Fast Iterative Solver for Neural Network Method: I. 1D Diffusion Problems

César Herrera[1]
Zhiqiang Cai[1] Anastassia Doktorova[1] Robert D. Falgout[2]

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

Student Colloquium, Purdue University
September 2024

Introduction: Neural Networks[1]

FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

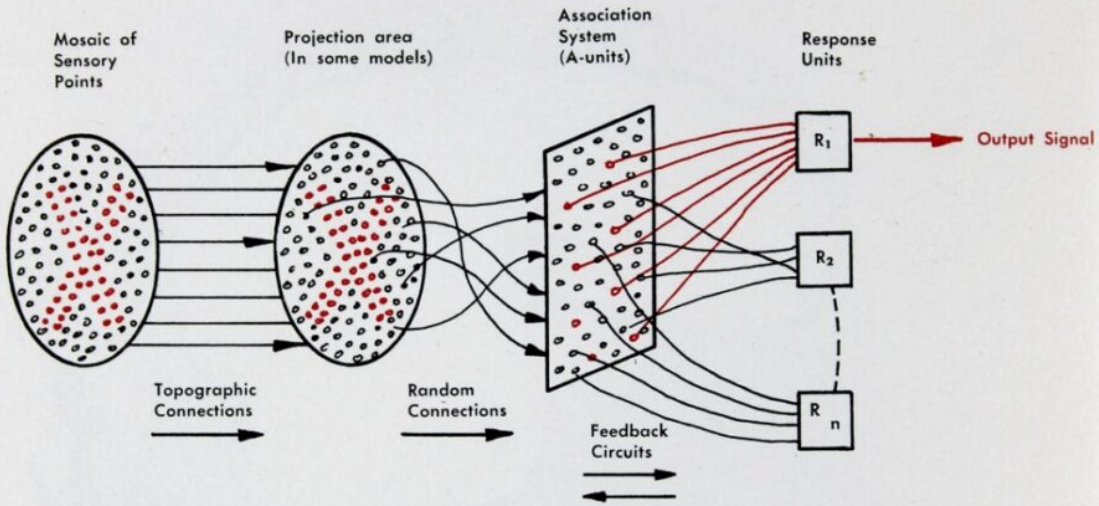


FIG. 2 — Organization of a perceptron.

[1] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain.". In: *Psychological Review* 65.6 (1958), pp. 386–408. ISSN: 0033-295X. DOI: 10.1037/h0042519. URL: http://dx.doi.org/10.1037/h0042519.

Introduction: Neural Networks (Fully-Connected Feed-Forward NN)

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

Output: $u(\mathbf{x}) = W^{(L)}_{c \times (n_{L-1}+1)} \begin{bmatrix} 1 \\ \mathbf{x}^{(L-1)} \end{bmatrix}$

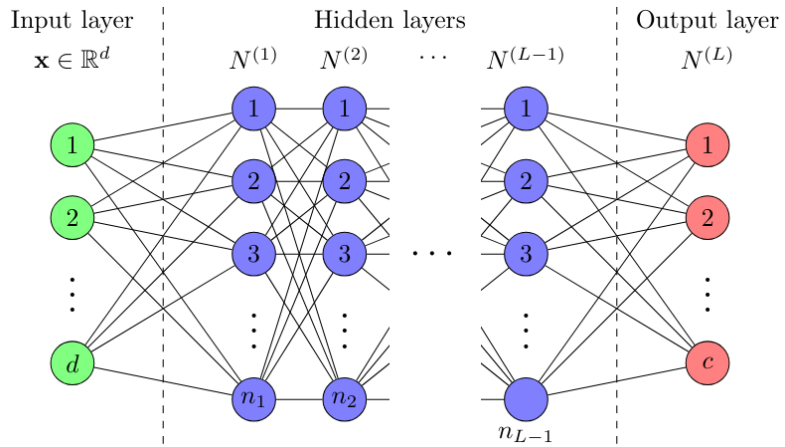


Figure: The neural network function structure[2]

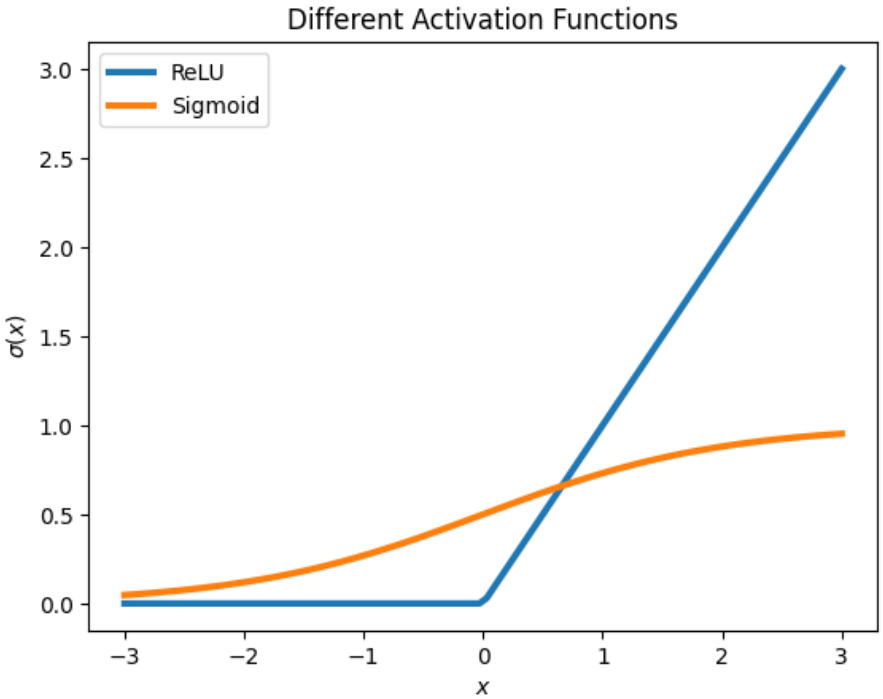
[2] Z. Cai, J. Choi, and M. Liu. *Least-Squares Neural Network (LSNN) Method For Linear Advection-Reaction Equation: Discontinuity Interface*. 2024. arXiv: 2301.06156 [math.NA].

