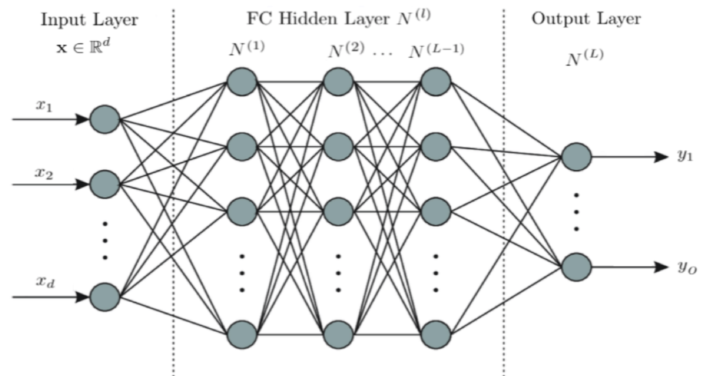
- **Objective** :

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

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

Neural Network (NN)

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



- **DNN function (models)**

$$\mathcal{N}(\mathbf{x}) = \boldsymbol{\omega}^{(L)} \left(N^{(L-1)} \circ \cdots \circ N^{(2)} \circ N^{(1)}(\mathbf{x}) \right) - b^{(L)}$$

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

- **The number of parameters**

$$N = \sum_{l=1}^{L} n_l \times (n_{l-1} + 1)$$

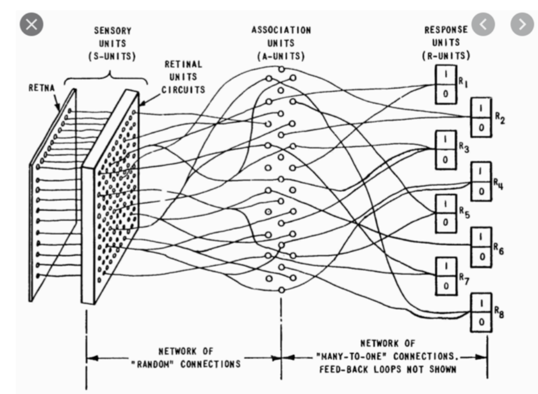


Figure I ORGANIZATION OF THE MARK I PERCEPTRON

- **Activate functions**
  - ReLU^k
  - Sigmoids
  - Etc.

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

$$\sigma(t) = \frac{1}{1 + e^x}$$

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, p. polynomials, finite elements, …?
  - Why using more than two layers?
  - How to design NN architecture?
  - ……

Part I. How to Design Competitive NN Method for Solving PDEs?

- **Conventional Wisdoms (current state)**

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

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

- **How to design competitive NN-based numerical methods???**

  Neural Net (a class of new approximating functions)

  high cost and uncertainty: non-convex optimization

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

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

  Equivalent Formulations of PDEs

PDE and Equivalent Optimization Formulations

- **Partial Differential Equation**
- **Variational Formulation**
- **Equivalent Optimization Formulations**
    - Energy functionals (a small class of problems)
      DeepRitz (E-Yu), Finite Neuron (Xu), DuNN (C.-Liu), ...
    - Various least-squares functionals
      DGM (Sirignano-Spiliopoulos), PINN (Karniadakis et. al.), LSNN (C.-Chen-Liu), ...

Least-squares Methods for Elliptic Partial Differential Equations

- **Elliptic Partial Differential Equations**

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

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

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

where the primitive least-square functional is given by

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

Least-Squares Methods Based on First-Order System

- **First-order system**

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

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, …)

Div 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

Scalar Hyperbolic Conservation Laws

- **Scalar Nonlinear Hyperbolic Conservation Laws**

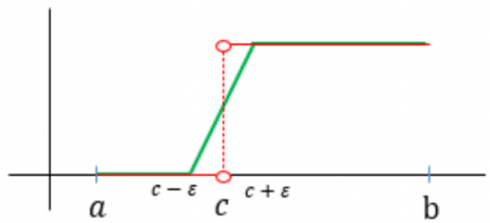
$$
\begin{cases}
u_t(\mathbf{x},t) + \nabla_{\mathbf{x}} \cdot \mathbf{f}(u) = 0, & \text{in } \Omega \times I, \\
u = g, & \text{on } \Gamma_-, \\
u(\mathbf{x},0) = u_0(\mathbf{x}), & \text{in } \Omega,
\end{cases}
$$

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

Approximation to Unit Step Function with Unknown Interface

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

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



- **Best continuous piece-wise linear 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 best CPL 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

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

- Approximations on fixed quasi-uniform mesh

  - very fine mesh-size

  - overshooting and oscillation

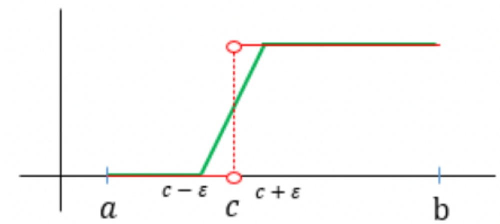
- Exploring new class of approximating functions

??? neural network ???

Approximation to Unit Step Function with Unknown Interface

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

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



- **Best neural network approximation**

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

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

- **Error estimate**

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

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

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

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**

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

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

LSNN Method

Numerical Issues

- Integration

  random sampling vs numerical integration

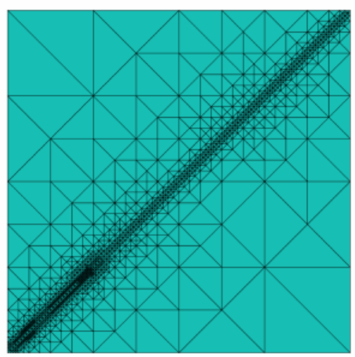
- Differentiation

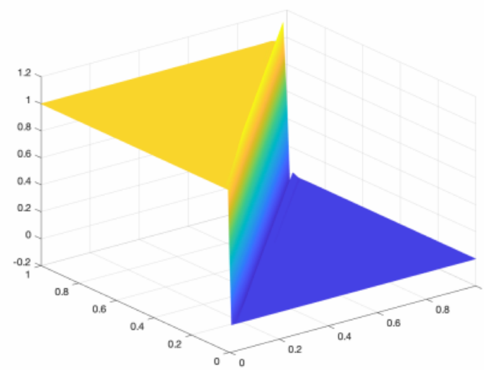
  automatic, exact, numerical, etc

- Algebraic solver (training NN)

  methods of gradient descent, ..., ???

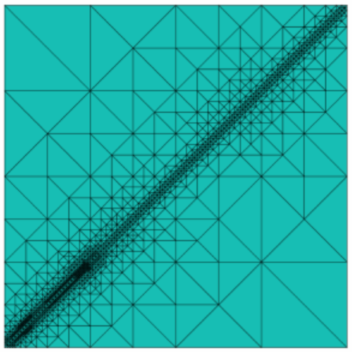
Linear scalar HCL: $f(u)=au$, i.e., $u_t+au_x=0$

