

## EFFICIENT SHALLOW RITZ METHOD FOR ONE-DIMENSIONAL DIFFUSION-REACTION PROBLEMS\*

ZHIQIANG CAI<sup>†</sup>, ANASTASSIA DOKTOROVA<sup>†</sup>, ROBERT D. FALGOUT<sup>‡</sup>,  
AND CÉSAR HERRERA<sup>†</sup>

**Abstract.** This paper studies the shallow Ritz method for solving one-dimensional diffusion-reaction problems. The method is capable of improving the order of approximation for nonsmooth problems. By following an approach similar to the one presented in [Z. Cai et al., *Efficient Shallow Ritz Method for 1D Diffusion Problems*, <https://arxiv.org/abs/2404.17750>, 2024], we present a damped block Newton (dBN) method to achieve nearly optimal order of approximation. The dBN method optimizes the Ritz functional by alternating between the linear and nonlinear parameters of the shallow ReLU neural network. For diffusion-reaction problems, new difficulties arise: (1) for the linear parameters, the mass matrix is dense and even more ill-conditioned than the stiffness matrix, and (2) for the nonlinear parameters, the Hessian matrix is dense and may be singular. This paper addresses these challenges, resulting in a dBN method with computational cost of  $\mathcal{O}(n)$ . The ideas presented for diffusion-reaction problems can also be applied to least-squares approximation problems. For both applications, starting with the nonlinear parameters as a uniform partition, numerical experiments show that the dBN method moves the mesh points to nearly optimal locations.

**Key words.** fast iterative solvers, neural network, Ritz formulation, ReLU activation, diffusion-reaction problems, least-squares approximation, Newton’s method

**MSC codes.** 65K10, 65F05

**DOI.** 10.1137/24M1672894

**1. Introduction.** Using neural networks (NNs) to solve partial differential equations (PDEs) has recently gained traction in scientific computing (see, e.g., [3, 6, 12, 13, 23, 26]). In one dimension, the shallow ReLU NN generates a class of approximating functions equivalent to free knot splines (FKS). FKS can significantly improve the order of approximation for nonsmooth functions and hence reduce the number of degrees of freedom dramatically (see [4] and discussion in [9]). However, determining the optimal knot locations (the nonlinear parameters of a shallow ReLU NN) leads to a complicated, computationally intensive nonconvex optimization problem. Moreover, the ReLU activation function induces dense and ill-conditioned algebraic systems.

For the one-dimensional diffusion problem, to realize optimal or nearly optimal order of the shallow Ritz approximation, we developed in [9] a damped block Newton (dBN) method that can efficiently move the uniformly distributed mesh points to nearly optimal locations. Due to the physical meaning of the parameters of the output and hidden layers, the dBN method employs the commonly used outer-inner iterative method by alternating between updates of the linear and nonlinear parameters (see, e.g., [16, 21, 25] since the 1970s and [1, 2, 11, 22] recently in the context of

---

\*Received by the editors July 1, 2024; accepted for publication (in revised form) July 25, 2025; published electronically October 14, 2025.

<https://doi.org/10.1137/24M1672894>

**Funding:** This work was supported in part by the National Science Foundation under grant DMS-2110571 and by the US Department of Energy under B665416. This work was performed under the auspices of the US Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-JRNL-865920).

<sup>†</sup>Department of Mathematics, Purdue University, West Lafayette, IN 47907 USA (caiz@purdue.edu, adoktoro@purdue.edu, herre125@purdue.edu).

<sup>‡</sup>Lawrence Livermore National Laboratory, Livermore, CA 94551 USA (rfalgout@llnl.gov).

PDEs). This step is natural but resolves none of the essential difficulties mentioned above. The key contributions of the dBN method consist of (1) the exact inversion of the dense and ill-conditioned linear system and (2) one step of a damped Newton method applied to a *reduced* nonlinear system with  $\mathcal{O}(n)$  computational cost per each iteration. Additionally, derivation of a modified Newton method for the nonlinear parameters is highly nontrivial due to the facts that the ReLU activation function is not differentiable everywhere and that the corresponding Hessian matrix could be singular.

The purpose of this paper is to extend the dBN method to a broader class of problems in one dimension, while maintaining the efficiency previously achieved for diffusion problems [9]. For diffusion-reaction problems as well as least-squares approximation, the mass matrix  $M(\mathbf{b})$  arising from the NN approximation must be inverted per iteration for solving the linear parameter. Just like the stiffness matrix  $A(\mathbf{b})$  in [9],  $M(\mathbf{b})$  depends on the nonlinear parameter  $\mathbf{b}$ . However,  $M(\mathbf{b})$  is much more ill-conditioned than  $A(\mathbf{b})$ . Specifically, the condition numbers  $\kappa(A(\mathbf{b}))$  and  $\kappa(M(\mathbf{b}))$  are bounded, respectively, by  $\mathcal{O}(nh_{\min}^{-1})$  and  $\mathcal{O}(nh_{\min}^{-3})$  (see Lemma 4.3 in [9] and Lemma 4.2), where  $n$  is the number of neurons and  $h_{\min}$  is the smallest distance between two neighboring breakpoints. For the ReLU NN approximation to nonsmooth functions,  $h_{\min}$  could be much smaller than the uniform mesh size  $1/n$  by several orders. To invert the dense matrices  $M(\mathbf{b})$  and  $A(\mathbf{b}) + M(\mathbf{b})$  efficiently, we introduce their factorizations whose inversions can be computed in  $\mathcal{O}(n)$  operations (see (4.3) and (4.4)).

Along with the possible nondifferentiability of the diffusion coefficient, the nonlinear parameters for this broader class of problems present additional challenges. First, the optimality conditions for the nonlinear parameters are no longer nearly decoupled nonlinear algebraic systems (see (3.5)), which imply dense Hessian matrices. Second, the Hessians could be singular and hence the Newton method is not applicable. To overcome the second difficulty, we remove some neurons, that are either unneeded or can be fixed, to obtain a reduced nonlinear system. A neuron is not needed if its contribution to the current approximation is small, and a neuron can be fixed if it is at a nearly optimal location. Sufficient conditions for the derivation of the reduced system are introduced and guarantee the invertibility of the Hessian for the reduced nonlinear system. To invert the dense Hessian matrix of the reduced system in  $\mathcal{O}(n)$  operations, we utilize its special structure and the explicit formula for the inverse of the stiffness matrix in [9]. Hence, the overall computational cost per iteration of dBN remains  $\mathcal{O}(n)$ , consistent with the complexity established in [9].

To enhance the efficiency of moving breakpoints even further, we combine the dBN method with the adaptive neuron enhancement (ANE) method [19]. Numerical examples demonstrate the ability of the aforementioned methods to move the breakpoints quickly and efficiently and to outperform L-BFGS for select examples.

The paper is structured as follows. In section 2, we first describe the shallow Ritz discretization for diffusion-reaction problems. Section 3 presents the optimality conditions of the shallow Ritz discretization. An upper bound for the condition number of the mass matrix, along with a fast inversion strategy, is derived in section 4. The dBN method is introduced in section 5, and the adaptive version of dBN is described in subsection 5.1. The ideas presented in this paper make it possible to develop a dBN method for least-squares approximation problems, as explained in section 6. Finally, numerical results are presented in section 7.

**2. Shallow Ritz method for diffusion-reaction problems.** Consider the following diffusion-reaction equation in one dimension:

$$(2.1) \quad \begin{cases} -(a(x)u'(x))' + r(x)u(x) = f(x) & \text{in } I = (0, 1), \\ u(0) = \alpha, \quad u(1) = \beta, \end{cases}$$

where the diffusion coefficient  $a(x)$ , the reaction coefficient  $r(x)$ , and  $f(x)$  are given real-valued functions defined on  $I$ . Assume that  $a(x) \in L^\infty(I)$  and  $r(x) \in L^\infty(I)$  are bounded below by the respective positive constant  $a_0 > 0$  and nonnegative constant  $r_0 \geq 0$  almost everywhere on  $I$ .

As in [9], the modified Ritz formulation of problem (2.1) is to find  $u \in H^1(I) \cap \{u(0) = \alpha\}$  such that

$$(2.2) \quad J(u) = \min_{v \in H^1(I) \cap \{v(0) = \alpha\}} J(v),$$

where the modified energy functional is given by

$$(2.3) \quad J(v) = \frac{1}{2} \int_0^1 a(x)(v'(x))^2 dx + \frac{1}{2} \int_0^1 r(x)(v(x))^2 dx - \int_0^1 f(x)v(x) dx + \frac{\gamma}{2}(v(1) - \beta)^2.$$

Here,  $\gamma > 0$  is a penalization constant.

The set of approximating functions generated by the shallow ReLU NN with  $n$  interior breakpoints over the domain  $I$  is denoted by

$$\mathcal{M}_n(I) = \left\{ c_{-1} + c_0 x + \sum_{i=1}^n c_i \sigma(x - b_i) : c_i \in \mathbb{R}, 0 = b_0 < b_1 < \dots < b_n < b_{n+1} = 1 \right\},$$

where  $\sigma(t) = \max\{0, t\}$  is the ReLU activation function. Then, the Ritz NN approximation is to find  $u_n \in \mathcal{M}_n(I) \cap \{u_n(0) = \alpha\}$  such that

$$(2.4) \quad J(u_n) = \min_{v \in \mathcal{M}_n(I) \cap \{v(0) = \alpha\}} J(v).$$

Next, we provide an error bound for the solution  $u_n$  to (2.4). The corresponding bilinear form of the modified energy functional is given by

$$a(u, v) := \int_0^1 a(x)u'(x)v'(x) dx + \int_0^1 r(x)u(x)v(x) dx + \gamma u(1)v(1)$$

for any  $u, v \in H^1(I)$ . Denote by  $\|\cdot\|_a$  the induced norm of the bilinear form.

**PROPOSITION 2.1.** *Let  $u$  and  $u_n$  be solutions of problems (2.2) and (2.4), respectively. Then*

$$(2.5) \quad \|u - u_n\|_a \leq \sqrt{3} \inf_{v \in \mathcal{M}_n(I) \cap \{v(0) = \alpha\}} \|u - v\|_a + 2\sqrt{2} |a(1)u'(1)| \gamma^{-1/2}.$$

Moreover, if  $\mathcal{M}_n(I)$  has the approximation property

$$(2.6) \quad \inf_{v \in \mathcal{M}_n(I)} \|u - v\|_{H^1(I)} \leq C(u) n^{-1},$$

then there exists a constant  $C$  depending on  $u$  such that

$$(2.7) \quad \|u - u_n\|_a \leq C \left( n^{-1} + \gamma^{-1/2} \right).$$

*Proof.* Inequality (2.5) may be proved in a similar fashion as that of Lemma 3.1 in [9], and (2.7) is a direct consequence of (2.5) and (2.6).  $\square$

**3. Optimality conditions.** This section derives systems of algebraic equations arising from the optimality conditions of (2.4).