Activation Functions and Shallow Neural Network

- **Shallow neural network (one hidden layer):**

$$\mathcal{M}_n(\sigma, d) = \left\{ c_{-1} + \sum_{i=0}^{n} c_i \sigma(\omega_i \cdot \mathbf{x} - b_i) : c_i, b_i \in \mathbb{R}, \omega_i \in \mathcal{S}^{d-1} \right\}$$

- **Some activation functions:**
  1. ReLU: $\sigma(x) = \max\{0, x\}$
  2. Sigmoid: $\sigma(x) = \frac{1}{1+e^x}$



A New Class of Approximating Functions

Universal Aproximation Theorem (Cybenko 1989, Hornik-Stinchcombe-White 1990)

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

A Priori Error Estimate

See, e.g., Daubechies-DeVore-Foucart-Hanin-Petrova 2021, Yarotsky 2017, DeVore-Hanin-Petrova 2021.

- C^0 **piecewise linear functions on a fixed mesh in** $[0,1]$**:**

$$S_1^0(\Delta) = \left\{ \sum_{i=1}^{n} c_i \phi_i(x) : c_i \in \mathbb{R} \right\},$$

where

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

- C^0 **piecewise linear functions on a moving mesh in** $[0,1]$**:**

$$S_1^0(n) = \left\{ \sum_{i=1}^{n} c_i \phi_i(x; x_{i-1}, x_i, x_{i+1}) : c_i \in \mathbb{R}, , x_i \in [0,1] \right\}$$

- **One-dimensional shallow neural network:**

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

- **Important inclusion[3]:**

$$S_1^0(n) \subset \mathcal{M}_n(I) \subset S_1^0(n+1)$$

[3] I. Daubechies et al. "Nonlinear Approximation and (Deep) ReLU Networks". English (US). in: *Constructive Approximation* 55.1 (Feb. 2022), pp. 127–172. ISSN: 0176-4276. DOI: 10.1007/s00365-021-09548-z.

Our Goals

- Solve PDEs using NNs
  - Design fast NN methods
  - Design suitable NN architectures
  - Utilize geometric interpretation of paramaters in the NN

Why Neural Networks?

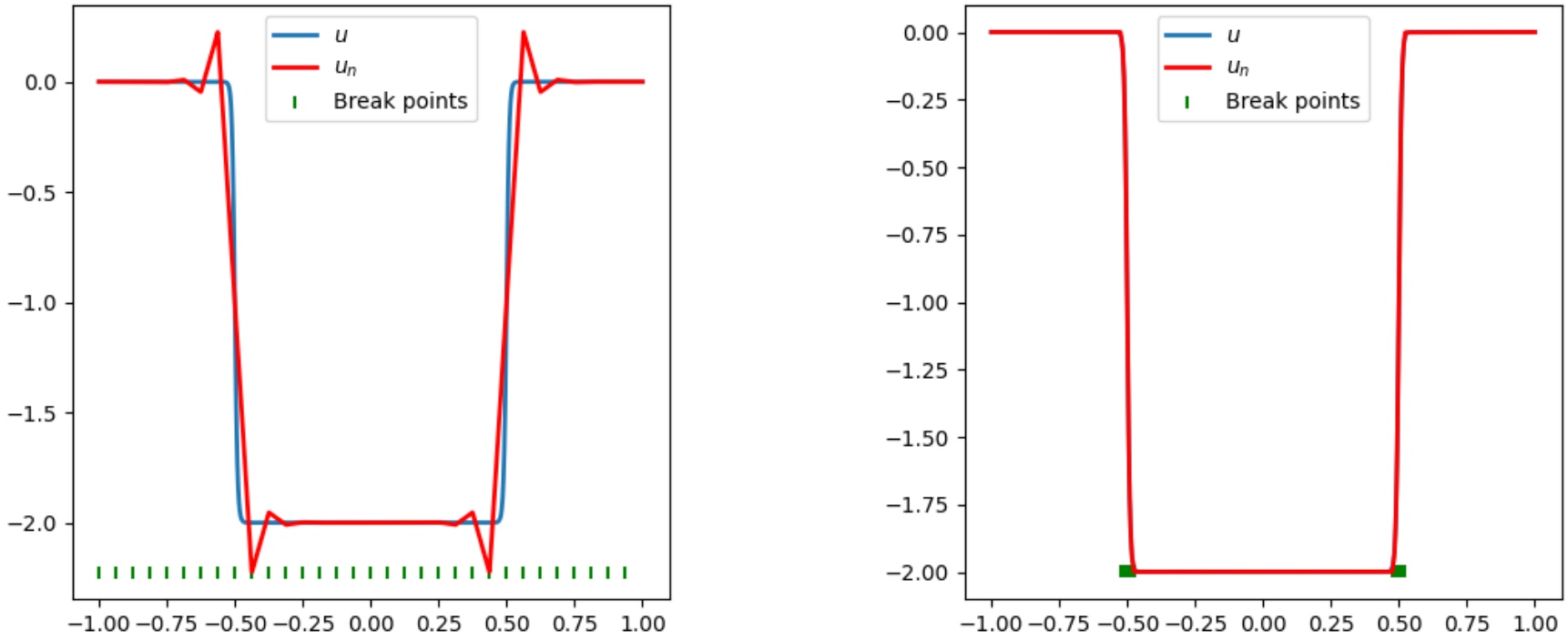


Figure: Solution $u(x)$ of a singularly perturbed reaction-diffusion equation approximated by NN. Left: 32 uniform breakpoints. Right: optimized NN model with 32 breakpoints, 500 iterations of the dBN method.

