Journal of Computational Physics 420 (2020) 109707

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs ☆

Zhiqiang Cai [a], Jingshuang Chen [a], Min Liu [b,*], Xinyu Liu [a]

[a] Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907-2067, United States of America
[b] School of Mechanical Engineering, Purdue University, 585 Purdue Mall, West Lafayette, IN 47907-2088, United States of America

ARTICLE INFO

Article history:
Received 7 November 2019
Received in revised form 2 June 2020
Accepted 3 July 2020
Available online 15 July 2020

Keywords:
Deep least-squares method
Neural network
Elliptic PDEs

ABSTRACT

This paper studies an unsupervised deep learning-based numerical approach for solving partial differential equations (PDEs). The approach makes use of the deep neural network to approximate solutions of PDEs through the compositional construction and employs least-squares functionals as loss functions to determine parameters of the deep neural network. There are various least-squares functionals for a partial differential equation. This paper focuses on the so-called first-order system least-squares (FOSLS) functional studied in [3], which is based on a first-order system of scalar second-order elliptic PDEs. Numerical results for second-order elliptic PDEs in one dimension are presented.

1. Introduction

Recently, deep neural network (DNN) models have had great success in computer vision, pattern recognition, and many other artificial intelligence tasks. A special feature of DNN is its new way to approximate functions through a composition of multiple linear and activation functions. This leads to some recent studies (see, e.g., [2,5,6,12]) on applications of deep learning to partial differential equations (PDEs).

The idea of solving differential equations using neural networks may be traced back to a paper in 1994 by Dissanayake and Phan-Thien [4]. For a differential equation $L(u) = 0$ defined on the domain Ω with boundary condition $B(u) = 0$ on $\partial\Omega$, a neural network was trained to minimize the following least-square functional

$$\tilde{\mathcal{L}}(v) = \int_{\Omega} \left| L(v)(x) \right|^2 dx + \int_{\partial\Omega} \left| B(v)(x) \right|^2 ds \equiv \|L(v)\|_{0,\Omega}^2 + \|B(v)\|_{0,\partial\Omega}^2, \tag{1.1}$$

where $\|\cdot\|_{0,S}$ is the L^2 norm over subdomain $S = \Omega$ or $\partial\Omega$. Several follow-up works use similar ideas with one hidden layer and sampling points from a mesh to numerically approximate the integrals in $\tilde{\mathcal{L}}$ at each iteration [9–11]. More recently, there is a limited emerging literature on the use of deeper hidden layers to solve PDEs [2,5,12]. It is also illustrated that the sampling points can be obtained by a random sampling of the domain rather than using a mesh, which is beneficial in higher-dimensional problem [2,12]. The least-squares functional defined in (1.1) is based on the original PDEs. For a

☆ Submitted to the editors DATE.
* Corresponding author.
E-mail addresses: caiz@purdue.edu (Z. Cai), chen2042@purdue.edu (J. Chen), liu66@purdue.edu (M. Liu), liu1957@purdue.edu (X. Liu).

https://doi.org/10.1016/j.jcp.2020.109707
0021-9991/© 2020 Elsevier Inc. All rights reserved.

second order PDE, the minimization of $\tilde{\mathcal{L}}(v)$ over admissible functions leads to a fourth-order PDE, which is a more difficult problem than the original one. Moreover, the interior and the boundary integrals in (1.1) are not balanced.

Another formulation of the loss function is to use the energy functional of the underlying PDEs, such as the resulting deep Ritz method recently introduced by E-Yu [6]. For a Poisson problem with Dirichlet boundary conditions, i.e.,

$$\begin{cases} -\Delta u = f, \text{ in } \Omega, \\ \quad u = 0, \text{ on } \partial\Omega, \end{cases}$$

the energy functional is given by

$$\tilde{\mathcal{J}}(v) = \int\limits_{\Omega} \left(\frac{1}{2} |\nabla v(x)|^2 - f(x) v(x) \right) dx. \tag{1.2}$$

This approach is applicable to problems having an underlying minimization principle.

The purpose of this paper is to study an unsupervised deep learning-based numerical approach for solving PDEs. The approach makes use of a deep neural network to approximate solutions of PDEs through the compositional construction and employs least-squares (LS) functionals as loss functions to determine parameters of the deep neural network. There are various least-squares functionals for a partial differential equation, this paper focuses on the FOSLS functional studied in [3], which is based on a first-order system of scalar second-order elliptic PDEs.

The LS methodology has been intensively studied for many PDEs including problems arising from solid and fluid dynamics, radiation transport, magnetohydrodynamics, etc. The method has many attractions. The two striking features are (i) it naturally symmetrizes and stabilizes the original problem; and (ii) value of the corresponding LS functional at the current approximation is an accurate a posteriori error estimator. The first property enables us to work on complex systems which might not have underlying minimization principles, and the second one provides feedback for automatically controlling numerical processes such as the number and the location of quadrature points for evaluating LS functional.

The paper is organized as follows. Section 2 describes the second order elliptic PDEs, the least-squares formulation based on a first-order system of the underlying problem introduced in [3], and proper treatment of boundary conditions when using energy, LS, and FOSLS functionals. Section 3 introduces deep neural network and corresponding deep FOSLS method. Finally, numerical results on three test problems in one dimension are presented in section 4. Moreover, a numerical comparison between uniformly distributed and adaptively obtained quadrature points is reported in section 4.4.

2. Problem formulation

Let Ω be a bounded domain in $\mathbb{R}^d$ with Lipschitz boundary $\partial\Omega = \bar{\Gamma}_D \cup \bar{\Gamma}_N$. Consider the following second-order scalar elliptic partial differential equation:

$$-\text{div}\,(A\nabla u) + Xu = f, \quad \text{in } \Omega \subset \mathbb{R}^d \tag{2.1}$$

with boundary conditions

$$u = g_{_D}, \quad \text{on } \Gamma_D \quad \text{and} \quad -\mathbf{n} \cdot A\nabla u = g_{_N}, \quad \text{on } \Gamma_N, \tag{2.2}$$

where $f \in L^2(\Omega)$, $g_{_D} \in H^{1/2}(\Gamma_D)$, $g_{_N} \in H^{-1/2}(\Gamma_N)$; $A(x)$ is a $d \times d$ symmetric matrix-valued function in $L^2(\Omega)^{d\times d}$; X is a linear differential operator of order at most one; and $\mathbf{n}$ is the outward unit vector normal to the boundary. We assume that A is uniformly positive definite. Possible choices for X include: $Xu = \text{div}\,(\mathbf{b}u)$ with $\mathbf{b} \in L^2(\Omega)^d$ and $Xu = \mathbf{a} \cdot \nabla u + cu$ with $\mathbf{a} \in L^2(\Omega)^d$, $c(x) \in L^2(\Omega)$.

Here and thereafter, we use the standard notation and definitions for the Sobolev space $H^s(\Omega)$ and $H^s(\Gamma)$ for a subset Γ in $\partial\Omega$. The standard associated inner product and norms are denoted by $(\cdot,\cdot)_{s,\Omega}$ and $(\cdot,\cdot)_{s,\Gamma}$ and by $\|\cdot\|_{s,\Omega}$ and $\|\cdot\|_{s,\Gamma}$, respectively. When $s = 0$, $H^0(\Omega)$ coincides with $L^2(\Omega)$. Denote the corresponding norms on product space $H^s(\Omega)^d$ by $\|\cdot\|_{s,\,\Omega,\,d}$ and $|\cdot|_{s,\,\Omega,\,d}$. When there is no ambiguity, the subscript Ω and d in the designation of norms will be suppressed.

2.1. Least-squares formulations

Problem (2.1)-(2.2) is non-symmetric in general and, hence, has no underlying minimization principle. To make use of the deep neural network, we will employ LS principles. There are many LS formulations for problem (2.1). For example, a direct application of the LS principle to problem (2.1) leads to a LS functional defined in (2.14) which is similar to that in (1.1) but with different boundary terms. In this section, we describe the FOSLS formulation introduced in [3] which is based on a first-order system of problem (2.1)-(2.2).

