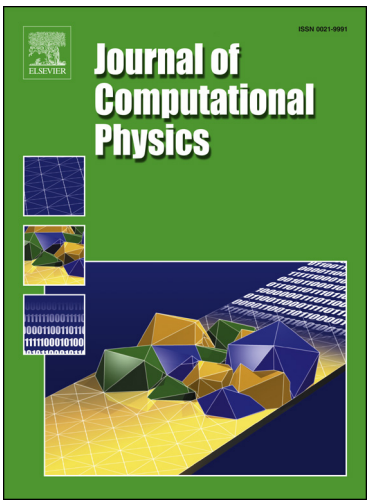
Journal Pre-proof



Int-Deep: A Deep Learning Initialized Iterative Method for Nonlinear Problems

Jianguo Huang, Haoqin Wang, Haizhao Yang

PII: S0021-9991(20)30449-6
DOI: https://doi.org/10.1016/j.jcp.2020.109675
Reference: YJCPH 109675

To appear in: *Journal of Computational Physics*

Received date: 8 December 2019
Revised date: 15 May 2020
Accepted date: 15 June 2020

Please cite this article as: J. Huang et al., Int-Deep: A Deep Learning Initialized Iterative Method for Nonlinear Problems, *J. Comput. Phys.* (2020), 109675, doi: https://doi.org/10.1016/j.jcp.2020.109675.

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2020 Published by Elsevier.

Journal Pre-proof

Highlights

- A deep learning initialized iterative method (Int-Deep) for low-dimensional nonlinear PDEs.
- O(1) deep learning computational cost w.r.t. the FEM mesh size h provides PDE solution ansatz.
- Traditional iterative methods converges quickly to the precision of FEM methods.

Journal Pre-proof

Journal Pre-proof

Int-Deep: A Deep Learning Initialized Iterative Method for Nonlinear Problems

Jianguo Huang and Haoqin Wang

School of Mathematical Sciences and MOE-LSC,
Shanghai Jiao Tong University, Shanghai 200240, China.

Haizhao Yang

Department of Mathematics, Purdue University*, West Lafayette, IN 47907, USA
Department of Mathematics, National University of Singapore†, Singapore

June 17, 2020

Abstract

This paper proposes a deep-learning-initialized iterative method (Int-Deep) for low-dimensional nonlinear partial differential equations (PDEs). The corresponding framework consists of two phases. In the first phase, an expectation minimization problem formulated from a given nonlinear PDE is approximately resolved with mesh-free deep neural networks to parametrize the solution space. In the second phase, a solution ansatz of the finite element method to solve the given PDE is obtained from the approximate solution in the first phase, and the ansatz can serve as a good initial guess such that Newton's method or other iterative methods for solving the nonlinear PDE are able to converge to the ground truth solution with high-accuracy quickly. Systematic theoretical analysis is provided to justify the Int-Deep framework for several classes of problems. Numerical results show that the Int-Deep outperforms existing purely deep learning-based methods or traditional iterative methods (e.g., Newton's method and the Picard iteration method).

Keywords. Deep learning, nonlinear problems, partial differential equations, eigenvalue problems, iterative methods, fast and accurate.

AMS subject classifications: 68U99, 65N30 and 65N25.

1 Introduction

This paper is concerned with the efficient numerical method for solving nonlinear partial differential equations (PDEs) including a class of eigenvalue problems as special cases, which is a ubiquitous and important topic in science and engineering [23, 3, 30, 36, 8, 42, 31, 15]. As far as we know, there have developed many traditional and typical numerical methods in this area, e.g., the finite difference method, the spectral method, and the finite element method [46]. The first two methods are generally used for solving problems over regular domains while the latter one is particularly suitable for solving problems over irregular domains [7, 67]. To achieve the numerical solution with

*Current institute.

†Part of the work was done in Singapore.

Journal Pre-proof

the desired accuracy, one is often required to numerically solve the discrete problem formulated as a large-scale nonlinear system of nonlinear equations, which is time-consuming. In this case, one of the most critical issues is to choose the feasible initial guess so that the numerical solver (e.g. Newton's method) is convergent. On the other hand, for reducing the computational cost, two grid methods are thereby devised in [58, 59], which only require to solve a small-sized nonlinear system arising from the finite element discretization based on a coarse triangulation. However, the difficulty is that we even can not ensure if the nonlinear system has a solution in theory if the mesh size of the coarse triangulation is large enough.

Recently, science and engineering have undergone a revolution driven by the success of deep learning techniques that originated in computer science. This revolution also includes broad applications of deep learning in computational and applied mathematics. Many new branches in scientific computing have emerged based on deep learning in the past five years including new methods for solving nonlinear PDEs. There are mainly two kinds of deep learning approaches for solving nonlinear PDEs: mesh-based [55, 56, 32, 34, 21, 20, 22] and mesh-free [9, 14, 28, 33, 48, 53, 47]. In the mesh-based methods, deep neural networks (DNNs) are constructed to approximate the solution operator of a PDE, e.g., seeking a DNN that approximates the map mapping the coefficients (or initial/boundary conditions) of a PDE to the corresponding solution. After construction, the DNN can be applied to solve a specific class of PDEs efficiently. In the mesh-free methods, which probably date back to 1990's (e.g., see [17, 38]), DNNs are applied as the parametrization of the solution space of a PDE; then the solution of the PDE is identified via seeking a DNN that fits the constraints of the PDE in the least-squares sense or minimizes a variational problem formulated from PDEs. The key to the success of these approaches is the universal approximation capacity of DNNs [37, 6, 61, 62, 52] even without the curse of dimensionality for a large class of functions [6, 43, 45, 44, 39].

Though the deep learning approach has made it possible to solve high-dimensional problems, which is a significant breakthrough in scientific computing, to the best of our knowledge, the advantage of deep learning approaches over traditional methods in the low-dimensional region is still not clear yet. The main concern is the computational efficiency of these frameworks: the number of iterations in deep learning methods is usually large or the accuracy is very limited (e.g., typically 10^{-2} to 10^{-4} relative error). In order to overcome this difficulty, one is tempted to set up a more efficient neural network architecture (e.g., incorporating physical information in the structure designing [49, 57, 64, 21, 27]), designing a more advanced learning algorithm for deep learning training [41, 63], or using a solution ansatz according to prior knowledge [38, 50, 40]. However, the overall performance of these frameworks for nonlinear problems without any prior knowledge may still not be very convincingly efficient.

This paper proposes the Int-Deep framework from a new point of view for designing highly efficient solvers of low-dimensional nonlinear PDEs with a finite element accuracy leveraging both the advantages of traditional algorithms and deep learning approaches. The Int-Deep framework consists of two phases as shown in Figure 1. In the first phase, an approximate solution to the given nonlinear PDE is obtained via deep learning approaches using DNNs of size $O(1)$ and $O(100)$ iterations, where $O(\cdot)$ means that the prefactor is independent of the final target accuracy in the Int-Deep framework, i.e., the accuracy of finite element methods. In particular, based on variational principles, we propose new methods to formulate the problem of solving nonlinear PDEs into an unconstrained minimization problem of an expectation over a function space parametrized via DNNs, which can be solved efficiently via batch stochastic gradient descent (SGD) methods due to the special form of expectation. Unlike previous methods in which the form of expectation is only derived for nonlinear PDEs related to variational equations, our proposed method can also handle those related to variational inequalities, providing a unified variational framework for a wider range

Journal Pre-proof

of nonlinear problems.