Table of Contents

1D Diffusion Problem

Consider the one-dimensional Poisson equation

$$\begin{cases} -(a(x)u'(x))' = f(x), & x \in I = (0,1), \\ u(0) = \alpha, \quad u(1) = \beta \end{cases}$$

Ritz formulation: find $u \in H^1(I)$ such that

$$u = \underset{\substack{v \in H^1(I) \\ v(0)=\alpha, v(1)=\beta}}{\arg\min} \left\{ \frac{1}{2} \int_0^1 a(x)(v'(x))^2 dx - \int_0^1 f(x)v(x)dx \right\}$$

Modified Ritz Formulation

Given $\gamma > 0$, let $J : H^1(I) \to \mathbb{R}$ be the modified energy functional given by

$$J(v) = \frac{1}{2}\int_0^1 a(x)(v'(x))^2 dx - \int_0^1 f(x)v(x)dx + \frac{\gamma}{2}(v(b) - \beta)^2$$

Let

$$\mathcal{M}_n(I) \quad = \quad \left\{ c_{-1} + \sum_{i=0}^{n} c_i \sigma(x - b_i) \; : \; c_i \in \mathbb{R}, 0 \leq b_i \leq 1, b_i < b_{i+1} \right\}$$

Ritz neural network approximation: find $u_n(x) \in \mathcal{M}_n(I)$ such that

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

Error Estimate

Proposition

Let u be the exact solution of the diffusion problem and $u_n \in \mathcal{M}_n(I)$ be the Ritz neural network approximation. Assume that $a \in L^\infty(I)$, then there exists a constant C depending on u such that

$$\|u - u_n\|_a \leq C \left(n^{-1} + \gamma^{-1/2} \right),$$

where $\|v\|_a^2 = \int_0^1 a(x)(v'(x))^2 dx + \gamma(v(1))^2$.

Table of Contents

Systems of Algebraic Equations

Let

$$u_n = u_n(x) = u_n(x; \mathbf{c}, \mathbf{b}) = \alpha + \sum_{i=0}^{n} c_i \sigma(x - b_i)$$

be a solution of the previous minimization problem. Then the linear and nonlinear parameters

$$\mathbf{c} = (c_0, \ldots, c_n)^T \quad \text{and} \quad \mathbf{b} = (b_0, \ldots, b_n)^T$$

satisfy the following system of algebraic equations

$$\nabla_{\mathbf{c}} J(u_n) = \mathbf{0} \quad \text{and} \quad \nabla_{\mathbf{b}} J(u_n) = \mathbf{0}.$$

Linear Parameters $\mathbf{c}$

The equation $\nabla_{\mathbf{c}} J(u_n) = 0$ has the form

$$\left(A(\mathbf{b}) + \gamma \mathbf{d}\mathbf{d}^T\right)\mathbf{c} = \mathbf{f}(\mathbf{b}) + \gamma(\beta - \alpha)\mathbf{d},$$

where

- $A(\mathbf{b}) = \int_0^1 a(x)\nabla_{\mathbf{c}} u_n'(x)\left(\nabla_{\mathbf{c}} u_n'(x)\right)^T dx$
  - i.e, $(A(\mathbf{b}))_{ij} = \int_0^1 a(x)\sigma'(x - b_{i-1})\sigma'(x - b_{j-1})dx$
- $\mathbf{f}(\mathbf{b}) = \int_0^1 f(x)\nabla_{\mathbf{c}} u_n(x)dx$
- $\mathbf{d} = (b - b_0, \ldots, b - b_n)^T$

Coefficient Matrix

$$A(\mathbf{b}) = \begin{pmatrix} \int_{b_0}^{1} a(x)dx & \int_{b_1}^{1} a(x)dx & \int_{b_2}^{1} a(x)dx & \cdots & \int_{b_n}^{1} a(x)dx \\ \int_{b_1}^{1} a(x)dx & \int_{b_1}^{1} a(x)dx & \int_{b_2}^{1} a(x)dx & \cdots & \int_{b_n}^{1} a(x)dx \\ \int_{b_2}^{1} a(x)dx & \int_{b_2}^{1} a(x)dx & \int_{b_2}^{1} a(x)dx & \cdots & \int_{b_n}^{1} a(x)dx \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \int_{b_n}^{1} a(x)dx & \int_{b_n}^{1} a(x)dx & \int_{b_n}^{1} a(x)dx & \cdots & \int_{b_n}^{1} a(x)dx \end{pmatrix},$$

particularly, when $a(x) = 1$

$$A(\mathbf{b}) = \begin{pmatrix} 1-b_0 & 1-b_1 & 1-b_2 & \cdots & 1-b_n \\ 1-b_1 & 1-b_1 & 1-b_2 & \cdots & 1-b_n \\ 1-b_2 & 1-b_2 & 1-b_2 & \cdots & 1-b_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1-b_n & 1-b_n & 1-b_n & \cdots & 1-b_n \end{pmatrix}$$

Condition Number

Lemma

Let $a(x) = 1$, then the condition number of the coefficient matrix $A(\mathbf{b})$ is bounded above by $\mathcal{O}\left(n/h_{\min}\right)$.

