STRUCTURE-GUIDED GAUSS-NEWTON METHOD: LSNN FOR LINEAR ADVECTION-REACTION EQUATION

César Herrera

Joint work with Zhiqiang Cai

41st Colorado Conference On Iterative and Multigrid Methods

PURDUE UNIVERSITY | Department of Mathematics

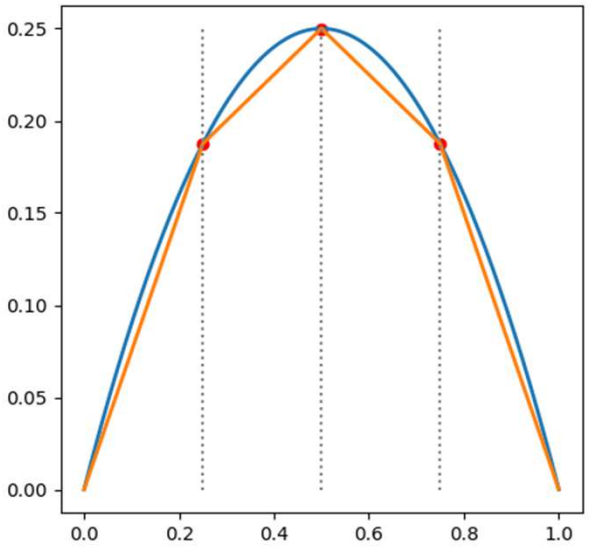
Approximation of Functions

How can we approximate a function?

Approximation of Functions

How can we approximate a function?

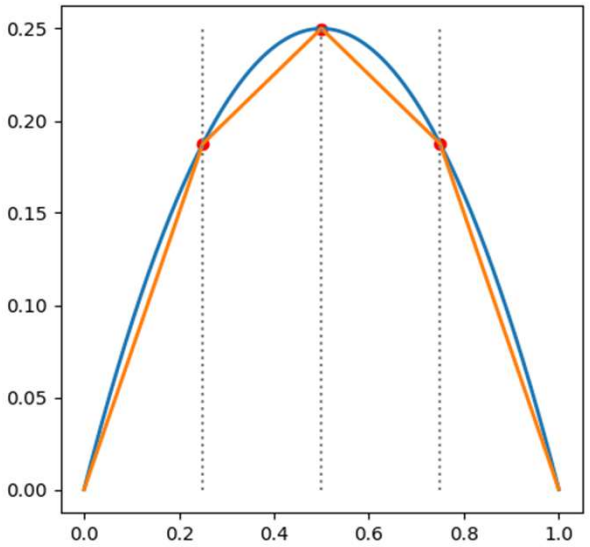
Simple and effective idea: **continuous piecewise linear functions.**



Approximation of Functions

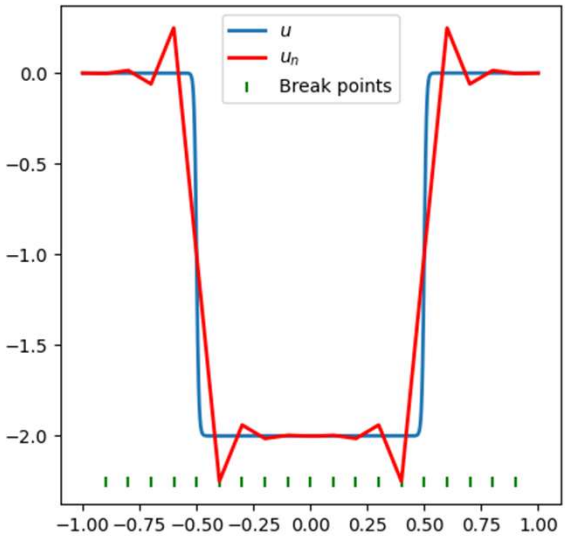
How can we approximate a function?

Simple and effective idea: **continuous piecewise linear functions.**



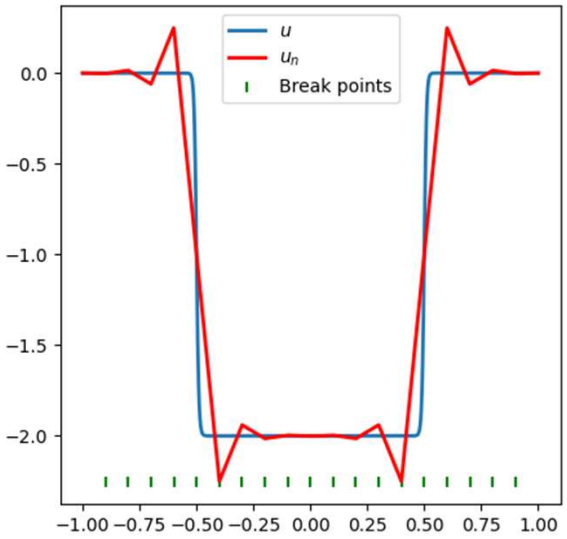
- **Finite Element Methods:** continuous piecewise linear approximation on a **fixed uniform mesh.**