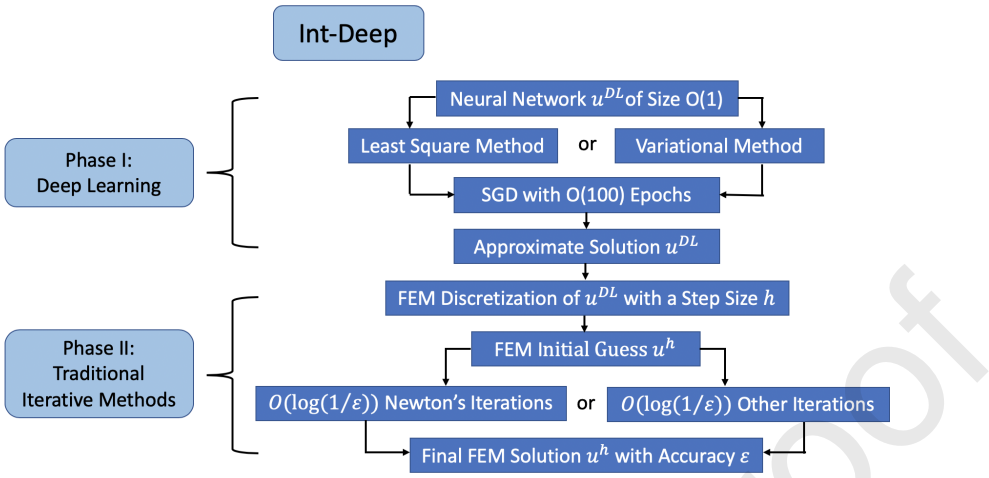


Figure 1: Computational flow of Int-Deep: Deep learning solvers in Phase I only require $O(1)$ computational cost since the network size is $O(1)$ and the number of epochs of the stochastic gradient descent (SGD) is $O(100)$. Empirically, traditional iterative methods converges to a solution with $O(\epsilon)$ accuracy in $O(\log(\frac{1}{\epsilon}))$ iterations in Phase II.

In the second phase, the approximate solution provided by deep learning approaches can serve as a good initial guess such that traditional iterative methods (e.g., Newton's method for solving nonlinear PDEs and the shifted power method for eigenvalue problems) converge quickly to the ground truth solution with high-accuracy. The hybrid algorithm substantially reduces the learning cost of deep learning approach while keeping the quality of initial guesses for traditional iterative methods; good initial guesses enable traditional iterative methods to converge in $O(\log(\frac{1}{\epsilon}))$ iterations to the ϵ precision of finite element methods. In each iteration of the traditional iterative method, the nonlinear problem has been linearized and hence traditional fast solvers for linear systems can be applied depending on the underlying structure of the linear system. Therefore, as we shall see in the numerical section, the Int-Deep framework outperforms existing purely deep learning-based methods or traditional iterative methods, e.g., Newton's method and Picard iteration. Furthermore, systematic theoretical analysis is provided to characterize the conditions under which the Int-Deep framework converges, serving as a trial to change current deep learning research from trial-and-error to a suite of methods informed by a principled design methodology.

This paper will be organized as follows. In Section 2, we briefly review the definitions of DNNs. In Section 3, we introduce the expectation minimization framework for deep learning-based PDE solvers in the first phase of the Int-Deep framework. In Section 4, as the second phase of Int-Deep, traditional iterative methods armed with good initial guesses provided by deep learning approaches will be introduced together with its theoretical convergence analysis, for clarity of presentation the proofs of which are left in Appendix. In Section 5, a set of numerical examples will be provided to demonstrate the efficiency of the proposed framework and to justify our theoretical analysis. Finally in Section 6, we summarize our paper with a short discussion.

Journal Pre-proof

2 Deep Neural Networks (DNNs)

Mathematically, DNNs are a form of highly non-linear function parametrization via function compositions using simple non-linear functions [26]. The validity of such an approximation method can be ensured by the universal approximation theorems of DNNs in [37, 6, 61, 62, 51, 52]. Let us introduce two basic neural network structures commonly used for solving PDEs below.

The first one is the so-called fully connected feed-forward neural network (FNN), which is a function in the form of a composition of L simple nonlinear functions as follows:

$$\phi(\boldsymbol{x};\boldsymbol{\theta}) := \boldsymbol{a}^T \boldsymbol{h}_L \circ \boldsymbol{h}_{L-1} \circ \cdots \circ \boldsymbol{h}_1(\boldsymbol{x}),$$

where $\boldsymbol{h}_\ell(\boldsymbol{x}) = \sigma\left(\boldsymbol{W}_\ell \boldsymbol{x} + \boldsymbol{b}_\ell\right)$ with $\boldsymbol{W}_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$, $\boldsymbol{b}_\ell \in \mathbb{R}^{N_\ell}$ for $\ell = 1, \ldots, L$, $\boldsymbol{a} \in \mathbb{R}^{N_L}$, σ is a non-linear activation function, e.g., a rectified linear unit (ReLU) $\sigma(x) = \max\{x, 0\}$ or hyperbolic tangent function $\tanh(x)$. Each $\boldsymbol{h}_\ell$ is referred as a hidden layer, N_ℓ is the width of the ℓ-th layer, and L is called the depth of the FNN. In the above formulation, $\boldsymbol{\theta} := \{\boldsymbol{a}, \boldsymbol{W}_\ell, \boldsymbol{b}_\ell : 1 \le \ell \le L\}$ denotes the set of all parameters in ϕ, which uniquely determines the underlying neural network.

The second one is the so-called residual neural network (ResNet) introduced by He, Zhang, Ren and Sun in [29]. We apply its variant defined recursively as follows:

$$\begin{aligned}
\boldsymbol{h}_0 &= \boldsymbol{V}\boldsymbol{x}, \\
\boldsymbol{g}_\ell &= \sigma(\boldsymbol{W}_\ell \boldsymbol{h}_{\ell-1} + \boldsymbol{b}_\ell), \qquad \ell = 1, 2, \ldots, L, \\
\boldsymbol{h}_\ell &= \bar{\boldsymbol{U}}_\ell \boldsymbol{h}_{\ell-2} + \boldsymbol{U}_\ell \boldsymbol{g}_\ell, \quad \ell = 1, 2, \ldots, L, \\
\phi(\boldsymbol{x};\boldsymbol{\theta}) &= \boldsymbol{a}^T \boldsymbol{h}_L,
\end{aligned}$$

where $\boldsymbol{V} \in \mathbb{R}^{N_0 \times d}$, $\boldsymbol{W}_\ell \in \mathbb{R}^{N_\ell \times N_0}$, $\bar{\boldsymbol{U}}_\ell \in \mathbb{R}^{N_0 \times N_0}$, $\boldsymbol{U}_\ell \in \mathbb{R}^{N_0 \times N_\ell}$, $\boldsymbol{b}_\ell \in \mathbb{R}^{N_\ell}$ for $\ell = 1, \ldots, L$, $\boldsymbol{a} \in \mathbb{R}^{N_0}$, $\boldsymbol{h}_{-1} = \boldsymbol{0}$. Throughout this paper, we consider $N_0 = N_\ell = N$ and $\boldsymbol{U}_\ell$ is set as the identity matrix in the numerical implementation of ResNets for the purpose of simplicity. Furthermore, as used in [19], we set $\bar{\boldsymbol{U}}_\ell$ as the identify matrix when ℓ is even and set $\bar{\boldsymbol{U}}_\ell = \boldsymbol{0}$ when ℓ is odd.

3 Phase I of Int-Deep: Variational Formulas for Deep Learning

In this section, we will present existing and our new analysis for reformulating nonlinear PDEs including eigenvalue problems to the minimization of expectation that can be solved by SGD. The analysis works for PDEs related to variational equations and variational inequalities, the latter of which is of special interest since there might be no available literature discussing this case to the best of our knowledge. Hence, our analysis could serve as a good reference for a wide range of problems.

Throughout this paper, we will use standard symbols and notations for Sobolev spaces and their norms/seminorms; we refer the reader to the reference [1] for details. Moreover, the standard $L^2(D)$-inner product for a bounded domain D is denoted by $(\cdot,\cdot)_D$. If the domain D is the solution domain Ω, we will drop out the dependence of Ω in all Sobolev norms/seminorms and $L^2(\Omega)$-inner product when there is no confusion caused.