To this end, let

$$u_n(x) = \alpha + c_0x + \sum_{i=1}^n c_i \sigma(x - b_i)$$

be a solution of (2.4) in  $\mathcal{M}_n(I) \cap \{u_n(0) = \alpha\}$ . Denote by  $\mathbf{c} = (c_0, \dots, c_n)^T$  and  $\mathbf{b} = (b_1, \dots, b_n)^T$  the respective linear and nonlinear parameters. Then, the first-order optimality conditions yield

$$(3.1) \quad \nabla_{\mathbf{c}} J(u_n) = \mathbf{0} \quad \text{and} \quad \nabla_{\mathbf{b}} J(u_n) = \mathbf{0},$$

where  $\nabla_{\mathbf{c}}$  and  $\nabla_{\mathbf{b}}$  denote the gradients with respect to  $\mathbf{c}$  and  $\mathbf{b}$ , respectively.

Let

$$H(t) = \begin{cases} 1, & t > 0, \\ \frac{1}{2}, & t = 0, \\ 0, & t < 0, \end{cases} \quad \text{and} \quad \delta(t) = \begin{cases} +\infty, & t = 0, \\ 0, & t \neq 0, \end{cases}$$

be the Heaviside (unit) step function and the Dirac delta function, respectively. Note that  $H(t) = \sigma'(t)$  everywhere except at  $t = 0$ , and similarly,  $\delta(t) = \sigma''(t)$ . Consider the mass matrix associated with the weight function  $\omega$  given by

$$(3.2) \quad M_{\omega}(\mathbf{b}) = (m_{ij}) \quad \text{with} \quad m_{ij} = \int_0^1 \omega(x) \sigma(x - b_{i-1}) \sigma(x - b_{j-1}) dx$$

for  $i, j = 1, \dots, n+1$ , and the stiffness matrix associated with  $\omega$  given by

$$A_{\omega}(\mathbf{b}) = (a_{ij}) \quad \text{with} \quad a_{ij} = \int_0^1 \omega(x) H(x - b_{i-k}) H(x - b_{j-k}) dx$$

for  $i, j = 1, \dots, n+k$ , where either  $k = 0$  or  $k = 1$  depending on the context.

Denote the right-hand-side vector by

$$\mathbf{f}(\mathbf{b}) = (f_i) \quad \text{with} \quad f_i = \int_0^1 (f(x) - \alpha r(x)) \sigma(x - b_{i-1}) dx$$

for  $i = 1, \dots, n+1$  and let  $\mathbf{d} = \nabla_{\mathbf{c}} u_n(1)$ . By a similar derivation as in [9], the first equation in (3.1) becomes

$$(3.3) \quad \mathcal{A}(\mathbf{b}) \mathbf{c} = \mathcal{F}(\mathbf{b}),$$

where

$$\mathcal{A}(\mathbf{b}) = A_a(\mathbf{b}) + M_r(\mathbf{b}) + \gamma \mathbf{d} \mathbf{d}^T \quad \text{and} \quad \mathcal{F}(\mathbf{b}) = \mathbf{f}(\mathbf{b}) + \gamma(\beta - \alpha) \mathbf{d}.$$

Comparing to (4.2) in [9], the additional term  $M_r(\mathbf{b}) \mathbf{c}$  in (3.3) results from the reaction term.

Next, to derive the second equation in (3.1), for  $j = 1, \dots, n$ , let

$$(3.4) \quad q_j = \int_{b_j}^1 (f(x) - r(x)u_n(x))dx - a(b_j)u'_n(b_j), \quad \text{where} \quad u'_n(b_j) := \sum_{i=0}^{j-1} c_i + \frac{c_j}{2}.$$

In a similar fashion as in [9], the second optimality condition in (3.1) becomes

$$(3.5) \quad \mathbf{0} = \nabla_{\mathbf{b}} J(u_n) = \mathbf{D}(\hat{\mathbf{c}}) (\mathbf{q} + \gamma(u_n(1) - \beta)\mathbf{1}),$$

where  $\mathbf{D}(\hat{\mathbf{c}})$  is an  $n \times n$  diagonal matrix with the  $i$ th diagonal element  $c_i$  and

$$\hat{\mathbf{c}} = (c_1, \dots, c_n)^T, \quad \mathbf{q} = (q_1, \dots, q_n)^T, \quad \text{and} \quad \mathbf{1} = (1, \dots, 1)^T.$$

*Remark 3.1.* If  $c_i = 0$ , then there is no  $i$ th equation in the optimality condition (3.5). Furthermore, the  $i$ th neuron has no contribution to the approximation  $u_n$ . Such neurons can be removed or redistributed.

**4. Mass matrix.** This section studies the mass matrix resulting from a shallow ReLU NN and the computation of its inversion.

While the stiffness matrix  $A_a(\mathbf{b})$  is dense, its inversion is a tridiagonal matrix with an explicit algebraic formula (see [9]). This property holds for matrices with the following structure:

$$(4.1) \quad \mathcal{M} = \begin{pmatrix} \alpha_1\beta_1 & \alpha_1\beta_2 & \alpha_1\beta_3 & \dots & \alpha_1\beta_k \\ \alpha_1\beta_2 & \alpha_2\beta_2 & \alpha_2\beta_3 & \dots & \alpha_2\beta_k \\ \alpha_1\beta_3 & \alpha_2\beta_3 & \alpha_3\beta_3 & \dots & \alpha_3\beta_k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_1\beta_k & \alpha_2\beta_k & \alpha_3\beta_k & \dots & \alpha_k\beta_k \end{pmatrix},$$

where  $\alpha_i$  and  $\beta_i$  are real numbers.

**LEMMA 4.1.** *Assume that  $\alpha_1 \neq 0$ ,  $\beta_k \neq 0$ , and  $\alpha_{i+1}\beta_i - \alpha_i\beta_{i+1} \neq 0$  for  $i = 1, \dots, k-1$ ; then the matrix  $\mathcal{M}$  defined in (4.1) is invertible. Moreover, its inverse  $\mathcal{M}^{-1}$  is symmetric and tridiagonal with nonzero entries given by*

$$\mathcal{M}_{ii}^{-1} = \frac{\alpha_{i+1}\beta_{i-1} - \alpha_{i-1}\beta_{i+1}}{(\alpha_i\beta_{i-1} - \alpha_{i-1}\beta_i)(\alpha_{i+1}\beta_i - \alpha_i\beta_{i+1})} \quad \text{and} \quad \mathcal{M}_{i,i+1}^{-1} = \frac{-1}{\alpha_{i+1}\beta_i - \alpha_i\beta_{i+1}} = \mathcal{M}_{i+1,i}^{-1},$$

where  $\alpha_0 = \beta_{k+1} = 0$  and  $\alpha_{k+1} = \beta_0 = 1$ .

*Proof.* The assumptions imply that  $\mathcal{M}_{i,i}^{-1}$  and  $\mathcal{M}_{i,i+1}^{-1} = \mathcal{M}_{i+1,i}^{-1}$  are well-defined. It is easy to verify that  $\mathcal{M}\mathcal{M}^{-1} = I$ .  $\square$

The stiffness matrix  $A_a(\mathbf{b})$  has the same structure as  $\mathcal{M}$  with

$$k = n, \quad \alpha_i = 1, \quad \text{and} \quad \beta_i = \int_{b_i}^1 a(x)dx.$$

The mass matrix  $M_r(\mathbf{b})$  is also dense due to the global support of the neurons, and its condition number is very large (see subsection 4.1). However, it can be factorized into matrices with structure as in (4.1). This section concludes with the inverse formulas of the mass matrix, and its derivations are presented in Appendix A algebraically and geometrically. By (4.3), application of  $M_r(\mathbf{b})^{-1}$  to any vector costs  $\mathcal{O}(n)$  operations.

**4.1. Condition number.** Let  $h_i = b_{i+1} - b_i$  for  $i = 0, \dots, n$ , and set

$$h_{\max} = \max_{0 \leq i \leq n} h_i \quad \text{and} \quad h_{\min} = \min_{0 \leq i \leq n} h_i.$$

It was shown in [9] that the condition number of  $A_a(\mathbf{b})$  is bounded by  $\mathcal{O}(nh_{\min}^{-1})$  for  $a(x) = 1$ . For the mass matrix, it was established in [17] that the condition number is both lower and upper bounded by  $\mathcal{O}(n^4)$  when the breakpoints are uniformly distributed over the domain  $\Omega = [0, 1]$ . Building on this, the authors in [7] extended the analysis to arbitrary breakpoint distributions, relating the condition number to the minimum pairwise distance between breakpoints.

LEMMA 4.2. *Let  $r(x) = 1$ ; then the condition number of the mass matrix  $M_r(\mathbf{b})$  is bounded by  $\mathcal{O}(n/h_{\min}^3)$ .*

*Proof.* For any vector  $\boldsymbol{\xi} = (\xi_0, \xi_1, \dots, \xi_n)^T \in \mathbb{R}^{n+1}$ , denote its magnitude by  $|\boldsymbol{\xi}| = (\sum_{i=0}^n \xi_i^2)^{1/2}$ . By the Cauchy–Schwarz inequality and the fact that  $\sigma(x - b_j) = 0$  for  $x \leq b_j$ , we have

$$\begin{aligned} (4.2) \quad \boldsymbol{\xi}^T M_r(\mathbf{b}) \boldsymbol{\xi} &= \int_0^1 \left( \sum_{i=0}^n \xi_i \sigma(x - b_i) \right)^2 dx \leq |\boldsymbol{\xi}|^2 \int_0^1 \left( \sum_{i=0}^n \sigma(x - b_i)^2 \right) dx \\ &= |\boldsymbol{\xi}|^2 \sum_{j=0}^n \int_{b_j}^{b_{j+1}} \sum_{i=1}^j \sigma(x - b_i)^2 dx = \frac{|\boldsymbol{\xi}|^2}{3} \sum_{j=0}^n \sum_{i=0}^j \{ (b_{j+1} - b_i)^3 - (b_j - b_i)^3 \} \\ &= \frac{|\boldsymbol{\xi}|^2}{3} \sum_{i=0}^n (b_{n+1} - b_i)^3 = \frac{|\boldsymbol{\xi}|^2}{3} \sum_{i=0}^n (1 - b_i)^3 \leq \frac{(n+1)}{3} |\boldsymbol{\xi}|^2. \end{aligned}$$

