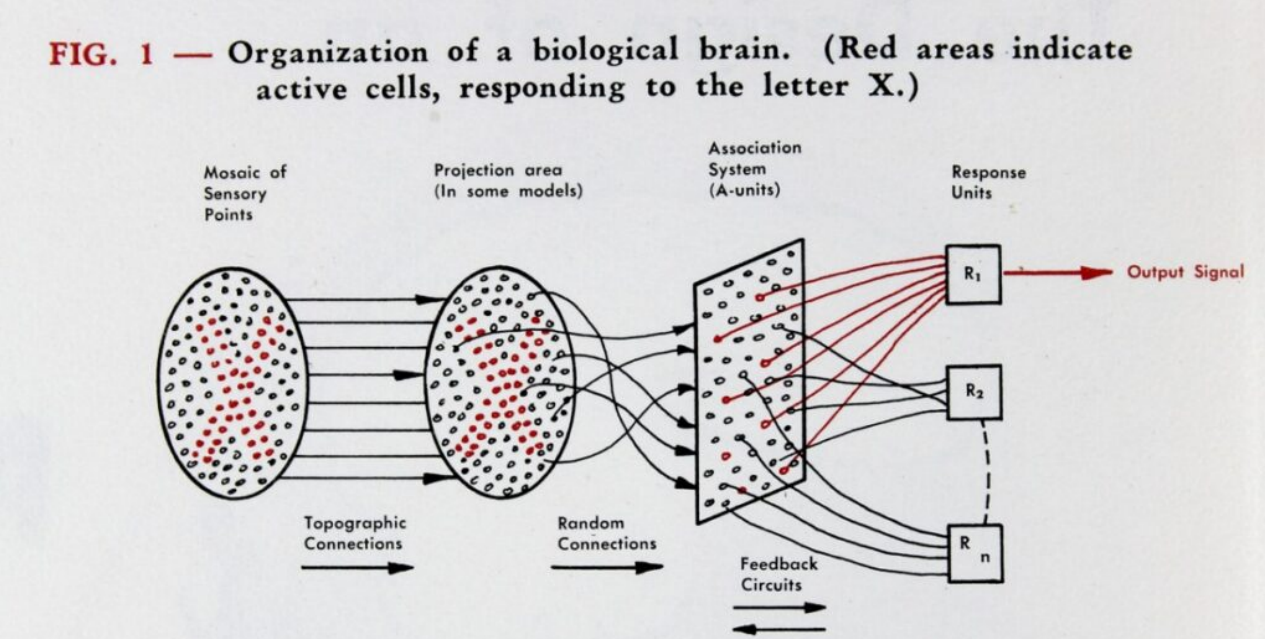
Fast Iterative Solver for Neural Network Method: I. 1D Diffusion Problems

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

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

Grad Student Research Day, Purdue University
November 2024

Introduction: Neural Networks[1]



[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 $\mathbf{x}^{(i)} = \quad W_{n_i \;\; (n_i \;\; _1+1)}^{(i)} \quad \begin{matrix} 1 \\ \mathbf{x}^{(i \;\; 1)} \end{matrix}$ for $i = 1;:::;L \quad 1.$

Output: $u(\mathbf{x}) = W_{c \;\; (n_L \;\; _1+1)}^{(L)} \quad \begin{matrix} 1 \\ \mathbf{x}^{(L \;\; 1)} \end{matrix}$

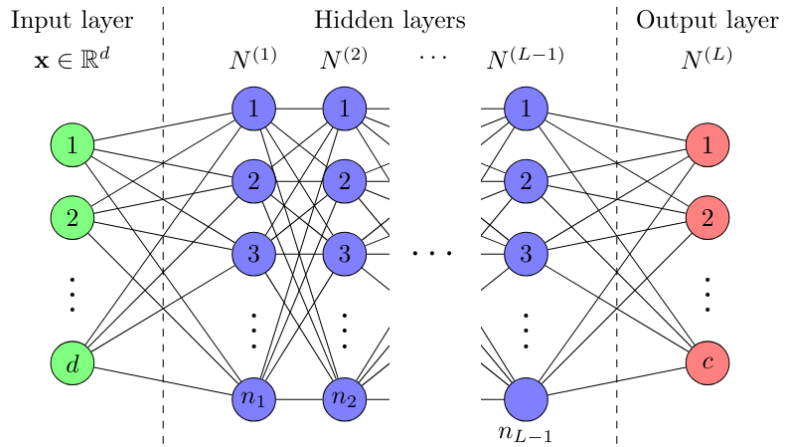


Figure: The neural network function structure[2]

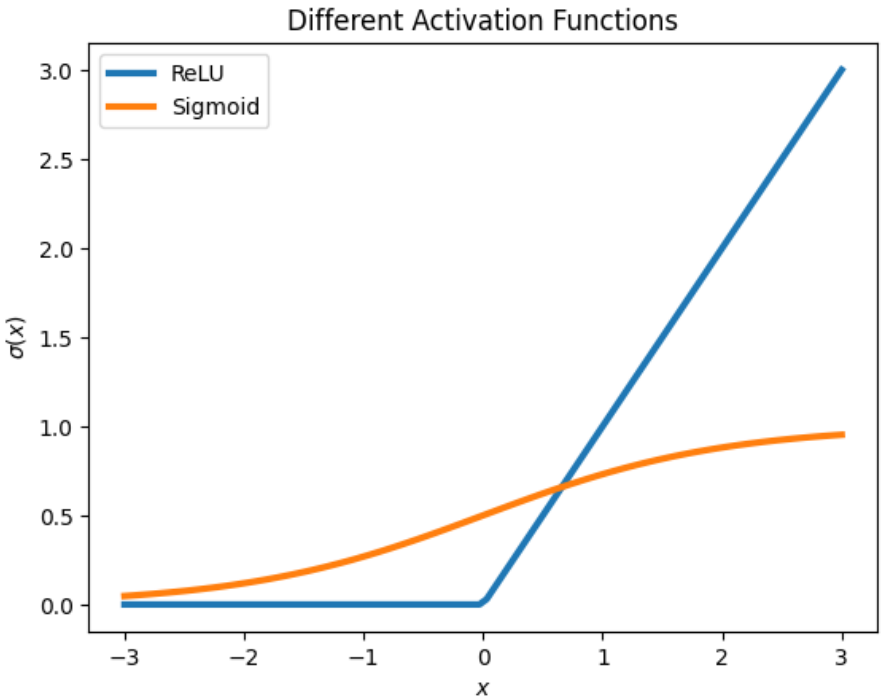
[2] Z. Cai, J. Choi, and M. Liu. *Least-Squares Neural Network (LSNN) Method For Linear Advection-Reaction Equation: Discontinuity Interface*. 2024. arXiv: 2301.06156 [math.NA].