Inverse of the Coefficient Matrix (Uniform Partition)

For the uniform mesh and $a(x) = 1$, the inverse of the coefficient matrix is given by

$$A(\mathbf{b})^{-1} = \begin{pmatrix} \frac{1}{n+1} & -\frac{1}{n+1} & 0 & 0 & \cdots & 0 & 0 \\ -\frac{1}{n+1} & \frac{2}{n+1} & -\frac{1}{n+1} & 0 & \cdots & 0 & 0 \\ 0 & -\frac{1}{n+1} & \frac{2}{n+1} & -\frac{1}{n+1} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{2}{n+1} & -\frac{1}{n+1} \\ 0 & 0 & 0 & 0 & \cdots & -\frac{1}{n+1} & \frac{2}{n+1} \end{pmatrix}$$

Inverse of the Coefficient Matrix

Lemma

The coefficient matrix is invertible and its inverse is given by

$$A(\mathbf{b})^{-1} = \begin{pmatrix} \frac{1}{s_1} & -\frac{1}{s_1} & 0 & 0 & \cdots & 0 & 0 \\ -\frac{1}{s_1} & \frac{1}{s_1}+\frac{1}{s_2} & -\frac{1}{s_2} & 0 & \cdots & 0 & 0 \\ 0 & -\frac{1}{s_2} & \frac{1}{s_2}+\frac{1}{s_3} & -\frac{1}{s_3} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{s_{n-1}}+\frac{1}{s_n} & -\frac{1}{s_n} \\ 0 & 0 & 0 & 0 & \cdots & -\frac{1}{s_n} & \frac{1}{s_n}+\frac{1}{s_{n+1}} \end{pmatrix},$$

where $s_i := \int_{b_{i-1}}^{b_i} a(x)dx$.

Nonlinear Parameters **b**

Lemma

For $j = 0, 1, \ldots, n$, let

$$g(b_j) = f(c_j) + a'(b_j)\left(\sum_{i=0}^{j-1} c_i + \frac{c_j}{2}\right).$$

Then the Hessian matrix $\nabla^2_{\mathbf{b}} J(u_n)$ has the form

$$\mathcal{H}(\mathbf{c}, \mathbf{b}) = \mathbf{B}(\mathbf{c}, \mathbf{b}) + \gamma \mathbf{c}\mathbf{c}^T,$$

where $\mathbf{B}(\mathbf{c}, \mathbf{b}) := \text{diag}(-c_0 g(b_0), \ldots, -c_n g(b_n))$

Table of Contents

A Damped Block Newton (dBN) Method

We want to solve

$$\nabla_{\mathbf{c}} J(u_n) = \mathbf{0} \quad \text{and} \quad \nabla_{\mathbf{b}} J(u_n) = \mathbf{0}.$$

The equation $\nabla_{\mathbf{c}} J(u_n) = 0$ has the form

$$\left(A(\mathbf{b}) + \gamma \mathbf{d}\mathbf{d}^T \right) \mathbf{c} = \mathbf{f}(\mathbf{b}) + \gamma(\beta - \alpha)\mathbf{d},$$

The Hessian matrix $\nabla^2_{\mathbf{b}} J(u_n)$ has the form

$$\mathcal{H}(\mathbf{c}, \mathbf{b}) = \mathbf{B}(\mathbf{c}, \mathbf{b}) + \gamma \mathbf{c}\mathbf{c}^T,$$

where $\mathbf{B}(\mathbf{c}, \mathbf{b})$ is a diagonal matrix.

The Sherman-Morrison Formula

Let $\mathcal{A}(\mathbf{b}) = A(\mathbf{b}) + \gamma \mathbf{d}\mathbf{d}^T$, by the Sherman-Morrison formula, we have

$$\mathcal{A}(\mathbf{b})^{-1} = A(\mathbf{b})^{-1} - \frac{\gamma A(\mathbf{b})^{-1}\mathbf{d}\mathbf{d}^T A(\mathbf{b})^{-1}}{1 + \gamma \mathbf{d}^T A(\mathbf{b})^{-1}\mathbf{d}}.$$

Lemma

If $c_i g(b_i) \neq 0$ for all $i = 0, 1, \ldots, n$ and $1 - \gamma \mathbf{c}^T \mathbf{B}^{-1}(\mathbf{c}, \mathbf{b})\mathbf{c} \neq 0$, then the Hessian matrix $\mathcal{H}(\mathbf{c}, \mathbf{b})$ is invertible. Moreover, its inverse is given by

$$\mathcal{H}^{-1}(\mathbf{c}, \mathbf{b}) = -\left[\mathbf{B}^{-1}(\mathbf{c}, \mathbf{b}) + \frac{\gamma \mathbf{B}^{-1}(\mathbf{c}, \mathbf{b})\mathbf{c}\mathbf{c}^T \mathbf{B}^{-1}(\mathbf{c}, \mathbf{b})}{1 - \gamma \mathbf{c}^T \mathbf{B}^{-1}(\mathbf{c}, \mathbf{b})\mathbf{c}}\right].$$

A Damped Block Newton (dBN) Method

Let $\left(\mathbf{c}^{(k)}, \mathbf{b}^{(k)}\right)$ be the previous iterate. We then compute the current state $\left(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k+1)}\right)$ by doing the following:

(i) Compute the linear parameters $\mathbf{c}^{(k+1)}$ by

$$\mathbf{c}^{(k+1)} = \mathcal{A}(\mathbf{b}^{(k)})^{-1}\left\{\mathbf{f}(\mathbf{b}^{(k)}) + \gamma(\beta - \alpha)\mathbf{d}(\mathbf{b}^{(k)})\right\}.$$

(ii) Assume that the Hessian matrix $\mathcal{H}(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k)})$ is invertible. Set the search direction

$$\mathbf{p}^{(k)} = -\mathcal{H}(\mathbf{c}^{(k+1)}, \mathbf{b}^{(k)})^{-1}\nabla_{\mathbf{b}} J(u_n(x; \mathbf{c}^{(k+1)}, \mathbf{b}^{(k)})).$$

(iii) Compute the stepsize η_k

$$\eta_k = \underset{\eta \in \mathbb{R}_+}{\operatorname{argmin}}\ J(u_n(x; \mathbf{c}^{(k+1)}, \mathbf{b}^{(k)} + \eta\mathbf{p}^{(k)})).$$

Set the current nonlinear parameters by

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + \eta_k \mathbf{p}^{(k)}.$$

Some Remarks

- If $\mathbf{c}_i^{(k+1)} g(\mathbf{b}_i^{(k)})$ vanishes for some $i \in \{0, \ldots, n\}$, then the corresponding nonlinear parameter will remain unchanged, i.e., $\mathbf{b}_i^{(k+1)} = \mathbf{b}_i^{(k)}$, and be removed at the step (ii) of the method.
- The computational cost per iteration is $\mathcal{O}(n)$. More specifically, the linear parameters $\mathbf{c}^{(k+1)}$ and the direction vector $\mathbf{p}^{(k)}$ are calculated in $8(n+1)$ and $4(n+1)$ operations, respectively.