Moving the Mesh

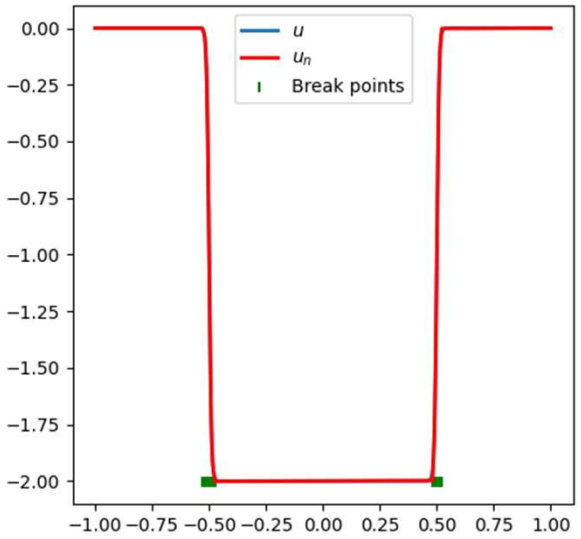


Fixed uniform mesh

Moving the Mesh

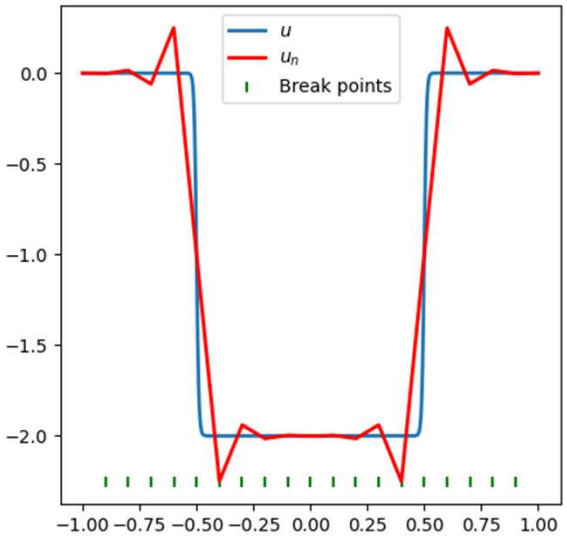


Fixed uniform mesh

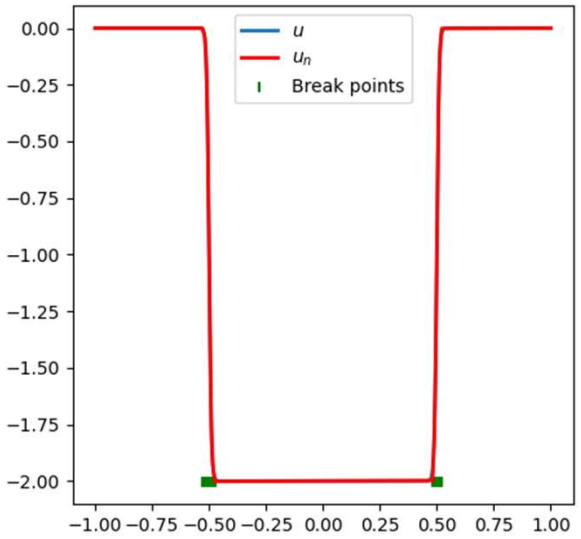


Moving mesh

Moving the Mesh



Fixed uniform mesh



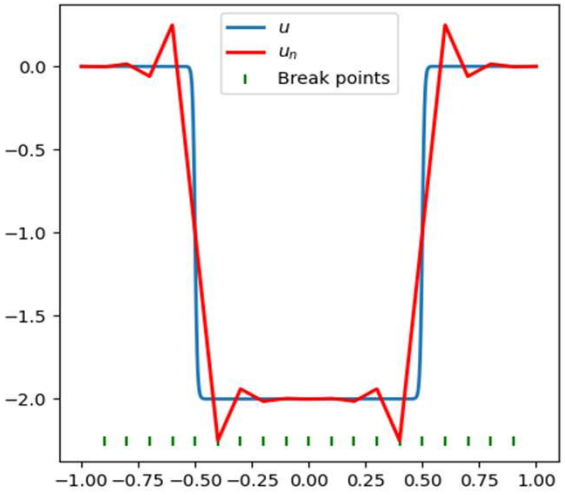
Moving mesh

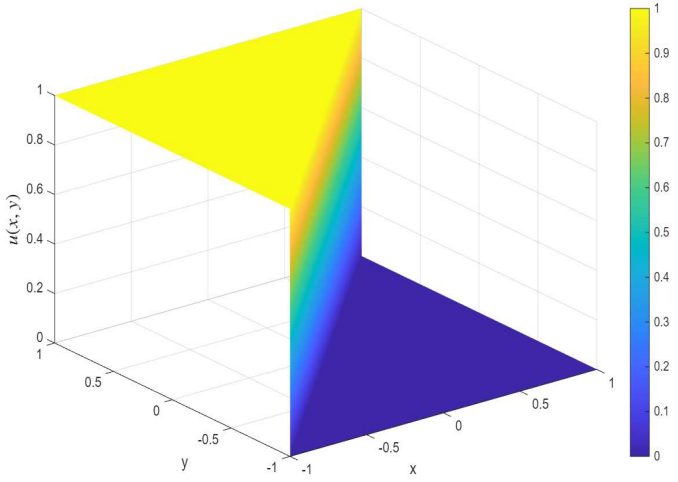
- **Neural networks: continuous piecewise linear approximations.**
- **Feature: moving the mesh.**

Broader Goals

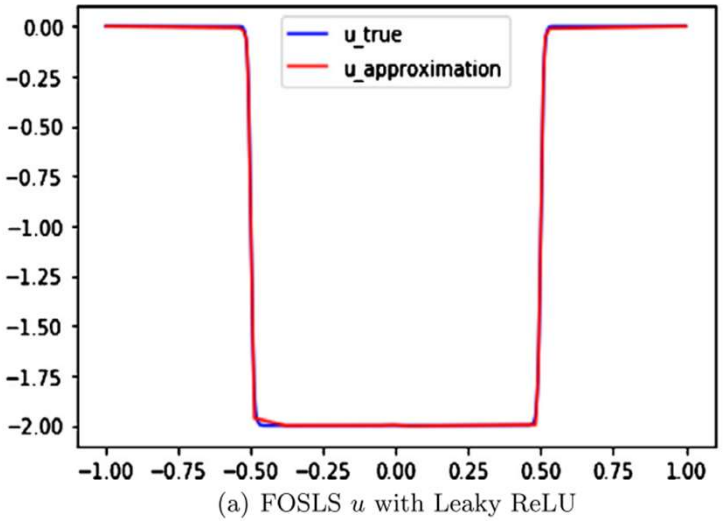
We aim to solve numerical PDEs with challenges such as

- **Boundary/interior layers:** e.g., singularly perturbed elliptic problems.
- **Shocks/discontinuidades and unknown interface**: e.g., hyperbolic conservation laws.





Efficient Neural Network Methods



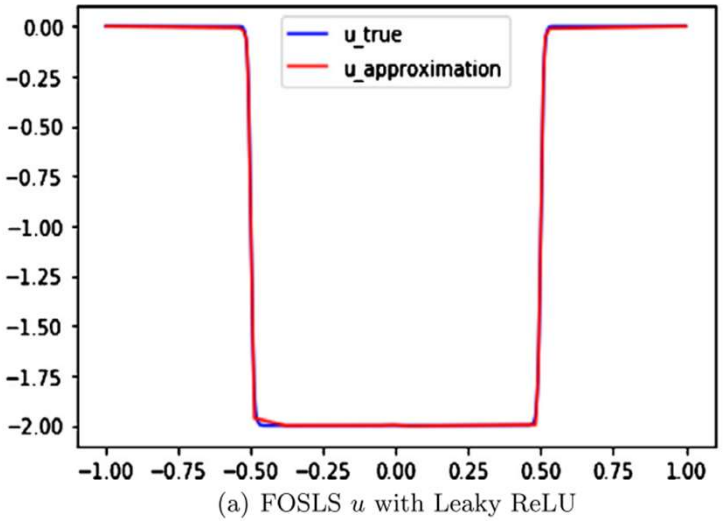
(a) FOSLS u with Leaky ReLU

2962 parameters, 20 hours[1]

- **Important: efficient optimization methods.**

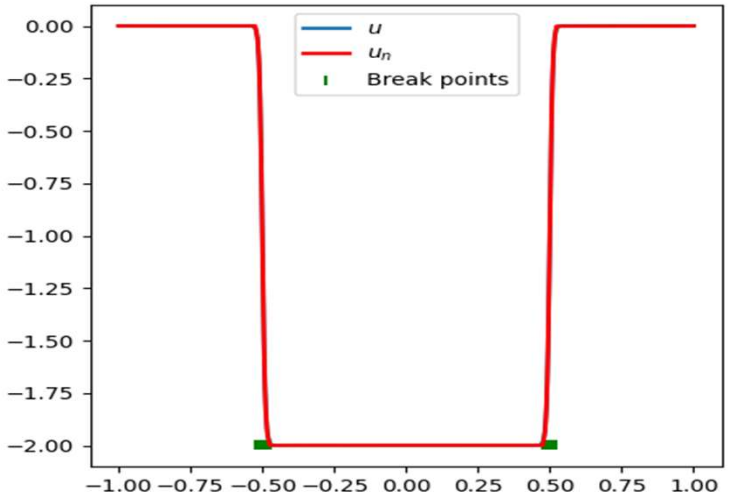
1. Zhiqiang Cai et al. "Deep least-squares methods: An unsupervised learning-based numericalmethod for solving elliptic PDEs". J. Comput. Phys.(2020)

Efficient Neural Network Methods



(a) FOSLS u with Leaky ReLU

2962 parameters, 20 hours[1]



41 parameters, 30 seconds

- **Important: efficient optimization methods.**

PURDUE UNIVERSITY | Department of Mathematics

1. Zhiqiang Cai et al. "Deep least-squares methods: An unsupervised learning-based numericalmethod for solving elliptic PDEs". J. Comput. Phys.(2020)