To estimate the lower bound of  $\boldsymbol{\xi}^T M_r(\mathbf{b}) \boldsymbol{\xi}$ , let

$$\tau_i(x) = \sum_{j=0}^i \xi_j \sigma(x - b_j) \quad \text{and} \quad a_{i-1} = \tau_i(b_i) = \sum_{j=0}^i \xi_j (b_i - b_j),$$

for  $i = 0, \dots, n + 1$ , where  $a_{-1} = 0$ . Then  $\tau_i(\frac{b_{i+1} + b_i}{2}) = \frac{a_{i-1} + a_i}{2}$ . Since  $\tau_i^2(x)$  is a quadratic function in each subinterval  $[b_i, b_{i+1}]$ , Simpson’s rule implies

$$\begin{aligned} \boldsymbol{\xi}^T M_r(\mathbf{b}) \boldsymbol{\xi} &= \sum_{i=0}^n \int_{b_i}^{b_{i+1}} \tau_i^2(x) dx = \frac{1}{6} \sum_{i=0}^n h_i \left[ \tau_i^2(b_i) + 4\tau_i^2\left(\frac{b_{i+1} + b_i}{2}\right) + \tau_i^2(b_{i+1}) \right] \\ &= \frac{1}{6} \sum_{i=0}^n h_i [a_{i-1}^2 + (a_{i-1} + a_i)^2 + a_i^2] \geq \frac{1}{6} h_{\min} |\mathbf{a}|^2, \end{aligned}$$

where  $\mathbf{a} = (a_0, a_1, \dots, a_n)^T$ . It is easy to see that  $\boldsymbol{\xi} = Q\mathbf{a}$ , where  $Q$  is an  $(n + 1)$ -order lower tridiagonal matrix given by

$$Q = \begin{pmatrix} \frac{1}{h_0} & 0 & 0 & \dots & 0 & 0 \\ -\left(\frac{1}{h_0} + \frac{1}{h_1}\right) & \frac{1}{h_1} & 0 & \dots & 0 & 0 \\ \frac{1}{h_1} & -\left(\frac{1}{h_1} + \frac{1}{h_2}\right) & \frac{1}{h_2} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{h_{n-1}} & 0 \\ 0 & 0 & 0 & \dots & -\left(\frac{1}{h_{n-1}} + \frac{1}{h_n}\right) & \frac{1}{h_n} \end{pmatrix}.$$

It is easy to verify that  $Q$  has spectral norm bounded by

$$\|Q\|_2 \leq \sqrt{\|Q\|_1 \|Q\|_\infty} \leq 4h_{\min}^{-1}.$$

Hence,

$$\boldsymbol{\xi}^T M_\omega(\mathbf{b}) \boldsymbol{\xi} \geq \frac{1}{6} h_{\min} \frac{|\boldsymbol{\xi}|^2}{\|Q\|_2^2} \geq \frac{1}{96} h_{\min}^3 |\boldsymbol{\xi}|^2,$$

which, together with the upper bound in (4.2), implies the validity of the lemma.  $\square$

**LEMMA 4.3.** *Under the assumption on the reaction coefficient  $r(x)$ , the condition number of the mass matrix  $M_r(\mathbf{b})$  is bounded by  $\mathcal{O}(nr_0^{-1}h_{\min}^{-3})$ .*

*Proof.* Since  $r \in L^\infty(I)$  and  $r(x) \geq r_0$  almost everywhere, in a similar fashion as the proof of Lemma 4.2, we have

$$\frac{1}{6} r_0 h_{\min}^3 |\boldsymbol{\xi}|^2 \leq \boldsymbol{\xi}^T M_r(\mathbf{b}) \boldsymbol{\xi} \leq C(n+1) |\boldsymbol{\xi}|^2,$$

which implies the validity of the lemma.  $\square$

Whereas the mass matrix associated with the ReLU NN is very ill-conditioned, it is well-known that the mass matrix for the finite element method is much better conditioned (see [14], for example). The following Lemma 4.4 reiterates the result in [14] with an alternate proof in a similar fashion as that of Lemma 4.2. To this end, for  $i = 1, \dots, n$ , denote the standard hat basis functions by

$$\varphi_i(x) = \begin{cases} (x - b_{i-1})/h_i, & x \in (b_{i-1}, b_i), \\ (b_{i+1} - x)/h_{i+1}, & x \in (b_i, b_{i+1}), \\ 0 & \text{otherwise.} \end{cases}$$

Let  $\boldsymbol{\varphi} = (\varphi_1, \dots, \varphi_n)^T$ ; then the corresponding mass matrix is given by

$$\tilde{M}(\mathbf{b}) = \int_0^1 \boldsymbol{\varphi} \boldsymbol{\varphi}^T dx.$$

**LEMMA 4.4.** *The condition number of the finite element mass matrix  $\tilde{M}(\mathbf{b})$  is bounded by  $\mathcal{O}(h_{\max}/h_{\min})$ .*

*Proof.* For any vector  $\boldsymbol{\xi} = (\xi_1, \dots, \xi_n)^T \in \mathbb{R}^n$ , in a similar fashion as that of Lemma 4.2, we get the equality

$$\boldsymbol{\xi}^T \tilde{M}(\mathbf{b}) \boldsymbol{\xi} = \sum_{j=0}^n \int_{b_j}^{b_{j+1}} (\xi_j \varphi_j + \xi_{j+1} \varphi_{j+1})^2 dx = \sum_{j=0}^n \frac{h_j}{6} [\xi_j^2 + (\xi_j + \xi_{j+1})^2 + \xi_{j+1}^2]$$

with  $\varphi_0(x) = \varphi_{n+1}(x) = \xi_0 = \xi_{n+1} = 0$ , which leads to the inequalities

$$\frac{1}{6} h_{\min} |\boldsymbol{\xi}|^2 \leq \boldsymbol{\xi}^T \tilde{M}(\mathbf{b}) \boldsymbol{\xi} \leq \frac{2}{3} h_{\max} |\boldsymbol{\xi}|^2.$$

This completes the proof of the lemma.  $\square$

*Remark 4.1.* It is well-known [15] that classic iterative solvers such as Richardson (i.e., the method of gradient descent), Jacobi, Gauss–Seidel, etc., for a system of linear equations converge slowly if the condition number  $\kappa(A)$  of the corresponding matrix is very large. The convergence rate of those methods is at most  $\frac{\kappa(A)-1}{\kappa(A)+1}$ .

**4.2. Inversion of the mass matrix.** Despite the ill-conditioning of the mass matrix  $M_r(\mathbf{b})$ , it can still be efficiently inverted. In fact, the following lemma states that  $M_r(\mathbf{b})$  can be factorized into a product of matrices that are easy to invert. The factorization can be derived by either an algebraic (see subsection A.1) or a geometric (see subsection A.2) approach.

LEMMA 4.5. *The mass matrix  $M_r(\mathbf{b})$  has the following factorization:*

$$(4.3) \quad M_r(\mathbf{b}) = T_1^{-1} T_2 T_3^{-1},$$

where  $T_1$ ,  $T_2$ , and  $T_3$  are tridiagonal matrices given in (A.1) and (A.6).

Remark 4.6. Additionally, for the given  $T_1, T_3$  in (A.6), the sum  $A_a(\mathbf{b}) + M_r(\mathbf{b})$  has the following factorization:

$$(4.4) \quad A_a(\mathbf{b}) + M_r(\mathbf{b}) = T_1^{-1} T_4 T_3^{-1},$$

where  $T_4$  is a tridiagonal matrix given in Remark A.5.

Remark 4.7. The Sherman–Morrison formula and (4.4) imply that the system of linear equations in (3.3) can be solved in  $\mathcal{O}(n)$  operations.

**5. A damped block Newton method.** As mentioned in [9], the system of nonlinear algebraic equations in (3.5) can be solved using the variable projection (VarPro) method [16]. In the VarPro approach, the solution  $\mathbf{c}$  of (3.3) is substituted into (3.5), resulting in a reduced nonlinear system for  $\mathbf{b}$ . However, this reduced system may exhibit a more complex structure.

In this section, we describe a dBN method for solving the minimization problem in (2.4). This method employs the block Gauss–Seidel method as an outer iteration for the linear and nonlinear parameters. Per each outer iteration, the linear and the nonlinear parameters are updated by exact inversion and one step of a damped Newton method, respectively.

To solve (3.3) for the linear parameter, the direct inversion is addressed in section 4. Now to apply Newton’s method to the optimality condition (3.5), we first must derive the Hessian for the nonlinear parameter and discuss its invertibility. To that end, assume that  $a(x)$  is differentiable at  $b_i$  for each  $i = 1, \dots, n$ , and define

$$g_i = \frac{\partial}{\partial b_i} q_i = r(b_i) u_n(b_i) - f(b_i) - a'(b_i) u_n'(b_i),$$

where  $q_i$  is defined in (3.4). Denote by  $\mathbf{D}(\mathbf{g}) = \text{diag}(g_1, \dots, g_n)$  the diagonal matrix with the  $i$ th diagonal element  $g(b_i)$ . A formula for the Hessian matrix  $\nabla_{\mathbf{b}}^2 J$  is given in the next lemma.

LEMMA 5.1. *Assume that  $c_i \neq 0$  and that  $a(x)$  is differentiable at  $x = b_i$  for all  $i = 1, \dots, n$ . Then the Hessian matrix  $\nabla_{\mathbf{b}}^2 J(u_n)$  has the form*

$$(5.1) \quad \mathbf{H}(\mathbf{c}, \mathbf{b}) = \mathbf{D}(\hat{\mathbf{c}})(\mathbf{D}(\mathbf{g}) + A_r(\mathbf{b})\mathbf{D}(\hat{\mathbf{c}}) + \gamma \mathbf{1}\hat{\mathbf{c}}^T).$$

*Proof.* Equation (5.1) can be derived in a similar fashion as that of Lemma 5.2 in [9]. The only difference here is the additional reaction term in (2.3). For that term, the computations shown in Lemma 4.1 from [8] can be used to obtain the second-order derivatives with respect to  $\mathbf{b}$ .  $\square$

The following lemma gives a sufficient condition for the invertibility of the Hessian  $\mathbf{H}(\mathbf{c}, \mathbf{b})$ .

LEMMA 5.2. Let  $\tilde{h}_i = \min\{h_{i-1}, h_i\}$ . If  $c_i \neq 0$ ,  $a(x)$  is differentiable at  $b_i$  and

$$(5.2) \quad \frac{g_i}{c_i} + \frac{r_0 \tilde{h}_i}{4} > 0 \quad \text{for all } i = 1, \dots, n,$$

then  $\mathbf{D}(\mathbf{g})\mathbf{D}(\hat{\mathbf{c}})^{-1} + A_r(\mathbf{b})$  is positive definite.

*Proof.* For any vector  $\boldsymbol{\xi} = (\xi_1, \dots, \xi_n)^T \in \mathbb{R}^n$ , an argument similar to the proof of Lemma 4.3 in [9] shows that

$$\boldsymbol{\xi}^T A_r(\mathbf{b}) \boldsymbol{\xi} \geq \frac{r_0}{4} \sum_{j=1}^n \tilde{h}_j \xi_j^2,$$

which implies that  $\boldsymbol{\xi}^T (\mathbf{D}(\mathbf{g})\mathbf{D}(\hat{\mathbf{c}})^{-1} + A_r(\mathbf{b})) \boldsymbol{\xi} \geq \sum_{j=1}^n (\frac{g_j}{c_j} + \frac{r_0 \tilde{h}_j}{4}) \xi_j^2$ . Now, the lemma is a direct consequence of the assumption in (5.2).  $\square$

COROLLARY 5.3. The matrix  $\mathbf{H}(\mathbf{c}, \mathbf{b})$  is invertible under the conditions of Lemma 5.2.