Numerical Experiments

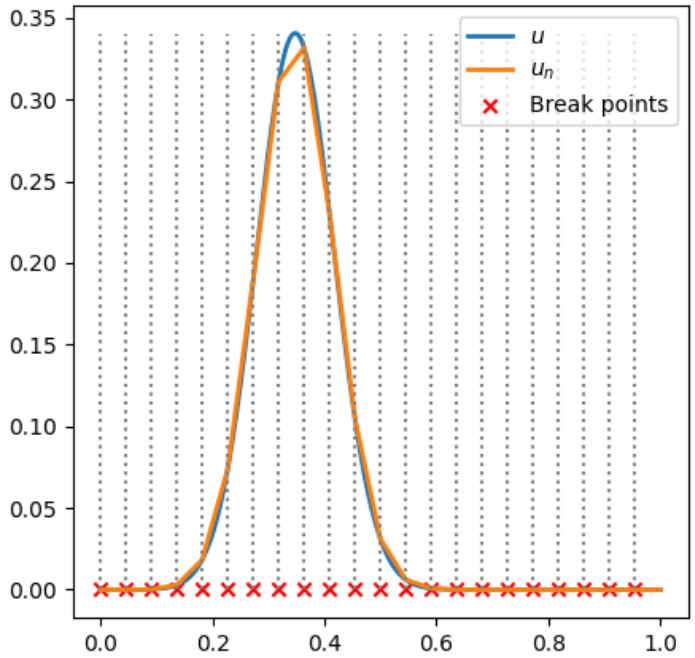
Example 1

$$u(x) = x\left(\exp\left(-\frac{(x-\frac{1}{3})^2}{0.01}\right) - \exp\left(-\frac{4}{9\times 0.01}\right)\right)$$

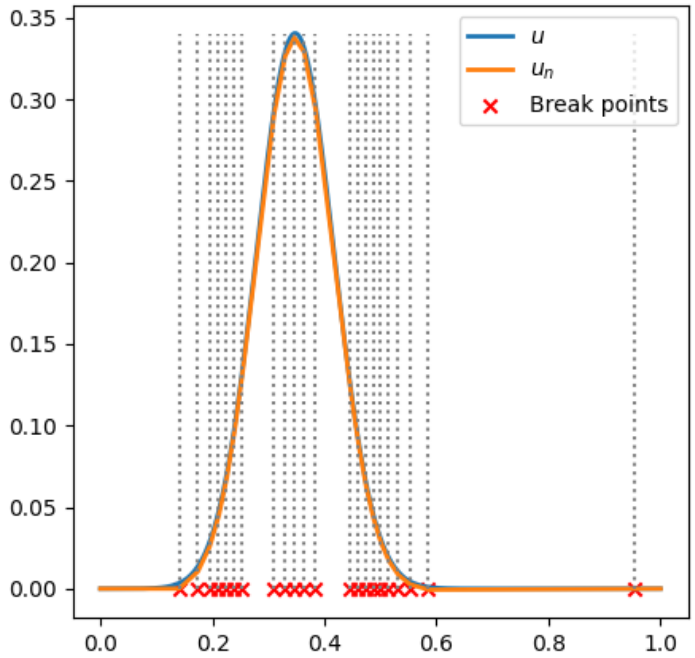
Consider the relative H^1 seminorm error given by

$$e_n = \frac{|u-u_n|_{H^1(I)}}{|u|_{H^1(I)}}$$

Numerical Experiments



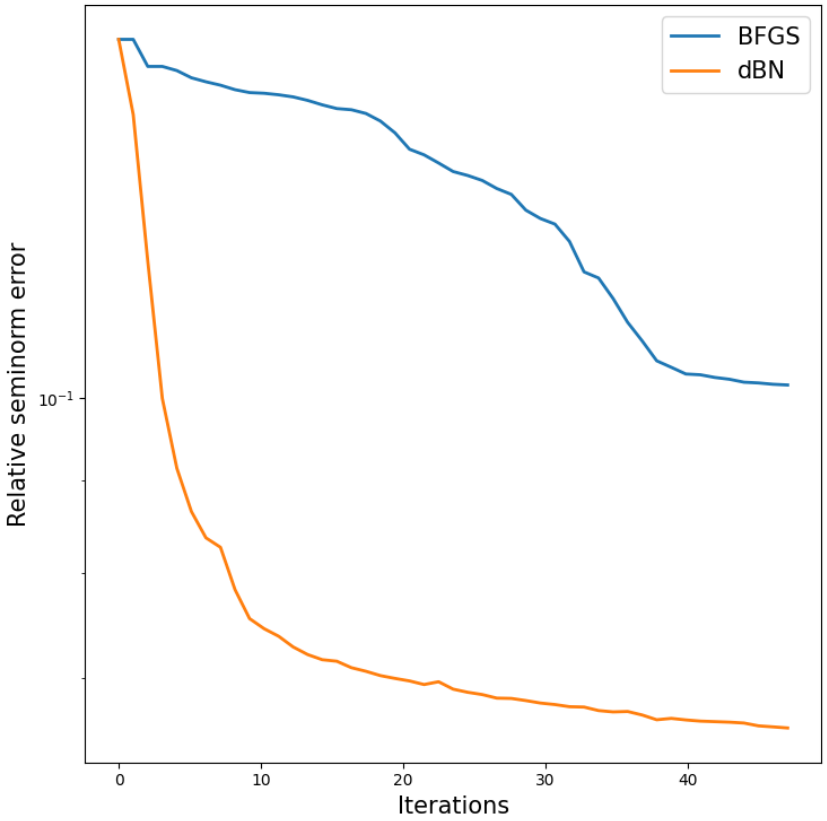
(a) Initial NN model with 22 uniform breakpoints, $e_n = 0.227$