OUTLINE

I. Neural networks for numerical PDEs

II. One-dimensional problems

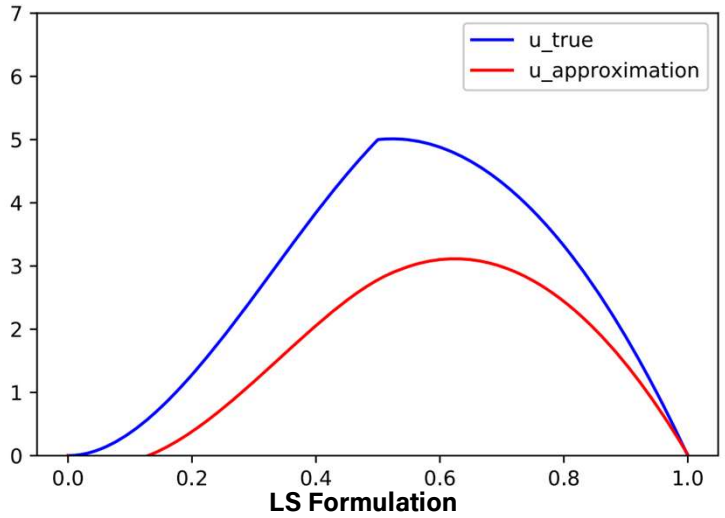
III. Two-dimensional linear advection-reaction equation

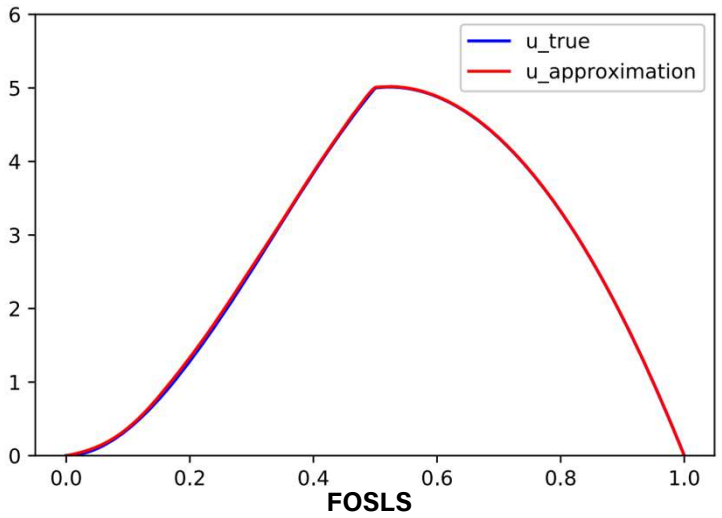
I. NEURAL NETWORKS FOR NUMERICAL PDES

Neural Networks for Numerical PDEs

What do we need to solve numerical PDEs using neural networks?

Neural Networks for Numerical PDEs

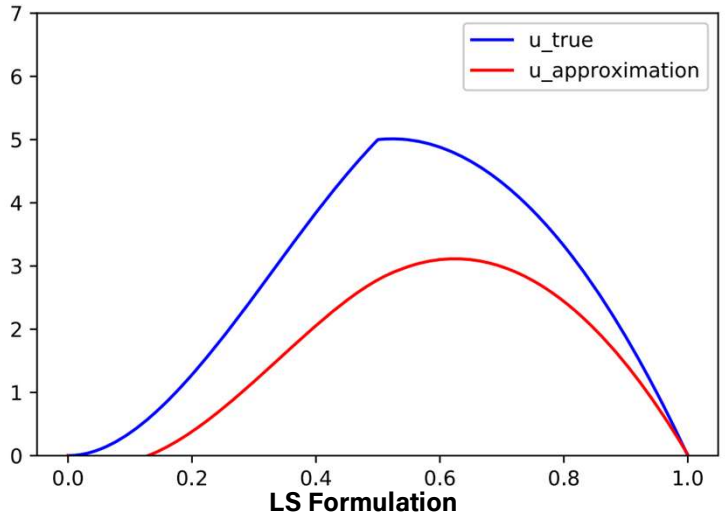


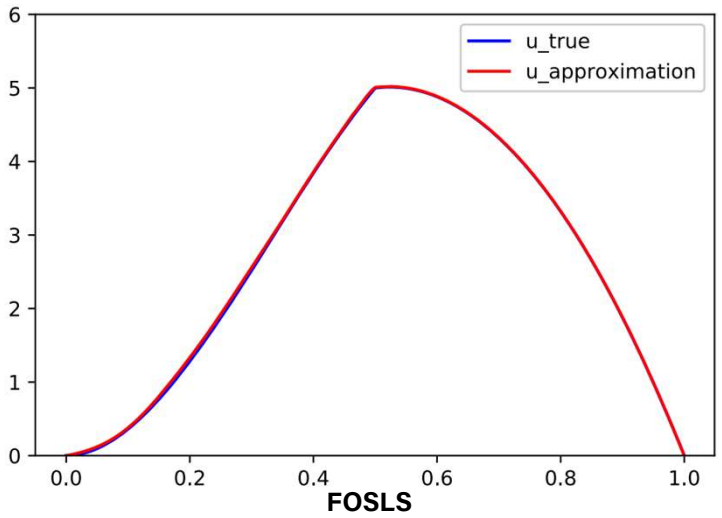


Same neural network, different formulation.[1]

1. Zhiqiang Cai et al. "Deep least-squares methods: An unsupervised learning-based numericalmethod for solving elliptic PDEs". J. Comput. Phys.(2020)

Neural Networks for Numerical PDEs





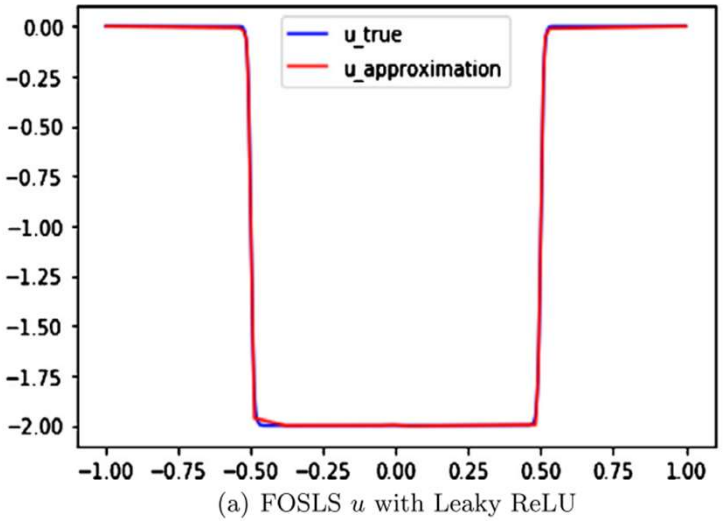
Same neural network, different formulation.[1]

- **Important: 1. PDE formulation.**

PURDUE UNIVERSITY | Department of Mathematics

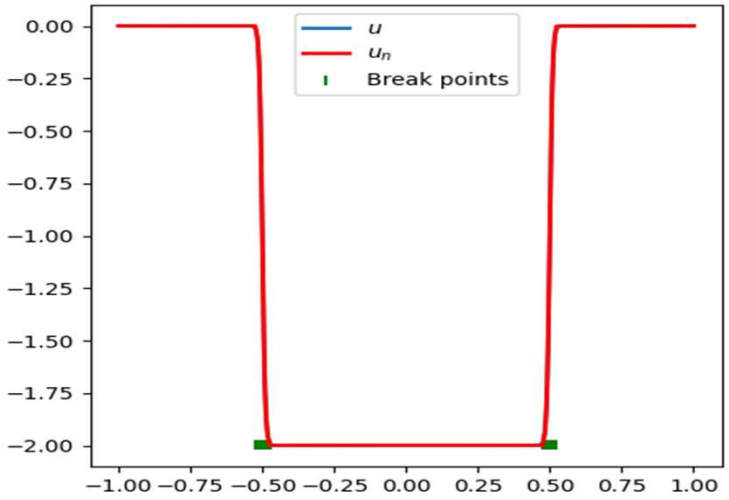
1. Zhiqiang Cai et al. "Deep least-squares methods: An unsupervised learning-based numericalmethod for solving elliptic PDEs". J. Comput. Phys.(2020)