*Remark 5.4.* Condition (5.2) is sufficient but not necessary. Therefore, for implementation purposes, we consider the relaxed condition

$$(5.3) \quad \frac{g_i}{c_i} + \tau_2^{-1} \frac{r_0 \tilde{h}_i}{4} > 0 \quad \text{for all } i = 1, \dots, n$$

for some parameter  $0 < \tau_2 \leq 1$ . In the numerical examples presented in section 7, setting  $\tau_2 = 1$  was not necessary to ensure invertibility, so a value  $\tau_2 < 1$  was used instead.

*Remark 5.5.* The action of  $\mathbf{H}(\mathbf{c}, \mathbf{b})^{-1}$  applied to any vector can be computed in  $\mathcal{O}(n)$  operations. This is due to the Sherman–Morrison formula and the facts that

$$\mathbf{D}(\mathbf{g}) + A_r(\mathbf{b})\mathbf{D}(\hat{\mathbf{c}}) = (\mathbf{D}(\mathbf{g})\mathbf{D}(\hat{\mathbf{c}})^{-1} A_r(\mathbf{b})^{-1} + I) A_r(\mathbf{b})\mathbf{D}(\hat{\mathbf{c}})$$

and that  $A_r(\mathbf{b})^{-1}$  is tridiagonal.

In cases where  $a'(b_i)$  is not defined for some  $i \in \{1, \dots, n\}$ , the breakpoint  $b_i$  lies on the interface and should be fixed without further update. If  $c_i = 0$ , then the breakpoint  $b_i$  could be either removed or redistributed (see Remark 3.1).

Overall, we can update the nonlinear parameter with a Newton step in  $\mathcal{O}(n)$  operations granted that the Hessian  $\mathbf{H}(\mathbf{c}, \mathbf{b})$  is invertible and the optimality condition (3.5) is well-posed. Thus, for the implementation of the dBN method, we set aside the neurons which violate these conditions and construct a reduced system with said neurons removed. To that end, let  $0 < \tau_1$  and  $0 < \tau_2 \leq 1$ , and define the set

$$S_1 = \{i \in \{1, \dots, n\} : |c_i| < \tau_1 \text{ or } b_i \notin I\}$$

of indices for the noncontributing neurons, and the set

$$S_2 = \left\{ i \in \{1, \dots, n\} \setminus S_1 : \frac{g_i}{c_i} + \tau_2^{-1} \frac{r_0 \tilde{h}_i}{4} \leq 0 \quad \text{or} \quad a'(b_i) \text{ DNE} \right\}$$

of indices for which the corresponding neurons do not satisfy the invertibility condition (5.3) or belong to the interface. Then

$$S = \{1, \dots, n\} \setminus (S_1 \cup S_2)$$

is the set of indices for the neurons which remain in the system.

Next, given a vector  $\mathbf{v} \in \mathbb{R}^n$ , we denote by  $\mathbf{v}_S \in \mathbb{R}^{n-|S^c|}$  the vector obtained by removing the  $i$ th entry from  $\mathbf{v}$  for every  $i \in S^c = \{1, \dots, n\} \setminus S$ . Similarly, for a matrix  $\mathbf{B} \in \mathbb{R}^{n \times n}$ , define  $\mathbf{B}_S \in \mathbb{R}^{(n-|S^c|) \times (n-|S^c|)}$  as the matrix obtained by removing the  $i$ th row and column of  $\mathbf{B}$  for each  $i \in S^c$ .

Finally, define the *reduced* search direction vector for the Newton step as

$$(5.4) \quad \mathbf{d}_R(\mathbf{c}, \mathbf{b}) = - \left[ (\mathbf{D}(\mathbf{g}) + A_r(\mathbf{b})\mathbf{D}(\hat{\mathbf{c}}) + \gamma \mathbf{1}\hat{\mathbf{c}}^T)_S \right]^{-1} \left( \mathbf{q} + \gamma(u_n(1) - \beta)\mathbf{1} \right)_S.$$

We are now ready to describe the dBN method (see Algorithm 5.1 for a pseudocode). Given prescribed tolerances  $0 < \tau_1$  and  $0 < \tau_2 \leq 1$ , let  $\mathbf{b}^{(k)}$  be the previous iterate; then the current iterate  $\mathbf{r}^{(k+1)} = (\mathbf{c}^{(k+1)}, \mathbf{b}^{(k+1)})$  is computed as follows:

(i) *Compute the linear parameters*

$$\mathbf{c}^{(k+1)} = \mathcal{A} \left( \mathbf{b}^{(k)} \right)^{-1} \mathcal{F} \left( \mathbf{b}^{(k)} \right).$$

(ii) *Compute the search direction*  $\mathbf{p}^{(k)} = (p_1^{(k)}, \dots, p_n^{(k)})^T$  by

$$(5.5) \quad (p_i^{(k)})_{i \in S} = d_R(\mathbf{c}, \mathbf{b}) \quad \text{and} \quad (p_i^{(k)})_{i \notin S} = \mathbf{0},$$

where  $d_R(\mathbf{c}, \mathbf{b})$  is the *reduced* Newton's direction vector defined in (5.4).

(iii) *Calculate the stepsize*  $\eta_k$  by performing a one-dimensional optimization

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

(iv) *Compute the nonlinear parameters*

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

(v) *Redistribute noncontributing breakpoints*  $b_i^{(k+1)}$  for all  $i \in S_1$  and sort  $\mathbf{b}^{(k+1)}$ .

*Remark 5.6.* The redistribution implemented for the numerical results shown in section 7 in step (v) was carried out as follows: For a neuron  $b_l^{(k+1)}$  satisfying  $l \in S_1$ , we set

$$b_l^{(k+1)} \leftarrow \frac{b_{m-1}^{(k+1)} + b_m^{(k+1)}}{2},$$

where  $m \in \{1, \dots, n+1\}$  is an integer chosen uniformly at random.

---

**Algorithm 5.1.** A dBN method for (2.4).

---

**Require:** Initial network parameters  $\mathbf{b}^{(0)}$

**Ensure:** Network parameters  $\mathbf{c}, \mathbf{b}$

**for**  $k = 0, 1 \dots$  **do**

▷ *Linear parameters*

$$\mathbf{c}^{(k+1)} \leftarrow \mathcal{A} \left( \mathbf{b}^{(k)} \right)^{-1} \mathcal{F} \left( \mathbf{b}^{(k)} \right)$$

▷ *Nonlinear parameters*

*Compute the search direction*  $\mathbf{p}^{(k)}$  *as in* (5.5)

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

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

▷ *Redistribute noncontributing neurons and sort*  $\mathbf{b}^{(k+1)}$

**end for**

---

*Remark 5.7.* The minimization problem in (2.4) is nonconvex and contains multiple local minima. Therefore, obtaining an initial approximation sufficiently close to the desired minimum is crucial. Since the nonlinear parameters  $\mathbf{b}$  correspond to the breakpoints that partition the interval  $I = [0, 1]$ , we initialize them following the approach in [5, 9, 18]. Specifically, the nonlinear parameters  $\mathbf{b}$  are set as a uniform partition of  $I$ , while the linear parameters  $\mathbf{c}$  are determined as the solution to the linear system in (3.3) for this uniform mesh.

**5.1. An adaptivity scheme.** For a fixed number of neurons, the dBN method for the diffusion-reaction equation moves the initial uniformly distributed breakpoints very efficiently to nearly optimal locations as shown in section 7. However, it was shown in [9] that introducing adaptivity results in a more optimal convergence rate.

In fact, the ANE method [18, 19] was employed in [9]. The ANE method starts with a relatively small NN and adaptively adds new neurons based on the previous approximation. At each adaptive step, the dBN method numerically solves the minimization problem in (2.4). Subsequently, the newly added neurons are initialized based on regions where the previous approximation is not accurate.

In the ANE method, new neurons are added according to a marking strategy. Below, we explain the local indicators and the marking strategy used in this paper.

Let  $\mathcal{K} = [c, d] \subseteq [0, 1]$  be a subinterval; then a modified local indicator of the ZZ type on  $\mathcal{K}$  (see, e.g., [10]) is defined by

$$\xi_{\mathcal{K}}^2 = \|a^{-1/2}(G(au'_n) - au'_n)\|_{L^2(\mathcal{K})}^2 + (d-c)^2 \|-G'(a^2u'_n) + u_n - f\|_{L^2(\mathcal{K})}^2,$$

where  $G(v)$  is the projection of  $v$  onto the space of the continuous piecewise linear functions. The corresponding relative error estimator is defined by  $\xi = \xi_I/|u_n|_{H^1(I)}$ .

Let  $u_n \in \mathcal{M}_n(I)$  be an NN with the breakpoints

$$0 = b_0 < b_1 < \dots < b_n < b_{n+1} = 1.$$

Define  $\mathcal{K}^i = [b_i, b_{i+1}]$ , so that the collection  $\mathcal{K}_n = \{\mathcal{K}^i\}_{i=0}^n$  defines a partition of the interval  $[0, 1]$ . To refine this partition, we define  $\widehat{\mathcal{K}}_n \subset \mathcal{K}_n$  by using the following average marking strategy:

$$(5.6) \quad \widehat{\mathcal{K}}_n = \left\{ K \in \mathcal{K}_n : \xi_K \geq \frac{1}{\#\mathcal{K}_n} \sum_{K \in \mathcal{K}_n} \xi_K \right\},$$

where  $\#\mathcal{K}_n$  is the number of elements in  $\mathcal{K}_n$ . The adaptive damped block Newton (AdBN) method is described in Algorithm 5.2.

**6. Least-squares approximation.** Given a function  $u(x) \in L^2(I)$ , the best least-squares approximation to  $u$  in  $\mathcal{M}_n(I)$  is to find  $u_n \in \mathcal{M}_n(I) \cap \{u_n(0) = u(0)\}$  such that

$$(6.1) \quad J(u_n) = \min_{v \in \mathcal{M}_n(I) \cap \{v(0)=u(0)\}} J(v),$$

where  $J(v)$  is the weighted continuous least-squares loss functional given by

$$(6.2) \quad J(v) = \frac{1}{2} \int_0^1 r(x) (v(x) - u(x))^2 dx = \frac{1}{2} \|v - u\|_r^2.$$

Let  $u_n(x) = u(0) + \sum_{i=0}^n c_i \sigma(x - b_i) \in \mathcal{M}_n(I)$  be a solution of (6.1). Then, the optimality conditions become

$$(6.3) \quad \mathbf{0} = \nabla_{\mathbf{c}} J(u_n) = M_r(\mathbf{b}) \mathbf{c} - \mathbf{f}(\mathbf{b}) \quad \text{and} \quad \mathbf{0} = \nabla_{\mathbf{b}} J(u_n) = \mathbf{D}(\hat{\mathbf{c}}) \mathbf{r},$$

**Algorithm 5.2.** AdBN method.

---

**Require:** Initial number of neurons  $n_0$ , parameters  $a(x)$ ,  $r(x)$ ,  $f(x)$ ,  $\alpha$ , and  $\beta$ , tolerance  $\epsilon$ ,