Activation Functions and Shallow Neural Network

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

$$M_n(\ ; d) = c_{\ 1} + \sum_{i=0} c_i\ (!_i\ \mathbf{x}\ b_i) : c_i; b_i\ 2\ \mathrm{R};\ !_i\ 2\ S^{d\ \ 1}$$

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



A New Class of Approximating Functions

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

$M(\ ;d) = fv(\mathbf{x})\ 2\ M_n(\ ;d) : n\ 2\ Z_+ g$ is dense in $C(K)$ for any compact set K R^d, provided that $2\ C(\mathrm{R})$ 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** $[0;1]$**:**

$$S_1^0(\Delta) = \left(\sum_{i=1} c_i \ _i(x) : c_i \ 2 \ \mathrm{R} \right) ;$$

where

$$_i(x) = \begin{cases} (x \quad x_i \ _1)=(x_i \quad x_i \ _1); & x \ 2 \ (x_i \ _1; x_i); \\ (x_{i+1} \quad x)=(x_{i+1} \quad x_i); & x \ 2 \ (x_i; x_{i+1}); \\ 0; & \text{otherwise.} \end{cases}$$

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

$$S_1^0(n) = \left(\sum_{i=1} c_i \ _i(x; x_i \ _1; x_i; x_{i+1}) : c_i \ 2 \ \mathrm{R}; ; x_i \ 2 \ [0;1] \right)$$

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

$$S_1^0(\Delta) = \left(\sum_{i=1} c_i \ _i(x) : c_i \ 2 \ \mathrm{R} \right) ;$$

where

$$_i(x) = \begin{cases} (x \quad x_i \ _1)=(x_i \quad x_i \ _1); & x \ 2 \ (x_i \ _1; x_i); \\ (x_{i+1} \quad x)=(x_{i+1} \quad x_i); & x \ 2 \ (x_i; x_{i+1}); \\ 0; & \text{otherwise.} \end{cases}$$

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

$$S_1^0(n) = \left(\sum_{i=1} c_i \ _i(x; x_i \ _1; x_i; x_{i+1}) : c_i \ 2 \ \mathrm{R}; ; x_i \ 2 \ [0; 1] \right)$$

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

$$M_n([0;1]) = M_n(I) = \left(c_1 + \sum_{i=0}^{n} c_i \, (x - b_i) : c_i \in \mathbb{R}; b_i \in [0;1] \right)$$

- **Important inclusion[3]:**

$$S_1^0(n) \quad M_n(I) \quad S_1^0(n+1)$$

[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.

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

$$M_n([0;1]) = M_n(I) = \left(c_1 + \sum_{i=0}^{n} c_i \, (x - b_i) : c_i \in \mathbb{R}; b_i \in [0;1] \right)$$

- **Important inclusion[3]:**

$$S_1^0(n) \quad M_n(I) \quad S_1^0(n+1)$$

[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.

Goals

Solve PDEs using NNs

- Design fast NN methods
- Design suitable NN architectures
- Utilize geometric interpretation of parameters in the NN

Why Neural Networks?