To this end, introducing the flux variable $\boldsymbol{\sigma} = -A\nabla u$, the second-order problem in (2.1) may be rewritten as a first-order system:

$$\begin{cases} \operatorname{div}\boldsymbol{\sigma} + Xu &= f, \text{ in } \Omega, \\ \boldsymbol{\sigma} + A\nabla u &= \mathbf{0}, \text{ in } \Omega \end{cases} \tag{2.3}$$

with boundary conditions

$$u = g_D, \quad \text{on } \Gamma_D \quad \text{and} \quad \mathbf{n}\cdot\boldsymbol{\sigma} = g_N, \quad \text{on } \Gamma_N. \tag{2.4}$$

Let

$$H(\text{div};\Omega) \equiv \left\{\mathbf{v}\in L^2(\Omega)^d : \operatorname{div}\mathbf{v}\in L^2(\Omega)\right\}.$$

Denote subsets of $H^1(\Omega)$ and $H(\text{div};\Omega)$ satisfying non-homogeneous boundary conditions by

$$H^1_{D,g}(\Omega) = \{v\in H^1(\Omega) : v|_{\Gamma_D} = g_D\} \text{ and } H_{N,g} = \{\boldsymbol{\tau}\in H(\text{div};\Omega) : \boldsymbol{\tau}\cdot\mathbf{n}|_{\Gamma_N} = g_N\}$$

respectively. When $g_D = 0$ and $g_N = 0$, these subsets become subspaces and are denoted by $H^1_D(\Omega)$ and $H_N(\text{div};\Omega)$. Let

$$\mathcal{V}_g = H_{N,g}(\text{div};\Omega)\times H^1_{D,g}(\Omega) \quad \text{and} \quad \mathcal{V}_0 = H_N(\text{div};\Omega)\times H^1_D(\Omega),$$

then the FOSLS formulation is to find $(\boldsymbol{\sigma}, u)\in\mathcal{V}_g$ such that

$$\tilde{\mathcal{G}}(\boldsymbol{\sigma}, u;\mathbf{f}) = \min_{(\boldsymbol{\tau},v)\in\mathcal{V}_g}\tilde{\mathcal{G}}(\boldsymbol{\tau}, v;\mathbf{f}), \tag{2.5}$$

where $\mathbf{f} = (f, g_D, g_N)$ and the FOSLS functional is defined by

$$\tilde{\mathcal{G}}(\boldsymbol{\tau}, v;\mathbf{f}) = \|\operatorname{div}\boldsymbol{\tau} + Xv - f\|^2_{0,\Omega} + \|A^{-1/2}\boldsymbol{\tau} + A^{1/2}\nabla v\|^2_{0,\Omega}. \tag{2.6}$$

It has been proved in [3] that the homogeneous FOSLS functional $\tilde{\mathcal{G}}(\boldsymbol{\tau}, v;\mathbf{0})$ is coercive and bounded in $\mathcal{V}_0$, i.e., there exist positive constants c_1 and c_2 such that

$$c_1|||(\boldsymbol{\tau}, v)|||^2 \le \tilde{\mathcal{G}}(\boldsymbol{\tau}, v;\mathbf{0}) \le c_2|||(\boldsymbol{\tau}, v)|||^2 \tag{2.7}$$

for all $(\boldsymbol{\tau}, v)\in\mathcal{V}_0$, where the FOSLS energy norm is given by

$$|||(\boldsymbol{\tau}, v)||| = \left(\|\boldsymbol{\tau}\|^2_{0,\Omega} + \|\operatorname{div}\boldsymbol{\tau}\|^2_{0,\Omega} + \|v\|^2_{1,\Omega}\right)^{1/2}.$$

The corcevity and boundedness of the homogeneous FOSLS functional further implies that the FOSLS minimization problem in (2.5) is well-posed, i.e., (2.5) has a unique solution (see [3] for a detail discussion).

2.2. Treatment of boundary conditions

Unlike finite element functions, it is not easy for a deep neural network function to satisfy a prescribed boundary condition. Such a difficulty was observed in [6] for the deep Ritz method. To circumvent this obstacle, for a Poisson equation (i.e., $A = I$ and $X = 0$) with pure Dirichlet boundary conditions (i.e., $\Gamma_N = \emptyset$), they add the essential boundary conditions to the energy functional:

$$\tilde{\mathcal{J}}(v) = \int_\Omega \left(\frac{1}{2}|\nabla v(x)|^2 - f(x)v(x)\right)dx + \beta\,\|v(x) - g_D\|^2_{0,\partial\Omega}, \tag{2.8}$$

where β is a parameter to be determined. When the data vanishes, i.e., $f = 0$ and $g_D = 0$, the modified energy functional becomes

$$\tilde{\mathcal{J}}(v) = \frac{1}{2}\,\|\nabla v\|^2_{0,\Omega} + \beta\|v(x)\|^2_{0,\partial\Omega}.$$

By the Sobolev trace theorem, the interior and boundary norms in the above formula are not in the same scale. Specifically, the boundary norm is 1/2-order weaker than the interior norm. This consideration suggests the following modified energy functional of (2.8)

$$\mathcal{J}(v;\mathbf{f}) = \int_\Omega \left(\frac{1}{2}|\nabla v(x)|^2 - f(x)v(x)\right)dx + \beta\,\|v(x) - g_D\|^2_{1/2,\partial\Omega}, \tag{2.9}$$

where $\mathbf{f} = (f, g_D)$ and β is a constant. For the Poisson equation with the mixed boundary conditions in (2.2), the energy functional becomes

$$\mathcal{J}(v;\mathbf{f}) = \frac{1}{2}\,\|\nabla v\|^2_{0,\Omega} - \left(\int\limits_{\Omega} f(x)v(x)\,dx + \int\limits_{\Gamma_N} g_{_N}\,v\,dS \right) + \beta\,\|v(x) - g_{_D}\|^2_{1/2,\Gamma_D} \tag{2.10}$$

where $\mathbf{f} = (f, g_{_D}, g_{_N})$ and β is a constant. The minimization problem based on the above energy functional is to find $u \in H^1(\Omega)$ such that

$$\mathcal{J}(u;\,\mathbf{f}) = \min_{v\in H^1(\Omega)} \mathcal{J}(v;\,\mathbf{f}). \tag{2.11}$$

For the FOSLS formulation defined in (2.5), both the Dirichlet and Neumann boundary conditions are essential boundary conditions and, hence, we need to add them to the FOSLS functional with proper scales:

$$\begin{aligned}\mathcal{G}(\boldsymbol{\tau},\,v;\mathbf{f}) = \|\mathrm{div}\,\boldsymbol{\tau} + Xv - f\|^2_{0,\Omega} + \|A^{-1/2}\boldsymbol{\tau} + A^{1/2}\nabla v\|^2_{0,\Omega}\\ +\alpha_D\|v - g_{_D}\|^2_{1/2,\Gamma_D} + \alpha_N\|\mathbf{n}\cdot\boldsymbol{\tau} - g_{_N}\|^2_{-1/2,\Gamma_N}\end{aligned} \tag{2.12}$$

for all $(\boldsymbol{\tau},\,v) \in \mathcal{V} \equiv H(\mathrm{div};\Omega) \times H^1(\Omega)$, where α_D and α_N are constants and may be chosen to be one. Now, the corresponding FOSLS formulation is to find $(\boldsymbol{\sigma},\,u) \in \mathcal{V}$ such that

$$\mathcal{G}(\boldsymbol{\sigma},\,u;\mathbf{f}) = \min_{(\boldsymbol{\tau},v)\in\mathcal{V}} \mathcal{G}(\boldsymbol{\tau},\,v;\mathbf{f}). \tag{2.13}$$

It has been proved that the homogeneous FOSLS functional $\mathcal{G}(\boldsymbol{\tau},\,v;\mathbf{0})$ is coercive and bounded in $\mathcal{V}$. This in turn implies that the LS minimization problem in (2.13) is well-posed in the space $\mathcal{V}$ without strongly enforced boundary conditions.