Neural Networks for Numerical PDEs



(a) FOSLS u with Leaky ReLU

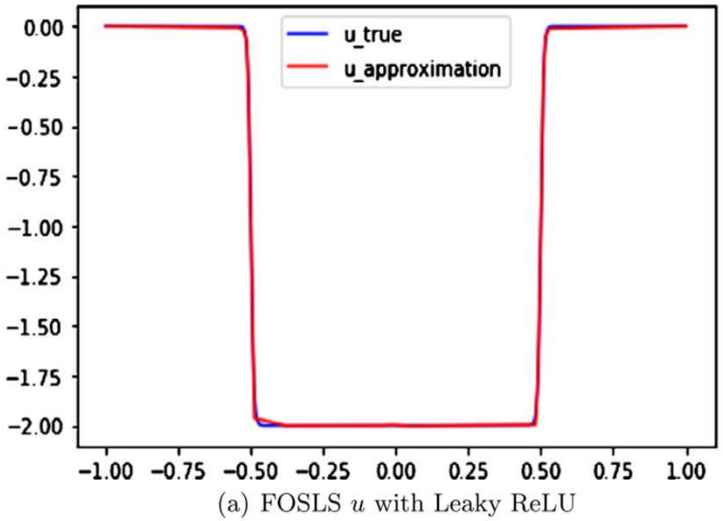
1-32-32-24-24-1, 2962 parameters, 20 hours[1]



1-20-1, 41 parameters, 30 seconds

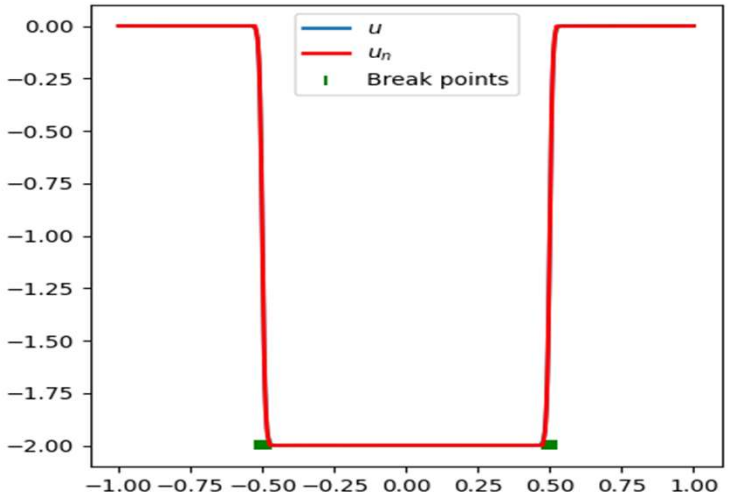
1. Zhiqiang Cai et al. "Deep least-squares methods: An unsupervised learning-based numericalmethod for solving elliptic PDEs". J. Comput. Phys.(2020)

Neural Networks for Numerical PDEs



(a) FOSLS u with Leaky ReLU

1-32-32-24-24-1, 2962 parameters, 20 hours[1]



1-20-1, 41 parameters, 30 seconds

- **Important:** 2. NN architecture.

  3. Efficient optimization methods.

PURDUE UNIVERSITY | Department of Mathematics

1. Zhiqiang Cai et al. "Deep least-squares methods: An unsupervised learning-based numericalmethod for solving elliptic PDEs". J. Comput. Phys.(2020)

Neural Networks for Numerical PDEs

What do we need to solve numerical PDEs using neural networks?

1. **Formulation:** finding a physics preserving formulation.
2. **Architecture:** number of layers and neurons.
3. **Optimization:** designing efficient optimization methods.

II. ONE-DIMENSIONAL PROBLEMS

One-dimensional Elliptic Problems

One-dimensional diffusion-reaction equation

$$\begin{cases} -\big(a(x)u'(x)\big)' + r(x)u(x) = f(x), & x \in (0,\ 1), \\ u(0) = u(1) = 0. \end{cases}$$

1. Formulation

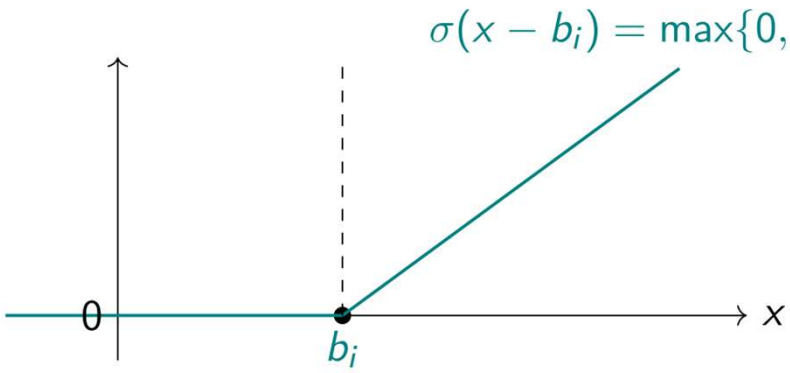
- **Ritz formulation:** find $u \in V$ such that

$$J(u) = \min_{v \in V} J(v), \qquad J(v) = \frac{1}{2}\int_0^1 [(av')^2 + rv^2]\, dx - \int_0^1 fv\, dx.$$

Architecture: One Hidden Layer

One-dimensional shallow (ReLU) neural network

$$\mathcal{M}_n(0,1) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(x - b_i) : c_i \in \mathbb{R}, b_i \in [0,1] \right\}$$

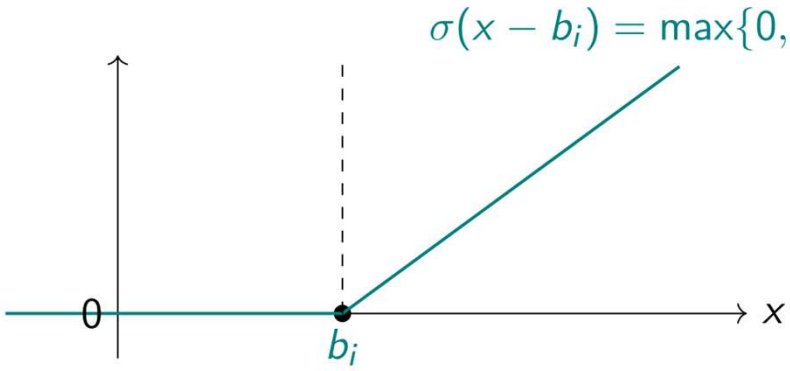


- Continuos piecewise linear function.
- Parameters:
  - c_i: **linear** parameters.
  - b_i: **non-linear** parameters (breaking points).

Architecture: One Hidden Layer

One-dimensional shallow (ReLU) neural network

$$\mathcal{M}_n(0,1) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(x - b_i) : c_i \in \mathbb{R}, b_i \in [0,1] \right\}$$



- Continuos piecewise linear function.
- Parameters:
  - c_i: **linear** parameters.
  - b_i: **non-linear** parameters (breaking points).
- Capable of approximating non-smooth functions.