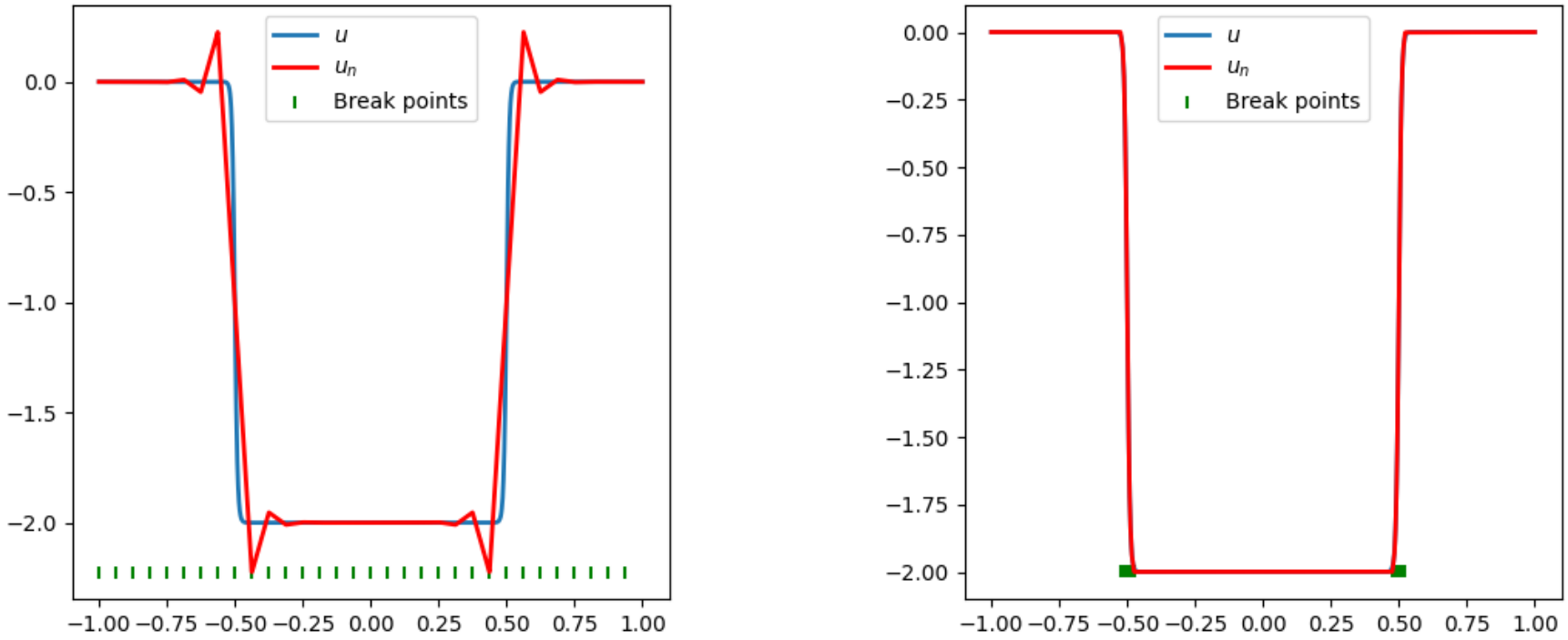


Figure: Solution $u(x)$ of a singularly perturbed reaction-diffusion equation approximated by NN. Left: 32 uniform breakpoints. Right: optimized NN model with 32 breakpoints, 500 iterations of the dBN method.

Table of Contents

1D Diffusion Problem

Consider the one-dimensional Poisson equation

$$\begin{cases} (a(x)u'(x))' = f(x); \quad x \in I = (0;1); \\ u(0) = \quad ; \quad u(1) = \end{cases}$$

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

$$u = \underset{\substack{v \in H^1(I) \\ v(0)= \ ;v(1)=}}{\arg\min} \frac{1}{2} \int_0^1 a(x)(v'(x))^2 dx \int_0^1 f(x)v(x)dx$$

Modified Ritz Formulation

Given $\quad > 0$, let $J : H^1(I) \; ! \; \mathrm{R}$ be the modified energy functional given by

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

Let

$$M_n(I) = \left(c_1 + \sum_{i=0}^{x} c_i \; (x \quad b_i) : c_i \; 2 \; \mathrm{R}; 0 \quad b_i \quad 1; b_i < b_{i+1} \right)$$

Ritz neural network approximation: find $u_n(x) \; 2 \; M_n(I)$ such that

$$J(u_n) = \min_{\substack{v \, 2 \, M_n(I) \\ v(0) =}} J(v)$$

Modified Ritz Formulation

Given $\quad > 0$, let $J : H^1(I) \; ! \; \mathrm{R}$ be the modified energy functional given by

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

Let

$$M_n(I) = \left(c_{\;1} + \sum_{i=0}^{\;x} c_i \; (x \quad b_i) : c_i \; 2 \; \mathrm{R}; 0 \quad b_i \quad 1; b_i < b_{i+1} \right)$$

Ritz neural network approximation: find $u_n(x) \; 2 \; M_n(I)$ such that

$$J(u_n) = \min_{\substack{v \, 2 \, M_n(I) \\ v(0) =}} J(v)$$

Error Estimate

Proposition

Let u be the exact solution of the diffusion problem and $u_n \ 2 \ M_n(I)$ be the Ritz neural network approximation. Assume that $a \ 2 \ L^{\ 1}(I)$, then there exists a constant C depending on u such that

$$ku \quad u_n k_a \quad C \quad n^{\ 1} + \quad ^{1=2} \quad ;$$

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

Systems of Algebraic Equations

Let

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

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

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

Linear Parameters $\mathbf{c}$

The equation $r_{\mathbf{c}} J(\mathbf{c}; \mathbf{b}) = 0$ has the form

$$A(\mathbf{b}) + \quad \mathbf{d}\mathbf{d}^T \quad \mathbf{c} = \mathbf{f}(\mathbf{b}) + \quad (\qquad)\mathbf{d};$$

where

- $A(\mathbf{b}) = \int_0^1 a(x)\, r_{\mathbf{c}} u_n^0(x) \quad r_{\mathbf{c}} u_n^0(x)^T dx$
  - i.e, $(A(\mathbf{b}))_{ij} = \int_0^1 a(x) \ '(x \quad b_{i-1}) \ '(x \quad b_{j-1}) dx$
- $\mathbf{f}(\mathbf{b}) = \int_0^1 f(x)\, r_{\mathbf{c}} u_n(x) dx$
- $\mathbf{d} = (b \quad b_0; : : : ; b \quad b_n)^T$

Coefficient Matrix

$$A(\mathbf{b}) = \begin{pmatrix} 0 & \int_1^{b_0} a(x)dx & \int_1^{b_1} a(x)dx & \int_1^{b_2} a(x)dx & \cdots & \int_1^{b_n} a(x)dx & 1 \\ \int_1^{b_0} a(x)dx & \int_1^{b_1} a(x)dx & \int_1^{b_1} a(x)dx & \int_1^{b_2} a(x)dx & & \int_1^{b_n} a(x)dx & \int_1^{b_n} a(x)dx \\ \int_1^{b_1} a(x)dx & \int_{b_2}^{b_1} a(x)dx & \int_{b_2}^{b_1} a(x)dx & \int_{b_2}^{b_2} a(x)dx & & \int_{b_n}^{b_n} a(x)dx & \int_{b_n}^{b_n} a(x)dx \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \int_1^{b_n} a(x)dx & \int_{b_n}^{b_n} a(x)dx & \int_{b_n}^{b_n} a(x)dx & \int_{b_n}^{b_n} a(x)dx & & \int_{b_n}^{b_n} a(x)dx & \int_{b_n}^{b_n} a(x)dx \end{pmatrix};$$