For the LS functional defined in (1.1), the norm on boundary conditions is weaker than that for the equation; moreover, the Dirichlet and the Neumann boundary conditions are not treated differently. A balanced LS functional for problem (2.1) is as follows:

$$\mathcal{L}(v;\,\mathbf{f}) = \|-\mathrm{div}\,(A\nabla\,v) + Xv - f\|^2_{0,\Omega} + \beta_D\|v - g_{_D}\|^2_{3/2,\Gamma_D} + \beta_N\|\mathbf{n}\cdot A\nabla\,v + g_{_N}\|^2_{1/2,\Gamma_N}, \tag{2.14}$$

where $\mathbf{f} = (f, g_{_D}, g_{_N})$. Now, the corresponding LS formulation is to find $u \in H^2(\Omega)$ such that

$$\mathcal{L}(u;\,\mathbf{f}) = \min_{v\in H^2(\Omega)} \mathcal{L}(v;\,\mathbf{f}). \tag{2.15}$$

Assume that the solution of problem (2.1)-(2.2) is H^2 regular. Then it is a direct consequence that the homogeneous LS functional $\mathcal{L}(v;\,\mathbf{0})$ is coercive and bounded in $H^2(\Omega)$. This implies that problem (2.15) is well-posed by Lax-Milgram theorem [3].

Remark 2.1. Note that the LS formulation (2.14)-(2.15) is only applicable to problems whose solutions are sufficiently smooth, more precisely, at least in $H^2(\Omega)$. This, in turn, implies that a DNN with non-piecewise-linear activation function is needed when using the LS functional as the loss function.

3. The deep FOSLS

This section describes deep neural network structures and the deep FOSLS method. Discussions on numerical evaluation of the FOSLS functional are, in principle, valid for both the energy and the LS functionals. Moreover, similar error bounds in (3.8) and (3.9) for the deep FOSLS is also valid for the energy and the LS functionals in the respective H^1 and H^2 norms.

3.1. Deep neural network structure

For convenience of audiences in numerical analysis, in this section we describe the DNN structure through functional terminology. A deep neural network defines a function

$$\mathcal{N}: x \in \mathbb{R}^d \longrightarrow y = \mathcal{N}(x) \in \mathbb{R}^c,$$

where d and c are dimensions of input $x \in \mathbb{R}^d$ and output $y = \mathcal{N}(x) \in \mathbb{R}^c$, respectively. The DNN function $\mathcal{N}(x)$ is typically represented as compositions of many different layers of functions:

$$y = \mathcal{N}(x) = \mathcal{N}^{(L)} \circ \cdots \mathcal{N}^{(2)} \circ \mathcal{N}^{(1)}(x), \tag{3.1}$$

where the symbol $\circ$ denotes the composition of functions: $f \circ g(x) = f(g(x))$, and L is the depth of the network. In this case, $\mathcal{N}^{(1)}$ is called the first layer of the network, $\mathcal{N}^{(2)}$ is called the second layer, and so on. All layers except the last one $\mathcal{N}^{(L)}$ are called hidden layers since they are hidden in between input and output (see Fig. 3.1).

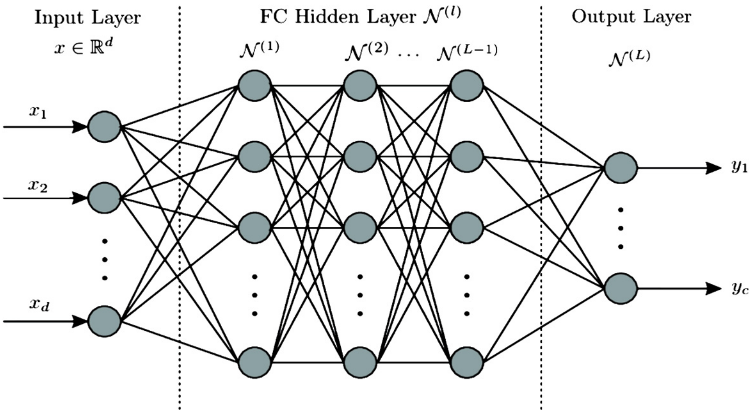


Fig. 3.1. Fully-Connected Neural Network.

Each layer is typically a vector-valued function. The choice of the function $\mathcal{N}^{(l)}(x)$ is guided by many mathematical and engineering disciplines. In this paper, we use fully connected (FC) hidden layers. A FC layer $\mathcal{N}^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ is defined as a composition of a linear transformation $T^l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ and an activation function $\psi^l : \mathbb{R} \to \mathbb{R}$ as follows:

$$\mathcal{N}^{(l)}(x^{l-1}) = \psi^l \circ T^l(x^{l-1}) = \psi^l(W^l x^{l-1} + b^l), \quad \text{for } x^{l-1} \in \mathbb{R}^{n_{l-1}}, \tag{3.2}$$

where $W^l = \left(w^l_{ij}\right)_{n_l \times n_{l-1}} \in \mathbb{R}^{n_l \times n_{l-1}}$, $b^l \in \mathbb{R}^{n_l}$, and application of ψ^l to a vector $z \in \mathbb{R}^{n_l}$ is defined component-wisely, i.e., $\psi^l(z) = \left(\psi^l(z_i)\right)_{n_l \times 1}$. Components of W^l and b^l are called weights and bias, respectively, and are parameters to be determined (trained). Each component of the vector-valued function $\mathcal{N}^{(l)}$ is interpreted as a neuron and the dimensionality n_l defines the width or the number of neurons of the lth layer in a network. The $n_0 = d$ and $n_L = c$ are the respective dimensions of input and output. There are $n_l \times (n_{l-1} + 1)$ parameters at the lth layer, and the total number of parameters of the DNN function $\mathcal{N}(x)$ defined in (3.1) is given by

$$N = \sum_{l=1}^{L} n_l \times (n_{l-1} + 1).$$

Choices of the activation function ψ have influences on the output of a model, its accuracy, and the computational efficiency of training. A commonly used activation function is the leaky ReLU defined as follows:

$$\psi(x) = \begin{cases} x, & \text{if } x > 0, \\ 0.01x, & \text{otherwise}, \end{cases} \tag{3.3}$$

which is a continuous piecewise linear function. A DNN with a piecewise linear activation function is capable of generating rich function classes. For instance, as discussed in [1,13], a DNN with at most $\lceil \log_2(d+1) \rceil$ hidden layers can represent piecewise linear function $\mathbb{R}^d \to \mathbb{R}$. Furthermore, by introducing some special network structures and adding more neurons as well as layers, DNN is able to approximate a large class of functions other than linear [14].

The sigmoid function is another commonly used activation function, which is defined by

$$\psi(x) = \frac{1}{1 + e^{-x}}, \quad x \in \mathbb{R}. \tag{3.4}$$

Both the leaky ReLU and the sigmoid activation functions are depicted in Fig. 3.2. The leaky ReLU is easier to compute than the non-linear sigmoid function. But using a smooth activation function such as the sigmoid function is essential for the deep LS method based on the LS functional defined in either (1.1) or (2.14). This is because functions generated by a DNN with a continuous piecewise linear activation function is only in $H^1(\Omega)$.

3.2. Deep FOSLS

The idea of the deep FOSLS is to employ DNN functions for approximating the solution $(\boldsymbol{\sigma}(x), u(x))$ of the FOSLS minimization problem in (2.5). More specifically, for each $x \in \Omega \subset \mathbb{R}^d$, a DNN is implemented to compute an approximation $(\hat{\boldsymbol{\sigma}}(x, \Theta), \hat{u}(x, \Theta))$ at the point x, where $\Theta \in \mathbb{R}^N$ stands for all parameters (weights and biases) in the DNN. A deep FOSLS approximation is to find $(\hat{\boldsymbol{\sigma}}(x, \Theta), \hat{u}(x, \Theta))$ such that

$$\mathcal{G}(\hat{\boldsymbol{\sigma}}(x, \Theta), \hat{u}(x, \Theta); \mathbf{f}) = \min_{\tilde{\Theta} \in \mathbb{R}^N} \mathcal{G}(\hat{\boldsymbol{\tau}}(x, \tilde{\Theta}), \hat{v}(x, \tilde{\Theta}); \mathbf{f}). \tag{3.5}$$

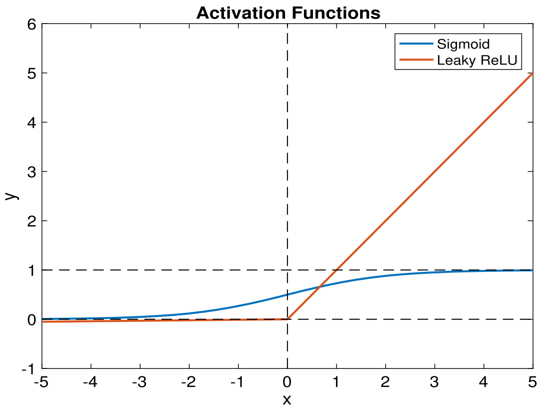


Fig. 3.2. Activation functions.