3.1 PDE Solvers Based on DNNs

The general idea of deep learning-based PDE solvers is to treat DNNs as an efficient parametrization of the solution space of a PDE and the solution of the PDE is identified via seeking a DNN that

Journal Pre-proof

fits the constraints of the PDE in the least-squares sense or minimizes the variational minimization problem related to the PDE. Let us use the following example to illustrate the main idea:

$$\begin{cases} \mathcal{D}(u) = f & \text{in } \Omega, \\ \mathcal{B}(u) = g & \text{on } \partial\Omega, \end{cases} \tag{3.1}$$

where $\mathcal{D}$ is a differential operator and $\mathcal{B}$ is a boundary operator.

In the least squares type methods (LSM), a DNN $\phi(\boldsymbol{x};\boldsymbol{\theta}^*)$ is constructed to approximate the solution $u(\boldsymbol{x})$ for $\boldsymbol{x} \in \Omega$ via minimizing the square loss

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{x}\in\Omega}\left[|\mathcal{D}\phi(\boldsymbol{x};\boldsymbol{\theta}) - f(\boldsymbol{x})|^2\right] + \gamma\mathbb{E}_{\boldsymbol{x}\in\partial\Omega}\left[|\mathcal{B}\phi(\boldsymbol{x};\boldsymbol{\theta}) - g(\boldsymbol{x})|^2\right], \tag{3.2}$$

with a positive parameter γ.

In the variational type methods (VM), (3.1) is solved via a variational minimization

$$u^* = \arg\min_{u\in H} J(u), \tag{3.3}$$

where the Hilbert space H is an admissible space, and $J(u)$ is a nonlinear functional over H. Then, the solution space H is parametrized via DNNs, i.e., $H \approx \{\phi(\boldsymbol{x};\boldsymbol{\theta})\}_{\boldsymbol{\theta}}$, where ϕ is a DNN with a fixed depth L and width N. After parametrization, (3.3) is approximated by the following problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J(\phi(\boldsymbol{x};\boldsymbol{\theta})). \tag{3.4}$$

In general, $J(\phi(\boldsymbol{x};\boldsymbol{\theta}))$ can be formulated as the sum of several integrals over several sets $\{\Omega_i\}_{i=1}^p$, each of which corresponds to one equation in (3.1):

$$J(\phi(\boldsymbol{x};\boldsymbol{\theta})) = \sum_{i=1}^{p}\int_{\Omega_i} F_i(\boldsymbol{x};\boldsymbol{\theta})\,\mathrm{d}x = \sum_{i=1}^{p}|\Omega_i|\mathbb{E}_{\boldsymbol{\xi}_i}\Big[F_i(\boldsymbol{\xi}_i;\boldsymbol{\theta})\Big], \tag{3.5}$$

where $F_i(\cdot;\boldsymbol{\theta})$ is a function related to a variational constraint on $\phi(\boldsymbol{x};\boldsymbol{\theta})$, $\boldsymbol{\xi}_i$ is a random vector produced by the uniform distribution over Ω_i, and $|\Omega_i|$ denotes the measure of Ω_i. In addition, $\boldsymbol{\xi}_i$ $(1 \le i \le p)$ are mutually independent. Based on (3.5), (3.4) can be expressed as

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{p}|\Omega_i|\mathbb{E}_{\boldsymbol{\xi}_i}\Big[F_i(\boldsymbol{\xi}_i;\boldsymbol{\theta})\Big]. \tag{3.6}$$

Both the LSM in (3.2) and VM in (3.4) can be reformulated to the expectation minimization problem in (3.6), which can be solved by the stochastic gradient descent (SGD) method or its variants (e.g., Adam [35]). In this paper, we refer to (3.6) as the expectation minimization framework for PDE solvers based on deep learning.

Although the convergence of SGD for minimizing the expectation in (3.6) is still an active research topic, empirical success shows that SGD can provide a good approximate local minimizer of (3.2) and (3.6). This completes the algorithm of using deep learning to solve nonlinear PDEs with equality constraints.

We would like to emphasize that when the PDE in (3.1) is nonlinear, its solutions might be the saddle points of (3.5) making it very challenging to identify its solutions via minimizing (3.5). Hence, we will use the LSM in (3.2) for PDEs associated with variational equations. It deserves to point out that (3.2) is also regarded as a variational formulation, referred to as the least-squares variational principle in [10] in contrast with the usual Ritz variational principle.

Journal Pre-proof

3.2 Minimization Problems for Variational Inequalities

Variational inequalities are a class of important nonlinear problems, frequently encountered in various industrial and engineering applications [18, 25]. Unlike previous methods in which the form of expectation is only derived for nonlinear PDEs with variational equations, we propose an expectation minimization framework also suitable for variational inequalities.

The abstract framework of an elliptic variational inequality of the second kind can be described as follows (cf. [25]). Find $u \in H$ such that

$$a(u, v-u) + j(v) - j(u) \geq \langle f, v-u\rangle, \quad v \in H, \tag{3.7}$$

where H is a Hilbert space equipped with the norm $\|\cdot\|_H$, and $\langle\cdot,\cdot\rangle$ stands for the duality pairing between H' and H, with H' being the dual space of H; $a(\cdot,\cdot)$ is a continuous, coercive and symmetric bilinear form over H; $j(\cdot) : H \to \overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ is a proper, convex and lower semi-continuous functional.

As shown in [25], the above problem has a unique solution under the stated conditions on the problem data. Moreover, it can be reformulated as the following minimization problem:

$$u = \arg\min_{v\in H} J(v) = \frac{1}{2}a(v,v) - \langle f, v\rangle + j(v), \tag{3.8}$$

which naturally falls into our expectation minimization framework in (3.6).

Let us discuss the simplified friction problem as an example, where the nonlinear PDE is given by

$$\begin{cases} -\Delta u + u = f & \text{in } \Omega, \\ |\partial_{\boldsymbol{n}} u| \leq g,\ u\partial_{\boldsymbol{n}} u + g|u| = 0 & \text{on } \Gamma_C, \\ u = 0 & \text{on } \Gamma_D, \end{cases} \tag{3.9}$$

where $\boldsymbol{n}$ is the unit outward normal to $\partial\Omega$; $\Gamma_C \subset \partial\Omega$ denotes the friction boundary, and $\Gamma_D = \partial\Omega \setminus \Gamma_C$; $f \in L^2(\Omega)$ and $g \in L^2(\Gamma_C)$ are two given functions. (3.9) can be expressed as an elliptic variational inequality in the form (3.7) or (3.8) by choosing

$$H = V_D = \{\phi \in H^1(\Omega) : \phi = 0 \text{ on } \Gamma_D\}$$

and

$$a(\phi,\chi) = (\nabla\phi, \nabla\chi) + (\phi,\chi), \quad j(\phi) = (g, |\phi|)_{\Gamma_C}, \quad \phi,\ \chi \in V_D.$$

Hence, the minimization problem of the nonlinear PDE (3.9) is given by

$$u = \arg\min_{\phi\in V_D} J_1(\phi), \tag{3.10}$$

where

$$\begin{aligned} J_1(\phi) &= \frac{1}{2}\|\phi\|_1^2 - (f,\phi) + j(\phi) \\ &= |\Omega|\mathbb{E}_{\boldsymbol{\xi}_1}\Big[\frac{1}{2}\big(|\nabla_{\boldsymbol{\xi}_1}\phi(\boldsymbol{\xi}_1;\boldsymbol{\theta})|^2 + \phi^2(\boldsymbol{\xi}_1;\boldsymbol{\theta})\big) - f(\boldsymbol{\xi}_1)\phi(\boldsymbol{\xi}_1;\boldsymbol{\theta})\Big] + |\Gamma_C|\mathbb{E}_{\boldsymbol{\xi}_2}\Big[g(\boldsymbol{\xi}_2)|\phi(\boldsymbol{\xi}_2;\boldsymbol{\theta})|\Big], \end{aligned}$$

where $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$ are random vectors following the uniform distribution over Ω and Γ_C, respectively.