- (1) Compute an approximation to the solution  $u_n$  of the optimization problem in (2.4) by the dBN method;
- (2) Compute the estimator  $\xi_n = (\sum_{K \in \mathcal{K}} \xi_K^2)^{1/2} / |u_n|_{H^1(I)}$ ;
- (3) If  $\xi_n \leq \epsilon$ , then stop; otherwise go to step (4);
- (4) Mark elements in  $\hat{\mathcal{K}}_n$  and denote by  $\#\hat{\mathcal{K}}_n$  the number of elements in  $\hat{\mathcal{K}}_n$ ;
- (5) Add  $\#\hat{\mathcal{K}}_n$  new neurons to the network and initialize them at the midpoints of elements in  $\hat{\mathcal{K}}_n$ , then go to step (1)

---

where  $M_r(\mathbf{b})$  is the mass matrix defined in (3.2),  $\mathbf{D}(\hat{\mathbf{c}}) = \text{diag}(c_1, \dots, c_n)$ , and  $\mathbf{f}(\mathbf{b})$  and  $\mathbf{r}$  are given respectively by

$$\mathbf{f}(\mathbf{b}) = (f_i)_{(n+1) \times 1} \quad \text{with } f_i = \int_0^1 r(x) (u(x) - u(0)) \sigma(x - b_{i-1}) dx,$$

$$\text{and } \mathbf{r} = (r_i)_{n \times 1} \quad \text{with } r_i = - \int_{b_i}^1 r(x) (u_n(x) - u(x)) dx.$$

Let  $w_i = r(b_i)(u_n(b_i) - u(b_i))$  for  $i = 1, \dots, n$ . In one dimension, Lemma 4.1 in [8] implies that the corresponding Hessian matrix is of the form

$$(6.4) \quad \nabla_{\mathbf{b}}^2 J(u_n) \equiv \mathbf{H}(\mathbf{c}, \mathbf{b}) = \mathbf{D}(\hat{\mathbf{c}}) (\mathbf{D}(\mathbf{w}) + A_r(\mathbf{b}) \mathbf{D}(\hat{\mathbf{c}})),$$

where  $\mathbf{D}(\mathbf{w}) = \text{diag}(w_1, \dots, w_n)$  is a diagonal matrix. Based on the optimality condition in (6.3) and the Hessian matrix in (6.4), we can then design the dBN and AdBN methods in a similar fashion as in sections 4 and 5, respectively.

We conclude this section by discussing the numerical integration of the functionals in (2.3) and (6.2). In one dimension, we may calculate the integrals analytically, as shown in several examples in section 7. In practice, integrals are often computed by various numerical integration methods. For an integrand (given as in (6.2) or unknown as in (2.3)), choosing an accurate quadrature rule with the least computational cost is highly nontrivial. An efficient and effective method for achieving this goal is the so-called adaptive quadrature (see, e.g., [24]), which was used in [20] for the deep Ritz method in linear elasticity.

An accurate but possibly inefficient numerical quadrature is, for example, the midpoint rule on a very fine uniform *integration* mesh. Specifically, let  $\{x_i = i/m\}_{i=0}^m$  be a partition of the interval  $I = [0, 1]$  with sufficiently large  $m$ . Then the least-squares functional  $J(v)$  is approximated by the following discrete least-squares functional:

$$J_m(v) = \frac{1}{2} \sum_{i=1}^m r(x_{i-1/2}) (v(x_{i-1/2}) - u(x_{i-1/2}))^2 = \frac{1}{2} \|v - u\|_{r,m}^2.$$

The accuracy of the approximation depends on both the complexity of the function  $u(x)$  and the size of  $m$ . Now, the discrete least-squares approximation to  $u$  in  $\mathcal{M}_n(I)$  is to find  $u_n \in \mathcal{M}_n(I) \cap \{u_n(0) = u(0)\}$  such that

$$(6.5) \quad J_m(u_n) = \min_{v \in \mathcal{M}_n(I) \cap \{v(0) = u(0)\}} J_m(v).$$

The dBN developed for the continuous optimization problem in (6.1) can be applied directly to the discrete optimization in (6.5) by replacing integration with summation when forming the matrices  $M_r(\mathbf{b})$  and  $A_r(\mathbf{b})$ . Since  $\|\cdot\|_{r,m}$  defines a norm in  $\mathbb{R}^m$ , then it is easy to see that  $M_r(\mathbf{b})$  and  $A_r(\mathbf{b})$  are positive definite (see, e.g., Lemma 4.1 of [8]).

Treating  $\{x_{i-1/2}, u(x_{i-1/2})\}_{i=1}^m$  as a training set, the minimization problem in (6.5) is similar to the standard machine learning problem in data science. Hence, the dBN method can be applied directly and viewed as an efficient training algorithm.

**7. Numerical experiments.** This section first presents numerical results using the dBN method to solve (6.1). Subsequently, results for the dBN and AdBN methods applied to (2.1) are shown in subsections 7.2 and 7.3. The parameters  $\tau_1$  and  $\tau_2$  were set to  $10^{-10}$  and  $10^{-2}$ , respectively. For diffusion-reaction problems, the penalization parameter  $\gamma$  was set to  $10^4$ . In the AdBN method, a refinement occurred when the difference in relative residuals between two consecutive iterates was less than  $10^{-6}$ .

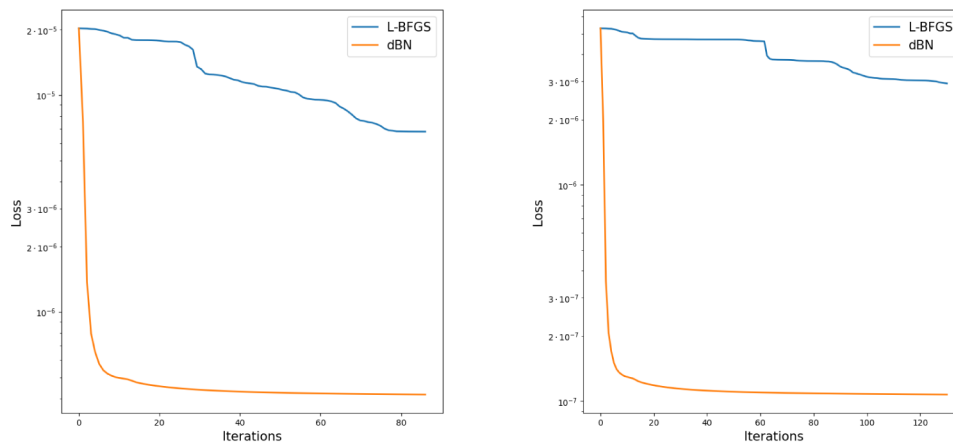
For each test problem of the diffusion-reaction equation, let  $u$  and  $u_n$  be the exact solution and its approximation in  $\mathcal{M}_n(I)$ , respectively. Denote the relative error by

$$(7.1) \quad e_n = \frac{|u - u_n|_{H^1(I)}}{|u|_{H^1(I)}}.$$

**7.1. Least-squares problem.** The first test problem is the function

$$(7.2) \quad u(x) = \sqrt{x}$$

as the target function for problem (6.1) with  $r(x) = 1$ . We aim to test the performance of dBN for least-squares approximation problems. Figure 1 presents a comparison between dBN and L-BFGS. In this comparison, we utilized a Python L-BFGS implementation from `scipy.optimize`. The initial network parameters for the two algorithms were set to be the uniform mesh for  $\mathbf{b}^{(0)}$  with  $\mathbf{c}^{(0)}$  given by solving (6.3). The computation times for selected iterations are reported in Table 1 (CPU: 12th Gen



(a) Loss versus number of iterations with 24 neurons. Final losses: L-BFGS -  $6.78 \times 10^{-6}$ , dBN -  $4.16 \times 10^{-7}$

(b) Loss versus number of iterations with 48 neurons. Final losses: L-BFGS -  $2.94 \times 10^{-6}$ , dBN -  $1.07 \times 10^{-7}$

FIG. 1. Comparison between L-BFGS and dBN for approximating function (7.2).

TABLE 1

Times at various iterations of dBN and L-BFGS for approximating function (7.2).

Method	$n$	Computation time (seconds)			
		20 itr.	40 itr.	60 itr.	80 itr.
dBN	24	0.16	0.29	0.43	0.56
L-BFGS	24	0.66	1.39	2.10	2.80
dBN	48	0.39	0.75	1.09	1.50
L-BFGS	48	2.67	5.01	7.59	10.11

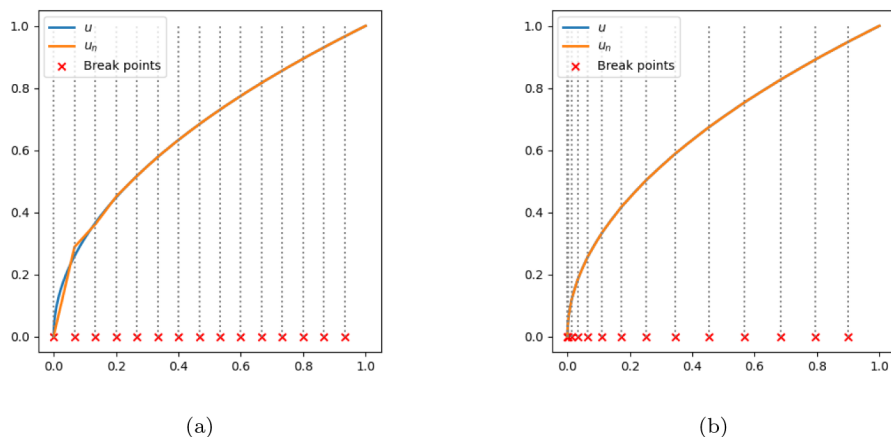


FIG. 2. (a) Initial NN model with 15 uniform breakpoints,  $J(u_n) = 5.64 \times 10^{-5}$ ; (b) optimized NN model with 15 breakpoints using dBN, 1000 iterations,  $J(u_n) = 1.20 \times 10^{-7}$ .

Intel Core i3-1215U, 1.20 GHz). In this example, our solver outperforms L-BFGS, achieving smaller losses in fewer iterations and less time.

Figure 2(a) illustrates the NN approximation of the function in (7.2), obtained using uniform breakpoints and determining the linear parameters through the solution of (6.3). Clearly, it is more optimal to concentrate more mesh points on the left side, where the curve is steeper. The dBN method is capable of making this adjustment, as illustrated in Figure 2(b). The loss functions confirm that the approximation improves substantially when the breakpoints are allocated according to the steepness of the function.

**7.2. Exponential solution.** The second test problem involves the function

$$(7.3) \quad u(x) = x \left( \exp \left( -\frac{(x - \frac{1}{3})^2}{0.01} \right) - \exp \left( -\frac{4}{9 \times 0.01} \right) \right),$$

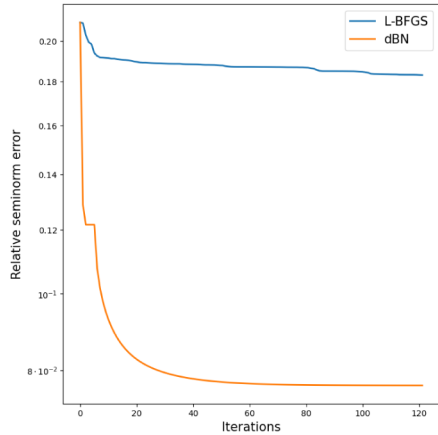
serving as a solution of (2.1) for  $a(x) = r(x) = 1$  and  $\alpha = \beta = 0$ .