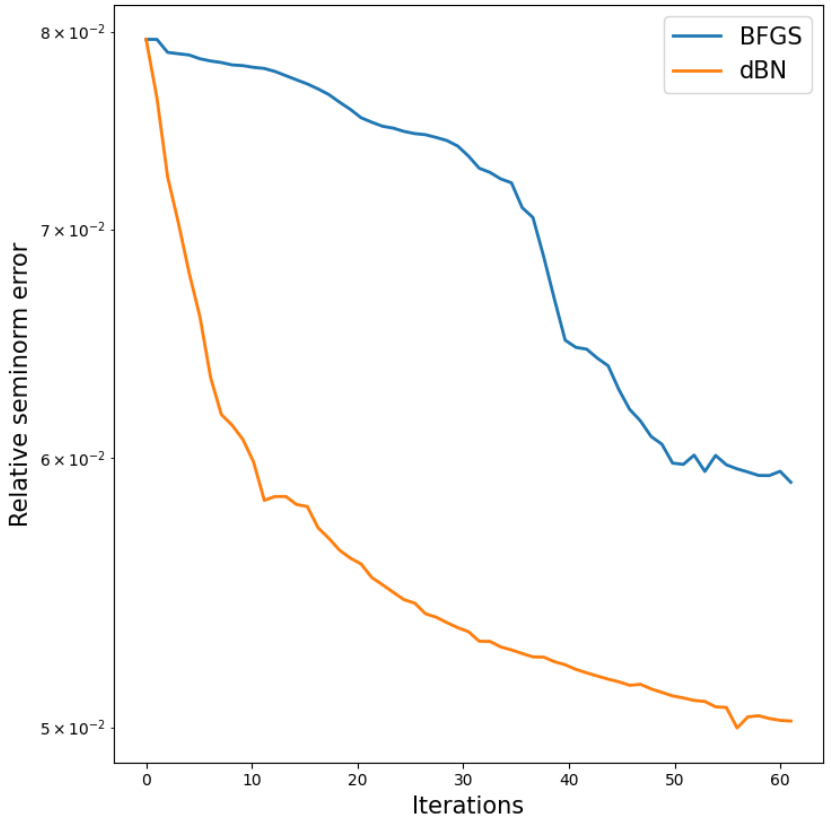
(b) Optimized NN model with 22 breakpoints, 500 iterations, $e_n = 0.100$

Figure: Using ReLU networks for approximating the function in Example 1

Numerical Experiments: dBN vs BFGS



(a) e_n vs number of iterations using 32 neurons, the ratio between the final errors is 0.673



(b) e_n vs number of iterations using 64 neurons, the ratio between the final errors is 0.783

Order of Convergence

n	e_n	r
60	4.65×10^{-2}	0.749
90	3.41×10^{-2}	0.751
120	2.49×10^{-2}	0.771
150	1.97×10^{-2}	0.783
180	1.78×10^{-2}	0.775
210	1.59×10^{-2}	0.775
240	1.27×10^{-2}	0.796
270	1.22×10^{-2}	0.787
300	1.09×10^{-2}	0.792
330	1.01×10^{-2}	0.792
360	9.94×10^{-3}	0.783
390	9.54×10^{-3}	0.780
420	8.07×10^{-3}	0.798

Table: Relative errors e_n and powers r for n neurons after 1000 iterations.

Table of Contents

Adaptivity

The true error $\|u_n - u\|_{a,K}$ on some interval $K \subseteq I$ is estimated by the ZZ-estimator:

$$\xi_K = \|a^{-1/2}\left(G(au_n') - au_n'\right)\|_{L^2(K)},$$

where $G(au_n')$ is the projection of au_n' onto the space of the continuous piecewise linear functions. For a given $u_n \in \mathcal{M}_n(I)$ with the breakpoints

$$0 = b_{-1} < b_0 < \ldots < b_n < b_{n+1} = 1,$$

let $\mathcal{K}^i = [b_{i-1}, b_i]$, then $\mathcal{K}_n = \left\{\mathcal{K}^i\right\}_{i=0}^{n+1}$ is a partition of the interval $[0, 1]$.

Adaptivity

Then, we define a subset $\widehat{\mathcal{K}_n} \subset \mathcal{K}_n$ using the average marking strategy[4][5]:

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

where $\#\mathcal{K}_n$ is the number of elements in $\mathcal{K}_n$. For a refinement step, a meshpoint gets added at the midpoint within each element of $\widehat{\mathcal{K}_n}$.

[4] M. Liu, Z. Cai, and J. Chen. "Adaptive two-layer ReLU neural network: I. Best least-squares approximation". In: *Computers & Mathematics with Applications* 113 (May 2022), pp. 34–44. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2022.03.005.

[5] M. Liu and Z. Cai. "Adaptive two-layer ReLU neural network: II. Ritz approximation to elliptic PDEs". In: *Computers & Mathematics with Applications* 113 (2022), pp. 103–116. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2022.03.010.