We can also use the penalty method to remove the constraint condition in the admissible space V_D, giving rise to an easier unconstrained minimization problem for deep learning. Then, the problem (3.10) is modified as

$$u = \arg\min_{\phi\in V_1} J_2(\phi), \tag{3.11}$$

Journal Pre-proof

where

$$V_1 = H^1(\Omega) \quad J_2(\phi) = J_1(\phi) + \gamma\|\phi\|^2_{0,\Gamma_D} = |\Omega|\mathbb{E}_{\boldsymbol{\xi}_1}\Big[\frac{1}{2}\big(|\nabla_{\boldsymbol{\xi}_1}\phi(\boldsymbol{\xi}_1;\boldsymbol{\theta})|^2 + \phi^2(\boldsymbol{\xi}_1;\boldsymbol{\theta})\big) - f(\boldsymbol{\xi}_1)\phi(\boldsymbol{\xi}_1;\boldsymbol{\theta})\Big] + |\Gamma_C|\mathbb{E}_{\boldsymbol{\xi}_2}\Big[g(\boldsymbol{\xi}_2)|\phi(\boldsymbol{\xi}_2;\boldsymbol{\theta})|\Big] + |\Gamma_D|\mathbb{E}_{\boldsymbol{\xi}_3}\Big[\gamma\phi^2(\boldsymbol{\xi}_3;\boldsymbol{\theta})\Big],$$

with γ denoting a penalty parameter to be determined feasibly and $\boldsymbol{\xi}_3$ being a random vector following the uniform distribution over Γ_D. The minimization problem in (3.11) is of the form of expectation minimization in (3.6).

3.3 Minimization Problems for Eigenvalue Problems

Now we discuss how to evaluate the smallest eigenvalue and its eigenfunction for a positive self-adjoint differential operator, e.g.,

$$\begin{cases} -\nabla(p(\boldsymbol{x})\nabla u) + q(\boldsymbol{x})u = \lambda u & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \tag{3.12}$$

where $p(\boldsymbol{x}) \in C^1(\bar{\Omega})$ and there exist two positive constants $p_1 \geq p_0$ such that $p_0 \leq p(\boldsymbol{x}) \leq p_1$ for all $\boldsymbol{x} \in \bar{\Omega}$, and $q(\boldsymbol{x}) \in C(\bar{\Omega})$ is nonnegative over Ω. Here and in what follows, $\bar{\Omega}$ denotes the closure of Ω.

The solution (λ, u) is governed by the following variational problem

$$u = \arg\min_{\phi \in V} J_3(\phi), \tag{3.13}$$

where $V = H^1_0(\Omega)$ and

$$J_3(\phi) = \frac{a(\phi,\phi)}{\|\phi\|^2_0} + \gamma(\phi(\boldsymbol{x}_0) - 1)^2; \quad a(\phi,\chi) = \int_\Omega [p(\boldsymbol{x})\nabla\phi\cdot\nabla\chi + q(\boldsymbol{x})\phi\ \chi]\,\mathrm{d}x, \quad \phi,\ \chi \in V;$$

γ is any given positive number, and $\boldsymbol{x}_0$ is any given point in the interior of Ω.

If $u_* \in V$ is a solution, we have by the variational principle for eigenvalue problems [2] that

$$u_* = \arg\min_{\phi \in V} \frac{a(\phi,\phi)}{\|\phi\|^2_0}. \tag{3.14}$$

Let α be a constant such that $(\alpha u_*)(\boldsymbol{x}_0) = 1$ and $u = \alpha u_*$. By (3.14), $J_3(u) \leq \frac{a(\phi,\phi)}{\|\phi\|^2_0} \leq J_3(\phi)$ for all $\phi \in V$. The converse can be proved similarly. Furthermore, it is easy to check that

$$J_3(\phi) = \frac{a(\phi(\boldsymbol{x};\boldsymbol{\theta}),\phi(\boldsymbol{x};\boldsymbol{\theta}))}{\|\phi(\boldsymbol{x};\boldsymbol{\theta})\|^2_0} + \gamma(\phi(\boldsymbol{x}_0;\boldsymbol{\theta}) - 1)^2 = \frac{\mathbb{E}_{\boldsymbol{\xi}}\Big[p(\boldsymbol{\xi})|\nabla_{\boldsymbol{\xi}}\phi(\boldsymbol{\xi};\boldsymbol{\theta})|^2 + q(\boldsymbol{\xi})\phi^2(\boldsymbol{\xi};\boldsymbol{\theta})\Big]}{\mathbb{E}_{\boldsymbol{\eta}}\Big[\phi^2(\boldsymbol{\eta};\boldsymbol{\theta})\Big]} + \gamma(\phi(\boldsymbol{x}_0;\boldsymbol{\theta}) - 1)^2,$$

where $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are i.i.d. random vectors following the uniform distribution over Ω.

In fact, (3.13) is a constrained minimization problem, that is

$$\begin{aligned} u &= \arg\min_{\phi \in V_1} J_3(\phi), \\ \text{s.t.} \quad u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

Journal Pre-proof

where $V_1 = H^1(\Omega)$ as before. Here, we exploit the idea from Jens Berg and Kaj Nyström [9] to reformulate the above problem as an unconstrained minimization one. To this end, we construct the neural network function as follows:

$$\phi(\boldsymbol{x};\boldsymbol{\theta}) = B(\boldsymbol{x})\psi(\boldsymbol{x};\boldsymbol{\theta}),$$

where $B(\boldsymbol{x})$ is a known smooth function such that the boundary $\partial\Omega$ can be parametrized by $B(\boldsymbol{x}) = 0$, and $\psi(\boldsymbol{x};\boldsymbol{\theta})$ is any function in V_1. So (3.13) becomes the expectation minimization:

$$u = \arg\min_{\psi\in V_1} J_3(B(\boldsymbol{x})\psi(\boldsymbol{x};\boldsymbol{\theta})). \tag{3.15}$$

Once we have obtained the eigenfunction $u(\boldsymbol{x})$, the corresponding eigenvalue is $\lambda = a(u,u)/\|u\|_0^2$. Note that the variational formulation developed here is different from the one devised by E and Yu [19] and the expectation form in our formulation makes it easier to implement SGD.

The proposed method evaluates the smallest eigenvalue and its eigenfunction not only for linear eigenvalue problems, but also for nonlinear eigenvalue problems. To generalize this idea for nonlinear eigenvalue problems, let us consider the nonlinear Schrödinger equation called the Gross-Pitaevskii (GP) equation as an example:

$$\begin{cases} -\Delta u + V(\boldsymbol{x})u + \beta u^3 = \lambda u & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \\ \|u\|_0 = 1, \end{cases} \tag{3.16}$$

where the real-valued potential function $V(\boldsymbol{x}) \in L^p(\Omega)$ for some real number $p > 1$, and β is a positive number.

According to [13], the ground state non-negative solution (λ, u) of (3.16) is unique and governed by the following variational problem

$$u = \arg\min\{J_4(\phi) : \phi \in H_0^1(\Omega), \|\phi\|_0 = 1\}, \tag{3.17}$$

where

$$J_4(\phi) = (\nabla\phi, \nabla\phi) + (V(\boldsymbol{x})\phi, \phi) + \frac{\beta}{2}(\phi^3, \phi).$$

To relax the constraint $\|\phi\|_0 = 1$, we consider the minimization problem for $\phi/\|\phi\|_0$ with $\phi \in V$, and reformulate the variational problem (3.17) in the form

$$u = \arg\min_{\phi\in H_0^1(\Omega)} J_5(\phi), \tag{3.18}$$

where

$$J_5(\phi) = \frac{(\nabla\phi, \nabla\phi) + (V(\boldsymbol{x})\phi, \phi)}{\|\phi\|_0^2} + \frac{\beta}{2}\frac{(\phi^3, \phi)}{\|\phi\|_0^4} + \gamma(\phi(\boldsymbol{x}_0) - 1)^2;$$

$\boldsymbol{x}_0$ is any given point in the interior of Ω. Moreover, the functional $J_5(\phi)$ can be written as the expectation framework as follow.