Similarly to Figure 1, we start by comparing our solver with L-BFGS. The initial network parameters for both algorithms were set to be the uniform mesh for  $\mathbf{b}^{(0)}$ , with  $\mathbf{c}^{(0)}$  given by the exact solution of (3.3). Table 2 reports the times at various iterations. From Figure 3 and Table 2, we observe that dBN outperforms L-BFGS in both accuracy and computation time.

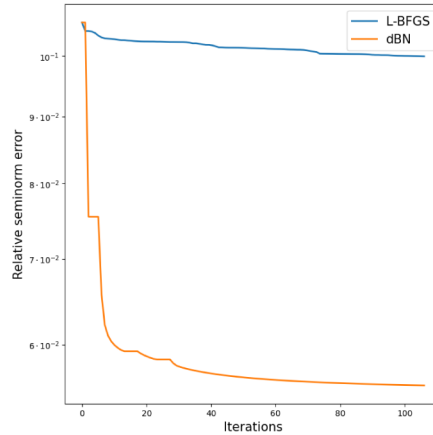
Figure 4(a) shows the initial NN approximation of the function in (7.3), obtained by using uniform breakpoints and determining the linear parameters through the solution of (3.3). The approximation generated by dBN is shown in Figure 4(b), while Figure 4(c) illustrates the approximation obtained by employing dBN with adaptivity

TABLE 2  
Times at various iterations of dBN and L-BFGS for approximating function (7.3).

Method	$n$	Computation time (seconds)			
		20 itr.	40 itr.	60 itr.	80 itr.
dBN	24	0.14	0.30	0.45	0.60
L-BFGS	24	0.80	1.48	2.16	2.90
dBN	48	0.35	0.71	1.08	1.43
L-BFGS	48	2.97	5.95	8.79	10.09



(a) Relative error  $e_n$  versus number of iterations with 24 breakpoints. Final relative errors: L-BFGS - 0.183, dBN - 0.076



(b) Relative error  $e_n$  versus number of iterations with 48 breakpoints. Final relative errors: L-BFGS - 0.099, dBN - 0.056

FIG. 3. Comparison between L-BFGS and dBN for approximating function (7.3).

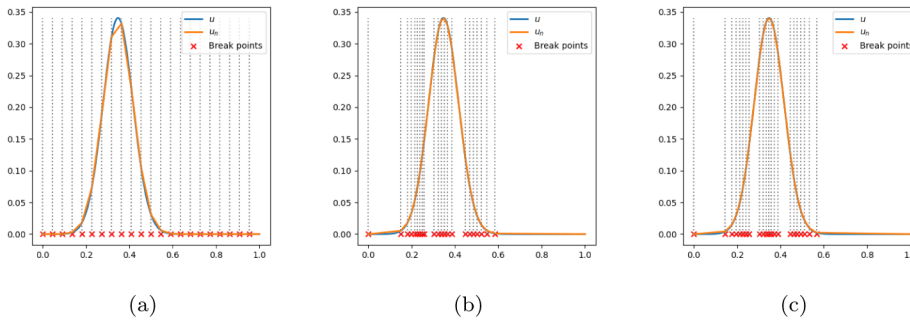


FIG. 4. (a) Initial NN model with 22 uniform breakpoints,  $e_n = 0.228$ ; (b) optimized NN model with 22 breakpoints, 500 iterations,  $e_n = 0.092$ ; (c) adaptive approximation ( $n = 8, 11, 14, 18, 22$ ),  $e_n = 0.083$ .

(AdBN). Notably, in both cases, the breakpoints are moved, and the approximation enhances the initial approximation.

Theoretically, from (2.7),  $\frac{1}{n}$  is the order of convergence of approximating a solution (7.3) by functions in  $\mathcal{M}_n$ . However, since (2.4) is a nonconvex optimization problem, the existence of local minimums makes it challenging to achieve this order. Therefore, given the NN approximation  $u_n$  to  $u$  provided by the dBN method, assume that

TABLE 3

Comparison of an adaptive network with fixed networks for relative error  $e_n$ , relative error estimators  $\xi_n$ , and powers  $r$ .

NN ( $n$ breakpoints)	$e_n$	$\xi_n$	$r$
Adaptive (18)	$1.11 \times 10^{-1}$	0.371	0.777
Adaptive (22)	$8.43 \times 10^{-2}$	0.335	0.813
Adaptive (26)	$6.99 \times 10^{-2}$	0.263	0.827
Adaptive (30)	$6.14 \times 10^{-2}$	0.231	0.828
Adaptive (34)	$5.55 \times 10^{-2}$	0.183	0.827
Adaptive (38)	$4.84 \times 10^{-2}$	0.159	0.839
Adaptive (42)	$4.35 \times 10^{-2}$	0.136	0.844
Adaptive (46)	$3.97 \times 10^{-2}$	0.130	0.848
Adaptive (50)	$3.70 \times 10^{-2}$	0.122	0.847
Adaptive (54)	$3.40 \times 10^{-2}$	0.114	0.852
Fixed (50)	$4.74 \times 10^{-2}$	0.147	0.783
Fixed (54)	$4.24 \times 10^{-2}$	0.121	0.796

$$e_n = \left(\frac{1}{n}\right)^r$$

for some  $r > 0$ . As in [9], we can use the AdBN method to improve the order of convergence of the dBN method (achieve an  $r$  closer to 1).

Table 3 illustrates AdBN starting with 18 neurons, refining 9 times, and reaching a final count of 54 neurons. The stopping tolerance was set to  $\epsilon = 0.12$ . The recorded data in Table 3 includes the relative seminorm error  $e_n$  and the relative error estimator  $\xi_n$  for each iteration of the adaptive process. Additionally, Table 3 provides the results for dBN with fixed 50 and 54 neurons. Comparing these results to the adaptive run with the same number of neurons, we observe a significant improvement in rate, error estimator, and seminorm error within the adaptive run.

**7.3. Singularly perturbed reaction-diffusion equation.** The third test problem is a singularly perturbed reaction-diffusion equation:

$$(7.4) \quad \begin{cases} -\varepsilon^2 u''(x) + u(x) = f(x), & x \in I = (-1, 1), \\ u(-1) = u(1) = 0. \end{cases}$$

For  $f(x) = -2(\varepsilon - 4x^2 \tanh(\frac{1}{\varepsilon}(x^2 - \frac{1}{4}))) / \cosh(\frac{1}{\varepsilon}(x^2 - \frac{1}{4}))^2 + \tanh(\frac{1}{\varepsilon}(x^2 - \frac{1}{4})) - \tanh(\frac{3}{4\varepsilon})$ , problem (7.4) has the following exact solution:

$$(7.5) \quad u(x) = \tanh\left(\frac{1}{\varepsilon}\left(x^2 - \frac{1}{4}\right)\right) - \tanh\left(\frac{3}{4\varepsilon}\right).$$

For some  $\nu = \varepsilon^2$ , these problems exhibit interior layers that make them challenging for mesh-based methods such as finite element and finite difference, leading to overshooting and oscillations. For  $\nu = 10^{-4}$ , Figure 5 illustrates the NN approximation of the function described in (7.5), using uniform breakpoints (a) and employing dBN to adjust the breakpoints (b). An interesting observation is that the resulting approximation from dBN does not exhibit overshooting or oscillations. This confirms that dBN is capable of successfully adjusting the breakpoints and may have the potential to accurately approximate solutions with boundary and/or interior layers.

It is worth mentioning that the relative  $L^2$ -norm error of the approximation depicted in Figure 5(b) is  $2.12 \times 10^{-3}$ . In [6], similar  $L^2$  errors were obtained using deep NNs with 2962 parameters. In our case, the number of parameters is only 41.

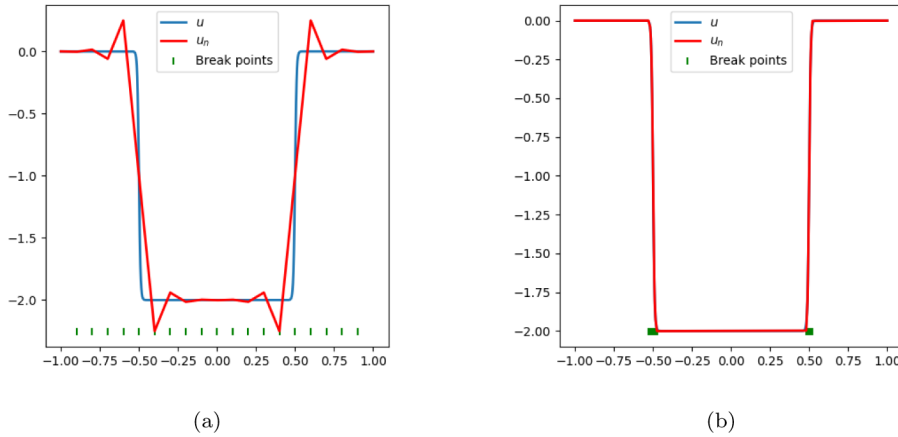


FIG. 5. For  $\nu = \varepsilon^2 = 10^{-4}$ , (a) initial NN model with 20 uniform breakpoints,  $e_n = 0.935$ ; (b) optimized NN model with 20 breakpoints, 200 iterations,  $e_n = 0.130$ .

TABLE 4

Relative error  $e_n$  for 32 breakpoints: initial approximation on uniform breakpoints and approximation after 500 iterations of dBN.

$\nu$	$e_n$ (initial)	$e_n$ (dBN)
$10^{-2}$	$1.63 \times 10^{-1}$	$6.72 \times 10^{-2}$
$10^{-3}$	$5.53 \times 10^{-1}$	$8.08 \times 10^{-2}$
$10^{-4}$	$8.89 \times 10^{-1}$	$7.65 \times 10^{-2}$
$10^{-5}$	$9.69 \times 10^{-1}$	$8.58 \times 10^{-2}$
$10^{-6}$	$9.90 \times 10^{-1}$	$8.92 \times 10^{-2}$

The resulting relative errors (measured in the  $H^1$  seminorm as given in (7.1)) obtained after using dBN for 32 breakpoints and various values of  $\nu$  are shown in Table 4. For each value of  $\nu$ , dBN considerably improves the initial approximation, and the error does not vary significantly with different values of  $\nu$ .

We also present the results using an adaptive finite element method (aFEM). Table 5 compares AdBN and aFEM. In this experiment, both AdBN and aFEM start with 12 breakpoints and are refined 6 times. The same marking strategy from (5.6) is used to determine the intervals for refinement in aFEM. After each refinement, the linear parameters were computed by solving (3.3). At the third refinement of AdBN, with 22 breakpoints, we obtain a more accurate approximation than the one at the final refinement of aFEM with 54 breakpoints. The resulting approximations are shown in Figure 6.

**8. Discussion and conclusion.** The shallow Ritz method improves the order of approximation for one-dimensional nonsmooth elliptic PDEs. The dBN method is an efficient iterative approach for solving the computationally intensive nonconvex optimization problem arising from finding the optimal breakpoints. Extending this method to diffusion-reaction problems presents additional difficulties.