Instead of evaluating the FOSLS functional analytically, in this paper we consider numerical approximation to the FOSLS functional. This means that we will use numerical quadrature to approximate integrals of the FOSLS functional. For simplicity and generality in high dimensions, we will adopt composite “mid-point” quadrature rule. To this end, let

$$\mathcal{T} = \{K \,:\, K \text{ is an open subdomain of } \Omega\}$$

be a partition of the domain Ω. Here, the partition means that union of all subdomains of $\mathcal{T}$ equal the whole domain Ω and that any two distinct subdomains of $\mathcal{T}$ have no intersection; more precisely,

$$\bar{\Omega} = \cup_{K\in\mathcal{T}} \bar{K} \quad \text{and} \quad K \cap T = \emptyset, \quad \forall\, K,\, T \in \mathcal{T}.$$

Denote by $\mathcal{E}_D = \{E \,:\, E = \partial K \cap \Gamma_D,\ \forall K \in \mathcal{T}\}$ and $\mathcal{E}_N = \{K \,:\, E = \partial K \cap \Gamma_N,\ \forall K \in \mathcal{T}\}$ the partitions of Γ_D and Γ_N associated with the partition $\mathcal{T}$, respectively. Let x_K and x_E be interior points of $K \in \mathcal{T}$ and $E \in \mathcal{E}_S$ with $S = D$ or N, respectively. The x_K and x_E will be used as quadrature points below. Note that quadrature points are fundamentally different from sampling points used in the setting of supervised learning.

Since Sobolev norms $\|\cdot\|_{1/2}$ and $\|\cdot\|_{-1/2}$ in the FOSLS functional are not computationally feasible, we will approximate them by weighted L^2 norms with local weights $h_E^{-1/2}$ and $h_E^{1/2}$, respectively, where h_E is the diameter of E. This idea leads to the following discrete FOSLS functional:

$$\begin{aligned}\hat{\mathcal{G}}(\hat{\boldsymbol{\tau}}(x,\Theta),\ \hat{v}(x,\Theta);\mathbf{f}) = &\sum_{K\in\mathcal{T}} \Big(\big(\operatorname{div}\hat{\boldsymbol{\tau}} + X\hat{v} - f\big)^2(x_K,\Theta) + \big(A^{-1/2}\hat{\boldsymbol{\tau}} + A^{1/2}\nabla\hat{v}\big)^2(x_K,\Theta)\Big)|K| \\ &+\alpha_D \sum_{E\in\mathcal{E}_D} \big(\hat{v} - g_D\big)^2(x_E,\Theta)|E|\,h_E^{-1} + \alpha_N \sum_{E\in\mathcal{E}_N} \big(\mathbf{n}\cdot\hat{\boldsymbol{\tau}} - g_N\big)^2(x_E,\Theta)|E|\,h_E,\end{aligned} \tag{3.6}$$

where $|K|$ and $|E|$ are the d and $d-1$ dimensional measures of K and E respectively; and α_D and α_N are two positive constants. For given data f, g_D, and g_N, the value of the discrete FOSLS functional at $(\hat{\boldsymbol{\tau}},\ \hat{v})$ is a function of the parameters Θ. Then the discrete deep FOSLS approximation is to find $(\hat{\boldsymbol{\sigma}}_{\mathcal{T}}(x,\Theta),\ \hat{u}_{\mathcal{T}}(x,\Theta))$ such that

$$\hat{\mathcal{G}}(\hat{\boldsymbol{\sigma}}_{\mathcal{T}}(x,\Theta),\ \hat{u}_{\mathcal{T}}(x,\Theta);\mathbf{f}) = \min_{\tilde{\Theta}\in\mathbb{R}^N} \hat{\mathcal{G}}(\hat{\boldsymbol{\tau}}(x,\tilde{\Theta}),\ \hat{v}(x,\tilde{\Theta});\mathbf{f}). \tag{3.7}$$

Remark 3.1. Similar to the discrete FOSLS functional defined in (3.6), the discrete energy and the discrete LS functionals are defined as follows:

$$\begin{aligned}\hat{\mathcal{J}}(\hat{v}(x,\Theta_u);\mathbf{f}) = &\sum_{K\in\mathcal{T}} \left(\frac{1}{2}|\nabla\hat{v}|^2 - f\hat{v}\right)(x_K,\Theta_u)|K| - \sum_{E\in\mathcal{E}_N} \big(g_N\hat{v}\big)(x_E,\Theta_u)|E| \\ &+\alpha_D \sum_{E\in\mathcal{E}_D} \big(\hat{v} - g_D\big)^2(x_E,\Theta_u)|E|\,h_E^{-1}\end{aligned}$$

and
$$\begin{aligned}\hat{\mathcal{L}}(\hat{v}(x,\Theta_u);\mathbf{f}) = &\sum_{K\in\mathcal{T}} \big(-\operatorname{div}(A\nabla\hat{v}) + X\hat{v} - f\big)^2(x_K,\Theta_u)|K| \\ &+\alpha_D \sum_{E\in\mathcal{E}_D} \big(\hat{v} - g_D\big)^2(x_E,\Theta_u)|E|h_E^{-3} + \alpha_N \sum_{E\in\mathcal{E}_N} \big(\mathbf{n}\cdot A\nabla\hat{v} + g_N\big)^2(x_E,\Theta_u)|E|h_E^{-1},\end{aligned}$$

respectively, where α_D and α_N are positive constants.

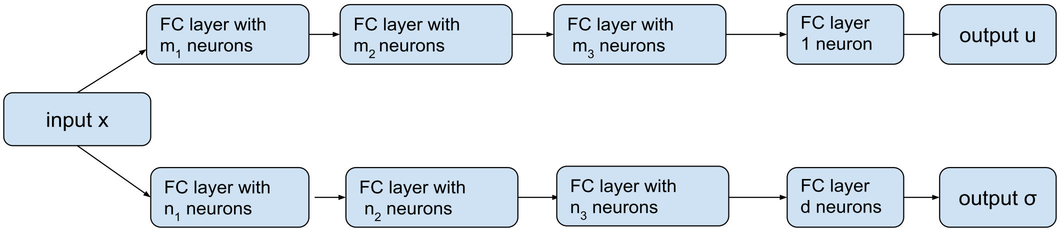

input x
FC layer with m$_1$ neurons
FC layer with m$_2$ neurons
FC layer with m$_3$ neurons
FC layer 1 neuron
output u
FC layer with n$_1$ neurons
FC layer with n$_2$ neurons
FC layer with n$_3$ neurons
FC layer d neurons
output σ

Fig. 4.1. Four-layer neural network for training $u(x)$ and $\boldsymbol{\sigma}(x)$. Each block consists of one fully-connected (FC) layer. x is an arbitrary point in the domain $\Omega \subset \mathbb{R}^d$, and m_l and n_l are the respective numbers of neurons in the upper and lower branches at the l^{th} layer.

To understand approximation property of the discrete deep FOSLS method, by the triangle inequality, we have

$$|||(\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}}_{\mathcal{T}}, u - \hat{u}_{\mathcal{T}})||| \le |||(\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}}, u - \hat{u})||| + |||(\hat{\boldsymbol{\sigma}} - \hat{\boldsymbol{\sigma}}_{\mathcal{T}}, \hat{u} - \hat{u}_{\mathcal{T}})|||, \tag{3.8}$$

where the first term represents the approximation error caused by the deep neural network and the second term is the numerical error by evaluating the FOSLS functional through numerical quadrature. How to estimate the former is still an open problem. The latter can be computed to a desired accuracy through either uniform or adaptive partition of the Ω, Γ_D, and Γ_N. A detailed algorithmic and theoretical discussions of the second term will be presented in a forthcoming paper.

