Efficient Neural Network Methods for Numerical PDEs: Singularly Perturbed Problems

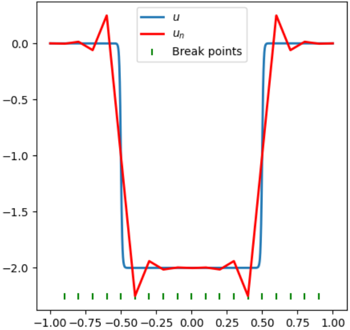
César Herrera [1]

Joint work with Zhiqiang Cai[1] Anastassia Doktorova[1] Robert D. Falgout[2]

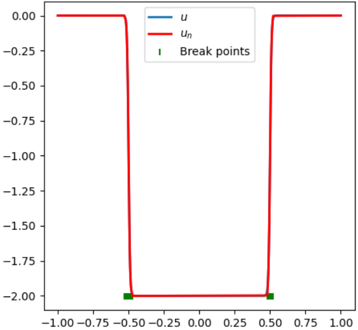
[1]Department of Mathematics, Purdue University
[2]Lawrence Livermore National Laboratory

Finite Element Circus
October 2025

A New Class of Approximating Functions



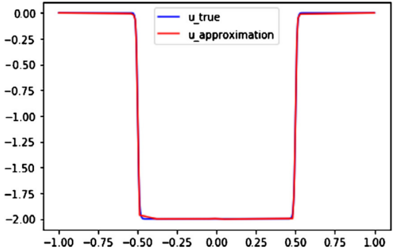
Fixed uniform mesh



Moving mesh

- **Feature: moving the mesh**

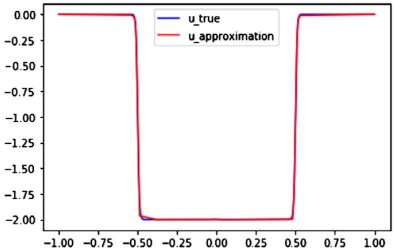
A New Class of Approximating Functions



(a) FOSLS u with Leaky ReLU

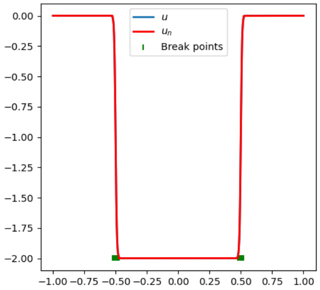
1-32-32-24-24-1, 2962 parameters, 20 hours[1]

A New Class of Approximating Functions



(a) FOSLS u with Leaky ReLU

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



1-20-1, 41 parameters, 30 seconds

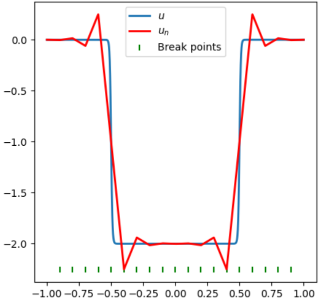
- **Important: efficient nonlinear solver**

[1] Zhiqiang Cai et al. "Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs". In: *Journal of Computational Physics* (2020)

Broader Goals

We look at problems with numerical challenges such as

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



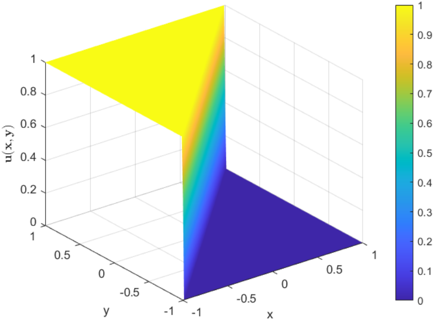


Table of Contents

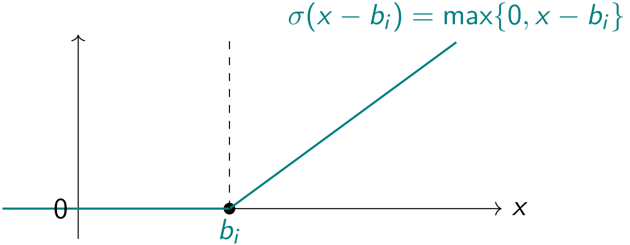
1. Shallow Neural Networks
2. Diffusion-Reaction Equation
3. Allen-Cahn Equation