$$\begin{aligned} J_5(\phi(\boldsymbol{x};\boldsymbol{\theta})) &= \frac{(\nabla_{\boldsymbol{x}}\phi(\boldsymbol{x};\boldsymbol{\theta}), \nabla_{\boldsymbol{x}}\phi(\boldsymbol{x};\boldsymbol{\theta})) + (V(\boldsymbol{x})\phi(\boldsymbol{x};\boldsymbol{\theta}), \phi(\boldsymbol{x};\boldsymbol{\theta}))}{\|\phi(\boldsymbol{x};\boldsymbol{\theta})\|_0^2} + \frac{\beta}{2}\frac{(\phi^3(\boldsymbol{x};\boldsymbol{\theta}), \phi(\boldsymbol{x};\boldsymbol{\theta}))}{\|\phi(\boldsymbol{x};\boldsymbol{\theta})\|_0^4} \\ &\quad + \gamma(\phi(\boldsymbol{x}_0;\boldsymbol{\theta}) - 1)^2 \\ &= \frac{\mathbb{E}_{\boldsymbol{\xi}}\Big[|\nabla_{\boldsymbol{\xi}}\phi(\boldsymbol{\xi};\boldsymbol{\theta})|^2 + V(\boldsymbol{\xi})\phi^2(\boldsymbol{\xi};\boldsymbol{\theta})\Big]}{\mathbb{E}_{\boldsymbol{\eta}}\Big[\phi^2(\boldsymbol{\eta};\boldsymbol{\theta})\Big]} + \frac{\beta}{2|\Omega|}\frac{\mathbb{E}_{\boldsymbol{\xi}}\Big[\phi^4(\boldsymbol{\xi};\boldsymbol{\theta})\Big]}{\mathbb{E}_{\boldsymbol{\eta};\boldsymbol{\zeta}}\Big[\phi^2(\boldsymbol{\eta};\boldsymbol{\theta})\phi^2(\boldsymbol{\zeta};\boldsymbol{\theta})\Big]} + \gamma(\phi(\boldsymbol{x}_0;\boldsymbol{\theta}) - 1)^2, \end{aligned}$$

Journal Pre-proof

where $\boldsymbol{\xi}, \boldsymbol{\eta}, \boldsymbol{\zeta}$ are i.i.d. random vectors produced by the uniform distribution over Ω.

Similarly, we can also eliminate the boundary constraint following the ideas in linear eigenvalue problems. Then the variational problem (3.17) is rewritten as

$$u = \arg\min_{\psi \in V_1} J_5(B(\boldsymbol{x})\psi(\boldsymbol{x};\theta)), \tag{3.19}$$

where $V_1 = H^1(\Omega)$ and $B(\boldsymbol{x})$ is the same as of linear eigenvalue problems.

It is noted that if u is the solution of (3.19), the eigenfunction is its normalization: $u/\|u\|_0$ (still denoted by u for simplicity). Thus, the smallest eigenvalue can be computed by

$$\lambda = (\nabla_{\boldsymbol{x}} u, \nabla_{\boldsymbol{x}} u) + (V(\boldsymbol{x})u, u) + \beta(u^3, u).$$

4 Phase II of Int-Deep: Traditional Iterative Methods

We propose to use deep learning solutions as initial guesses so as to achieve a high-accurate solution by traditional iterative methods in a few iterations (e.g., Newton's method for solving nonlinear systems [54][24] or the two grid methods in the context of finite elements [58, 59, 60][12]). In fact, these ideas have led to high-performance methods for solving semilinear elliptic problems and eigenvalue problems with the effectiveness shown by mathematical theory and numerical simulation. The key analysis of this idea is to characterize the conditions under which deep learning solutions can help traditional iterative methods converge quickly. We will provide several classes of examples and the corresponding analysis to support this idea as follows. In what follows, u^{DL} denotes the numerical solution obtained by the deep learning algorithm, and I_h is the usual nodal interpolation operator (cf. [11, 16]).

4.1 Semilinear Elliptic Equations with Equality Constrains

Consider the following semilinear elliptic equation

$$\begin{cases} -\Delta u + f(\boldsymbol{x}, u) = 0 & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \tag{4.1}$$

where Ω is a bounded convex polygon in $\mathbb{R}^d$ $(d = 1, 2)$, and $f(\boldsymbol{x}, u)$ is a sufficiently smooth function. For simplicity, we use $f(u)$ for $f(\boldsymbol{x}, u)$ and f' for f_u in what follows.

Let $V = H_0^1(\Omega)$. Then the variational formulation of problem (4.1) is to find $u \in V$ such that

$$a(u, \chi) := (\nabla u, \nabla \chi) + (f(u), \chi) = 0, \quad \chi \in V. \tag{4.2}$$

For any $v \in L^\infty(\Omega)$, define

$$a_v(\phi, \chi) := (\nabla \phi, \nabla \chi) + (f'(v)\phi, \chi), \quad \phi,\ \chi \in V. \tag{4.3}$$

As in [58, 59], we assume that problem (4.1) (equivalently, problem (4.2)) satisfies the following conditions:

A1 Any solution u of (4.1) has the regularity $u \in W^{2,\infty}(\Omega)$.

A2 Let $u \in W^{2,\infty}(\Omega)$ be a solution of (4.1). Then there exists a positive constant C^u such that

$$C^u \|\phi\|_1 \le \sup_{\chi \in V} \frac{a_u(\phi, \chi)}{\|\chi\|_1}, \quad \phi \in V.$$

Journal Pre-proof

We next consider the finite element method for solving problem (4.2). Let $\mathcal{T}_h$ be a quasi-uniform and shape regular triangulation of Ω into K. Write $h_K = \mathrm{diam}(K)$ and $h := \max_{K\in\mathcal{T}_h} h_K$. Introduce the Courant element space by

$$V_h = \{\phi \in C(\bar{\Omega}) : \phi|_K \in \mathbb{P}_1(K) \text{ for all } K \in \mathcal{T}_h\} \cap V, \tag{4.4}$$

where $\mathbb{P}_1(K)$ denotes the function space consisting of all linear polynomials over K. Then the finite element method is to find $u^h \in V_h$ such that

$$a(u^h, \chi) = 0, \quad \chi \in V_h. \tag{4.5}$$

Now, let us introduce Int-Deep for solving the problem in (4.5). Concretely speaking, we choose $I_h u^{DL}$ as the initial guess and obtain a highly accurate approximate solution by Newton's method.

Algorithm 1 A hybrid Newton's method for semilinear problems

Input: the target accuracy ϵ, the maximum number of iterations $N_{\max}$, the approximate solution in a form of a DNN u^{DL} in Phase I of Int-Deep.
Output: $u^h = u^h_{k+1}$.

Initialization: Let $u^h_0 = I_h u^{DL}$, $k = 0$, and $e_k = 1$;
while $e_k > \epsilon$ and $k < N_{\max}$ **do**
Find $v^h_k \in V_h$ such that

$$(\nabla v^h_k, \nabla\chi) + (f'(u^h_k)v^h_k, \chi) = -(\nabla u^h_k, \nabla\chi) - (f(u^h_k), \chi), \quad \chi \in V_h.$$

Let $u^h_{k+1} = u^h_k + v^h_k$.
$e_{k+1} = \|u^h_{k+1} - u^h_k\|_0 / \|u^h_k\|_0$, $k = k + 1$.
end while

Next, we turn to discuss the convergence of the Algorithm 1. In the theorem below, we introduce a discrete maximum norm $\|\cdot\|_{0,\infty,h}$ to quantify errors. Let Ω_h consist of all the vertices of the triangulation $\mathcal{T}_h$ of Ω. Then for any $v \in C(\bar{\Omega})$, $\|v\|_{0,\infty,h} = \max_{\boldsymbol{x}\in\Omega_h} |v(\boldsymbol{x})|$.

In what follows, to simplify the presentation, for any two quantities a and b, we write "$a \lesssim b$" for "$a \le Cb$", where C is a generic positive constant independent of h, which may take different values at different occurrences. Moreover, any symbol C or c (with or without superscript or subscript) denote positive generic constants independent of the finite element mesh size h.