particularly, when $a(x) = 1$

$$A(\mathbf{b}) = \begin{pmatrix} 0 & 1 & b_0 & 1 & b_1 & 1 & b_2 & \cdots & 1 & b_n & 1 \\ 1 & b_0 & 1 & b_1 & 1 & b_2 & & & 1 & b_n & \\ 1 & b_1 & 1 & b_1 & 1 & b_2 & & & 1 & b_n & \\ 1 & b_2 & 1 & b_2 & 1 & b_2 & & & 1 & b_n & \\ \vdots & & \vdots & & \vdots & & \ddots & & \vdots & & \\ 1 & b_n & 1 & b_n & 1 & b_n & & & 1 & b_n & \end{pmatrix}$$

Condition Number

Lemma

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

Inverse of the Coefficient Matrix (Uniform Partition)

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

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

Inverse of the Coefficient Matrix

Lemma

The coefficient matrix is invertible and its inverse is given by

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

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

Nonlinear Parameters **b**

Lemma

For $j = 0; 1; : : : ; n$, let

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

Then the Hessian matrix $r^2_{\mathbf{b}} J(\mathbf{c}; \mathbf{b})$ has the form

$$H(\mathbf{c}; \mathbf{b}) = \mathbf{B}(\mathbf{c}; \mathbf{b}) + \quad \mathbf{c}\mathbf{c}^T;$$

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

Table of Contents

A Damped Block Newton (dBN) Method

We want to solve

$$r_{\mathbf{c}} J(\mathbf{c}; \mathbf{b}) = \mathbf{0} \quad \text{and} \quad r_{\mathbf{b}} J(\mathbf{c}; \mathbf{b}) = \mathbf{0};$$

The equation $r_{\mathbf{c}} J(\mathbf{c}; \mathbf{b}) = 0$ has the form

$$A(\mathbf{b}) + \quad \mathbf{d}\mathbf{d}^T \quad \mathbf{c} = \mathbf{f}(\mathbf{b}) + \quad (\qquad)\mathbf{d};$$

The Hessian matrix $r_{\mathbf{b}}^2 J(\mathbf{c}; \mathbf{b})$ has the form

$$H(\mathbf{c}; \mathbf{b}) = \mathbf{B}(\mathbf{c}; \mathbf{b}) + \quad \mathbf{c}\mathbf{c}^T;$$

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

The Sherman-Morrison Formula

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

$$A(\mathbf{b})^{\ 1} = A(\mathbf{b})^{\ 1} \quad \frac{A(\mathbf{b})^{\ 1}\mathbf{d}\mathbf{d}^T A(\mathbf{b})^{\ 1}}{1 + \quad \mathbf{d}^T A(\mathbf{b})^{\ 1}\mathbf{d}}:$$

Lemma

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

$$H^{\ 1}(\mathbf{c}; \mathbf{b}) = \quad \mathbf{B}^{\ 1}(\mathbf{c}; \mathbf{b}) + \frac{\mathbf{B}^{\ 1}(\mathbf{c}; \mathbf{b})\mathbf{c}\mathbf{c}^T \mathbf{B}^{\ 1}(\mathbf{c}; \mathbf{b})}{1 \quad \mathbf{c}^T \mathbf{B}^{\ 1}(\mathbf{c}; \mathbf{b})\mathbf{c}} \quad :$$

The Sherman-Morrison Formula

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

$$A(\mathbf{b})^{\ 1} = A(\mathbf{b})^{\ 1} \quad \frac{A(\mathbf{b})^{\ 1}\mathbf{d}\mathbf{d}^T A(\mathbf{b})^{\ 1}}{1 + \quad \mathbf{d}^T A(\mathbf{b})^{\ 1}\mathbf{d}}:$$

Lemma

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

$$H^{\ 1}(\mathbf{c}; \mathbf{b}) = \quad \mathbf{B}^{\ 1}(\mathbf{c}; \mathbf{b}) + \frac{\mathbf{B}^{\ 1}(\mathbf{c}; \mathbf{b})\mathbf{c}\mathbf{c}^T \mathbf{B}^{\ 1}(\mathbf{c}; \mathbf{b})}{1 \quad \mathbf{c}^T \mathbf{B}^{\ 1}(\mathbf{c}; \mathbf{b})\mathbf{c}} \quad :$$

A Damped Block Newton (dBN) Method

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

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

$$r_{\mathbf{c}} J(\mathbf{c}; \mathbf{b}^{(k)}) = 0;$$

(ii) Set the search direction

$$\mathbf{p}^{(k)} = \quad H(\mathbf{c}^{(k+1)}; \mathbf{b}^{(k)}) \quad ^{1} r_{\mathbf{b}} J(\mathbf{c}^{(k+1)}; \mathbf{b}^{(k)});$$

(iii) Compute the stepsize $_{k}$

$$_{k} = \underset{2\mathrm{R}_{+}}{\arg\min} J(\mathbf{c}^{(k+1)}; \mathbf{b}^{(k)} + \quad \mathbf{p}^{(k)});$$

Set the current nonlinear parameters by

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + \quad _{k}\mathbf{p}^{(k)};$$

A Damped Block Newton (dBN) Method

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

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

$$r_{\mathbf{c}} J(\mathbf{c}; \mathbf{b}^{(k)}) = 0;$$

(ii) Set the search direction

$$\mathbf{p}^{(k)} = \quad H(\mathbf{c}^{(k+1)}; \mathbf{b}^{(k)}) \quad ^{1} r_{\mathbf{b}} J(\mathbf{c}^{(k+1)}; \mathbf{b}^{(k)});$$