Table of Contents

1. Shallow Neural Networks
2. Diffusion-Reaction Equation
3. Allen-Cahn Equation

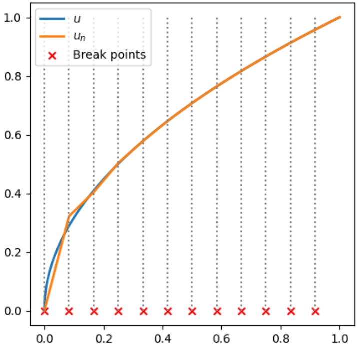
- **One-dimensional shallow (ReLU) neural network:**

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

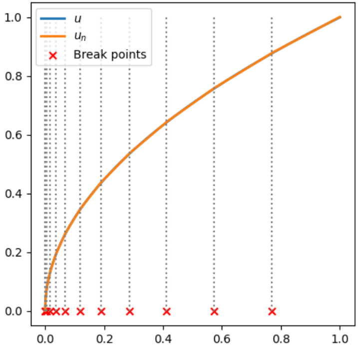


- **Piecewise linear function**
- **Parameters**
  1. $c_i \to$ **linear parameters**
  2. $b_i \to$ **non-linear parameters (breaking points)**

Example



Fixed uniform mesh



Moving mesh

Fig: NN approximation to $u(x) = \sqrt{x}$ with 12 breakpoints

Shallow Neural Networks

- **In fact the error, is (L^∞ norm)**
  - $\mathcal{O}(n^{-1/2})$ **on a fixed uniform mesh.**
  - $\mathcal{O}(n^{-1})$ **on a moving mesh.**
- **Drawback: Finding optimal breaking points $\leftrightarrow$ solving a high-dimensional non-convex optimization problem**

Table of Contents

1 Shallow Neural Networks

2 Diffusion-Reaction Equation

3 Allen-Cahn Equation

Neural Network Method for 1D Diffusion–Reaction

Diffusion-Reaction problem:

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

Ritz formulation:

$$u = \underset{v \in H_0^1(0,1)}{\arg\min}\ J(v), \qquad J(v) = \frac{1}{2}\int_0^1 \left[(v')^2 + v^2\right] dx - \int_0^1 fv\, dx.$$

Neural network approximation:

$$u_n = \underset{\substack{v \in \mathcal{M}_n(0,1) \\ v(0)=v(1)=0}}{\arg\min}\ J(v).$$

Neural Network Method for 1D Diffusion–Reaction

Diffusion-Reaction problem:

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

Ritz formulation:

$$u = \underset{v \in H_0^1(0,1)}{\arg\min}\ J(v), \qquad J(v) = \frac{1}{2}\int_0^1 \left[(v')^2 + v^2\right] dx - \int_0^1 fv\, dx.$$

Neural network approximation:

$$u_n = \underset{\substack{v \in \mathcal{M}_n(0,1) \\ v(0)=v(1)=0}}{\arg\min}\ J(v).$$

First-Order Optimality Conditions

Minimization problem:

$$u_n = \underset{\substack{v \in \mathcal{M}_n(0,1) \\ v(0)=v(1)=0}}{\arg\min} \; J(v)$$

Neural network representation:

$$u_n(x; \mathbf{c}, \mathbf{b}) = c_0 + \sum_{i=1}^{n} c_i \, \sigma(x - b_i),$$

where

$$\mathbf{c} = (c_0, \ldots, c_n)^T, \qquad \mathbf{b} = (b_1, \ldots, b_n)^T.$$

First-order optimality conditions:

$$\nabla_{\mathbf{c}} J(\mathbf{c}, \mathbf{b}) = \mathbf{0}, \qquad \nabla_{\mathbf{b}} J(\mathbf{c}, \mathbf{b}) = \mathbf{0}.$$

Optimality Conditions

For a given fixed $\mathbf{b}$

$$\nabla_{\mathbf{c}} J(\mathbf{c}, \mathbf{b}) = \mathbf{0}$$

is a system of **linear** equations for $\mathbf{c}$.

Difficulties:

- **Coefficient matrix**

$$A(\mathbf{b}) = \left(\int_0^1 \sigma'(x - b_i)\sigma'(x - b_j)dx \right)_{ij}, \quad \kappa(A) = \mathcal{O}(nh_{\min}^{-1})$$

- **Mass matrix**

$$M(\mathbf{b}) = \left(\int_0^1 \sigma(x - b_i)\sigma(x - b_j)dx \right)_{ij}, \quad \kappa(M) = \mathcal{O}(nh_{\min}^{-3})$$

are both **dense** and **ill-conditioned**.