Let us define a neighborhood of u as follows:

$$B(u) = \{v \in V : \|v - u\|_{0,\infty} \le C_1^u\}, \tag{4.6}$$

where C_1^u is a positive constant given in Lemma A.1. Then we have the following theorem showing the convergence of Algorithm 1.

Theorem 4.1. *Let $u \in W^{2,\infty}(\Omega)$ be a solution of (4.1) and u^h_k the function sequence formed by Algorithm 1. Let $B(u)$ be a neighborhood of u given by (4.6). Assume $u^h_k \in B(u)$ for $k = 0, 1, \cdots$. Write $\delta = \|u - u^{DL}\|_{0,\infty,h}$. Then there exist two positive constants $\bar{h}_0$ ($\bar{h}_0 < h_2$) and $\bar{\delta}_0$ such that if $h < \bar{h}_0$ and $\delta < \bar{\delta}_0$, there holds*

$$\|u - u^h_k\|_{0,\infty} \lesssim h^2 + \beta_1^{2^k} \qquad \text{for } d = 1,$$
$$\|u - u^h_k\|_{0,\infty} \lesssim h^2|\log(h)| + h^{-2/p}\beta_2^{2^k} \quad \text{for } d = 2,$$

Journal Pre-proof

where h_2 is a positive constant given in Lemma A.4,

$$\beta_1 = c_0 c_1 (h^2 + \delta) < 1, \quad \beta_2 = c_2 c_3 (h^2 |\log(h) + \delta|) < 1,$$

with c_i $(0 \le i \le 3)$ as positive constants.

The proof of Theorem 4.1 can be found in Appendix A. According to the second-order convergence in β_1 and β_2 in the above theorem, one can use only a few Newton's iterations to achieve a numerical solution with the same accuracy as in the finite element solution u^h. The forthcoming numerical results will demonstrate this theoretical estimate. As shown in reference [58] (see also Lemma A.3 in Appendix A), we find that under the condition $h < \bar{h}_0 < h_2$ given in the above theorem, the finite element method (4.5) has a unique solution u^h around a given exact solution u. Moreover, the analysis developed here can be applied to high order finite element methods.

4.2 Eigenvalue Problems

In this subsection, we propose and analyze Int-Deep for solving eigenvalue problems in the similar spirit of the two grid method due to Xu and Zhou [60]. We first discuss how to design efficient methods for linear eigenvalue problems. For simplicity, let us consider the following problem

$$\begin{cases} -\Delta u = \lambda u & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{4.7}$$

The variational problem for (4.7) is given as follows. Find the smallest number λ and a nonzero function $u \in V$ such that

$$a(u, \chi) = \lambda(u, \chi), \quad \chi \in V, \tag{4.8}$$

where $V = H_0^1(\Omega)$ and $a(u, \chi) = (\nabla u, \nabla \chi)$. Without loss of generality, assume $\|u\|_0 = 1$.

In the discretization, we use the same finite element space V_h as given in the last subsection. Hence, the finite element method for (4.8) is to find the smallest number λ^h and a nonzero function $u^h \in V_h$ such that

$$a(u^h, \chi) = \lambda^h (u^h, \chi), \quad \chi \in V_h. \tag{4.9}$$

Now, let us introduce Int-Deep for solving the problem in (4.9). Concretely speaking, we choose the normalized $I_h u^{DL}$ as the initial guess and obtain a highly accurate approximate solution by a certain iterative scheme. The iterative scheme is motivated by the two grid method for eigenvalue problems due to Xu and Zhou [60]. The method is essentially the power method for eigenvalue problems and the eigenvalue computation is accelerated by the Rayleigh quotient.

In each iteration step of the "while" loop in Algorithm 2, typical numerical methods would normalize the approximate eigenfunction to enforce $\|u_k^h\|_0 = 1$, which might help to speedup the convergence. However, we will only prove the convergence of Algorithm 2 without the normalization step as follows for simplicity. We define the elliptic projection operator P_h that projects any $u \in V$ to $P_h u \in V_h$ such that $a(P_h u, \chi) = a(u, \chi), \quad \chi \in V_h$. The analysis of this operator is well-known (cf. [11, 16]) but we quote an important result (cf. [5, 4]) below.

Lemma 4.1. *Let (λ, u) be an eigenpair of (4.8). For any $\phi \in H_0^1(\Omega) \setminus \{0\}$, there holds*

$$\frac{a(\phi, \phi)}{(\phi, \phi)} - \lambda = \frac{a(\phi - u, \phi - u)}{(\phi, \phi)} - \lambda \frac{(\phi - u, \phi - u)}{(\phi, \phi)}.$$

With the help of the above results, we can derive the following theorem.

Journal Pre-proof

Algorithm 2 Int-Deep for linear eigenvalue problems

Input: the target accuracy ϵ, the maximum number of iterations $N_{\max}$, the approximate solution in a form of a DNN u^{DL} in Phase I of Int-Deep.

Output: $u^h = u^h_{k+1}, \lambda^h = \lambda^h_{k+1}$.

Initialization: Let $u^h_0 = I_h u^{DL}/\|I_h u^{DL}\|_0$, $\lambda^h_0 = |u^h_0|^2_1/\|u^h_0\|^2_0$, $k = 0$, $e_k = 1$;

while $e_k > \epsilon$ and $k < N_{\max}$ **do**

Find $u^h_{k+1} \in V_h$ such that

$$a(u^h_{k+1}, \chi) = \lambda^h_k (u^h_k, \chi), \quad \chi \in V_h,$$
$$\lambda^h_{k+1} = \frac{|u^h_{k+1}|^2_1}{\|u^h_{k+1}\|^2_0};$$

$e_{k+1} = \|u^h_{k+1} - u^h_k\|_0/\|u^h_k\|_0$, $k = k+1$.

end while

Theorem 4.2. *Let λ be the smallest eigenvalue of (4.7) and $u \in H^2(\Omega)$ be the corresponding eigenfunction with $\|u\|_0 = 1$, respectively. Denote $\tilde{\delta} = \max\{|\lambda - \lambda^h_0|, \|u - u^{DL}\|_{0,\infty,h}\}$. Assume that the sequence $\|u^h_k\|_0$ is bounded below and above by two positive constants ε_0 and ε_1, respectively. Then there exist two positive constants $\tilde{h_0}$ and $\tilde{\delta}_0$ such that if $h < \tilde{h_0}$ and $\tilde{\delta} < \tilde{\delta}_0$, there holds*

$$|\lambda - \lambda^h_k| + \|u - u^h_k\|_0 \lesssim \beta_3^{2^k} + h^2, \tag{4.10}$$

where $\beta_3 = c_4 c_6(\tilde{\delta} + h^2)$, c_4 and c_6 are generic positive constants independent of the finite element mesh size h.

Note that $\|u^h_0\|_0 = 1$ and only very few iterations are required to derive the desired numerical solution in Algorithm 2 in practical implementation. Hence, it is reasonable to assume the sequence $\|u^h_k\|_0$ is bounded below and above by two positive constants. This assumption is also verified by our numerical experiments.

The proof of Theorem 4.2 can be found in Appendix B. Similar to Theorem 4.1, in view of the second-order convergence in β_3 in the above theorem, one can use only a few iterations to achieve a numerical solution with the same accuracy as for the related finite element method. The forthcoming numerical results will demonstrate this theoretical estimate.

Now, let us turn to the nonlinear eigenvalue problem. For simplicity consider the Gross-Pitaevskii equation (3.16). The weak form of (3.16) is given as follow. Find the smallest number λ and a non-negative function $u \in H^1_0(\Omega)$ such that

$$\begin{cases} a(u, \chi) + (f(u), \chi) = \lambda(u, \chi), \quad \chi \in H^1_0(\Omega), \\ \|u\|_0 = 1, \end{cases} \tag{4.11}$$

where

$$a(u, \chi) = (\nabla u, \nabla \chi) + (V(\boldsymbol{x})u, \chi), \quad f(u) = \beta u^3.$$

Once we have obtained the eigenfunction $u(x)$, then the corresponding eigenvalue is $\lambda = a(u, u) + (f(u), u)$.