Adaptive Damped Block Newton (AdBN) method

Given a tolerance ϵ, start with a small number of neurons n_0, then

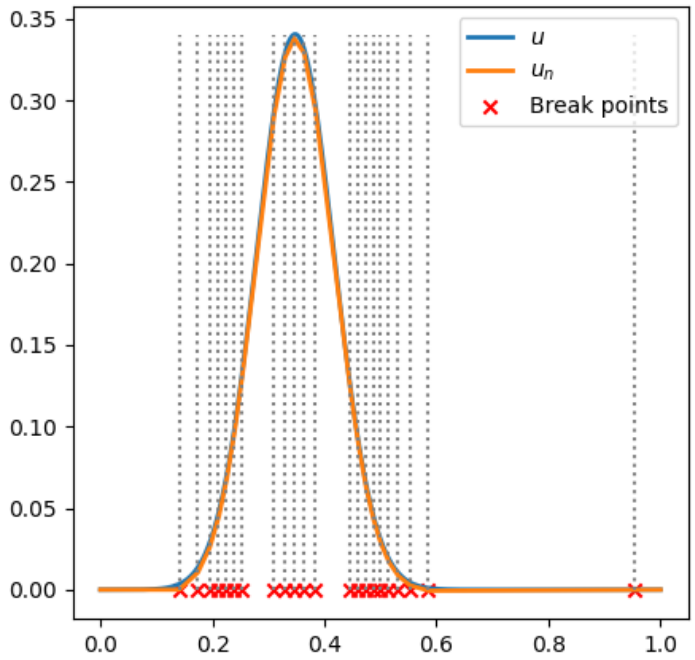
(i) Compute the solution u_n using dBN

(ii) Estimate the total error by computing $\xi = \left(\sum_{K \in \mathcal{K}} \xi_K^2 \right)^{1/2} / |u_n|_{H^1(I)}$

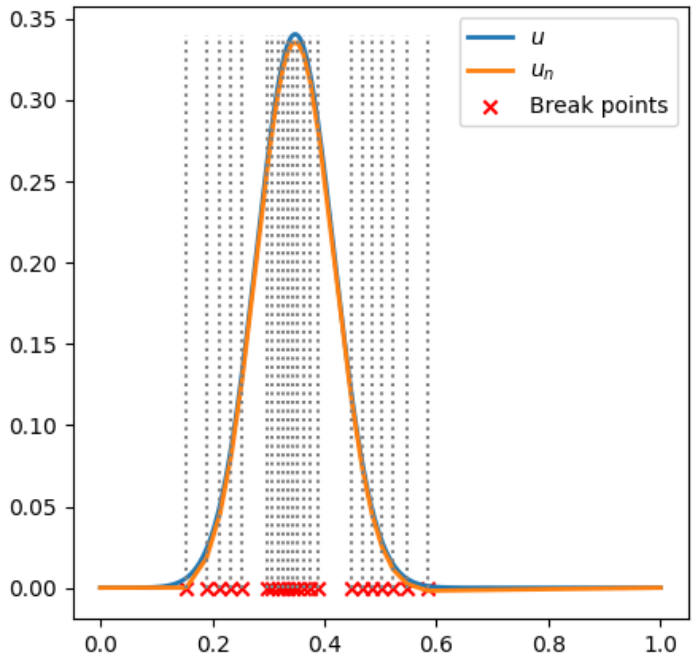
(iii) If $\xi \leq \epsilon$, then stop; otherwise go to step (iv)

(iv) Add new neurons to the network by using the network enhancement strategy, then go to step (i)

Numerical Experiments: dBN vs AdBN



(a) Optimized NN model with 22 breakpoints, 500 iterations, $e_n = 0.100$



(b) Adaptive NN model with 22 breakpoints: 10 initial breakpoints, 2 refinements (13, 22 neurons), $e_n = 0.083$

Figure: Using ReLU networks for approximating the function in Example 1

Numerical Experiments: dBN vs AdBN

NN (n neurons)	e_n	ξ_n	r
Adaptive (19)	1.04×10^{-1}	0.149	0.768
Adaptive (27)	6.55×10^{-2}	0.089	0.827
Adaptive (32)	5.71×10^{-2}	0.075	0.826
Adaptive (45)	4.21×10^{-2}	0.054	0.832
Adaptive (68)	2.76×10^{-2}	0.032	0.851
Adaptive (94)	2.01×10^{-2}	0.023	0.860
Adaptive (137)	1.39×10^{-2}	0.015	0.869
Adaptive (187)	1.04×10^{-2}	0.011	0.873
Adaptive (280)	6.93×10^{-3}	0.007	0.882
Fixed (187)	1.87×10^{-2}	0.020	0.761
Fixed (280)	1.12×10^{-2}	0.013	0.781

Table: Comparison of an adaptive network with fixed networks for relative error e_n, relative error estimators ξ_n, and rates r

Numerical Experiments: Non-smooth Solution

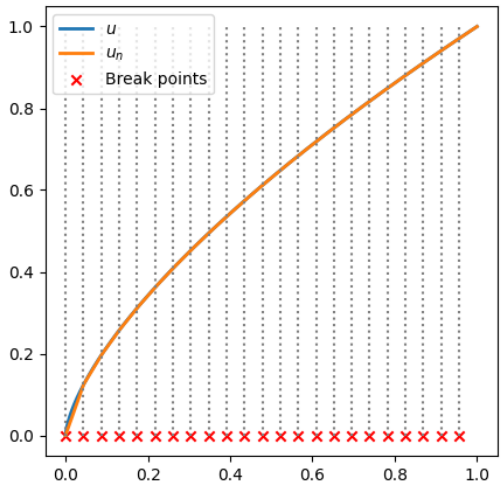
Example 2

$$u(x) = x^{2/3}$$

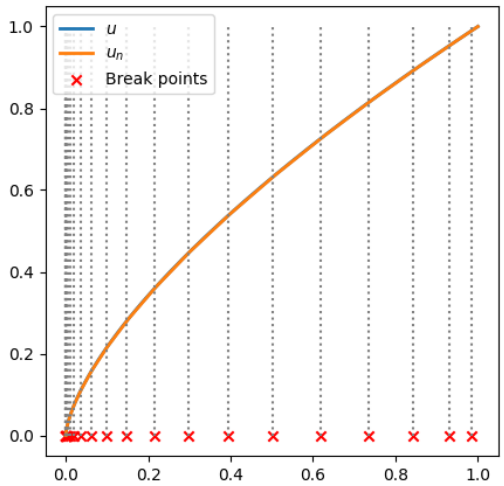
This function belongs to $H^{1+\frac{1}{6}-\epsilon}(I)$ for any $\epsilon > 0$. We highlight that the order of convergence for approximating this solution with n uniform breakpoints is at most $\mathcal{O}\left(n^{-1/6}\right)$.