Optimality Conditions

For a given fixed $\mathbf{c}$

$$\nabla_{\mathbf{b}} J(\mathbf{c}, \mathbf{b}) = 0$$

is a system of **nonlinear** algebraic equations for $\mathbf{b}$.

Difficulties:

- ReLU is **not differentiable** at 0.
- The Hessian matrix $\nabla^2_{\mathbf{b}} J(\mathbf{c}, \mathbf{b})$ could be **singular**.

How do we overcome these challenges?

- The mass and coefficient matrices can be **factorized** as products of **tri-diagonal** matrices.
- A **reduced nonlinear system** is obtained by removing breakpoints that make the Hessian **singular**.
- **Exact inversions** can be done in $\mathcal{O}(n)$ operations.

Iterative Method

Given the parameters $(\mathbf{c}^{(k)}, \mathbf{b}^{(k)})$, we compute $(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k+1)})$ as follows:

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

$$\nabla_{\mathbf{c}} J(\mathbf{c}, \mathbf{b}^{(k)}) = \mathbf{0}.$$

(ii) Compute $\mathbf{b}^{(k+1)}$ using a **Newton iteration**

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} - \left[\nabla^2_{\mathbf{b}} J(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k)})\right]^{-1} \nabla_{\mathbf{b}} J(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k)}).$$

The **computational cost** of each iteration is $\mathcal{O}(n)$.

Iterative Method

Given the parameters $(\mathbf{c}^{(k)}, \mathbf{b}^{(k)})$, we compute $(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k+1)})$ as follows:

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

$$\nabla_{\mathbf{c}} J(\mathbf{c}, \mathbf{b}^{(k)}) = \mathbf{0}.$$

(ii) Compute $\mathbf{b}^{(k+1)}$ using a **Newton iteration**

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} - \left[\nabla^2_{\mathbf{b}} J(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k)})\right]^{-1} \nabla_{\mathbf{b}} J(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k)}).$$

The **computational cost** of each iteration is $\mathcal{O}(n)$.

Iterative Method

Given the parameters $(\mathbf{c}^{(k)}, \mathbf{b}^{(k)})$, we compute $(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k+1)})$ as follows:

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

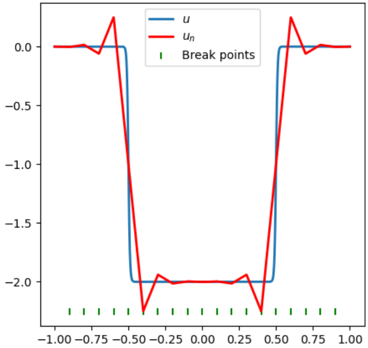
$$\nabla_{\mathbf{c}} J(\mathbf{c}, \mathbf{b}^{(k)}) = \mathbf{0}.$$

(ii) Compute $\mathbf{b}^{(k+1)}$ using a **Newton iteration**

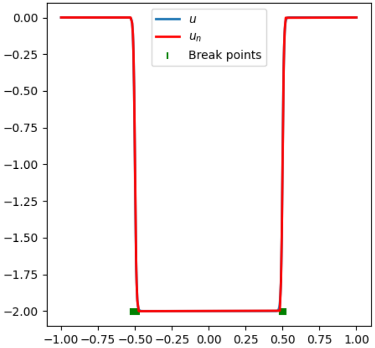
$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} - \left[\nabla^2_{\mathbf{b}} J(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k)})\right]^{-1} \nabla_{\mathbf{b}} J(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k)}).$$

The **computational cost** of each iteration is $\mathcal{O}(n)$.

Numerical Experiment



Initial NN Approximation



Optimized NN Approximation-200 itr

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

Table of Contents

1 Shallow Neural Networks

2 Diffusion-Reaction Equation

3 Allen-Cahn Equation

Allen-Cahn Equation

- Scalar **Allen-Cahn** equation

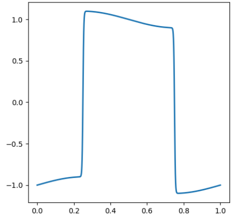
$$u_t(x,t) = \varepsilon^2 u_{xx}(x,t) - \big(u(x,t)^3 - u(x,t)\big)$$

- First order CSS

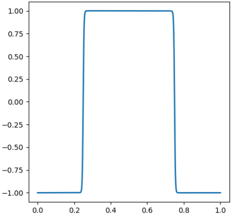
$$\frac{u^k - u^{k-1}}{\Delta t} = \varepsilon^2 u^k_{xx} - (u^k)^3 + u^{k-1}$$

  - Absolute stable.
  - **Semilinear** Diffusion-reaction problem for each time step.