To discretize the above problem, we use the same finite element space V_h in (4.4). Hence, the finite element method for (4.11) is to find the smallest number λ^h and a non-negative function

Journal Pre-proof

$u^h \in V_h$ such that

$$\begin{cases} a(u^h, \chi) + (f(u^h), \chi) = \lambda^h (u^h, \chi), \quad \chi \in V_h, \\ \|u_h\|_0 = 1. \end{cases} \tag{4.12}$$

The error analysis of the finite element method for (4.12) can be found in [65, 13].

Finally, let us introduce an accelerated iteration method for solving (4.12). Though an iteration scheme can be derived by the two grid method for nonlinear elliptic eigenvalue problems due to Cancès, Chakir, and He [12], we propose another iteration scheme motivated by Newton's method for nonlinear eigenvalue problems due to Gao, Yang and Meza [24]. Here, we only derive the algorithm for nonlinear eigenvalue problems and we will analyze the convergence of Algorithm 3 in future work.

Algorithm 3 A hybrid Newton's method for nonlinear eigenvalue problems

Input: the target accuracy ϵ, the maximum number of iterations $N_{\max}$, the approximate solution in a form of a DNN u^{DL} in Phase I of Int-Deep.

Output: $u^h = u^h_{k+1}, \lambda^h = \lambda^h_{k+1}$.

Initialization: Let $u^h_0 = I_h u^{DL}/\|I_h u^{DL}\|_0$, $\lambda^h_0 = a(u^h_0, u^h_0) + (f(u^h_0), u^h_0)$, $k = 0$, $e_k = 1$;

while $e_k > \epsilon$ and $k < N_{\max}$ **do**

Find $(v^h_k, \mu^h_k) \in V_h \times \mathbb{R}$ such that

$$a(v^h_k, \chi) + (f'(u^h_k)v^h_k, \chi) - \lambda^h_k (v^h_k, \chi) - \mu^h_k (u^h_k, \chi) = -a(u^h_k, \chi) - (f(u^h_k), \chi) + \lambda^h_k (u^h_k, \chi), \ \chi \in V_h,$$
$$2(v^h_k, u^h_k) = 1 - (u^h_k, u^h_k).$$

Let $u^h_{k+1} = u^h_k + v^h_k, \lambda^h_{k+1} = \lambda^h_k + \mu^h_k$.

$e_{k+1} = \|u^h_{k+1} - u^h_k\|_0/\|u^h_k\|_0$, $k = k + 1$.

end while

5 Numerical Experiments

This section consists of two parts. In the first part, we provide various numerical examples to illustrate the performance of the proposed deep learning-based methods. As we shall see, deep learning approaches are capable of providing approximate solutions to nonlinear problems in $O(100)$ iterations. However, continuing the iteration cannot further improve accuracy. In the second part, we will investigate the numerical performance of the Int-Deep framework in terms of network hyper-parameters and the convergence analysis in the previous section. Though the hyper-parameters of Int-Deep to guarantee a good initial guess should be problem-dependent, as we shall see in various numerical examples in this section, DNNs of size $O(1)$ and trained with $O(100)$ iterations can provide good initial guesses enabling traditional iterative methods to converge in $O(\log(\frac{1}{\epsilon}))$ iterations to the ϵ precision of finite element methods.

In our numerical experience, the ResNet has a better numerical performance than FNN. Hence, without especial explanation, we always use the ResNet of width 50 and depth 6 with an activation function $\sigma(x) = \max\{x^3, 0\}$. Neural networks are trained by Adam optimizer [35] with a learning rate $\eta = 1\mathrm{e}-03$. The batch size is 512 for all 1D examples and 1024 for all 2D examples. Deep learning algorithms are implemented by Python 3.7 using PyTorch 1.0. and a single NVIDIA Quadro P6000 GPU card. All finite element methods are implemented in MATLAB 2018b in an Intel Core i7, 2.6GHz CPU on a personal laptop with a 32GB RAM.

Journal Pre-proof

Let us first summarize notations used in this section. Suppose u is the exact solution to a problem and u_k^h is the approximation evaluated by the Int-Deep framework in the k-th iteration in the second phase. We denote the absolute difference between the ground truth and the numerical solution as

$$e_k^h := u - u_k^h,$$

which will be measured with different norms to obtain the accuracy of our framework. After completing iterations, u^h is used as the approximate solution by traditional iterative methods, and e^h is denoted as its absolute difference. Similarly, e^{DL} is denoted as the absolute difference between the exact solution and the approximate solution by deep learning. In eigenvalue problems, let λ, λ^{DL} and λ^h denote the target eigenvalue, the approximate one by deep learning, and the approximate one by traditional methods, respectively. #Epoch means the number of epochs in the Adam for deep learning and #K stands for the number of iterations in the traditional iterative methods in Int-Deep. We always apply the Courant element method to solve the weak form in Algorithms 1 to 3.

We also summarize the numerical examples in this section in Table 1 below, which could help the reader to better understand the structure of extensive numerical examples.

Table 1: Summary of numerical examples.

Phase I		Phase II with an initial guess by Phase I	
Example 5.1	Phase I, Section 3.1	Example 5.4	Phase II, Section 4.1
Example 5.2	Phase I, Section 3.2	Example 5.5	Phase II, Section 4.2, linear eigenvalue problem
Example 5.3	Phase I, Section 3.3	Example 5.6	Phase II, Section 4.2, nonlinear eigenvalue problem

5.1 Phase I of Int-Deep: Deep Learning Methods

Let us first provide numerical examples to illustrate the performance of the proposed variational formulations for deep learning in Section 3.

5.1.1 Linear PDEs

Example 5.1. *Consider the second-order linear elliptic equation with the Dirichlet boundary condition in one dimension:*

$$\begin{cases} -\frac{\mathrm{d}}{\mathrm{d}x}\left(p(x)\frac{\mathrm{d}u}{\mathrm{d}x}\right) + x^2 u(x) = f(x), & x \in (-1,1), \\ u = 0, & x = -1 \text{ or } 1, \end{cases}$$

with $p(x) = 1 + x^2$ *and* $f(x) = \pi^2(1+x^2)\sin(\pi x) - 2\pi x\cos(\pi x) + x^2\sin(\pi x)$. *The exact solution of this problem is*

$$u(x) = \sin(\pi x).$$

In this experiment, the variational formulation in (3.6) with a penalty constant $\gamma = 500$ is applied in the deep learning method. To evaluate the test error, we adopt a mesh size $h = \frac{2}{512}$ to generate the uniform triangulation $\mathcal{T}_h$ as the test locations. The test errors during training are summarized in Table 2 and Figure 2. From Table 2 and Figure 2, we know the absolute discrete maximum error is reduced to order 1e-3 after 300 epochs. However, the error oscillates after about 1000 epochs and it is difficult to get a highly accurate solution when #Epoch increases.

Journal Pre-proof

Table 2: Example 5.1

#Epoch	$\|e^{DL}\|_{0,\infty,h}$
300	1.16e-3
500	6.09e-4
1000	9.36e-4
5000	3.09e-3
10000	2.49e-3

Table 3: Example 5.2.

#Epoch	$\|e^{DL}\|_{0,\infty,h}$
300	6.55e-2
500	5.67e-2
1000	6.78e-3
5000	4.27e-3
10000	2.22e-3

Table 4: Example 5.3.

#Epoch	$\|\lambda - \lambda^{DL}\|$	$\|u - u^{DL}\|_{0,\infty,h}$
300	1.40e-1	5.06e-2
500	2.31e-2	1.37e-2
1000	1.71e-2	1.66e-2
5000	4.85e-3	1.04e-2
10000	3.20e-3	1.45e-2

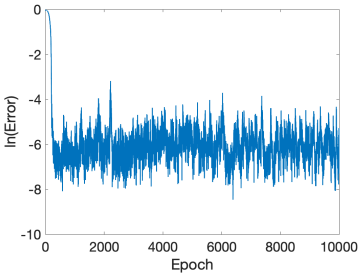


Figure 2: e^{DL} of Example 5.1.