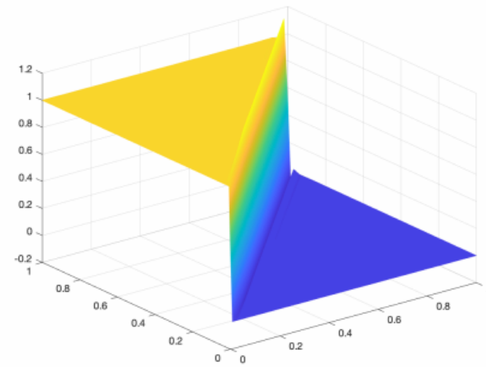


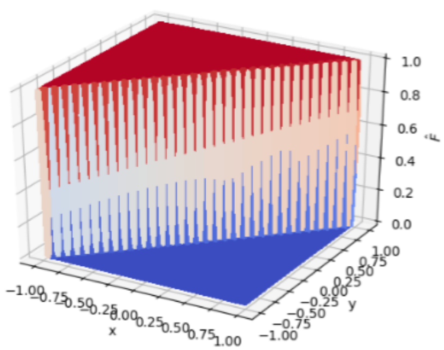


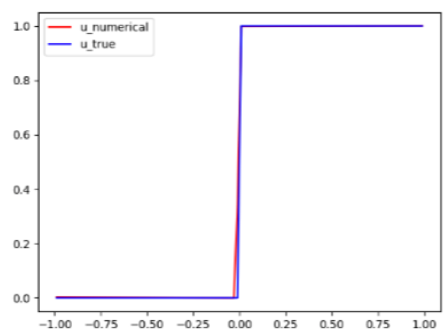
Liu-Zhang, CMAME, 2020

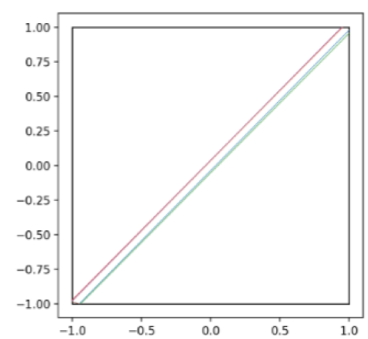
Famous Transport Equation $u_t+u_x=0$







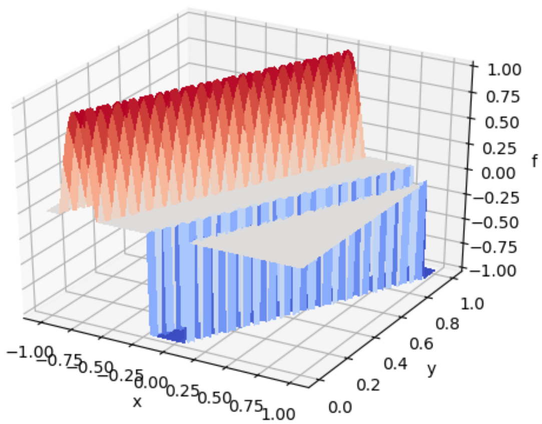




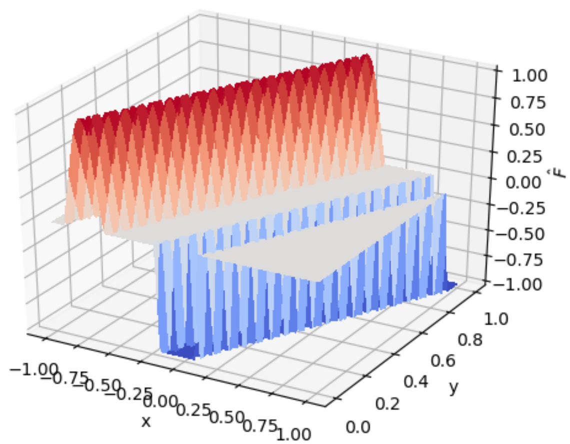
(2-6-1)

C.-Chen-Liu, JCP, 2021

Two discontinuous interfaces

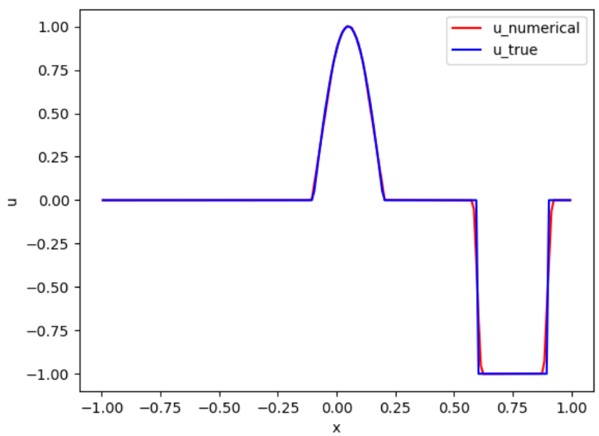


(a) Exact solution u

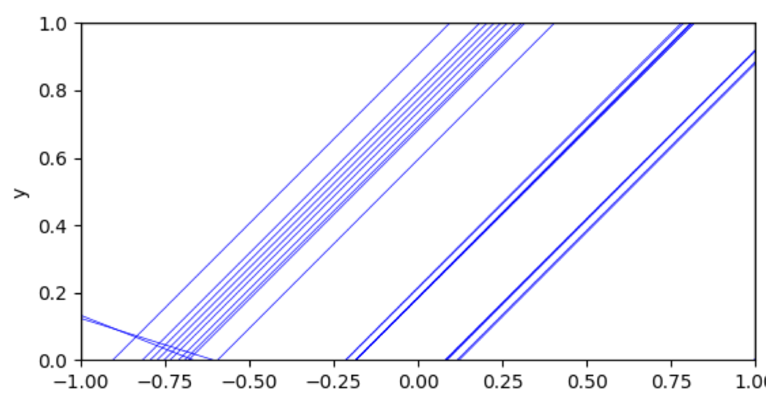


(b) Network approximation $\bar{u}_{\mathcal{T}}^{N}$

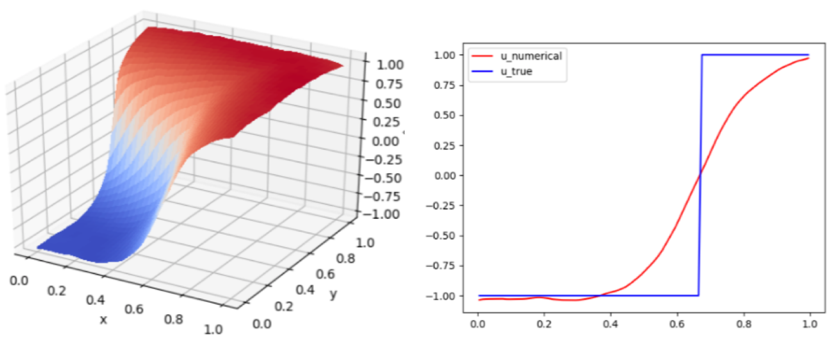
(2-31-1)



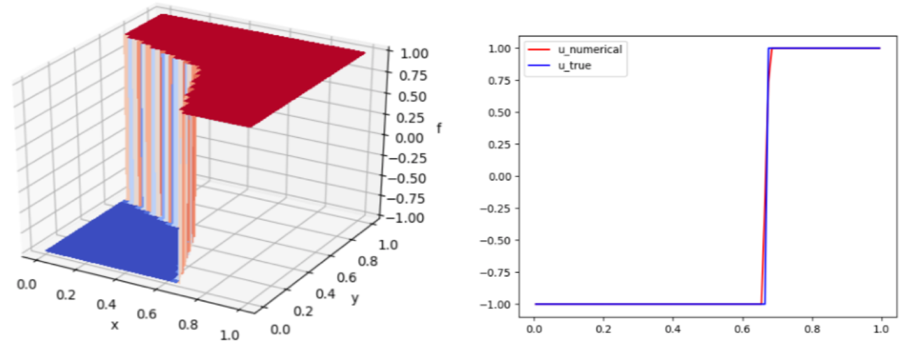
(c) Traces of the exact solution and approximation $\bar{u}_{\mathcal{T}}^{N}$ on the plane $y = 0.8$



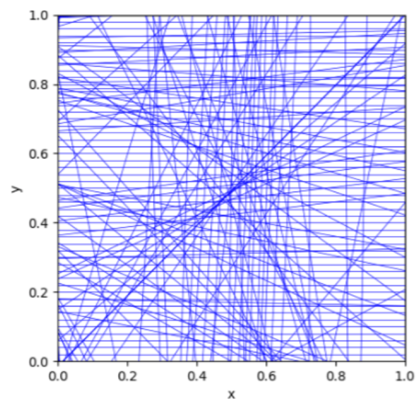
(d) Network breaking lines



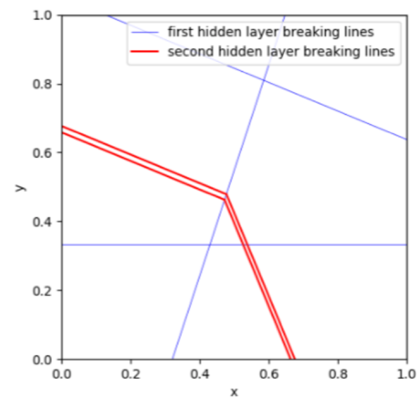
(2-200-1)



(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.