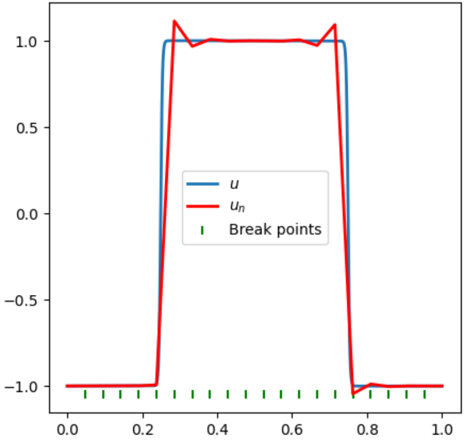
Numerical Experiment: Finite Element Approximation



Initial condition



Final state



Finite element approximation, 20 uniform breakpoints

Semilinear Diffusion-Reaction problem:

$$\begin{cases} -u''(x) + (u(x))^3 = f(x), \quad x \in (0,1), \\ u(0) = u(1) = 0. \end{cases}$$

Neural Network Method Semilinear Diffusion–Reaction

Semilinear Diffusion-Reaction problem:

$$\begin{cases} -u''(x) + (u(x))^3 = f(x), & x \in (0,1), \\ u(0) = u(1) = 0. \end{cases}$$

Can we extend our iterative NN method for these problems?

Semilinear Diffusion-Reaction problem:

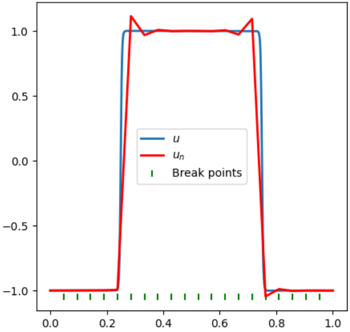
$$\begin{cases} -u''(x) + (u(x))^3 = f(x), \quad x \in (0,1), \\ u(0) = u(1) = 0. \end{cases}$$

Can we extend our iterative NN method for these problems?

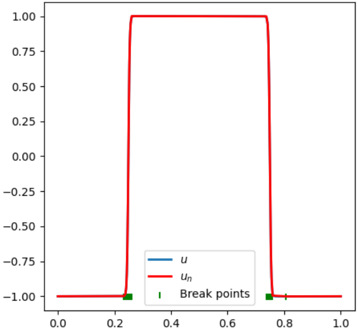
Yes!

The method can be generalized **without losing efficiency**: each iteration still costs $\mathcal{O}(n)$.

Numerical Experiment: NN Approximation



20 **fixed** breakpoints



20 **moving** breakpoints

Fig: Solving $u_t = \varepsilon^2 u_{xx} - \left(u^3 - u\right)$, $\varepsilon^2 = 10^{-5}$

Conclusions and Remarks

- Key points of our NN method for 1D diffusion-reaction problems.
  - Efficiently **moves the mesh** for singularly perturbed problems.
  - **Local convergence** was analyzed.
  - Can be extended to semilinear problems and Allen-Cahn equation.
- Ongoing work: 2D Advection-Reaction Equation.
- For more details see:
  - Z. Cai, A. Doktorova, R. D. Falgout, and C. Herrera. **Efficient Shallow Ritz Method For 1D Diffusion Problems.** arXiv:2404.17750.
  - Z. Cai, A. Doktorova, R. D. Falgout, and C. Herrera. **Efficient Shallow Ritz Method For 1D Diffusion-Reaction Problems.** SISC, 2025.
  - Website: math.purdue.edu/herre125/

Conclusions and Remarks

- Key points of our NN method for 1D diffusion-reaction problems.
  - Efficiently **moves the mesh** for singularly perturbed problems.
  - **Local convergence** was analyzed.
  - Can be extended to semilinear problems and Allen-Cahn equation.
- Ongoing work: 2D Advection-Reaction Equation.
- For more details see:
  - Z. Cai, A. Doktorova, R. D. Falgout, and C. Herrera. **Efficient Shallow Ritz Method For 1D Diffusion Problems.** arXiv:2404.17750.
  - Z. Cai, A. Doktorova, R. D. Falgout, and C. Herrera. **Efficient Shallow Ritz Method For 1D Diffusion-Reaction Problems.** SISC, 2025.
  - Website: math.purdue.edu/herre125/