One-dimensional Elliptic Problems

1. Formulation

- **Ritz formulation:** find $u \in V$ such that

$$J(u) = \min_{\substack{v \in V \\ v(0) = v(1) = 0}} J(v).$$

2. Architecture

- **One hidden layer**

$$\mathcal{M}_n(0,1) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(x - b_i) : c_i \in \mathbb{R}, b_i \in [0,1] \right\}$$

One-dimensional Elliptic Problems

1. Formulation

- **Ritz formulation:** find $u \in V$ such that

$$J(u) = \min_{\substack{v \in V \\ v(0) = v(1) = 0}} J(v).$$

2. Architecture

- **One hidden layer**

$$\mathcal{M}_n(0,1) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(x - b_i) : c_i \in \mathbb{R}, b_i \in [0,1] \right\}$$

Discretization: find $u_n \in \mathcal{M}_n(0,1)$ that minimizes J:

$$J(u_n) = \min_{\substack{v \in \mathcal{M}_n(0,1) \\ v(0) = v(1) = 0}} J(v)$$

One-dimensional Elliptic Problems

1. Formulation

- **Ritz formulation:** find $u \in V$ such that

$$J(u) = \min_{\substack{v \in V \\ v(0) = v(1) = 0}} J(v).$$

2. Architecture

- **One hidden layer**

$$\mathcal{M}_n(0,1) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(x - b_i) : c_i \in \mathbb{R}, b_i \in [0,1] \right\}$$

Discretization: find $u_n \in \mathcal{M}_n(0,1)$ that minimizes J:

$$J(u_n) = \min_{\substack{v \in \mathcal{M}_n(0,1) \\ v(0) = v(1) = 0}} J(v)$$

3. Optimization?

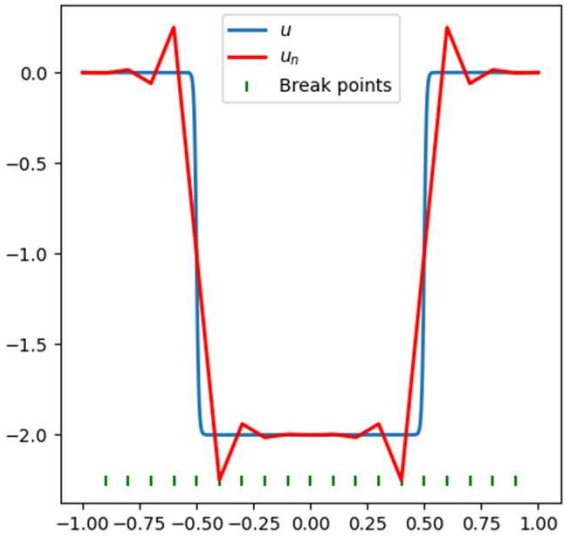
Efficient Iterative Method

3. Optimization

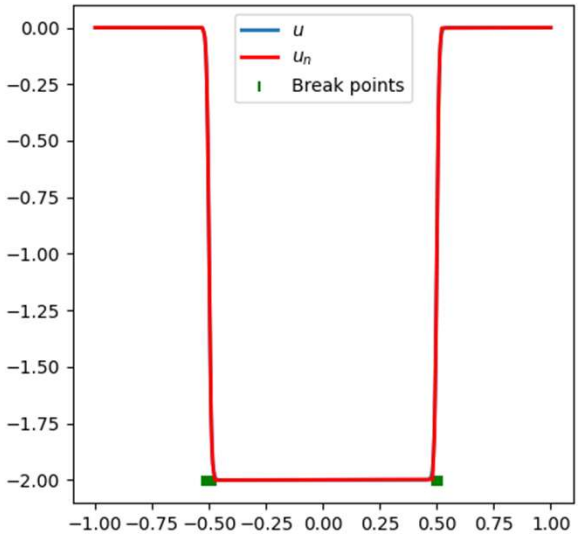
- Block Newton methods:
  - Z. Cai, A. Doktorova, R. D. Falgout, and C.Herrera. *Efficient Shallow Ritz Method for 1D Diffusion Problems.* Comput. Math. Appl, 2025
  - Z. Cai, A. Doktorova, R. D. Falgout, and C.Herrera. *Efficient Shallow Ritz Method for 1D Diffusion-Reaction Problems.* SISC, 2025
  - **Efficient methods ($\mathcal{O}(n)$ per iteration).**
  - Capable of **moving** the mesh.

Numerical Experiment

Solving $-\varepsilon^2 u''(x) + u(x) = f(x),\ \varepsilon^2 = 10^{-4}$.



Initial NN Approximation



Optimized NN Approximation-200 itr.

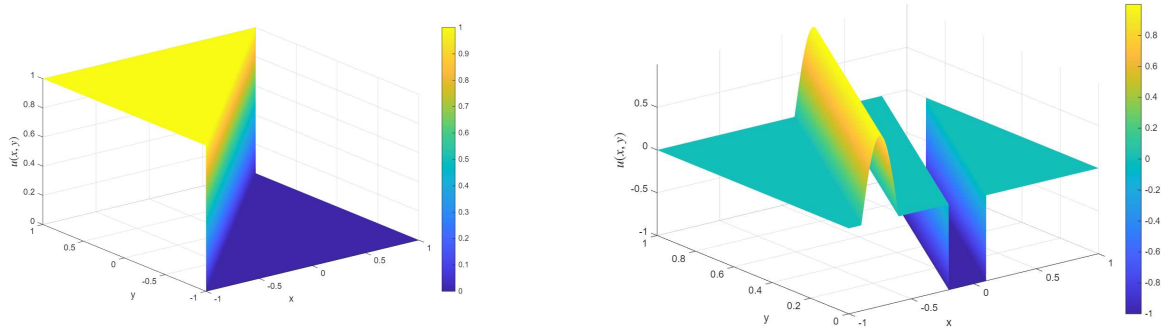
III. TWO-DIMENSIONAL LINEAR ADVECTION-REACTION EQUATION

Linear Advection-Reaction Equation

Two-dimensional advection-reaction equation

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

where $u_{\boldsymbol{\beta}}$ denotes the **directional derivative** along the velocity field $\boldsymbol{\beta}$.



Goal: approximating **discontinuous** solutions without using the **known** interface.

Linear Advection-Reaction Equation

Two-dimensional advection-reaction equation

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

1. Formulation

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

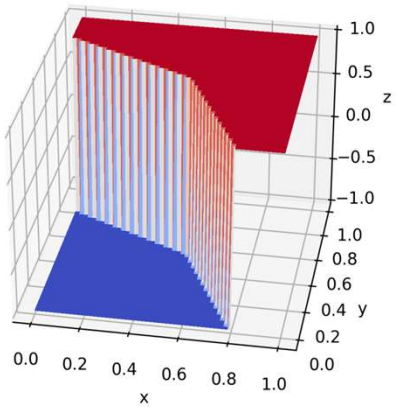
$$\mathcal{L}(u;\, \boldsymbol{f}) = \min_{v \in V_{\boldsymbol{\beta}}} \mathcal{L}\,(v;\, \boldsymbol{f}) \qquad \mathcal{L}(v;\, \boldsymbol{f}) = \int_{\Omega} (v_{\boldsymbol{\beta}} + \gamma u - f)^2\, d\boldsymbol{x} + \int_{\Gamma_{-}} |\boldsymbol{\beta} \cdot \boldsymbol{n}| (v - g)^2 ds\,.$$

- **Coercivity and continuity (De Sterck, Manteuffel, McCormick, Olson, 2004):** there exists positive constants α and M such that

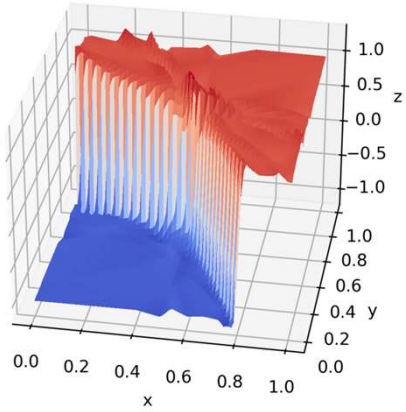
$$\alpha \|v\|_{\boldsymbol{\beta}} \leq \mathcal{L}(v;\, \mathbf{0}) \leq M \|v\|_{\boldsymbol{\beta}}$$

One vs Two Hidden Layers

2. Architecture



Solution [2]