LSNN method for scalar nonlinear HCLs

- **Scalar nonlinear hyperbolic conservation laws**

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

- **Least-squares formulation**

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

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

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

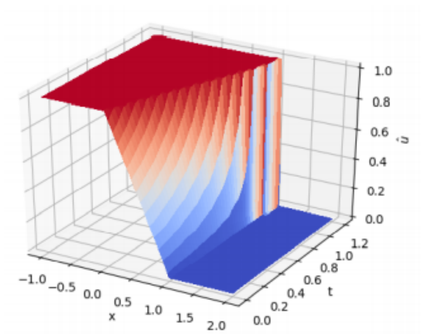
$$0 = u_t + \nabla_{\mathbf{x}} \cdot \mathbf{f}(u) = \mathbf{div}\,\big(u, \mathbf{f}(u)\big) = \mathbf{div}\,\mathbf{F}(u)$$

- **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]*)

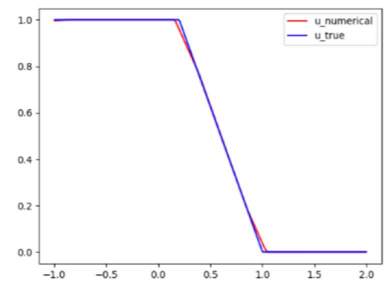
Let $\mathcal{T}$ be a partition of the domain $\Omega \subset \mathbb{R}^{d+1}$.

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

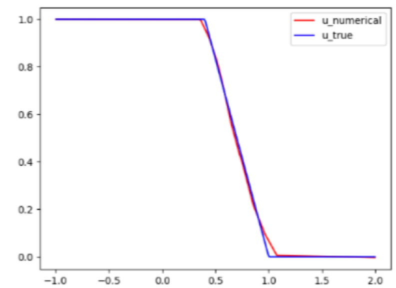
$$\mathbf{div}_{\mathcal{T}}\mathbf{F}\big(u(\mathbf{z}_K)\big) \approx \mathrm{avg}_K \mathbf{div}\,\mathbf{f}(u) = \frac{1}{|K|}\int_{\partial K} \mathbf{F}(u) \cdot \mathbf{n}\, dS$$



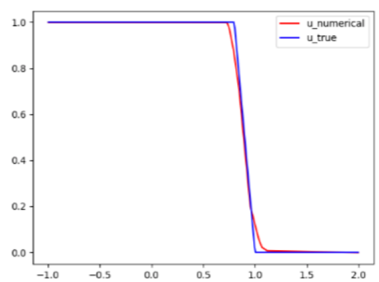
(a) Exact solution u on $\Omega \times I$



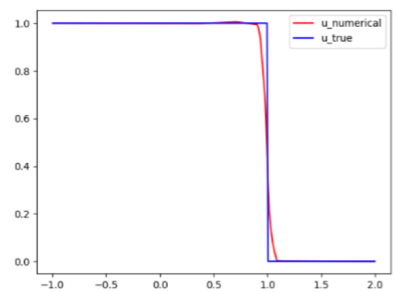
(a) Traces of exact solution and approximation $u_{1,\mathcal{T}}$ on the plane $t = 0.2$



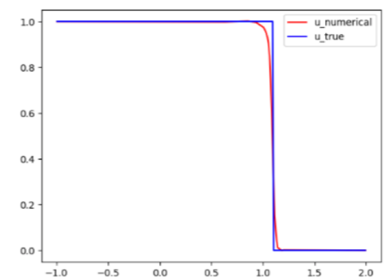
(c) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ on the plane $t = 0.4$



(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 $t = 1.0$

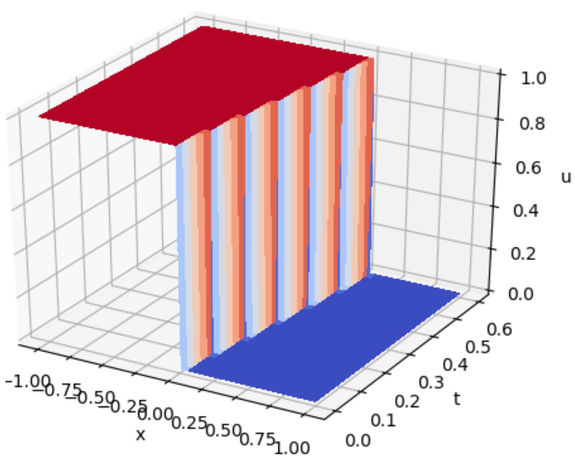


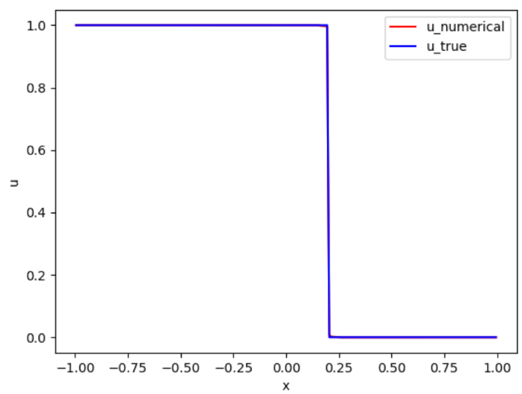
(f) Traces of exact solution and approximation $u_{6,\mathcal{T}}$ on the plane $t = 1.2$

C.-Chen-Liu, Applied Numer. Math., 174 (2022), 163-176

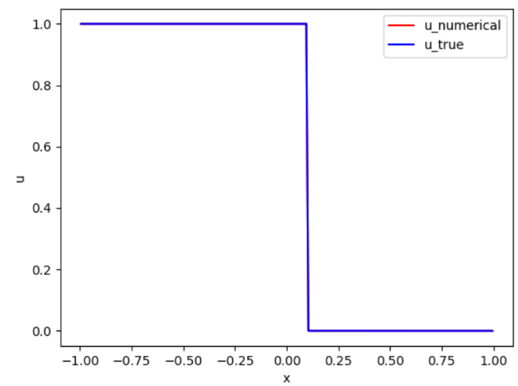
Inviscid Burger Equation $f(u) = \frac{1}{2}u^2$

Riemann Problem Shock formation: exact solution

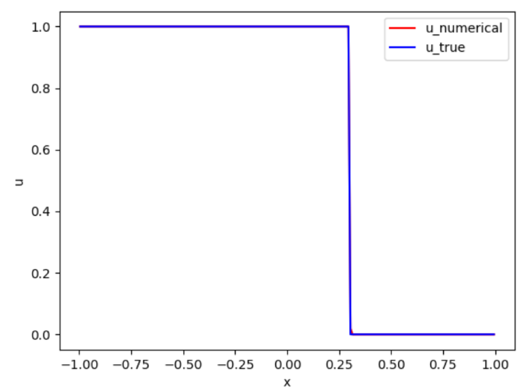




t=0.2



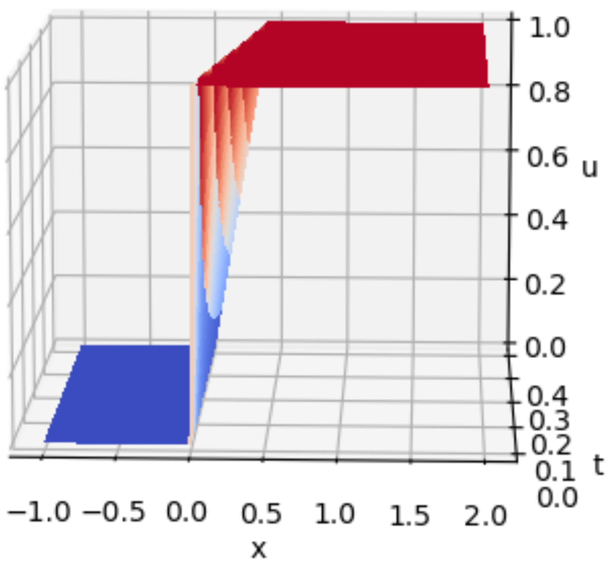
t=0.4

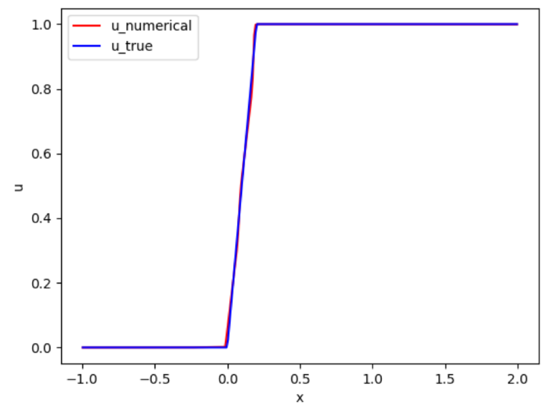


t=0.6 (2-10-10-1)