(iii) Compute the stepsize $_{k}$

$$_{k} = \operatorname*{argmin}_{2\mathrm{R}_{+}} J(\mathbf{c}^{(k+1)}; \mathbf{b}^{(k)} + \quad \mathbf{p}^{(k)});$$

Set the current nonlinear parameters by

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + \quad _{k}\mathbf{p}^{(k)};$$

A Damped Block Newton (dBN) Method

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

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

$$r_{\mathbf{c}} J(\mathbf{c}; \mathbf{b}^{(k)}) = 0.$$

(ii) Set the search direction

$$\mathbf{p}^{(k)} = \quad H(\mathbf{c}^{(k+1)}; \mathbf{b}^{(k)}) \quad ^{1} r_{\mathbf{b}} J(\mathbf{c}^{(k+1)}; \mathbf{b}^{(k)}).$$

(iii) Compute the stepsize $\quad_k$

$$_k = \operatorname*{argmin}_{2\mathrm{R}_+} J(\mathbf{c}^{(k+1)}; \mathbf{b}^{(k)} + \quad \mathbf{p}^{(k)}).$$

Set the current nonlinear parameters by

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

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 $O(n)$. 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.

Table of Contents

Numerical Experiments

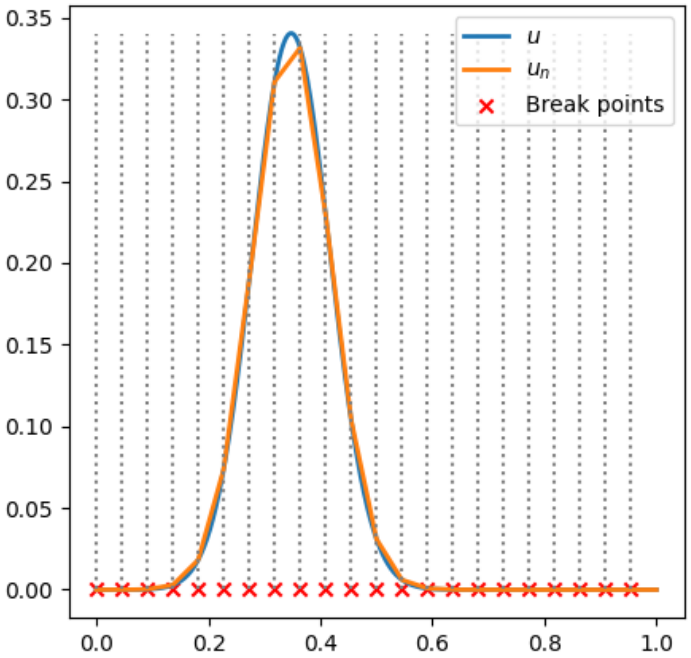
Example 1

$$u(x) = x \quad \exp\left(\frac{(x \quad \frac{1}{3})^2}{0{:}01}\right) \quad \exp\left(\frac{4}{9 \quad 0{:}01}\right)$$

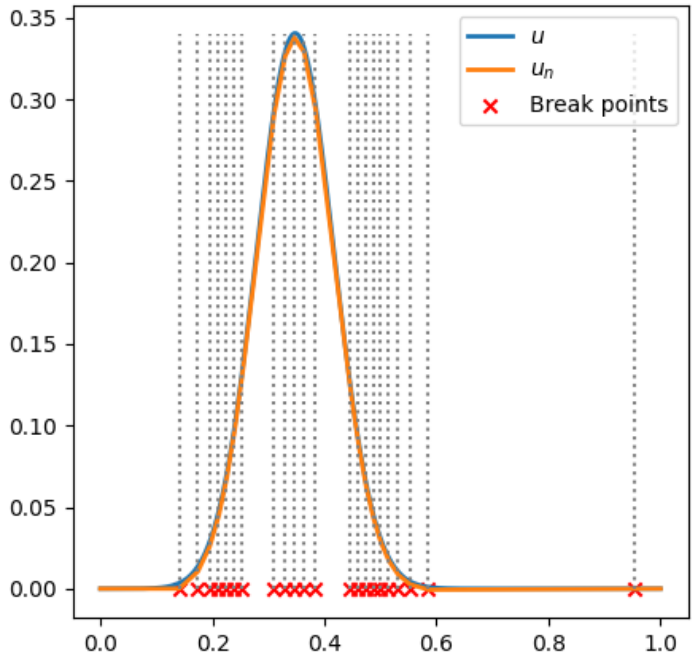
Consider the relative H^1 seminorm error given by