In (3.8), $(\hat{\boldsymbol{\sigma}}_{\mathcal{T}}(x, \Theta), \hat{u}_{\mathcal{T}}(x, \Theta))$ is assumed to be the exact solution of the minimization problem in (3.7). In practice, problem (3.7) is solved numerically by an iterative method such as the method of (stochastic) gradient decent. Let $(\hat{\boldsymbol{\sigma}}^k_{\mathcal{T}}(x, \Theta), \hat{u}^k_{\mathcal{T}}(x, \Theta))$ be the algebraic approximation at the kth iterate, then the total error of the discrete deep FOSLS method is bounded by the sum of the DNN approximation error, the quadrature error, and the algebraic error as follows:

$$\left|\left|\left|(\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}}^k_{\mathcal{T}}, u - \hat{u}^k_{\mathcal{T}})\right|\right|\right| \le |||(\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}}, u - \hat{u})||| + |||(\hat{\boldsymbol{\sigma}} - \hat{\boldsymbol{\sigma}}_{\mathcal{T}}, \hat{u} - \hat{u}_{\mathcal{T}})||| + \left|\left|\left|(\hat{\boldsymbol{\sigma}}_{\mathcal{T}} - \hat{\boldsymbol{\sigma}}^k_{\mathcal{T}}, \hat{u}_{\mathcal{T}} - \hat{u}^k_{\mathcal{T}})\right|\right|\right|. \tag{3.9}$$

Again, (3.9) is obtained by the triangle inequality.

4. Numerical experiments

The solution $u(x)$ and the flux $\boldsymbol{\sigma}(x)$ in the FOSLS formulation are independent variables. This observation implies that an efficient DNN structure is to approximate them separately. Hence, a DNN to be employed consists of two branches: the upper and lower branches for the respective u and $\boldsymbol{\sigma}$ (see Fig. 4.1). These two branches have no neuron connection. For numerical experiments in this paper, we use a four-layer neural network. Within each branch, a fully connected layer is implemented.

Let Θ_u and $\Theta_{\boldsymbol{\sigma}}$ represent all parameters in the upper and lower branches, respectively. Denote by $\mathcal{N}^l_u$ and $\mathcal{N}^l_{\boldsymbol{\sigma}}$ the fully connected layer defined in (3.2) for the respective upper and lower branches. The four-layer neural network (see Fig. 4.1) defines two functions $u(x, \Theta_u)$ and $\boldsymbol{\sigma}(x, \Theta_{\boldsymbol{\sigma}})$ by the upper and lower branches:

$$u(x, \Theta_u) = \mathcal{N}^4_u \circ \mathcal{N}^3_u \circ \mathcal{N}^2_u \circ \mathcal{N}^1_u(x) \quad \text{and} \quad \boldsymbol{\sigma}(x, \Theta_{\boldsymbol{\sigma}}) = \mathcal{N}^4_{\boldsymbol{\sigma}} \circ \mathcal{N}^3_{\boldsymbol{\sigma}} \circ \mathcal{N}^2_{\boldsymbol{\sigma}} \circ \mathcal{N}^1_{\boldsymbol{\sigma}}(x),$$

respectively. Activation functions for the hidden and the output layers are usually different depending on the underlying application. In this paper, we use the same activation function for the hidden layers and identity for the output layer. In the numerical experiments, both the leaky ReLU and sigmoid functions are tested for the deep Ritz and the FOSLS methods, while the leaky ReLU activation function may not be used for the deep LS method as discussed in section 3.1. Now, the deep FOSLS method is to find $(\boldsymbol{\sigma}(x, \Theta_{\boldsymbol{\sigma}}), u(x, \Theta_u))$ by minimizing the discrete FOSLS functional defined in (3.6) over parameters $\Theta = (\Theta_u, \Theta_{\boldsymbol{\sigma}})$. The deep LS and Ritz methods are to find $u(x, \Theta_u)$ (using only the upper branch) by minimizing the corresponding discrete LS and energy functionals over parameters Θ_u (Remark 3.1).

To train (numerically compute) parameters Θ associated with the DNN functions $u(x, \Theta_u)$ and $\boldsymbol{\sigma}(x, \Theta_{\boldsymbol{\sigma}})$, the Adam optimizer version of gradient descent [8] is implemented as an iterative method to numerically solve the minimization problem in (3.7). The iterative parameter (may vary at each iteration) of the method of gradient decent is called the step size or learning rate.

Test problems in this section consist of a Poisson, a singularly perturbed reaction-diffusion equation, and an interface problem, all in one dimension. As discussed in section 3.2, the FOSLS functional, similarly the energy and the LS functionals, are evaluated numerically based on a partition of the domain. For numerical results reported in sections 4.1, 4.2, and 4.3, we use a uniform partition of interval $[a, b]$: $a = x_0 < x_1 < \cdots < x_n = b$ with $x_i = a + ih$ and $h = (b - a)/n$ for $i = 0, 1, \ldots, n$. Quadrature points in (3.6) are chosen to be the midpoints of subintervals: $x_{i-1/2} = a + h(2i - 1)/2$ for $i = 1, 2, \ldots, n$. First-order derivative at midpoints in the functionals are approximated by the forward finite difference quotient, $\dfrac{v(x_{i-1/2}) - v(x_{i-1/2} - \tau)}{\tau}$ with $\tau = h/2$.

All experiments are replicated three times to reduce variability of random initialization of the method of gradient decent and the medians of three training results are reported. Numerical results are reported through the true error in the relative

Table 4.1
Relative errors of Poisson equation with different number of quadrature points.

Relative errors Quadrature points	$\frac{\|u-\bar{u}_\tau\|_0}{\|u\|_0}$	$\frac{\lvert u-\bar{u}_\tau\rvert_1}{\lvert u\rvert_1}$	$\frac{\|\boldsymbol{\sigma}-\bar{\boldsymbol{\sigma}}_\tau\|_0}{\|\boldsymbol{\sigma}\|_0}$	$\frac{G^{1/2}(\bar{\boldsymbol{\sigma}}_\tau,\bar{u}_\tau;\mathbf{f})}{\vert\vert\vert(\boldsymbol{\sigma},u)\vert\vert\vert}$
200	0.065238	0.109056	0.056508	0.098030
400	0.048421	0.167703	0.026564	0.095498
800	0.025238	0.106552	0.020481	0.068702
1600	0.024631	0.114932	0.020091	0.063403

L^2 norm and the H^1 seminorm (or the energy norm) (see Tables 4.2, 4.3, and 4.4). Moreover, the exact solution vs numerical approximations are depicted in Figs. 4.2, 4.3, and 4.4. Note that only the figures for the FOSLS functional are presented as reference in Figs. 4.2 and 4.3 since results for the energy and the LS functionals are similar. For the deep FOSLS method, we also report numerical results on the approximation to the flux variable $\boldsymbol{\sigma}$ in the relative L^2 norm and the relative value of the FOSLS functional. A PyTorch implementation is released at https://github.com/janiechen8/DeepLSMethod.

4.1. Poisson equation

The first test problem is a one-dimensional Poisson equation used in [7]:

$$\begin{cases} -u''(x) \;=\; f(x), \;\; x \in \Omega = (0,\,1), \\ \qquad\;\; u \;=\; 0, \qquad x \in \partial\Omega = \{0,\,1\} \end{cases} \tag{4.1}$$

with $f = -40000(x^3 - 2x^2/3 + 173x/1800 + 1/300)e^{-100(x-1/3)^2}$. Problem (4.1) has the following exact solution

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

A four-layer neural network ($m_1 = n_1 = 24$ and $m_2 = m_3 = n_2 = n_3 = 14$) with total 1246 parameters is implemented for the deep FOSLS method.