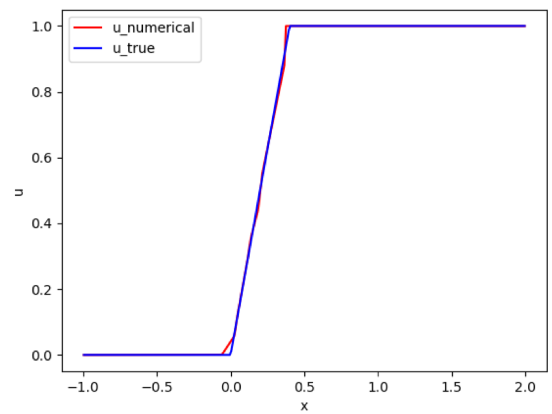
C.-Chen-Liu, arXiv: 2110.10895 [math.NA]

Riemann Problem
Rarefaction wave: exact solution





t=0.2

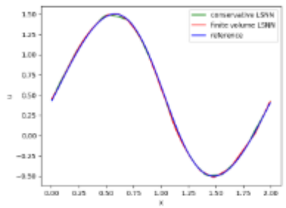


t=0.4

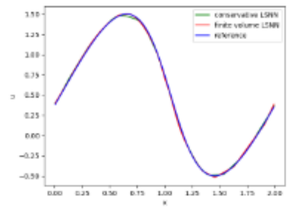
(2-10-10-1)

Inviscid Burgers equation with smooth initial

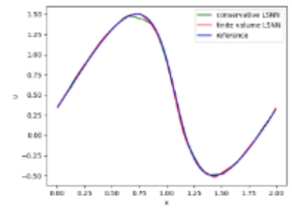
$$u_0(x) = 0.5 + \sin(\pi x).$$



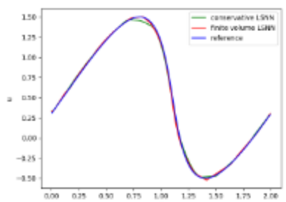
(a) Traces of reference and numerical solutions $u_{1,\mathcal{T}}$ on the plane $t = 0.05$



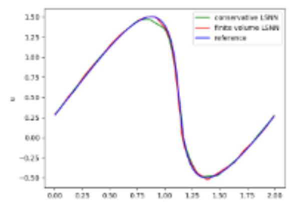
(b) Traces of reference and numerical solutions $u_{2,\mathcal{T}}$ on the plane $t = 0.1$



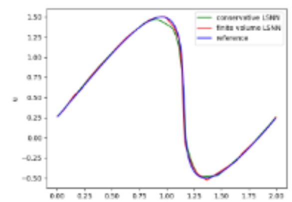
(c) Traces of reference and numerical solutions $u_{3,\mathcal{T}}$ on the plane $t = 0.15$



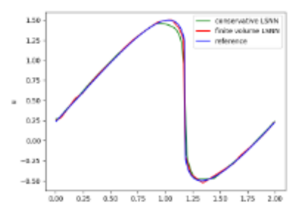
(d) Traces of reference and numerical solutions $u_{4,\mathcal{T}}$ on the plane $t = 0.2$



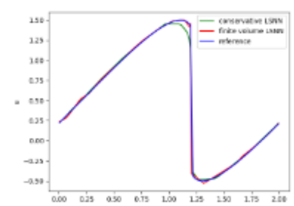
(e) Traces of reference and numerical solutions $u_{5,\mathcal{T}}$ on the plane $t = 0.25$



(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 $t = 0.35$

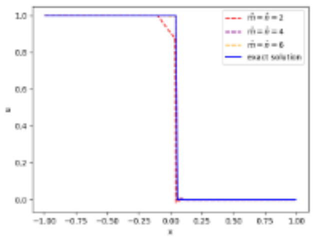


(h) Traces of reference and numerical solutions $u_{8,\mathcal{T}}$ on the plane $t = 0.4$

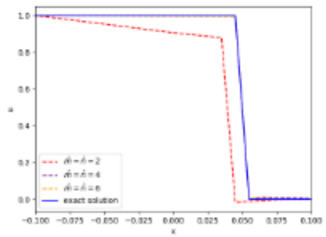
(2-30-30-1)

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

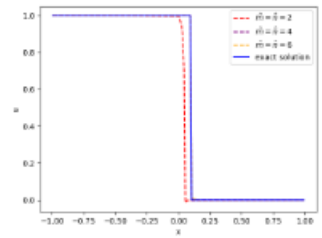
Riemann Problem with Higher order flux $f(u) = \frac{1}{4}u^4$



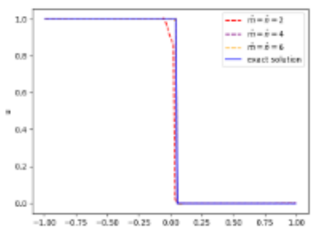
(a) Traces of exact and numerical solutions $u_{1,\mathcal{T}}$ using the trapezoidal rule on the plane $t = 0.2$



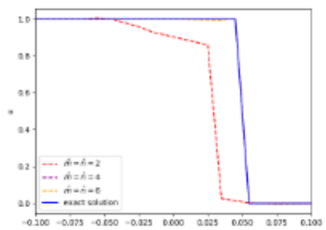
(b) Zoom-in plot near the discontinuous interface of sub-figure (a)



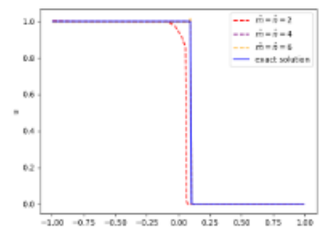
(c) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ using the trapezoidal rule on the plane $t = 0.4$



(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)

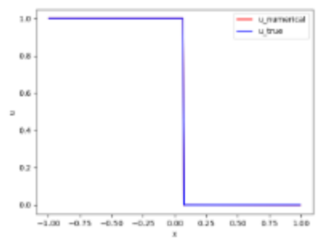


(f) Traces of exact and numerical solutions $u_{2,\mathcal{T}}$ using the mid-point rule on the plane $t = 0.4$

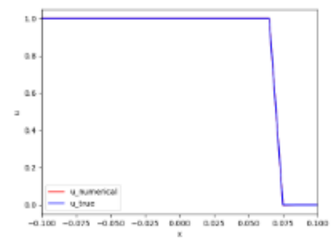
(2-10-10-1)

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

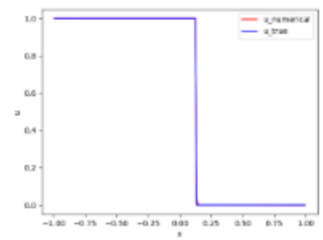
Riemann Problem with Non-convex flux $f(u) = \frac{1}{3}u^3$



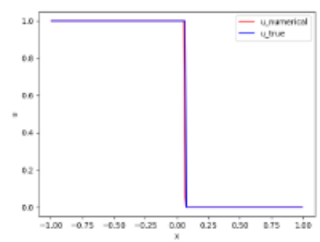
(a) Traces of exact and numerical solutions $u_{1,\mathcal{T}}$ on the plane $t = 0.2$



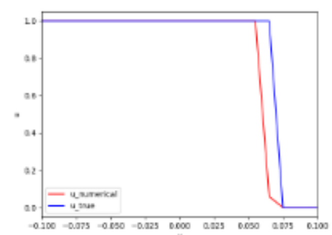
(b) Zoom-in plot near the discontinuous interface of sub-figure (a)