$$e_n = \frac{ju \quad u_n j_{H^1(I)}}{ju j_{H^1(I)}}$$

Numerical Experiments



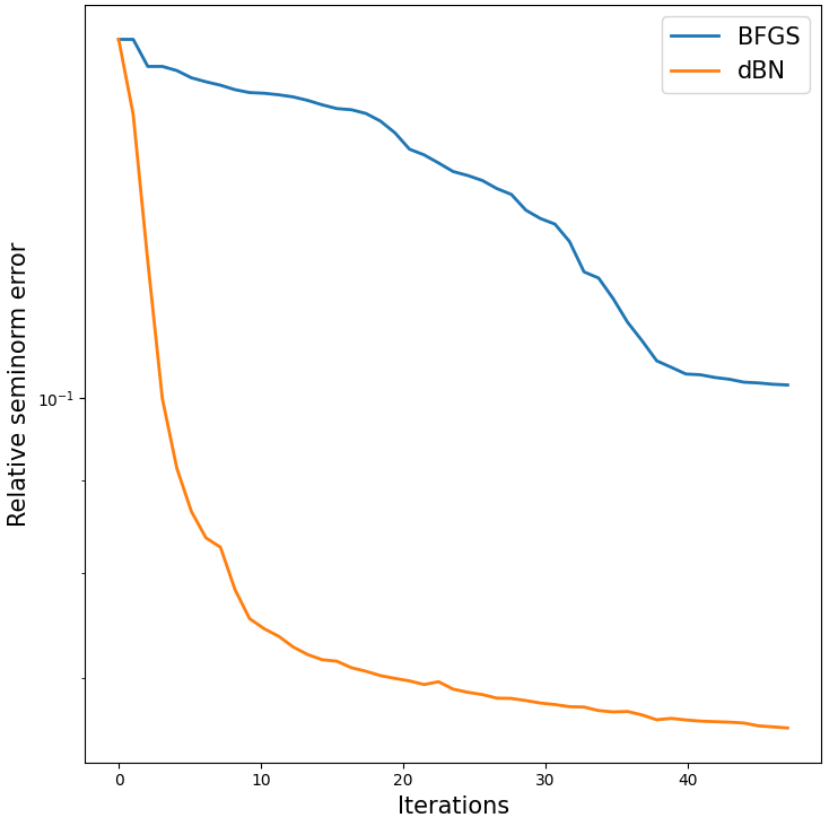
(a) Initial NN model with 22 uniform breakpoints, $e_n = 0.227$



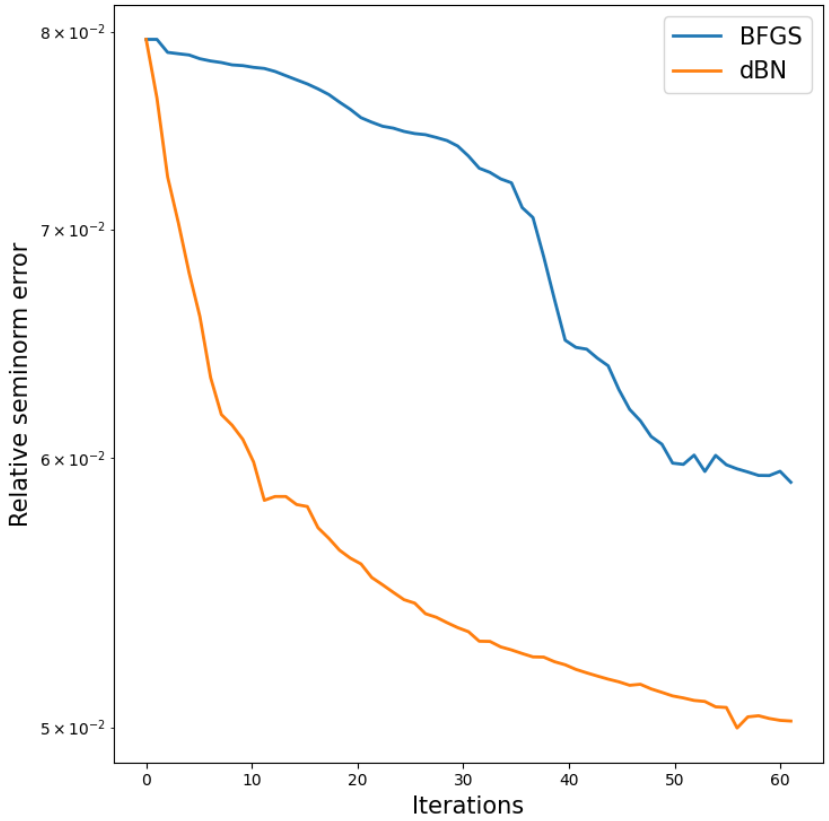
(b) Optimized NN model with 22 breakpoints, 500 iterations, $e_n = 0.100$

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

Numerical Experiments: Non-smooth Solution

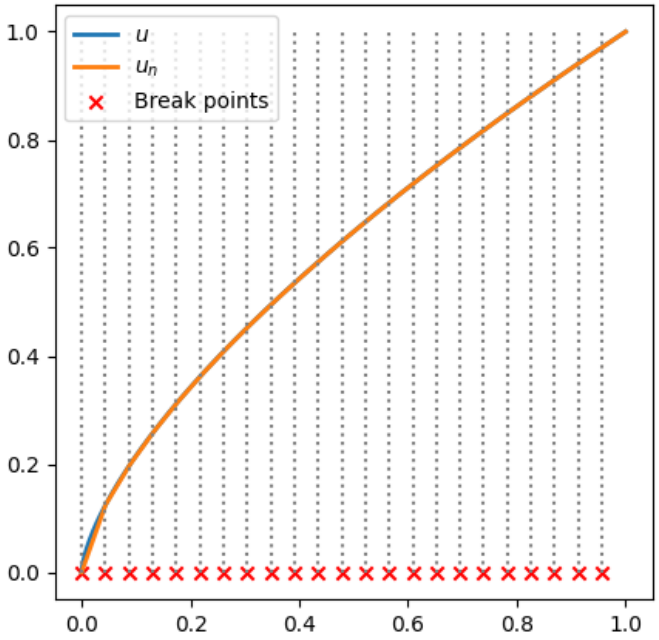
Example 2

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

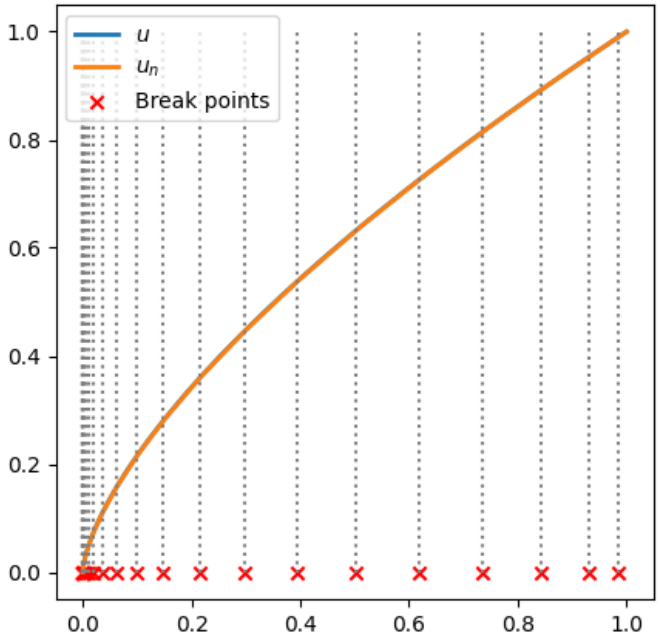
This function belongs to $H^{1+\frac{1}{6}}$ (I) for any > 0. We highlight that the order of convergence for approximating this solution with n uniform breakpoints is at most $O\ n^{\ 1=6}$.