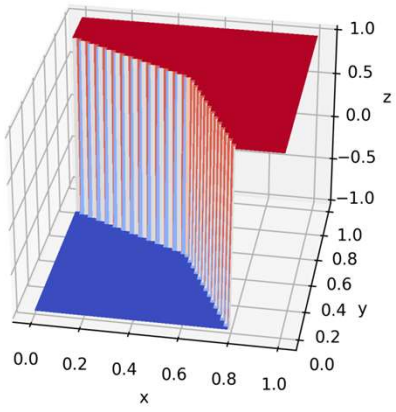
1 hidden layer [2]
2-158-1

PURDUE UNIVERSITY | Department of Mathematics

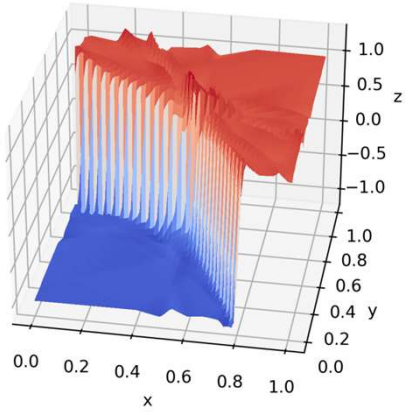
2. Z. Cai, J. Choi, and M. Liu. "Least-Squares Neural Network (LSNN) Method for Linear Advection-Reaction Equation: Discontinuity Interface". SISC (2024)

One vs Two Hidden Layers

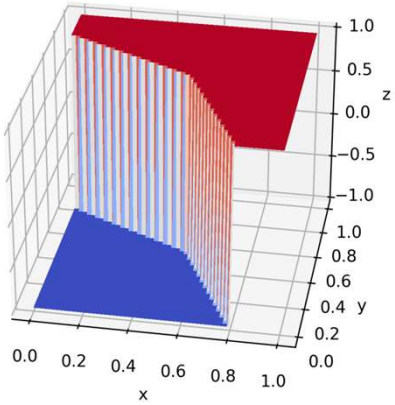
2. Architecture



Solution [2]



1 hidden layer [2]
2-158-1



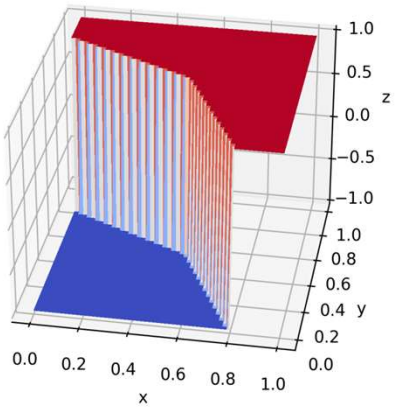
2 hidden layers [2]
2-4-4-1

2. Z. Cai, J. Choi, and M. Liu. "Least-Squares Neural Network (LSNN) Method for Linear Advection-Reaction Equation: Discontinuity Interface". SISC (2024)

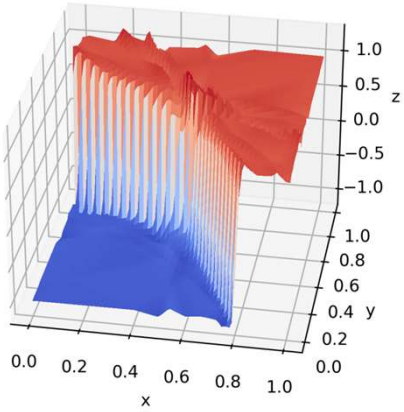
One vs Two Hidden Layers

2. Architecture

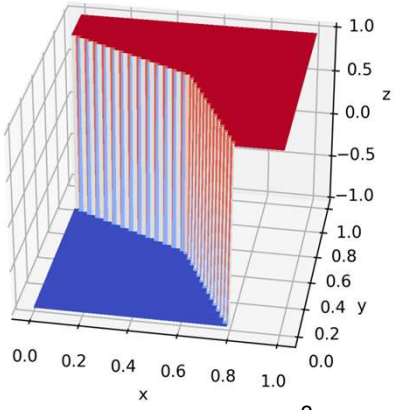
Z. Cai, J. Choi, M. Liu (2024): Two hidden layers are necessary and sufficient for two or more dimensions.



Solution [2]



1 hidden layer [2]
2-158-1



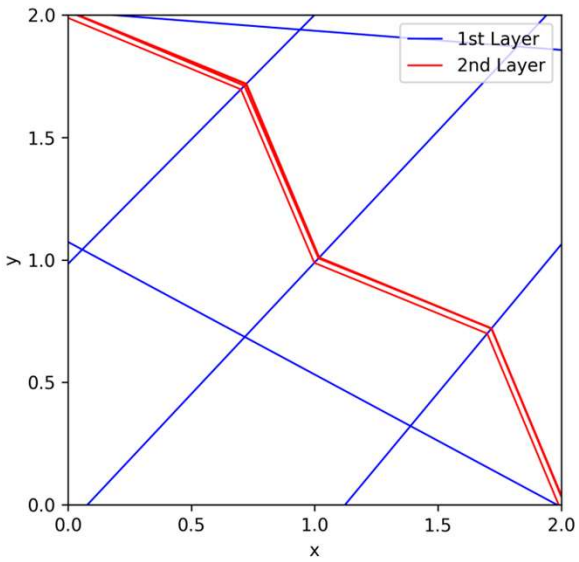
2 hidden layers [2]
2-4-4-1

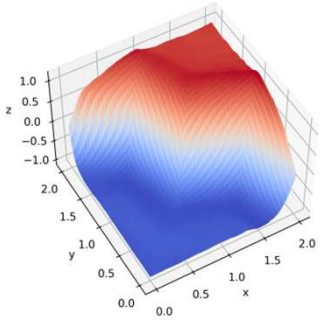
2. Z. Cai, J. Choi, and M. Liu. "Least-Squares Neural Network (LSNN) Method for Linear Advection-Reaction Equation: Discontinuity Interface". SISC (2024)

One vs Two Hidden Layers

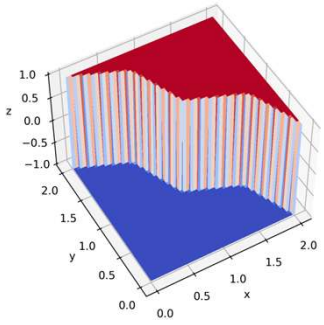
2. Architecture

- **1 hidden layer:** interface is a line segment
- **2 hidden layers:** interface is a curve

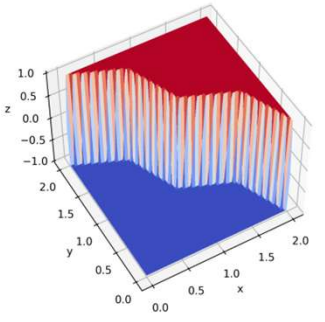




1 hidden layers[2]
2-300-1



Solution[4]



2 hidden layers[2]
2-6-6-1

PURDUE UNIVERSITY | Department of Mathematics

2. Z. Cai, J. Choi, and M. Liu. "Least-Squares Neural Network (LSNN) Method for Linear Advection-Reaction Equation: Discontinuity Interface". SISC (2024)

Shallow Neural Network

One-dimensional shallow (ReLU) neural network

$$\mathcal{M}_n(0,1) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(x - b_i) : c_i \in \mathbb{R}, b_i \in [0,1] \right\}$$

- Continuos piecewise linear function.
- Parameters:
  - c_i: **linear** parameters.
  - b_i: **non-linear** parameters (breaking points).