(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 $t = 0.2$

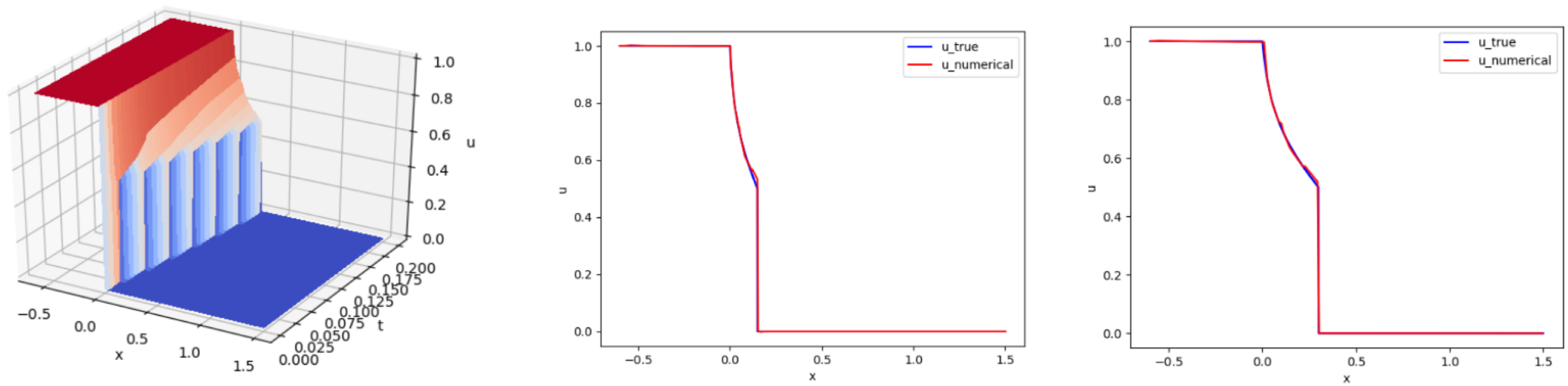


(e) Zoom-in plot near the discontinuous interface of sub-figure (d)

(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*

Part II. How to Design Neural Architecture?

- **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**

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

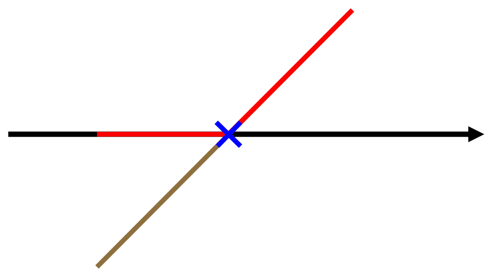
where $\hat{\mathcal{K}}_{k-1}$ is the set of marked physical subdomains

Physical Partition

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

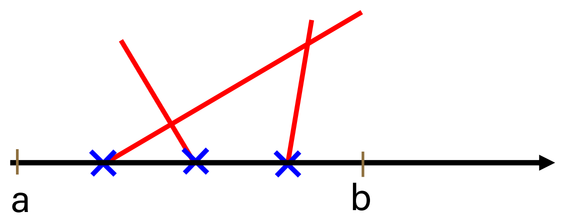
- Free Hyper-planes (Linear part of neurons)

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



- Hyper-break 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

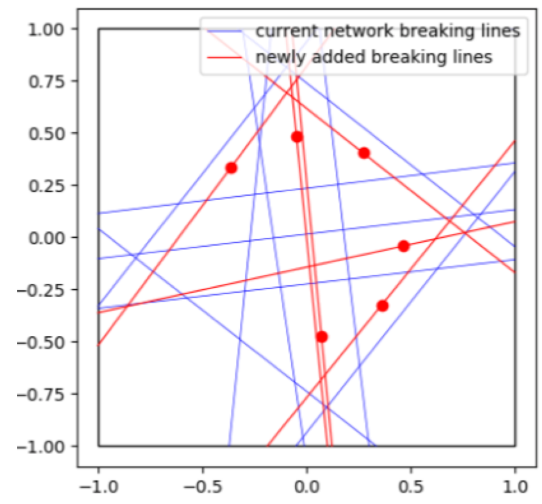
$$n_k = n_{k-1} + \#\hat{\mathcal{K}}_{k-1}$$

- **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

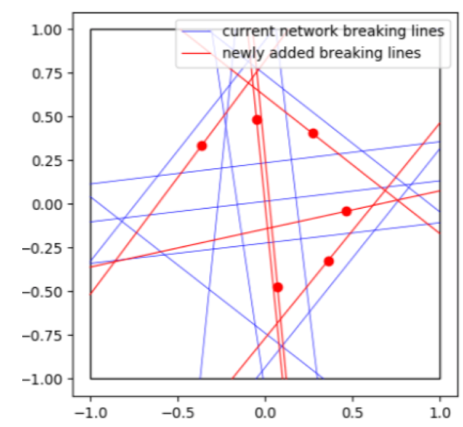
- **Non-convex optimization**

  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

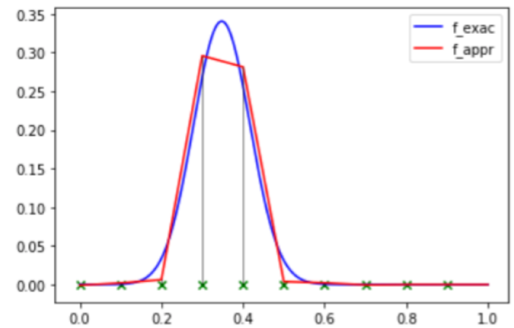


Adaptive 2-Layer NN

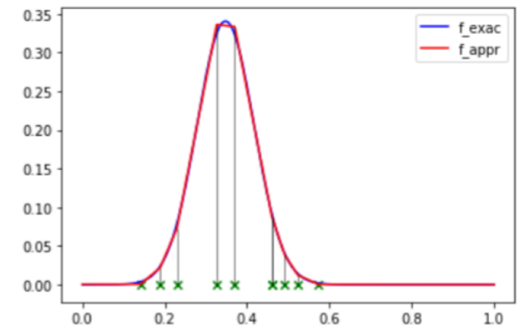
$$f(x) = x\left(e^{-(x-\frac{1}{3})^2/k} - e^{-\frac{4}{9}/k}\right)$$

Comparing adaptive neural network with fixed networks for testing problem (

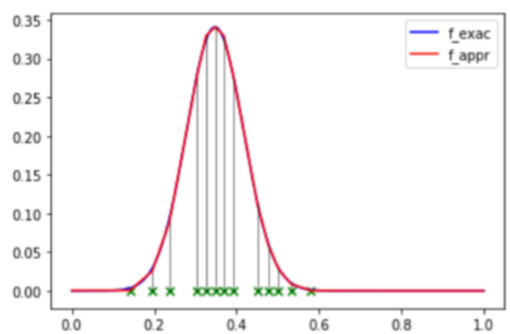
Network (neurons)	# Parameters	$\|f - f_\mathcal{T}\|_\mathcal{T}/\|f\|$
Fixed (20)	41	0.007644
Fixed (38)	77	0.003762
Adaptive (10→13→20)	41	0.003837



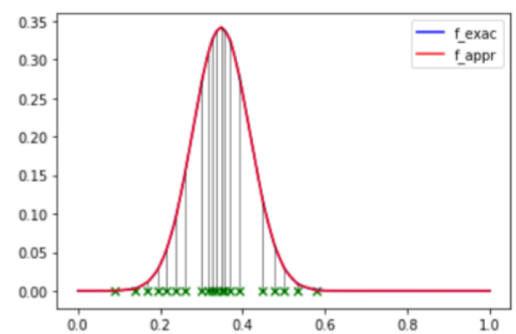
(a) Initial NN model with 10 uniform break points