The first numerical experiment is to show that with sufficient quadrature points for evaluating the FOSLS functional, accuracy of the deep FOSLS method is determined by the approximation property of the DNN structure (3.8). Denote $\bar{u}_\tau$ and $\bar{\boldsymbol{\sigma}}_\tau$ as the network outputs of u and $\boldsymbol{\sigma}$, respectively. Using the leaky ReLU activation function, a fixed learning rate of 0.0005 and 10000 iterations, Table 4.1 shows that 800 quadrature points are enough to accurately evaluate the FOSLS functional.

The goal of the second numerical experiment is to report numerical performances when using different functionals as well as activation functions. With the same learning rate and iteration number, Table 4.2 and Fig. 4.2 show that all three methods are able to accurately approximate the solution of the Poisson equation. Due to smoothness of the exact solution, the deep LS method performs slightly better than the other two methods; moreover, the sigmoid function is more accurate than the leaky ReLU function possibly because of exponential feature of the exact solution.

4.2. Singularly perturbed reaction-diffusion equation

The second test problem is a singularly perturbed reaction-diffusion equation:

$$\begin{cases} -\varepsilon^2 u''(x) + u(x) \;=\; f(x), \;\; x \in \Omega = (-1,\,1), \\ \qquad\qquad\qquad\;\; u \;=\; 0, \qquad x \in \partial\Omega = \{-1,\,1\}. \end{cases} \tag{4.2}$$

For $f = -2\left(\varepsilon - 4x^2\tanh(\frac{1}{\varepsilon}(x^2-\frac{1}{4}))\right)\left(1/\cosh(\frac{1}{\varepsilon}(x^2-\frac{1}{4}))\right)^2 + \tanh(\frac{1}{\varepsilon}(x^2-\frac{1}{4})) - \tanh(\frac{3}{4\varepsilon})$, problem (4.2) has the following exact solution

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

With $\boldsymbol{\sigma} = -\varepsilon^2 u'$, the corresponding FOSLS functional defined in (2.12) is of the form

$$\mathcal{G}(\boldsymbol{\tau}, v; f) = \|\boldsymbol{\tau}' + v - f\|_{0,\Omega}^2 + \left\|\boldsymbol{\tau}/\epsilon + \epsilon v'\right\|_{0,\Omega}^2 + \alpha\,\|u\|_{1/2,\partial\Omega}^2,$$

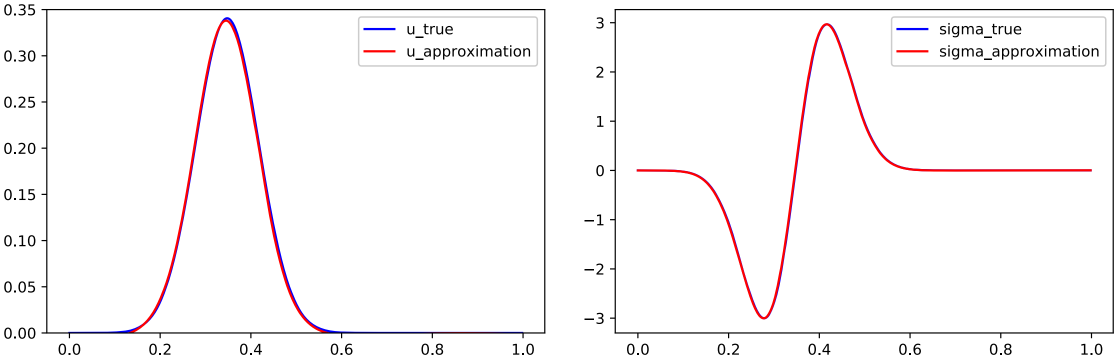
and the corresponding energy norms are $\vert\vert\vert(\boldsymbol{\tau},\,v)\vert\vert\vert = \left(\vert\vert\vert\boldsymbol{\tau}\vert\vert\vert^2 + \vert\vert\vert v\vert\vert\vert^2\right)^{1/2}$ with

$$\vert\vert\vert v\vert\vert\vert = \left(\|v\|_{0,\Omega}^2 + \|\epsilon v'\|_{0,\Omega}^2\right)^{1/2} \quad \text{and} \quad \vert\vert\vert\boldsymbol{\tau}\vert\vert\vert = \left(\|\boldsymbol{\tau}/\epsilon\|_{0,\Omega}^2 + \|\boldsymbol{\tau}'\|_{0,\Omega}^2\right)^{1/2}.$$

The goal of this numerical experiment is to test the performance of deep learning based method for problems with boundary and/or interior layers which pose difficulty for mesh-based methods such as finite element, finite difference, etc.

Table 4.2
Relative errors of Poisson equation with different functionals, activation functions and quadrature points.

Loss and activation \ Relative errors	$\frac{\|u-\bar{u}_\tau\|_0}{\|u\|_0}$	$\frac{\|u-\bar{u}_\tau\|_1}{\|u\|_1}$	$\frac{\|\boldsymbol{\sigma}-\bar{\boldsymbol{\sigma}}_\tau\|_0}{\|\boldsymbol{\sigma}\|_0}$	$\frac{G^{1/2}(\bar{\boldsymbol{\sigma}}_\tau,\bar{u}_\tau;\mathbf{f})}{\|\|(\boldsymbol{\sigma},u)\|\|}$
Energy (LeakyReLU & 800 points)	0.029161	0.160666	–	–
FOSLS (LeakyReLU & 800 points)	0.025238	0.106552	0.020481	0.068702
Energy (Sigmoid & 200 points)	0.013144	0.026246	–	–
LS (Sigmoid & 200 points)	0.008876	0.009108	–	–
FOSLS (Sigmoid & 200 points)	0.013505	0.019830	0.008897	0.045650



(a) FOSLS u with Sigmoid activation

(b) FOSLS $\boldsymbol{\sigma}$ with Sigmoid activation

Fig. 4.2. Poisson equation approximation results with FOSLS functional and sigmoid activation.

The four-layer neural network depicted in Fig. 4.1 is implemented with the following setting: $m_1 = n_1 = 32$ and $m_2 = m_3 = n_2 = n_3 = 24$. This network has 2962 parameters. Uniformly distributed 2000 quadrature points are used for evaluating different cost functionals. The learning rate starts with 0.001, and is reduced by half for every 5000 iterations. This learning rate decay strategy is adopted for accelerating the training (iterative) process.

For $\varepsilon = 0.01$ and $\alpha = 1$, after 20000 iterations, the median results are reported in Table 4.3 and Fig. 4.3. All three methods exhibit accurate approximation to the solution with interior layers. For both the leaky ReLU and sigmoid activation functions, the deep FOSLS method is more accurate than the deep Ritz method. Again, the DNN using the sigmoid function is more accurate than that using the leaky ReLU function, possibly due to exponential feature of the exact solution.

An interesting observation from Fig. 4.1 is that the DNN-based methods do not produce overshooting and oscillations, unlike mesh-based traditional numerical methods without strategies such as limiter, etc. This could indicate that the deep FOSLS, LS, and Ritz methods have potential to accurately approximate problems with boundary and/or interior layers.