Numerical Experiments: Non-smooth Solution

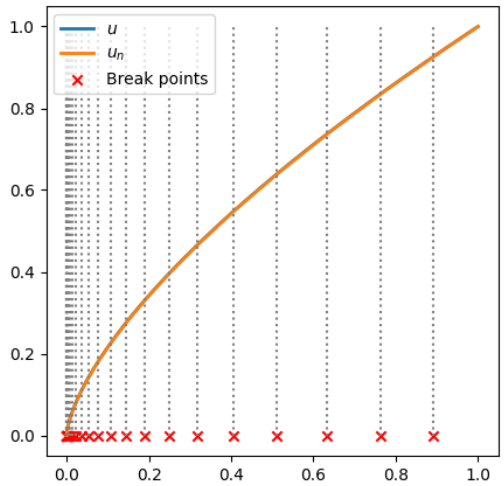
Figure: Results of using ReLU networks for approximating $u(x) = x^{2/3}$



(a) Initial NN model with 23 uniform breakpoints, $e_n = 0.284$



(b) Optimized NN model with 23 breakpoints, 500 iterations, $e_n = 0.056$



(c) Adaptive NN model with 23 breakpoints: 9 initial break points, 2 refinements (14, 23 neurons), $e_n = 0.042$

Table of Contents

Summary

- The key points of our method:
  - Good initialization.
    - Non-linear parameters $\longleftrightarrow$ uniform mesh.
    - Linear parameters $\longleftrightarrow$ best linear parameters for the uniform mesh.
  - Coefficient matrix has a sparse, tridiagonal inverse.
  - Hessian matrix (from the non linear system) is diagonal.

Fast Iterative Solver for Neural Network Method: II. 1D General Elliptic Problems and Data Fitting

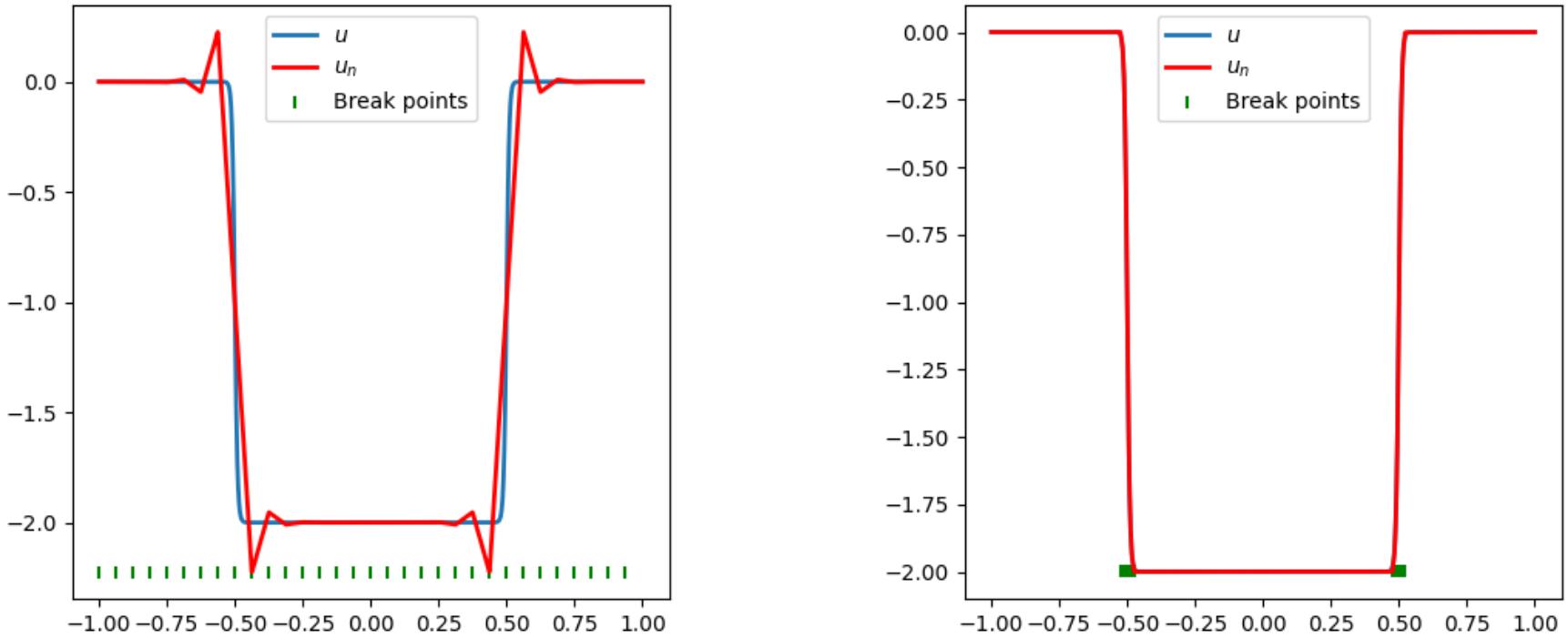


Figure: Solution $u(x)$ of a singularly perturbed reaction-diffusion equation approximated by NN. Left: 32 uniform breakpoints. Right: optimized NN model with 32 breakpoints, 500 iterations of the dBN method.

Current and future work

- Convergence analysis of our methods.
- Design a similar method for 1D elliptic problems with nonsymmetric variational formulation, such as $-u'' + u' + u = f$
- 2D elliptic problems

Thanks!