First, unlike the finite element mass matrix, the NN mass matrix is dense and very ill-conditioned. This difficulty is overcome in one dimension through a special factorization of the mass matrix, which was done using both algebraic and geometrical approaches. This factorization enables the  $\mathcal{O}(n)$  computational cost for the inversion of the mass matrix.

TABLE 5  
Comparison of AdBN and aFEM for relative errors  $e_n$ .

Method ( $n$ breakpoints)	$e_n$
AdBN (12)	$2.89 \times 10^{-1}$
AdBN (16)	$1.71 \times 10^{-1}$
AdBN (17)	$1.66 \times 10^{-1}$
AdBN (22)	$1.32 \times 10^{-1}$
AdBN (28)	$9.42 \times 10^{-2}$
AdBN (29)	$8.10 \times 10^{-2}$
AdBN (30)	$7.88 \times 10^{-2}$
aFEM (12)	$9.63 \times 10^{-1}$
aFEM (13)	$9.63 \times 10^{-1}$
aFEM (18)	$9.19 \times 10^{-1}$
aFEM (26)	$8.18 \times 10^{-1}$
aFEM (34)	$5.77 \times 10^{-1}$
aFEM (42)	$2.73 \times 10^{-1}$
aFEM (54)	$1.33 \times 10^{-1}$

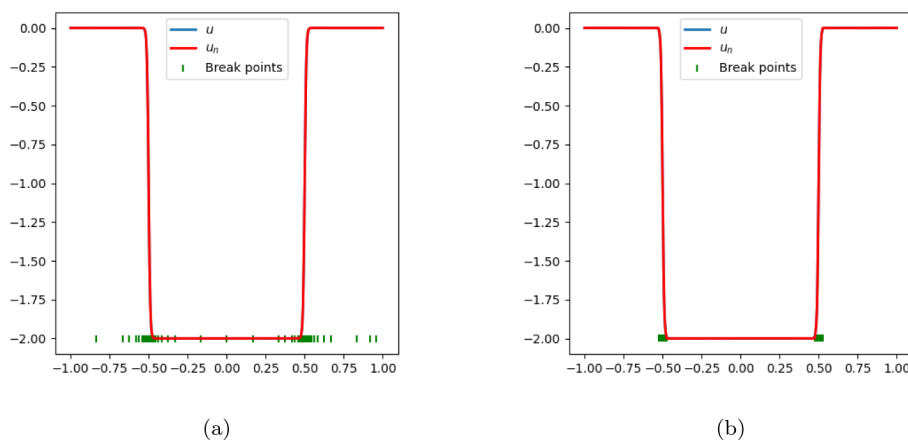


FIG. 6. For  $\nu = \varepsilon^2 = 10^{-4}$ , (a) aFEM approximation ( $n = 12, 13, 18, 26, 34, 42, 54$ ),  $e_n = 0.133$ ; (b) AdBN approximation ( $n = 12, 16, 17, 22, 28, 29, 30$ ),  $e_n = 0.079$ .

Second, the optimality conditions for the nonlinear parameters are no longer nearly decoupled nonlinear algebraic systems, which leads to dense Hessian matrices. Additionally, the Hessians may be singular. To address this, we define a reduced nonlinear system in which the Hessian matrix is invertible. The inverse of the reduced system's Hessian matrix can be computed in  $\mathcal{O}(n)$  operations by exploiting the fact that the inverse of the stiffness matrix is tridiagonal. Consequently, the resulting dBN method is implemented with a computational cost of only  $\mathcal{O}(n)$  per iteration.

Overall, the numerical results demonstrate the efficiency of our method in terms of not only the number of iterations but also the cost per iteration, making a compelling case to pursue the construction of similar solvers for higher dimensional problems. Of particular interest is the application of dBN methods to the singularly perturbed reaction-diffusion problem. For a fixed number of mesh points  $n$ , dBN appears to achieve an accuracy independent of the diffusion coefficient  $\varepsilon^2$ . Furthermore, when adding in adaptivity, AdBN seems to be comparable to FE methods using mesh refinement.

The ideas developed for the dBN method in diffusion-reaction problems are readily applicable to least-squares approximation problems. For both problems (elliptic PDEs and least-squares approximation), local convergence can be guaranteed; a detailed convergence analysis will be presented in a forthcoming paper.

### Appendix A. Inverting the mass matrix.

**A.1. Algebraic approach.** This section derives an inverse formula of the mass matrix through a decomposition into two matrices. The decomposition is based on the fact that matrices with the structure of  $\mathcal{M}$  in (4.1) have tridiagonal inverses.

For  $1 \leq i \leq j \leq n+1$ , let  $m_{ij}$  be the  $(i, j)$ -element of the mass matrix  $M_r(\mathbf{b})$ ; then

$$\begin{aligned} m_{ij} &= \int_0^1 r(x) \sigma(x - b_{i-1}) \sigma(x - b_{j-1}) dx = \int_{b_{j-1}}^1 r(x) (x - b_{i-1}) (x - b_{j-1}) dx \\ &= \int_{b_{j-1}}^1 r(x) (x - 1) (x - b_{j-1}) dx + (1 - b_{i-1}) \int_{b_{j-1}}^1 r(x) (x - b_{j-1}) dx \equiv m_{ij}^1 + m_{ij}^2, \end{aligned}$$

which implies the following decomposition:

$$M_r(\mathbf{b}) = M^1(\mathbf{b}) + M^2(\mathbf{b}) \equiv (m_{ij}^1)_{n \times n} + (m_{ij}^2)_{n \times n}.$$

Both  $M^1(\mathbf{b})$  and  $M^2(\mathbf{b})$  have the same structure as  $\mathcal{M}$  in (4.1) with

$$m_{ij}^1 = \beta_{\max\{i,j\}}^1 \quad \text{and} \quad m_{ij}^2 = \alpha_{\min\{i,j\}}^2 \beta_{\max\{i,j\}}^2,$$

where

$$\begin{aligned} \beta_k^1 &= \int_{b_{k-1}}^1 r(x) (x - 1) (x - b_{k-1}) dx, \quad \alpha_k^2 = 1 - b_{k-1}, \quad \text{and} \\ \beta_k^2 &= \int_{b_{k-1}}^1 r(x) (x - b_{k-1}) dx. \end{aligned}$$

**PROPOSITION A.1.** *The inverse of the mass matrix  $M_r(\mathbf{b})$  is given by*

$$(A.1) \quad M_r(\mathbf{b})^{-1} = M^2(\mathbf{b})^{-1} (M^2(\mathbf{b})^{-1} + M^1(\mathbf{b})^{-1})^{-1} M^1(\mathbf{b})^{-1}.$$

*Proof.* (A.1) is a direct consequence of the fact that

$$M_r(\mathbf{b}) = M^1(\mathbf{b}) (M^2(\mathbf{b})^{-1} + M^1(\mathbf{b})^{-1}) M^2(\mathbf{b}). \quad \square$$

**Remark A.2.** Since  $M^1(\mathbf{b})^{-1}$  and  $M^2(\mathbf{b})^{-1}$  are tridiagonal, so is  $M^1(\mathbf{b})^{-1} + M^2(\mathbf{b})^{-1}$ . Hence,  $M_r(\mathbf{b})^{-1}$  in (A.1) applied to any vector can be computed in  $\mathcal{O}(n)$  operations.

**A.2. Geometric approach.** This section presents another way to invert the mass matrix, based on a factorization of  $M_r(\mathbf{b})$  into the product of three matrices easy to invert. The factorization arises from expressing the global ReLU basis functions in terms of local discontinuous basis functions.

To this end, for  $k = 0, \dots, n$ , let  $I_k = [b_k, b_{k+1})$  and define the local basis functions

$$\varphi_k^0(x) = \begin{cases} 1, & x \in I_k, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad \varphi_k^1(x) = \begin{cases} h_k^{-1}(x - b_k), & x \in I_k, \\ 0 & \text{otherwise.} \end{cases}$$

Since  $\sum_{k=0}^n \varphi_k^0(x) \equiv 1$  in  $I$ , we have

$$(A.2) \quad \text{span}\{1, \sigma(x - b_0), \dots, \sigma(x - b_n)\} \subset \text{span}\{\varphi_k^0(x)\}_{k=0}^n \cup \text{span}\{\varphi_k^1(x)\}_{k=0}^n.$$

Set

$$(A.3) \quad \boldsymbol{\psi}(x) = (\psi_0(x), \dots, \psi_n(x))^T \quad \text{and} \quad \boldsymbol{\varphi}_i(x) = (\varphi_0^i(x), \dots, \varphi_n^i(x))^T,$$

where  $\psi_k(x) = \sigma(x - b_k)$ , and let  $\mathbf{D}(\mathbf{h}) = \text{diag}(h_0, \dots, h_n)$ ,

$$G = \begin{pmatrix} 1 & & & & \\ -1 & 1 & & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix}_{n+1 \times n+1} \quad \text{and} \quad G^{-1} = \begin{pmatrix} 1 & & & & \\ 1 & 1 & & & \\ \vdots & \vdots & \ddots & & \\ 1 & 1 & \dots & 1 \end{pmatrix}_{n+1 \times n+1}.$$

LEMMA A.3. *There exist mappings  $B_0 : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^{n+1}$  and  $B_1 : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^{n+1}$  such that*

$$(A.4) \quad \boldsymbol{\psi} = B_0 \boldsymbol{\varphi}_0 + B_1 \boldsymbol{\varphi}_1.$$

Moreover, we have

$$B_0 = G^{-T} \mathbf{D}(\mathbf{h}) (G^{-T} - I) \quad \text{and} \quad B_1 = G^{-T} \mathbf{D}(\mathbf{h}),$$

where  $I$  is the  $(n+1)$ -order identity matrix.

*Proof.* (A.2) implies that there exist  $B_0$  and  $B_1$  such that (A.4) is valid. To determine  $B_0$  and  $B_1$ , for any  $\mathbf{c} = (c_0, \dots, c_n)^T \in \mathbb{R}^{n+1}$ , let  $v(x) = \mathbf{c}^T \boldsymbol{\psi}(x)$ , then

$$v(x) = \mathbf{c}^T \boldsymbol{\psi}(x) = \mathbf{c}^T B_0 \boldsymbol{\varphi}_0(x) + \mathbf{c}^T B_1 \boldsymbol{\varphi}_1(x).$$

On each  $I_k$ , using the fact that  $v'(x)$  and  $\mathbf{c}^T B_0 \boldsymbol{\varphi}_0(x)$  are both constant, we have

$$\begin{aligned} (\mathbf{c}^T B_1)_{k+1} &= v(b_{k+1}) - v(b_k) = \sum_{i=0}^k c_i \left( \sum_{j=i}^k h_j \right) - \sum_{i=0}^{k-1} c_i \left( \sum_{j=i}^{k-1} h_j \right) \\ &= \sum_{i=0}^k c_i h_k = (\mathbf{D}(\mathbf{h}) G^{-1} \mathbf{c})_{k+1}, \end{aligned}$$

which, together with the arbitrariness of  $\mathbf{c}$ , implies that  $B_1 = G^{-T} \mathbf{D}(\mathbf{h})$ .