Shallow Neural Network

Two-dimensional shallow (ReLU) neural network

$$\mathcal{M}_n(\Omega) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(\boldsymbol{\omega}_i \cdot \boldsymbol{x} + b_i) : c_i \in \mathbb{R}, b_i \in \mathbb{R}, \boldsymbol{\omega}_i \in \mathbb{R}^2 \right\}$$

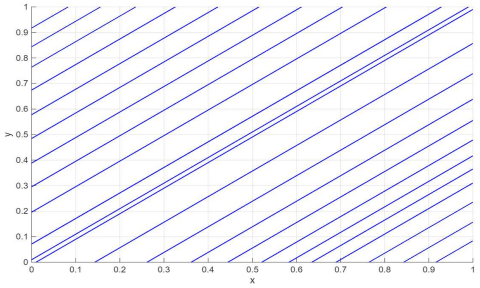
- Continuos piecewise linear function.
- Parameters:
  - c_i: **linear** parameters.
  - $\boldsymbol{\omega}_i$, b_i: **non-linear** parameters
  - **Breaking lines**: $\{\boldsymbol{x} \in \Omega: \boldsymbol{\omega}_i \cdot \boldsymbol{x} + b_i = 0\}$

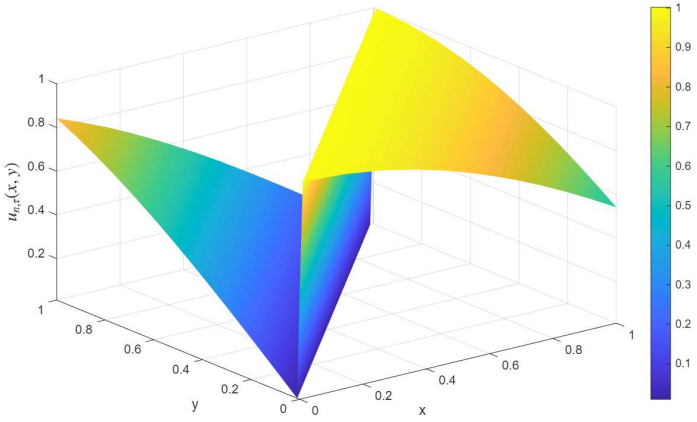
Shallow Neural Network

Two-dimensional shallow (ReLU) neural network

$$\mathcal{M}_n(\Omega) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(\boldsymbol{\omega}_i \cdot \boldsymbol{x} + b_i) : c_i \in \mathbb{R}, b_i \in \mathbb{R}, \boldsymbol{\omega}_i \in \mathbb{R}^2 \right\}$$

- Continuos piecewise linear function.
- Parameters:
  - c_i: **linear** parameters.
  - $\boldsymbol{\omega}_i$, b_i: **non-linear** parameters
  - **Breaking lines**: $\{\boldsymbol{x} \in \Omega: \boldsymbol{\omega}_i \cdot \boldsymbol{x} + b_i = 0\}$





Linear Advection-Reaction Equation

1. Formulation

- **Least-squares formulation:** find $u \in V_{\boldsymbol{\beta}}$ that minimizes

$$\mathcal{L}(v) = \int_{\Omega} (v_{\boldsymbol{\beta}} + \gamma u - f)^2 \, d\boldsymbol{x} + \int_{\Gamma_-} |\boldsymbol{\beta} \cdot \boldsymbol{n}| (v - g)^2 ds \,.$$

2. Architecture

- **One hidden layer**

$$\mathcal{M}_n(\Omega) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(\boldsymbol{\omega}_i \cdot \boldsymbol{x} + b_i) : c_i \in \mathbb{R}, b_i \in \mathbb{R}, \boldsymbol{\omega}_i \in \mathbb{R}^2 \right\}$$

Linear Advection-Reaction Equation

1. Formulation

- **Least-squares formulation:** find $u \in V_{\boldsymbol{\beta}}$ that minimizes

$$\mathcal{L}(v) = \int_{\Omega} (v_{\boldsymbol{\beta}} + \gamma u - f)^2 \, d\boldsymbol{x} + \int_{\Gamma_-} |\boldsymbol{\beta} \cdot \boldsymbol{n}| (v - g)^2 ds \,.$$

2. Architecture

- **One hidden layer**

$$\mathcal{M}_n(\Omega) = \left\{ c_0 + \sum_{i=1}^{n} c_i \sigma(\boldsymbol{\omega}_i \cdot \boldsymbol{x} + b_i) : c_i \in \mathbb{R}, b_i \in \mathbb{R}, \boldsymbol{\omega}_i \in \mathbb{R}^2 \right\}$$

Discretization: find a neural network $u_n \in \mathcal{M}_n(\Omega)$ that minimizes $\mathcal{L}$:

$$\mathcal{L}(u_n) = \min_{v \in \mathcal{M}_n(\Omega)} \mathcal{L}(v)$$

Optimality Conditions

3. Optimization

Non-convex optimization problem: find a neuronal network u_n that minimizes the functional $\mathcal{L}$.

Difficulties:

- Initialization.
- ReLU is not pointwise differentiable.
- Possible matrix singularities.

Optimality Conditions

3. Optimization

Non-convex optimization problem: find a neuronal network u_n that minimizes the functional $\mathcal{L}$.

Neural network representation:

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

where

$$\boldsymbol{c} = (c_0, c_1, \dots, c_n)^T, \qquad \boldsymbol{r} = (b_1, \boldsymbol{\omega}_1, \dots, b_n, \boldsymbol{\omega}_n)^T.$$

First-order optimality conditions:

$$\nabla_{\boldsymbol{c}} \mathcal{L}(\boldsymbol{c}, \boldsymbol{r}) = \mathbf{0}, \qquad \nabla_{\boldsymbol{r}} \mathcal{L}(\boldsymbol{c}, \boldsymbol{r}) = \mathbf{0}.$$

Mass and Gauss-Newton Matrices

- **Linear parameters**

$$\mathbf{0} = \nabla_c \mathcal{L}(c, r) = A(r)c - F(r)$$

  - Mass matrix $A(r)$ is spd.

Mass and Gauss-Newton Matrices

- **Linear parameters**

$$\mathbf{0} = \nabla_c \mathcal{L}(c, r) = A(r)c - F(r)$$

  - Mass matrix $A(r)$ is spd.

- **Non-linear parameters**

$$\mathbf{0} = \nabla_r \mathcal{L}(c, r) = (D(c) \otimes I_3) G(c, r)$$

Mass and Gauss-Newton Matrices

- **Linear parameters**

$$\mathbf{0} = \nabla_c \mathcal{L}(c, r) = \boldsymbol{A}(r)c - \boldsymbol{F}(r)$$

  - Mass matrix $\boldsymbol{A}(r)$ is spd.

- **Non-linear parameters**

$$\mathbf{0} = \nabla_r \mathcal{L}(c, r) = (\boldsymbol{D}(c) \otimes \boldsymbol{I_3})\boldsymbol{G}(c, r)$$

  - Gauss-Newton matrix.

$$(\boldsymbol{D}(c) \otimes \boldsymbol{I_3})\mathcal{H}(c, r)(\boldsymbol{D}(c) \otimes \boldsymbol{I_3})$$

  - Layer Gauss-Newton $\mathcal{H}(c, r)$ matrix is spd.
  - If $c_i = 0$, we do not need to update $r_i = (b_i, \omega_i)$.

Structure Guided Gauss Newton Method (SgGN)

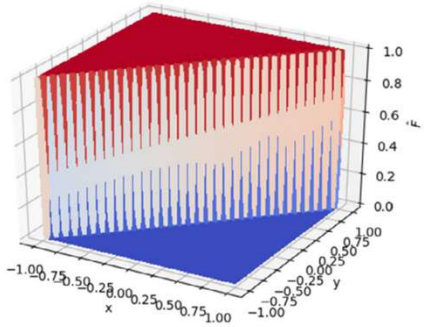
Iterative method:

1. **Inicialization:** $r^{(0)}$-> **uniform** partition.