4.3. Interface problem

To test the performance of three cost functionals for non-smooth problems, we experimented a one-dimensional interface equation as follows.

$$\begin{cases} -\left(au'(x)\right)' = f(x), & x \in \Omega = (0,1), \\ \qquad\quad u = 0, & x \in \partial\Omega = \{0,1\}, \end{cases} \tag{4.3}$$

where $a = 1$ for $x \in (0, \frac{1}{2})$ and $a = k$ for $x \in (\frac{1}{2}, 1)$. It is well-known that solutions of interface problems are not smooth, in particular, not in $H^2(\Omega)$. For

$$f(x) = \begin{cases} 8k(3x-1), & x \in (0, \frac{1}{2}), \\ 4k(k+1), & x \in (\frac{1}{2}, 1), \end{cases}$$

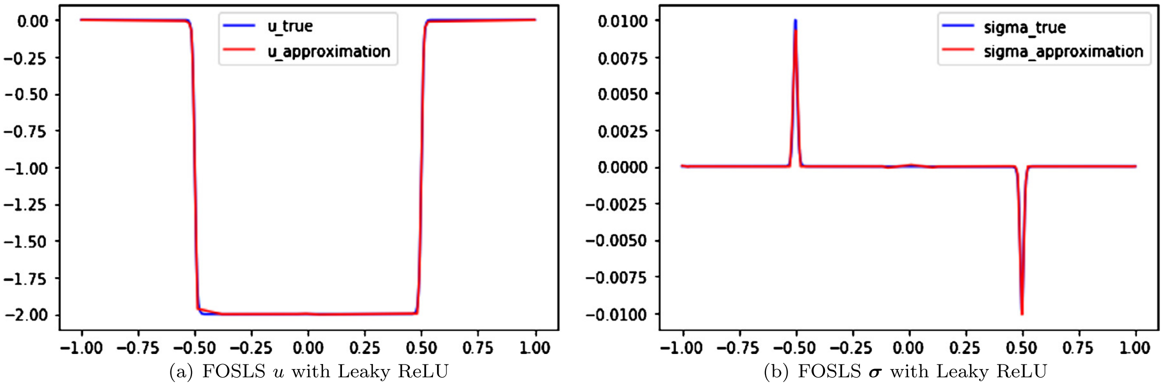
problem (4.3) has the following exact solution

$$u(x) = \begin{cases} 4kx^2(1-x), & x \in (0, \frac{1}{2}), \\ [2(k+1)x-1](1-x), & x \in (\frac{1}{2}, 1). \end{cases}$$

Note that derivative of the true solution is discontinuous at point $x = 0.5$. With $\boldsymbol{\sigma} = -au'$, the corresponding FOSLS functional defined in (2.3) has the form

Table 4.3
Relative errors of singularly perturbed equation with different loss and activation functions.

Relative errors / Loss and activation	$\frac{\|u-\bar{u}_\tau\|_0}{\|u\|_0}$	$\frac{\|\!\|u-\bar{u}_\tau\|\!\|}{\|\!\|u\|\!\|}$	$\frac{\|\boldsymbol{\sigma}-\bar{\boldsymbol{\sigma}}_\tau\|_0}{\|\boldsymbol{\sigma}\|_0}$	$\frac{G^{1/2}(\bar{\boldsymbol{\sigma}}_\tau, \bar{u}_\tau; \mathbf{f})}{\|\!\|(\boldsymbol{\sigma}, u)\|\!\|}$
Energy functional (LeakyReLU)	0.011316	0.026179	–	–
FOSLS functional (LeakyReLU)	0.006654	0.020810	0.099863	0.031482
Energy functional (Sigmoid)	0.003019	0.004612	–	–
LS functional (Sigmoid)	0.000910	0.002088	–	–
FOSLS functional (Sigmoid)	0.001403	0.001711	0.211490	0.014825



(a) FOSLS u with Leaky ReLU (b) FOSLS $\boldsymbol{\sigma}$ with Leaky ReLU

Fig. 4.3. Singularly perturbed equation: approximation results with FOSLS functional and Leaky ReLU.

$$\mathcal{G}(\boldsymbol{\tau}, v; f) = \|\boldsymbol{\tau}' - f\|_{0,\Omega}^2 + \left\|a^{-1/2}\boldsymbol{\tau} + a^{1/2}v'\right\|_{0,\Omega}^2 + \alpha\, \|u\|_{1/2,\partial\Omega}^2.$$

The same network structure is implemented as the one used in section 4.2. Numerical evaluations of the functionals are done on a uniform partition of the interval [0, 1] with $h = 0.002$. A same learning rate decay strategy is adopted here as described in section 4.2.

For $k = 10$ and $\alpha = 1$, the numerical result after 20000 iterations are reported in Table 4.4 and Fig. 4.4. The results show that the deep FOSLS method is significantly better than the deep Ritz method, while the deep LS method fails to approximate the solution well. This verifies Remark 2.1, i.e., the deep LS method is only applicable to sufficiently smooth problems. Moreover, since the true solution of this problem is a piecewise polynomial, as expected that the leaky ReLU activation function gives a better performance than the sigmoid function. This indicates that the choice of activation function is problem dependent, and we may use the relative value of the FOSLS functional to guide this choice in real-world applications where the true solutions are unknown.

4.4. Adaptive numerical quadrature

Numerical results reported in the previous sections employed uniform quadrature points. As discussed in section 1, one appealing feature of FOSLS function is that the value of the corresponding FOSLS functional is an accurate a posteriori error estimator which can be used to guide an adaptive control of the quadrature points selection. In this section, we report numerical results of the deep FOSLS method with the leaky ReLU using local and global refined partitions for the test problem in section 4.1. The same network structure and learning rate as those in section 4.1 are used.

To this end, we first describe adaptive numerical quadrature. Let $\mathcal{T}^{old}$ be the current partition of the domain Ω. For each subdomain $K \in \mathcal{T}^{old}$, let $x_K \in K$ be the quadrature point (e.g., the centroid of K). Let $(\boldsymbol{\sigma}(x, \Theta_{\boldsymbol{\sigma}}), u(x, \Theta_u))$ be the deep FOSLS approximation associated with the current partition $\mathcal{T}^{old}$. If the relative value of the FOSLS functional at $(\boldsymbol{\sigma}(x, \Theta_{\boldsymbol{\sigma}}), u(x, \Theta_u))$ is not within the prescribed tolerance, we create a new partition $\mathcal{T}^{new}$ by refining the old partition $\mathcal{T}^{old}$ as follows:

- for each $K \in \mathcal{T}^{old}$, compute local indicator

$$\eta(x_K) = \Big(\big(\mathrm{div}\,\boldsymbol{\sigma} + Xu - f\big)^2(x_K, \Theta) + \big(A^{-1/2}\boldsymbol{\sigma} + A^{1/2}\nabla u\big)^2(x_K, \Theta)\Big)\,|K|,$$

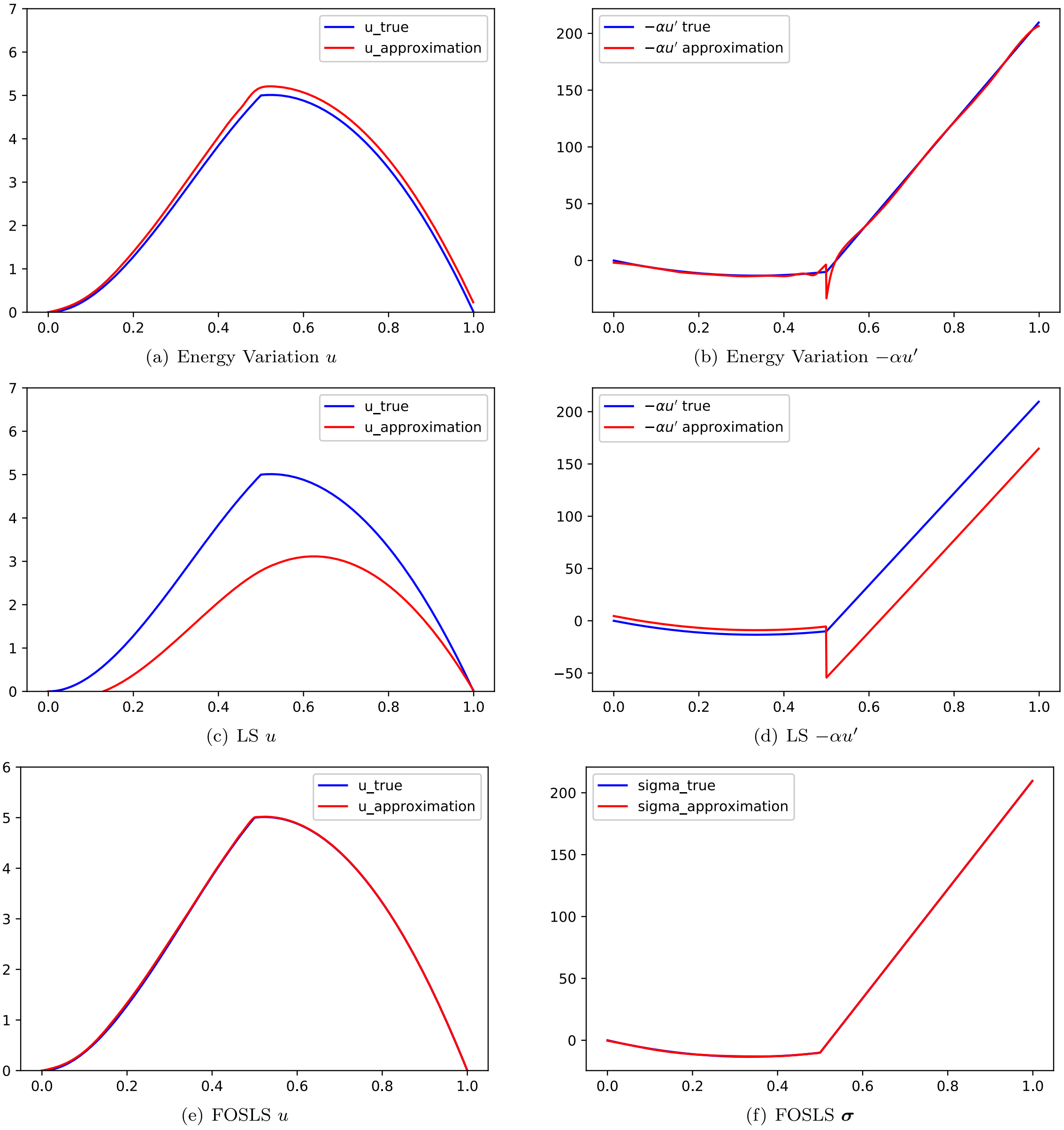
- refine subdomain $K \in \mathcal{T}^{old}$ if $\eta(x_K)$ is among the top 10% of the largest indicators.