(b) Optimized NN model with 10 neurons



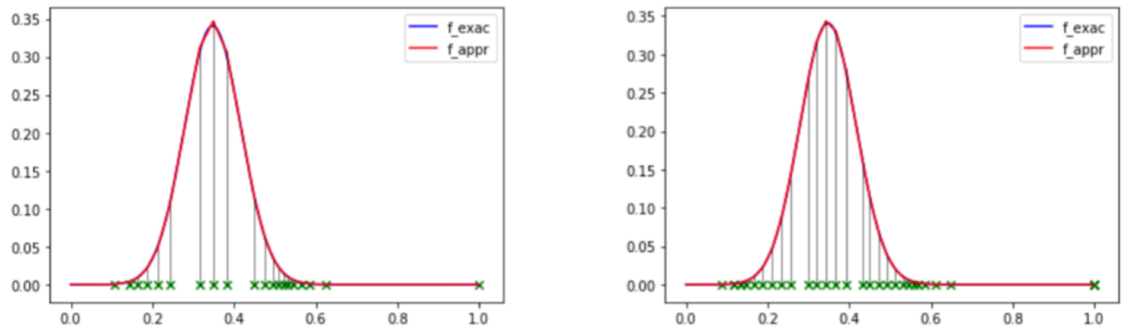
(c) Optimized NN model with 13 neurons using ANE



(d) Optimized NN model with 20 neurons using ANE

Adaptive 2-Layer NN

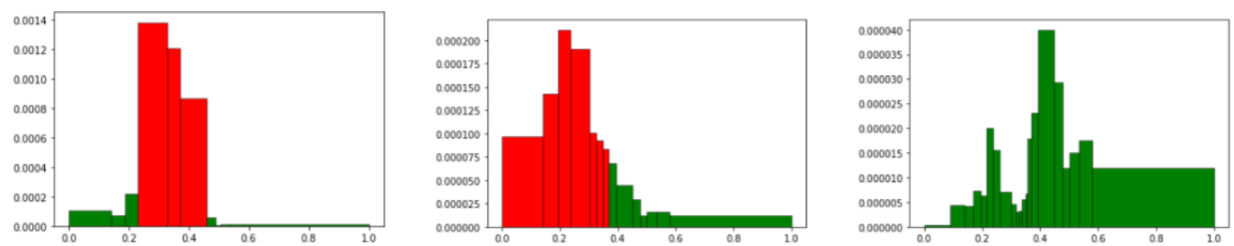
$$f(x) = x\left(e^{-(x-\frac{1}{3})^2/k} - e^{-\frac{4}{9}/k}\right)$$



(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

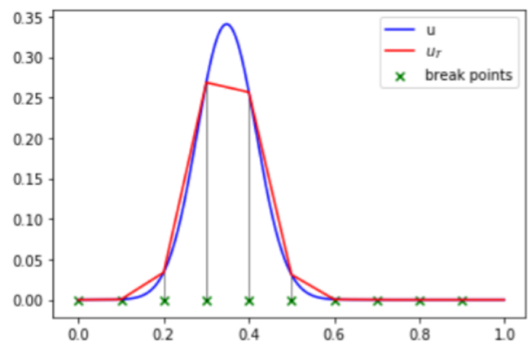
(c) 20 neurons

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.*

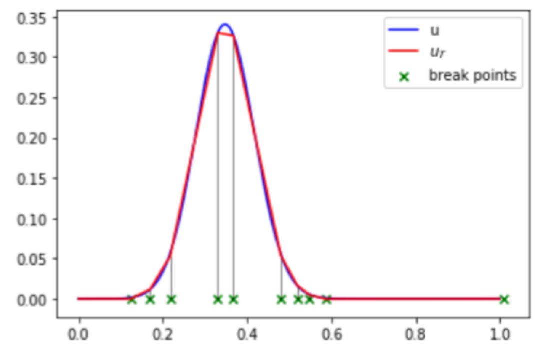
Adaptive 2-Layer NN for One-dimensional Poisson Problem

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	-

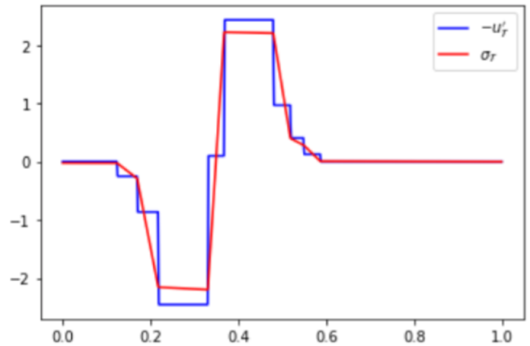


(a) Initial model $u_\mathcal{T}$ with 10 neurons $\frac{\|u'-u'_\mathcal{T}\|_0}{\|u'\|_0}$=0.522380

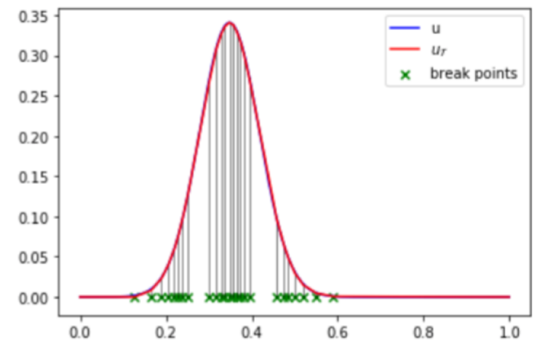


(b) Optimized model $u_\mathcal{T}$ with 10 neurons, $\frac{\|u'-u'_\mathcal{T}\|_0}{\|u'\|_0}$=0.229533

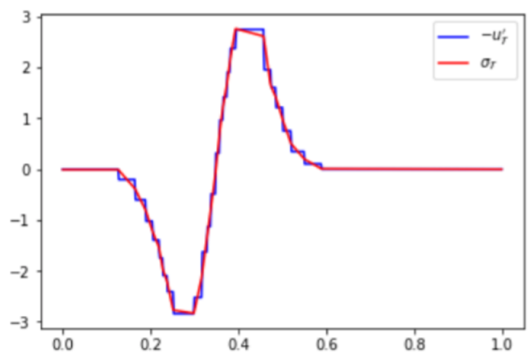
Adaptive 2-Layer NN for One-dimensional Poisson Problem



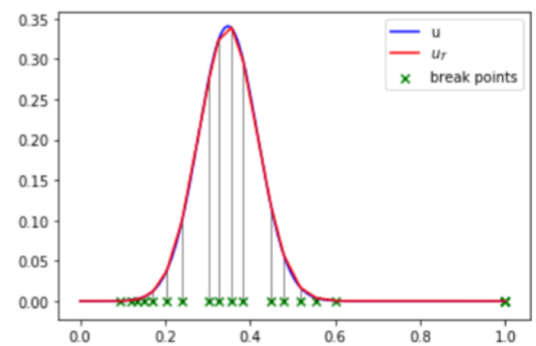
(c) Recovered flux $\sigma_{\mathcal{T}}$ and the calculated $-u'_{\mathcal{T}}$ of 10 neurons, $\dfrac{\|\sigma_{\mathcal{T}} + u'_{\mathcal{T}}\|_0}{\|\sigma_{\mathcal{T}}\|_0}$=0.278647



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

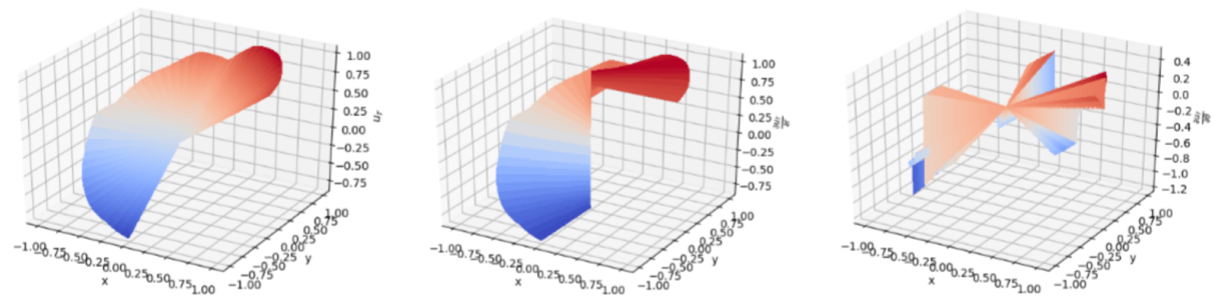


(e) Recovered flux $\sigma_{\mathcal{T}}$ and the calculated $-u'_{\mathcal{T}}$ of 25 neurons, $\dfrac{\|\sigma_{\mathcal{T}} + u'_{\mathcal{T}}\|_0}{\|\sigma_{\mathcal{T}}\|_0}$=0.076366