By the definitions of  $\boldsymbol{\varphi}_0(x)$  and  $\boldsymbol{\varphi}_1(x)$  and the fact that  $v(b_0) = 0$ , we have

$$\begin{aligned} (\mathbf{c}^T B_0)_{k+1} &= v(b_k) = (v(b_k) - v(b_{k-1})) + (v(b_{k-1}) - v(b_{k-2})) + \dots + (v(b_1) - v(b_0)) \\ &= (\mathbf{D}(\mathbf{h}) G^{-1} \mathbf{c})_k + (\mathbf{D}(\mathbf{h}) G^{-1} \mathbf{c})_{k-1} + \dots + (\mathbf{D}(\mathbf{h}) G^{-1} \mathbf{c})_1 \\ &= ((G^{-1} - I) \mathbf{D}(\mathbf{h}) G^{-1} \mathbf{c})_k, \end{aligned}$$

which, together with the arbitrariness of  $\mathbf{c}$ , implies that  $B_0 = G^{-T} \mathbf{D}(\mathbf{h}) (G^{-T} - I)$ . This completes the proof of the lemma.  $\square$

For  $i, j = 0, 1$ , let

$$\mathbf{D}_{ij}(r) = \int_0^1 r(x) \boldsymbol{\varphi}_i \boldsymbol{\varphi}_j^T dx.$$

For  $k = 0, 1, 2$ , let

$$(A.5) \quad \mathbf{D}_r(\mathbf{s}^k) = \text{diag}(s_0^k(r), \dots, s_n^k(r)) \quad \text{with} \quad s_i^k(r) = \int_{b_i}^{b_{i+1}} r(x)(x - b_i)^k dx.$$

Then, together with  $\mathbf{D}(\mathbf{h}) = \text{diag}(h_0, \dots, h_n)$ , it is easy to see that

$$\mathbf{D}_{00}(r) = \mathbf{D}_r(\mathbf{s}^0), \quad \mathbf{D}_{01}(r) = \mathbf{D}_{10}(r) = \mathbf{D}(\mathbf{h})^{-1} \mathbf{D}_r(\mathbf{s}^1), \quad \text{and} \quad \mathbf{D}_{11}(r) = \mathbf{D}(\mathbf{h})^{-2} \mathbf{D}_r(\mathbf{s}^2).$$

**THEOREM A.4.** *Let  $Q = G\mathbf{D}(\mathbf{h})^{-1}G$  and let*

$$T_{M_r} = (I - G^T)\mathbf{D}_{00}(r)(I - G) + (I - G^T)\mathbf{D}_{01}(r)G + G^T\mathbf{D}_{10}(r)(I - G) + G^T\mathbf{D}_{11}(r)G;$$

*then the mass matrix  $M_r(\mathbf{b})$  defined in (3.2) has the following factorization:*

$$(A.6) \quad M_r(\mathbf{b}) = Q^{-T} T_{M_r} Q^{-1}.$$

*Proof.* By (A.4) and the fact that  $B_0 = B_1(G^{-T} - I)$ , we have

$$\begin{aligned} M_r(\mathbf{b}) &= \int_0^1 r(x) \boldsymbol{\psi} \boldsymbol{\psi}^T dx = B_0 \mathbf{D}_{00} B_0^T + B_0 \mathbf{D}_{01} B_1^T + B_1 \mathbf{D}_{10} B_0^T + B_1 \mathbf{D}_{11} B_1^T \\ &= B_1 \{ (G^{-T} - I) \mathbf{D}_{00} (G^{-1} - I) + (G^{-T} - I) \mathbf{D}_{01} + \mathbf{D}_{10} (G^{-1} - I) + \mathbf{D}_{11} \} B_1^T \\ &= B_1 G^{-T} \{ (I - G^T) \mathbf{D}_{00} (I - G) + (I - G^T) \mathbf{D}_{01} G \\ &\quad + G^T \mathbf{D}_{10} (I - G) + G^T \mathbf{D}_{11} G \} G^{-1} B_1^T, \end{aligned}$$

which, together with the fact that  $Q^{-1} = G^{-1} B_1^T$ , implies (A.6).  $\square$

*Remark A.5.* The transformation in (A.4) leads to a similar factorization of the stiffness matrix as  $A_a(\mathbf{b}) = Q^{-T} T_{A_a} Q^{-1}$  with  $T_{A_a} = G^T \mathbf{D}(\mathbf{h})^{-2} \mathbf{D}_a(\mathbf{s}^0) G$  being tridiagonal, where  $\mathbf{D}_a(\mathbf{s}^0)$  is defined similarly as in (A.5). Therefore, the sum of the stiffness matrix and the mass matrix satisfies that  $A_a(\mathbf{b}) + M_r(\mathbf{b}) = Q^{-T} (T_{A_a} + T_{M_r}) Q^{-1}$ .

#### REFERENCES

- [1] M. AINSWORTH AND Y. SHIN, *Plateau phenomenon in gradient descent training of ReLU networks: Explanation, quantification and avoidance*, SIAM J. Sci. Comput., 43 (2020), pp. A3438–A3468, <https://doi.org/10.1137/20M1353010>.
- [2] M. AINSWORTH AND Y. SHIN, *Active neuron least squares: A training method for multivariate rectified neural networks*, SIAM J. Sci. Comput., 44 (2022), pp. A2253–A2275, <https://doi.org/10.1137/21M1460764>.
- [3] J. BERG AND K. NYSTRÖM, *A unified deep artificial neural network approach to partial differential equations in complex geometries*, Neurocomputing, 317 (2018), pp. 28–41, <https://doi.org/10.1016/j.neucom.2018.06.056>.
- [4] H. G. BURCHARD, *Splines (with optimal knots) are better*, Appl. Anal., 3 (1974), pp. 309–319, <https://doi.org/10.1080/00036817408839073>.
- [5] Z. CAI, J. CHEN, AND M. LIU, *Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation*, J. Comput. Phys., 443 (2021), 110514, <https://doi.org/10.1016/j.jcp.2021.110514>.
- [6] Z. CAI, J. CHEN, M. LIU, AND X. LIU, *Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs*, J. Comput. Phys., 420 (2020), 109707, <https://doi.org/10.1016/j.jcp.2020.109707>.
- [7] Z. CAI, T. DING, M. LIU, X. LIU, AND J. XIA, *Matrix Analysis for Shallow ReLU Neural Network Least-Squares Approximations*, manuscript.
- [8] Z. CAI, T. DING, M. LIU, X. LIU, AND J. XIA, *A Structure-Guided Gauss-Newton Method for Shallow ReLU Neural Network*, arXiv:2404.05064v1 [cs.LG], 2024.

- [9] Z. CAI, A. DOKTOROVA, R. D. FALGOUT, AND C. HERRERA, *Efficient Shallow Ritz Method for 1D Diffusion Problems*, <https://arxiv.org/abs/2404.17750>, 2024.
- [10] Z. CAI AND S. ZHANG, *Recovery-based error estimators for interface problems: Conforming linear elements*, *SIAM J. Numer. Anal.*, 47 (2009), pp. 2132–2156, <https://doi.org/10.1137/080717407>.
- [11] E. C. CYR, M. A. GULIAN, R. G. PATEL, M. PEREGO, AND N. A. TRASK, *Robust training and initialization of deep neural networks: An adaptive basis viewpoint*, in *Proceedings of the First Mathematical and Scientific Machine Learning Conference*, *Proc. Mach. Learn. Res.* 107, PMLR, 2020, pp. 512–536, <http://proceedings.mlr.press/v107/cyr20a/cyr20a.pdf>.
- [12] T. DOCKHORN, *A Discussion on Solving Partial Differential Equations Using Neural Networks*, arXiv:1904.07200 [cs.LG], 2019.
- [13] W. E AND B. YU, *The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems*, *Commun. Math. Stat.*, 6 (2018), pp. 1–12, <https://doi.org/10.1007/s40304-018-0127-z>.
- [14] I. FRIED, *The  $l_2$  and  $l_\infty$  condition numbers of the finite element stiffness and mass matrices, and the pointwise convergence of the method*, in *The Mathematics of Finite Elements and Applications*, J. Whiteman, ed., Academic Press, New York, 1973, pp. 163–174, <https://doi.org/10.1016/B978-0-12-747250-8.50013-8>.
- [15] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 4th ed., Johns Hopkins Stud. Math. Sci., Johns Hopkins University Press, Baltimore, 2013.
- [16] G. H. GOLUB AND V. PEREYRA, *The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate*, *SIAM J. Numer. Anal.*, 10 (1973), pp. 413–432, <https://doi.org/10.1137/0710036>.
- [17] Q. HONG, J. W. SIEGEL, Q. TAN, AND J. XU, *On the Activation Function Dependence of the Spectral Bias of Neural Networks*, arXiv:2208.04924 [cs.LG], 2022.
- [18] M. LIU AND Z. CAI, *Adaptive two-layer ReLU neural network: II. Ritz approximation to elliptic PDEs*, *J. Comput. Math.*, 113 (2022), pp. 103–116, <https://doi.org/10.1016/j.camwa.2022.03.010>.
- [19] M. LIU, Z. CAI, AND J. CHEN, *Adaptive two-layer ReLU neural network: I. Best least-squares approximation*, *J. Comput. Math.*, 113 (2022), pp. 34–44, <https://doi.org/10.1016/j.camwa.2022.03.005>.
- [20] M. LIU, Z. CAI, AND K. RAMANI, *Deep Ritz method with adaptive quadrature for linear elasticity*, *Comput. Methods Appl. Mech. Engrg.*, 415 (2023), 116229, <https://doi.org/10.1016/j.cma.2023.116229>.
- [21] X. LIU AND Y. YUAN, *On the separable nonlinear least squares problems*, *J. Comput. Math.*, 26 (2008), pp. 390–403, <http://www.jstor.org/stable/43693546>.
- [22] J. PARK, J. XU, AND X. XU, *A Neuron-Wise Subspace Correction Method for the Finite Neuron Method*, arXiv:2211.12031 [math.NA], 2022.
- [23] M. RAISSI, P. PERDIKARIS, AND G. KARNIADAKIS, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, *J. Comput. Phys.*, 378 (2019), pp. 686–707, <https://doi.org/10.1016/j.jcp.2018.10.045>.
- [24] J. R. RICE, *A metalgorithm for adaptive quadrature*, *J. ACM*, 22 (1975), pp. 61–82, <https://doi.org/10.1145/321864.321870>.
- [25] A. RUHE AND P. A. WEDIN, *Algorithms for separable nonlinear least squares problems*, *SIAM Rev.*, 22 (1980), pp. 318–337, <https://doi.org/10.1137/1022057>.
- [26] J. SIRIGNANO AND K. SPILIOPOULOS, *DGM: A deep learning algorithm for solving partial differential equations*, *J. Comput. Phys.*, 375 (2018), pp. 1339–1364, <https://doi.org/10.1016/j.jcp.2018.08.029>.