A subdomain may be refined, e.g., by bisection in low dimensions or by some aggressive refinements in high dimensions.

Starting with a uniform partition of interval [0, 1] with $h = 0.005$, Table 4.5 reports relative values of the FOSLS functional at the current approximations on both local and global refined, and uniformly distributed partitions. All three methods

Table 4.4
Relative errors of interface problem with different loss and activation functions.

Relative errors Loss function	$\frac{\|u-\bar{u}_\tau\|_0}{\|u\|_0}$	$\frac{\|\boldsymbol{\sigma}-\bar{\boldsymbol{\sigma}}_\tau\|_0}{\|\boldsymbol{\sigma}\|_0}$	$\frac{G^{1/2}(\bar{\boldsymbol{\sigma}}_\tau,\bar{u}_\tau;\mathbf{f})}{\|\|(\boldsymbol{\sigma},u)\|\|}$
Energy functional (Sigmoid)	0.054705	–	–
LS functional (Sigmoid)	0.397965	–	–
FOSLS functional (Sigmoid)	0.007137	0.001870	0.005073
Energy functional (Leaky ReLU)	0.041087	–	–
FOSLS functional (Leaky ReLU)	0.002840	0.000686	0.001406



(a) Energy Variation u

(b) Energy Variation $-\alpha u'$

(c) LS u

(d) LS $-\alpha u'$

(e) FOSLS u

(f) FOSLS $\boldsymbol{\sigma}$

Fig. 4.4. Interface problem approximation results using different loss functions (all with Sigmoid activation function).

used a total of 10000 iterations. The local refinement method refines the quadrature points adaptively at every 2000 iterations, and global refinement method refines only once after 5000 iterations. Clearly, Table 4.5 shows that locally refined partition is better than globally uniform partitions.

Table 4.5
Comparison of locally refined and uniform partitions.

Relative errors Methods	$\dfrac{G^{1/2}(\bar{\boldsymbol{\sigma}}_\tau,\, \bar{u}_\tau; \mathbf{f})}{\vert\vert\vert(\bar{\boldsymbol{\sigma}}_\tau,\, \bar{u}_\tau)\vert\vert\vert}$
Local refinement of 200 to 292 quadrature points	0.085691
Global refinement of 200 to 400 quadrature points	0.100553
Uniform distribution of 292 quadrature points	0.102849

5. Discussion and conclusion

We proposed the deep FOSLS method by using DNNs to approximate solutions of PDEs and modified the deep Ritz and the deep LS methods by treating boundary conditions in a balance way. While the deep Ritz and LS methods are applicable to problems having underlying minimization principle and smooth problems, respectively, the deep FOSLS method is applicable to a much larger class of problems.

Both the deep LS and FOSLS methods are based on the least-squares principle applied to the respective original PDEs and a first-order system of the original PDEs. A striking feature of the least-squares principle is that values of the LS and FOSLS functionals provide feedback for automatically controlling numerical processes such as the numbers of neurons and layers in DNN, the number and the location of quadrature points for evaluating the functionals. Adaptive control first on numerical evaluation of the least-squares functionals (see preliminary numerical results in section 4.4) and then on DNN structure will be topics of our further study on the deep least-squares methods. Finally, unlike finite elements, DNN generates function in $H^2(\Omega)$ when using smooth activation functions. This means that the deep LS method is a competitive method for smooth problems.

With limited knowledge on approximation theory of DNNs, in order to accurately evaluate the functionals, inequality (3.8) and similar inequalities in the H^1 and H^2 norms for the respective deep Ritz and LS methods shed some lights on how to adaptively choose quadrature points for a fixed DNN structure. Similarly, inequality (3.9) plus an algebraic error estimator provides a guidance on when to terminate the iterative process.

Comparing with traditional mesh-based numerical methods such as finite difference, finite volume, and finite element, etc., DNN provides a new class of functions that is meshless and “pointless” and that has the attractive feature of the moving mesh method. This explains why the deep FOSLS, LS, and Ritz methods approximate well the singularly perturbed reaction diffusion equation with a sharp interior layer (see section 4.2); in particular, the DNN approximations exhibit no overshooting and no oscillation which are common numerical defects for mesh-based traditional numerical methods without strategies such as limiter, etc.

CRediT authorship contribution statement

Category 1: Conception and design of study: Z. Cai, J. Chen, M. Liu, X. Liu. Acquisition of data: J. Chen, M. Liu, X. Liu. Analysis and/or interpretation of data: Z. Cai, J. Chen, M. Liu, X. Liu.

Category 2: Drafting the manuscript: Z. Cai, J. Chen, M. Liu, X. Liu. Revising the manuscript critically for important intellectual content: Z. Cai, J. Chen, M. Liu.

Category 3: Approval of the version of the manuscript to be published: Z. Cai, J. Chen, M. Liu, X. Liu.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R. Arora, A. Basu, P. Mianjy, A. Mukherjee, Understanding deep neural networks with rectified linear units, in: International Conference on Representation Learning, Vancouver, BC, Canada, 2018.

[2] J. Berg, K. Nystrom, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41.

[3] Z. Cai, R. Lazarov, T.A. Manteuffel, S.F. McCormick, First-order system least squares for second-order partial differential equations: part I, SIAM J. Numer. Anal. 31 (6) (1994) 1785–1799.

[4] M.W.M.G. Dissanayake, N. Phan-Thien, Neural network based approximations for solving partial differential equations, Commun. Numer. Methods Eng. 10 (3) (1994) 195–201.

[5] T. Dockhorn, A discussion on solving partial differential equations using neural networks, CoRR, arXiv:1904.07200 [abs], 2019.

[6] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 3.

[7] J. He, L. Li, J. Xu, C. Zheng, Relu deep neural networks and linear finite elements, arXiv preprint, arXiv:1807.03973, 2018.

[8] D.P. Kingma, J.Ba. Adam, A method for stochastic optimization, in: International Conference on Representation Learning, San Diego, 2015.

[9] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.

[10] I.E. Lagaris, A.C. Likas, D.G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, IEEE Trans. Neural Netw. 11 (5) (2000) 1041–1049.

[11] K.S. McFall, J.R. Mahan, Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions, IEEE Trans. Neural Netw. 20 (8) (2009) 1221–1233.
[12] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1139–1364.
[13] J. Tarela, M. Martinez, Region configurations for realizability of lattice piecewise-linear models, Math. Comput. Model. 30 (11–12) (1999) 17–27.
[14] D. Yarotsky, Error bounds for approximations with deep relu networks, Neural Netw. 94 (2017) 103–114.