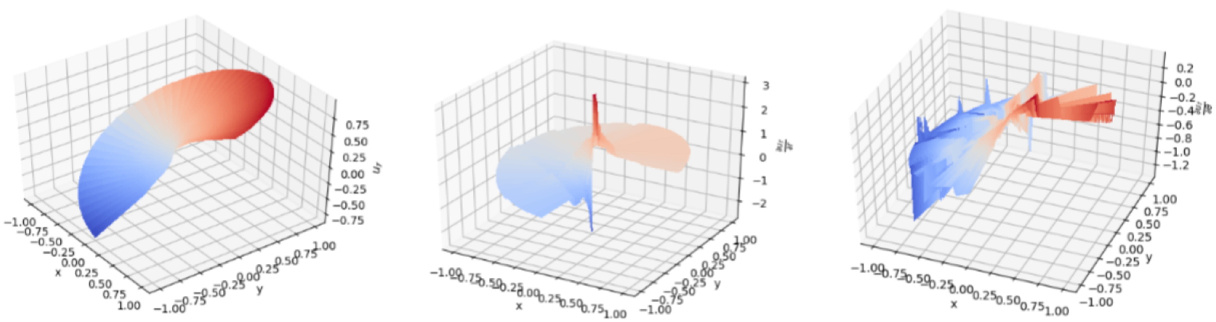


(f) A fixed model $u_{\mathcal{T}}$ with 25 neurons, $\dfrac{\|u' - u'_{\mathcal{T}}\|_0}{\|u'\|_0}$=0.151279

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

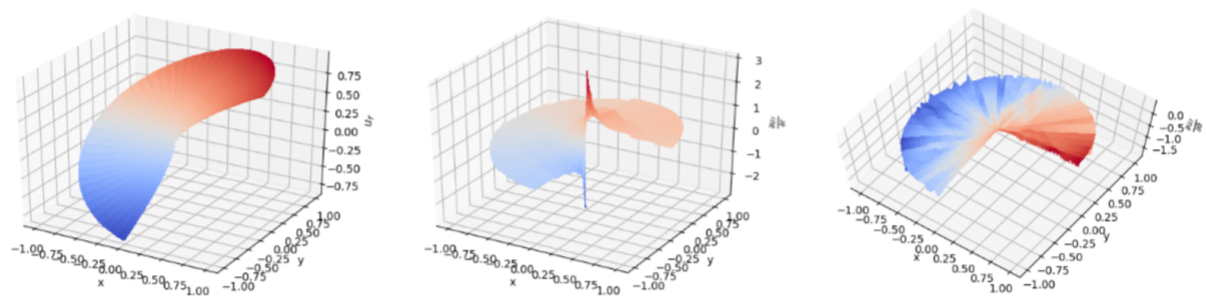


(d) Initial $u_{\mathcal{T}}$ (20 neurons) (e) Initial $\partial_r u_{\mathcal{T}}$ (20 neurons) (f) Initial $\partial_\theta u_{\mathcal{T}}$ (20 neurons)

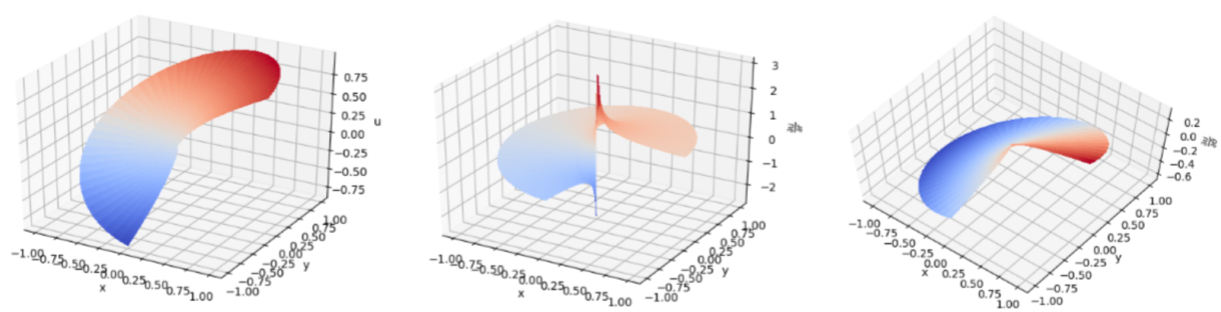


(g) Optimal $u_{\mathcal{T}}$ (20 neurons) (h) $\partial_r u_{\mathcal{T}}$ (20 neurons) (i) $\partial_\theta u_{\mathcal{T}}$ (20 neurons)

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

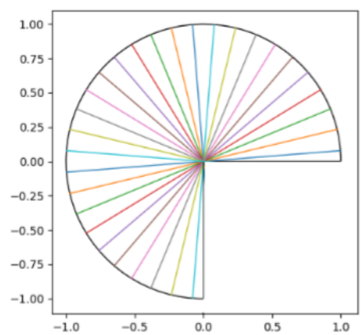


(j) Adaptive NN of $u_{\mathcal{T}}$ (86 neurons) (k) $\partial_r u_{\mathcal{T}}$ (86 neurons) (l) $\partial_\theta u_{\mathcal{T}}$ (86 neurons)

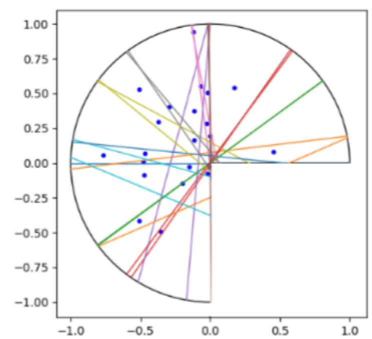


(a) The exact solution u (b) The exact $\partial_r u$ (c) The exact $\partial_\theta u$

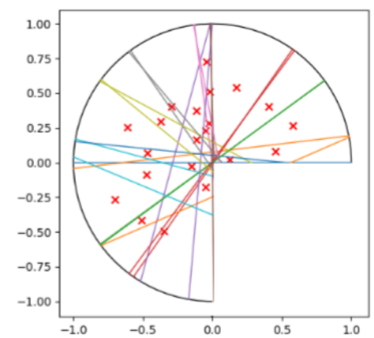
Adaptive 2-Layer NN for Poisson equation in L-shape domain



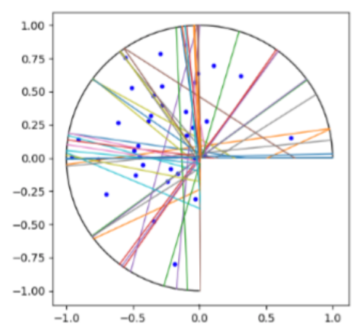
(a) Initial break lines of 20 neurons



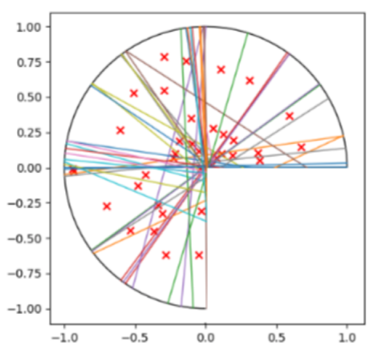
(b) Optimal break lines of 20 neurons with marked elements using (5.2)



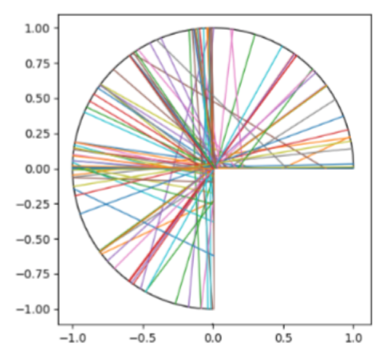
(c) Elements marked with the exact local error