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

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 parameters) is diagonal.

arXiv Preprint

Mathematics > Numerical Analysis

[Submitted on 27 Apr 2024]

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

Zhiqiang Cai, Anastassia Doktorova, Robert D. Falgout, César Herrera

The discretization of the deep Ritz method [18] for the Poisson equation leads to a high-dimensional non-convex minimization problem, that is difficult and expensive to solve numerically. In this paper, we consider the shallow Ritz approximation to one-dimensional diffusion problems and introduce an effective and efficient iterative method, a damped block Newton (dBN) method, for solving the resulting non-convex minimization problem.

The method employs the block Gauss-Seidel method as an outer iteration by dividing the parameters of a shallow neural network into the linear parameters (the weights and bias of the output layer) and the non-linear parameters (the weights and bias of the hidden layer). Per each outer iteration, the linear and the non-linear parameters are updated by exact inversion and one step of a damped Newton method, respectively. Inverses of the coefficient matrix and the Hessian matrix are tridiagonal and diagonal, respectively, and hence the cost of each dBN iteration is $\mathcal{O}(n)$. To move the breakpoints (the non-linear parameters) more efficiently, we propose an adaptive damped block Newton (AdBN) method by combining the dBN with the adaptive neuron enhancement (ANE) method [25]. Numerical examples demonstrate the ability of dBN and AdBN not only to move the breakpoints quickly and efficiently but also to achieve a nearly optimal order of convergence for AdBN. These iterative solvers are capable of outperforming BFGS for select examples.

Subjects: **Numerical Analysis (math.NA)**

MSC classes: 65N99

Cite as: arXiv:2404.17750 **[math.NA]**

(or arXiv:2404.17750v1 **[math.NA]** for this version)

https://doi.org/10.48550/arXiv.2404.17750

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

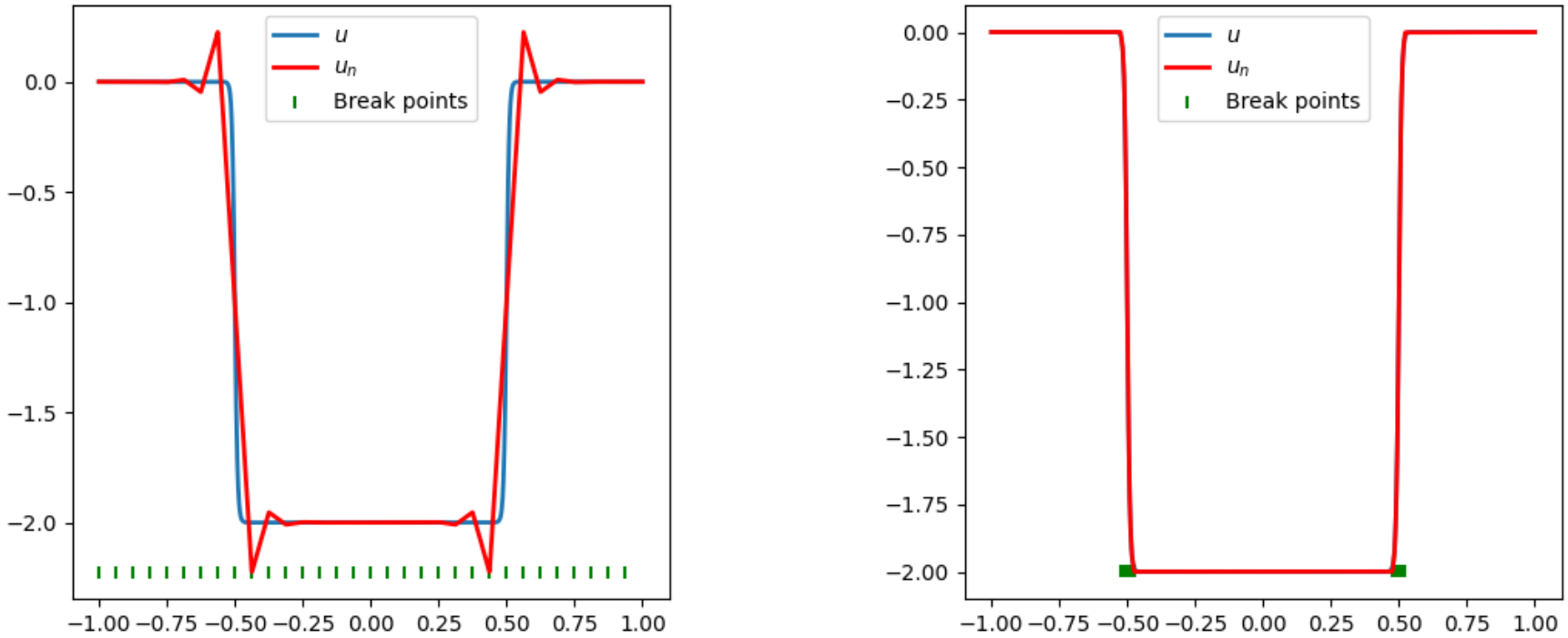
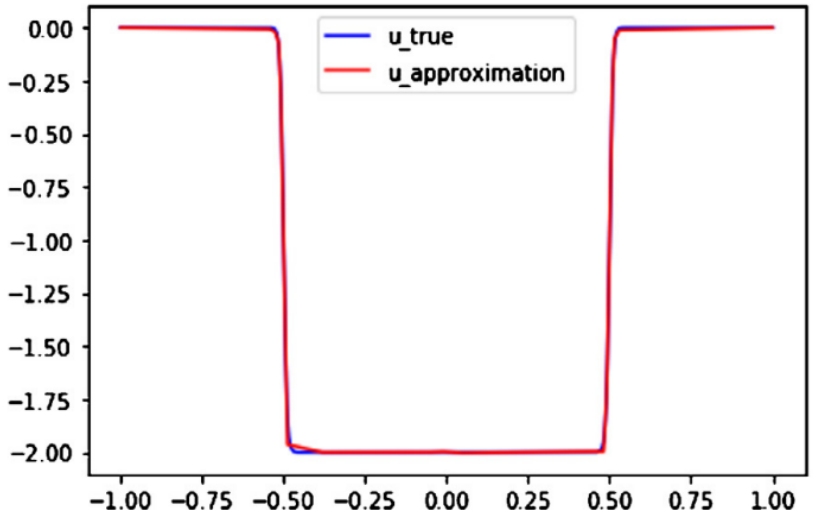