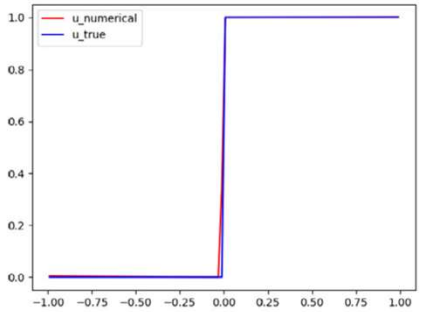
2. **Iterate:** given $\left(c^{(k)}, r^{(k)}\right)$**,** we compute $\left(c^{(k+1)}, r^{(k+1)}\right)$

   - **Linear parameters:** compute $c^{(k+1)}$ by solving the **system of linear equations** $\nabla_c \mathcal{L}\left(c, r^{(k)}\right) = \mathbf{0}$.
   - **Non-linear parameters:** compute $r^{(k+1)}$ using one **Gauss-Newton iteration.**

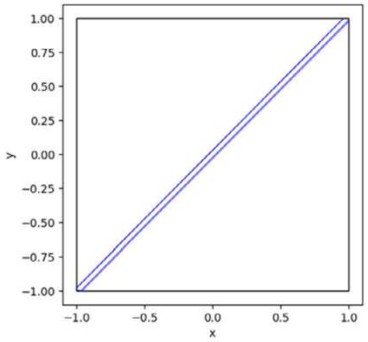
Transport equation $u_x + u_t = 0$



(a) Network approximation $\bar{u}_T^N$

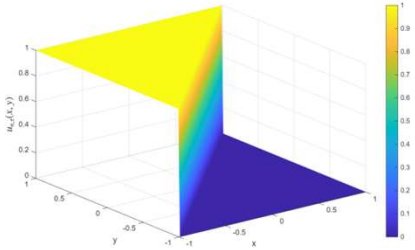


(b) Vertical cross section along $y = -x$

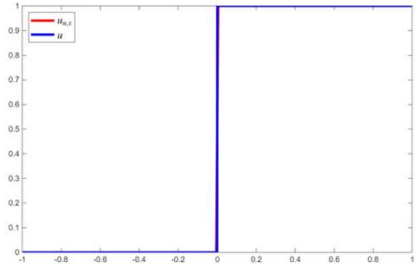


(c) Network breaking lines

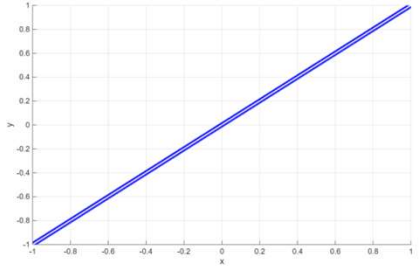
20000 ADAM iterations; the relative L^2 error is 5.8×10^{-2}



(a) NN approximation $u_{n,\tau}$



(b) Traces on $y = -x$



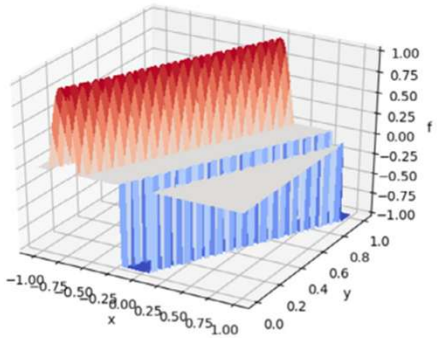
(c) NN breaking lines

100 SgGN iterations; the relative L^2 error is 5.3×10^{-11}

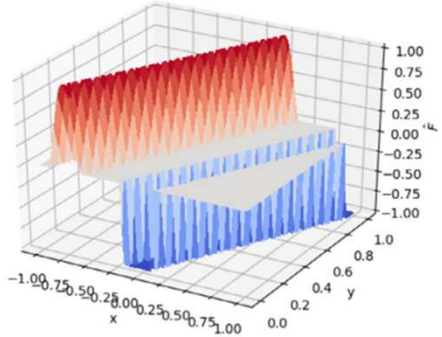
PURDUE UNIVERSITY | Department of Mathematics

5. Z. Cai, J. Chen, and M. Liu. "Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation". J. Comput. Phys (2021)

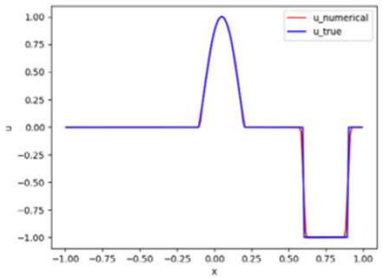
Two Discontinuous Interfaces



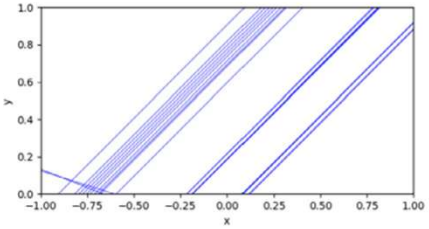
(a) Exact solution u



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



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

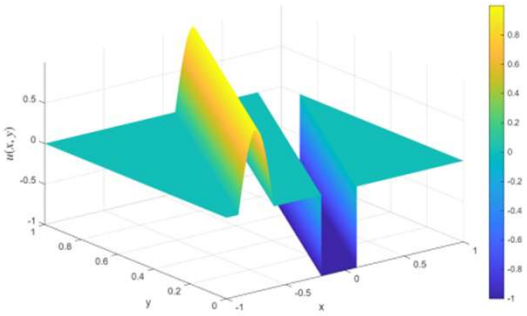


(d) Network breaking lines

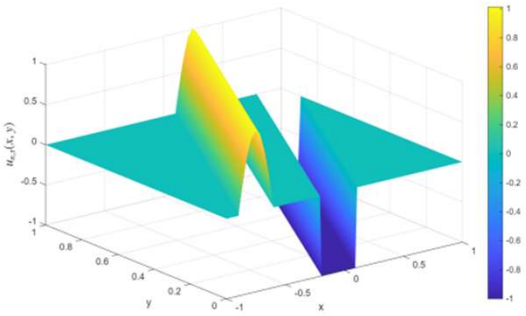
80000 ADAM iterations[5]; the relative L^2 error is 1.17×10^{-1}

5. Z. Cai, J. Chen, and M. Liu. "Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation". J. Comput. Phys (2021)

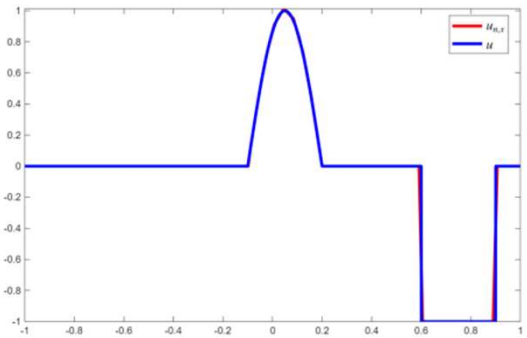
Two Discontinuous Interfaces



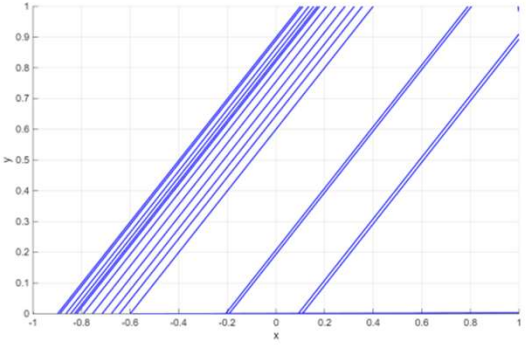
(a) Exact Solution



(b) NN Approximation



(c) Traces on $y = 0.8$



(d) NN breaking lines

100 SgGN iterations; the relative L^2 error is 2.80×10^{-3}

Conclusions and Final Remarks

- We can **accurately approximate discontinuos solutions** using neural networks.
- SgGN method **significantly reduces the number of iterations** compared to standard first order methods.
- Mass and Gauss-Newton matrices are **ill-conditioned**. **Efficient linear solvers?**
- Actual and future work:
  - Scalar hyperbolic conservation laws.
  - Iterative methods for 2 hidden layers NN.

THANKS

website

PURDUE UNIVERSITY | Department of Mathematics