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



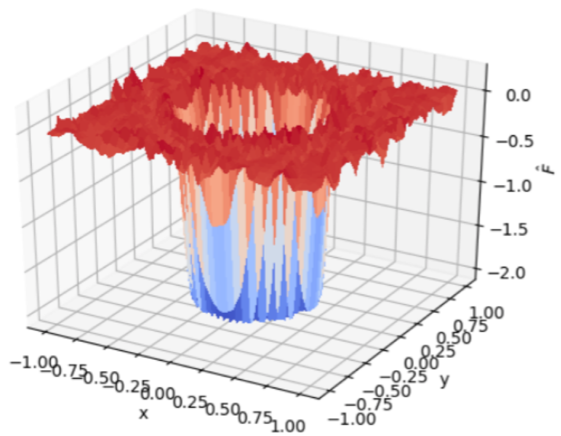
(e) Elements marked with the exact local error



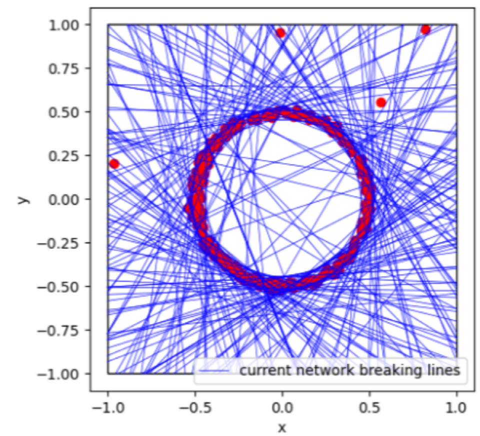
(f) Final break lines of 86 neurons

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**

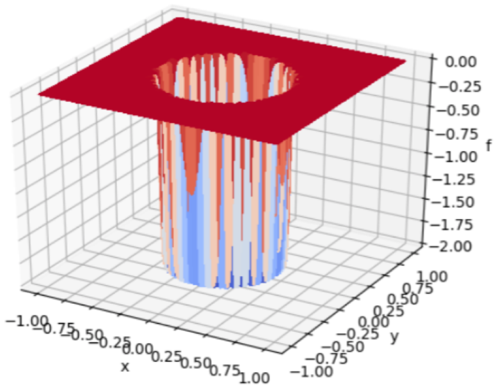
$$\eta_r \leq \delta,$$

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

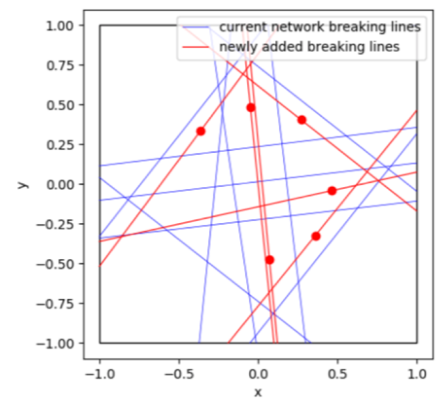
Liu-C.-Chen, JCP, 455 (2022), 111021.

Adaptive 3-Layer NN

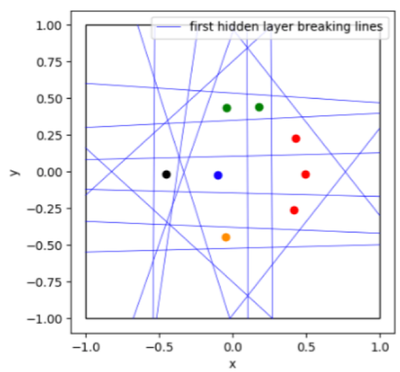
$$f(x,y) = \tanh\left(\frac{1}{\alpha}(x^2+y^2-\frac{1}{4})\right) - \tanh\left(\frac{3}{4\alpha}\right)$$

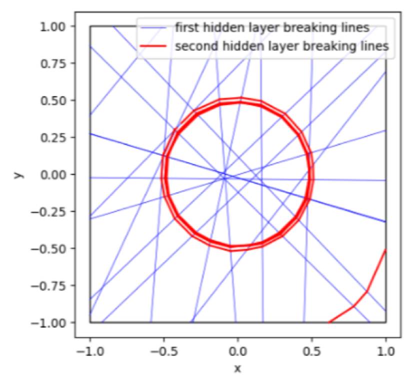


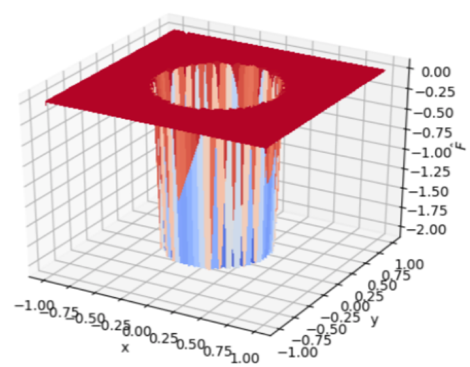
(a) The target function f with a circular transitional layer



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

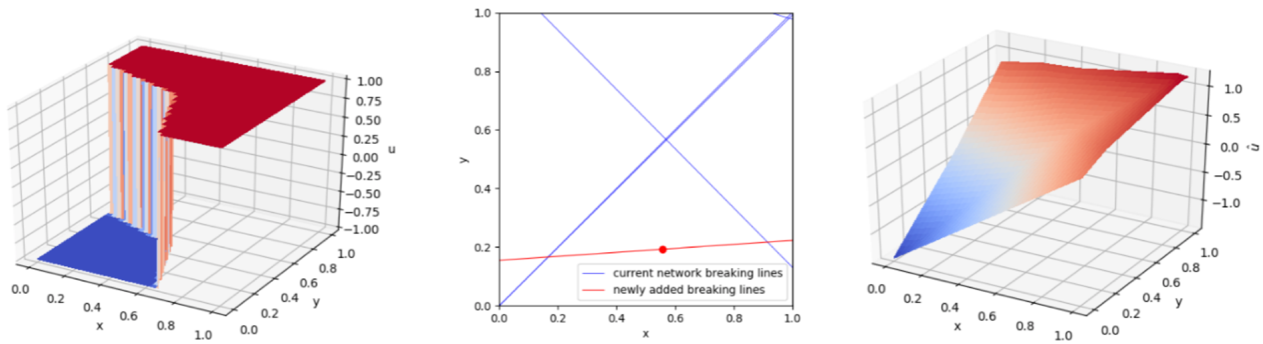






(2-18-5-1)

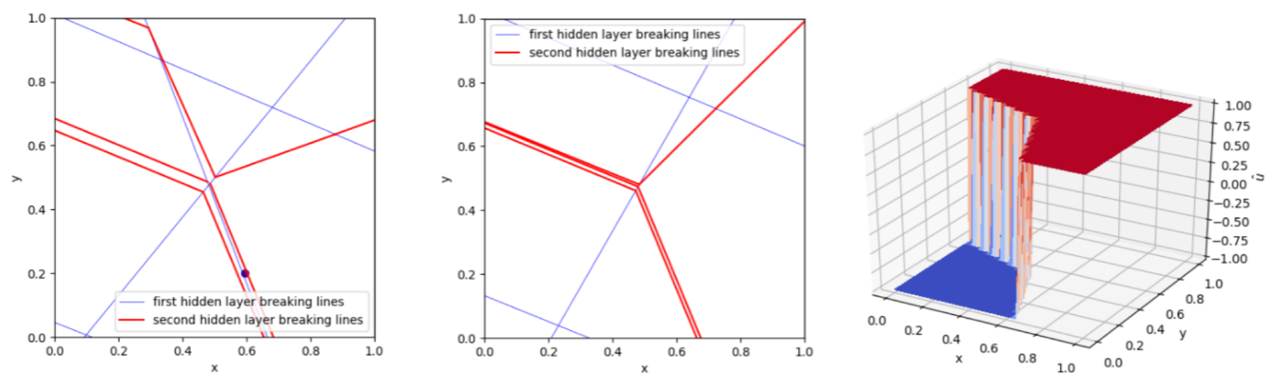
Adaptive 3-Layer NN for Linear Advection-Reaction Problem



(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 functions**

  Free mesh vs uniform mesh and adaptive mesh

- **Non-convex optimization**

  Bottleneck, the method of continuation, ...

- **Scalar hyperbolic conservation laws**

  Neural Net is the best class of 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