Figure: Solution $u(x)$ of a singularly perturbed reaction-diffusion equation approximated by NN. Left: 32 uniform breakpoints. Right: optimized NN model with 32 breakpoints, 500 iterations of the dBN method.

Singularly Perturbed Reaction-Diffusion Equation



(a) FOSLS u with Leaky ReLU

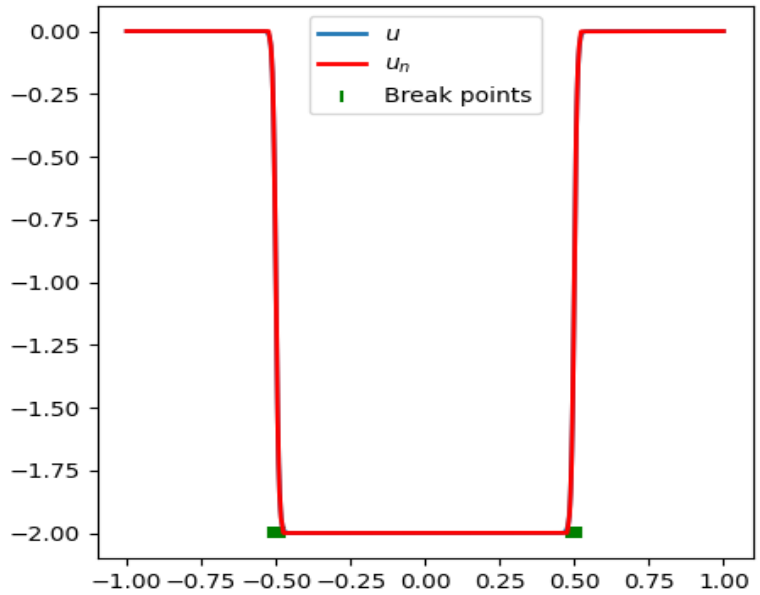


Figure: $u(x)$ approximated by NN. Left: Scientific Machine Learning approach[4],1-32-32-24-24-1, 2962 parameters, about 14 hours. Right: dBN method, 1-32-1, 64 parameters, 2 minutes (100 iterations).

[4] Zhiqiang Cai et al. \Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs". In: *Journal of Computational Physics* 420 (2020), p. 109707. issn: 0021-9991.

arXiv Preprint

Mathematics > Numerical Analysis

[Submitted on 1 Jul 2024]

Fast Iterative Solver For Neural Network Method: II. 1D Diffusion-Reaction Problems And Data Fitting

Zhiqiang Cai, Anastassia Doktorova, Robert D. Falgout, César Herrera

This paper expands the damped block Newton (dBN) method introduced recently in [4] for 1D diffusion-reaction equations and least-squares data fitting problems. To determine the linear parameters (the weights and bias of the output layer) of the neural network (NN), the dBN method requires solving systems of linear equations involving the mass matrix. While the mass matrix for local hat basis functions is tri-diagonal and well-conditioned, the mass matrix for NNs is dense and ill-conditioned. For example, the condition number of the NN mass matrix for quasi-uniform meshes is at least $\mathcal{O}(n^4)$. We present a factorization of the mass matrix that enables solving the systems of linear equations in $\mathcal{O}(n)$ operations. To determine the non-linear parameters (the weights and bias of the hidden layer), one step of a damped Newton method is employed at each iteration. A Gauss-Newton method is used in place of Newton for the instances in which the Hessian matrices are singular. This modified dBN is referred to as dBGN. For both methods, the computational cost per iteration is $\mathcal{O}(n)$. Numerical results demonstrate the ability dBN and dBGN to efficiently achieve accurate results and outperform BFGS for select examples.

Subjects: **Numerical Analysis (math.NA)**; Machine Learning (cs.LG)
MSC classes: 65K10, 65F05
Cite as: arXiv:2407.01496 **[math.NA]**
(or arXiv:2407.01496v1 **[math.NA]** for this version)
https://doi.org/10.48550/arXiv.2407.01496

Current and future work

- Convergence analysis of our methods.
- Design a similar method for 1D elliptic problems with nonsymmetric variational formulation, such as $u'' + u' + u = f$
- Design fast iterative solvers for multi-dimensional problems
  - 2D elliptic problems
  - Advection-reaction equation
  - Hyperbolic conservation laws

Thanks!