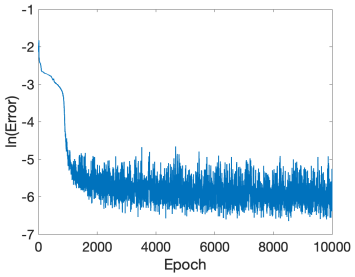


Figure 3: e^{DL} of Example 5.2.

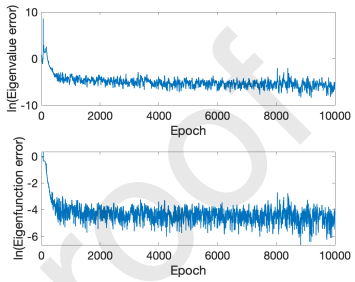


Figure 4: Test errors of Example 5.3.

5.1.2 Variational Inequalities

Example 5.2. *Consider the simplified friction problem*

$$\begin{cases} -\Delta u + u = f & \text{in } \Omega, \\ |\partial_{\boldsymbol{n}} u| \le g, \ u\partial_{\boldsymbol{n}} u + g|u| = 0 & \text{on } \Gamma_C, \\ u = 0 & \text{on } \Gamma_D, \end{cases}$$

where $\boldsymbol{n}$ *is the outer normal vector,* $\Omega = (0,1)^2$, $\Gamma_C = \{1\} \times [0,1]$ *and* $\Gamma_D = \partial\Omega \setminus \Gamma_C$. *We set* $g = 1$ *and choose* f *such that the problem has an exact solution* $u(x,y) = (\sin x - x \sin 1)\sin(2\pi y)$.

In this experiment, the variational formula in (3.11) with a penalty constant $\gamma = 500$ is applied in the deep learning method. To evaluate the test error, we adopt a mesh size $h = \frac{1}{128}$ to generate the uniform triangulation $\mathcal{T}_h$ as the test locations. The test errors during training are summarized in Table 3 and Figure 3. From Table 3 and Figure 3, we know the absolute discrete maximum error is reduced to order 1e-3 after 1000 epochs. However, the error oscillates after 2000 epochs and it is difficult to get a highly accurate solution when #Epoch increases.

5.1.3 Eigenvalue Problems

Example 5.3. *Consider the following problem*

$$\begin{cases} -u'' = \lambda u, & x \in (0,1), \\ u = 0, & x = 0 \text{ or } 1. \end{cases}$$

The smallest eigenvalue is π^2 *and the corresponding eigenfunction is* $u(x) = \sin(\pi(x-1))$.

In this experiment, the variational formula in (3.15) with $x_0 = 0.5$ and $\gamma = 100$ is applied in the deep learning method. To evaluate the test error, we adopt a mesh size $h = \frac{1}{256}$ to generate the

Journal Pre-proof

uniform triangulation $\mathcal{T}_h$ as the test locations. The test errors during training are summarized in Table 4 and Figure 4. From Table 4 and Figure 4, we know the absolute discrete maximum error of the eigenfunction is reduced to order 1e-2 after 500 epochs and the eigenvalue error is reduced to order 1e-3 after 5000 epochs. Both errors oscillate after 1000 epochs and it is hard to to get high-accuracy as #Epoch increases.

5.2 Phase II of Int-Deep: Traditional Iterative Methods

Now we use the approximate solution by deep learning methods as the initial guess in traditional iterative methods. We will show that DNNs of size $O(1)$ and trained with $O(100)$ iterations are good enough for traditional iterative methods to converge in at most $O(\log(\frac{1}{\epsilon}))$ iterations to the ϵ precision of finite element methods.

5.2.1 Semilinear PDEs

We consider semilinear elliptic equations with homogeneous Dirichlet boundary conditions to demonstrate the efficiency of the Int-Deep framework in Algorithm 1 (deep learning combined with Newton's method for semilinear PDE) and to verify Theorem 4.1.

Example 5.4. *Consider the following semilinear elliptic equations*

$$\begin{cases} -\Delta u - (u-1)^3 + (u+2)^2 = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases}$$

where $\Omega = (0,1)^d$ *with* $d = 1$ *or* 2. *We choose* f *such that the problem has an exact solution* $u(x) = 3\sin(2\pi x)$ *for* $d = 1$ *and* $u(x) = 3\sin(2\pi x)\sin(2\pi y)$ *for* $d = 2$.

Case $d = 1$.

First of all, we show that Newton's method cannot converge to a solution without a good initial guess u_0 and the deep learning method can provide a good initial guess. We apply Newton's iteration in Algorithm 1 with several types of initial guesses u_0 listed in Table 5, and the other parameters are taken as $h = \frac{1}{1024}$, $N_{\max} = 15$, and $\epsilon = 0.01 \times h^2$. u^{DL} is obtained via the deep learning approach based on the variational formula (3.6) with $\gamma = 500$ and 200 epochs in the Adam. Without good knowledge of the ground truth solution or u^{DL}, Newton's method fails to converge to a good numerical solution.

Table 5: The performance of Newton's method with different initial guesses u_0. ω stands for a Gaussian random noise with mean zero and unit variance.

u_0	#K	$\|e_0^h\|_{0,\infty,h}$	$\|e^h\|_{0,\infty,h}$	u_0	#K	$\|e_0^h\|_{0,\infty,h}$	$\|e^h\|_{0,\infty,h}$
1	5	4.00e+0	1.95e+0	ω	6	5.68e+0	3.05e+0
2	5	5.00e+0	1.95e+0	$1+\omega$	5	6.33e+0	1.95e+0
5	15	8.00e+0	1.28e+5	$-1+\omega$	15	6.95e+0	2.65e+6
-1	15	4.00e+0	9.50e+4	$u+\omega$	6	4.03e+0	1.82e-5
-2	12	5.00e+0	4.66e+0	$u+2.5\times\omega$	15	8.89e+0	2.10e+5
-5	15	8.00e+0	6.45e+4	u^{DL}	5	2.97e-1	1.82e-5

Besides, we also test the performance of the Picard's iteration in this example. We apply Picard's iteration with several types of initial guesses u_0 listed in Table 6, and the other parameters are taken as $h = \frac{1}{1024}$, $N_{\max} = 15$, and $\epsilon = 0.01 \times h^2$. Picard's iteration fails to converge to the right solution in all tests.

Journal Pre-proof

Table 6: The performance of Picard's method with different initial guesses u_0. ω stands for a Gaussian random noise with mean zero and unit variance.

u_0	#K	$\|e_0^h\|_{0,\infty,h}$	$\|e^h\|_{0,\infty,h}$	u_0	#K	$\|e_0^h\|_{0,\infty,h}$	$\|e^h\|_{0,\infty,h}$
1	15	4.00e+0	1.95e+0	ω	15	5.92e+0	1.95e+0
0	15	3.00e+0	1.95e+0	$u+\omega$	10	3.59e+0	nan

Furthermore, we show that the two-grid method also needs a good initial guess u_0 for Newton's method in the coarse grid stage; otherwise, the two-grid method cannot converge well. Denote the mesh size of the coarse grid as H and the mesh size of the fine grid as h. The numerical results in Table 7 summarize the performance of the two-grid method with several types of initial guesses u_0 and different mesh sizes H and h. The two grid method can converge to the correct solution only in the case when the initial guess of the coarse grid stage is provided by the deep learning method.

Table 7: The performance of the two-grid method for Example 5.4 in 1D with different initial guesses and mesh sizes.

u_0	H	h	$\|e^h\|_{0,\infty,h}$	H	h	$\|e^h\|_{0,\infty,h}$	H	h	$\|e^h\|_{0,\infty,h}$
1	2^{-4}	2^{-8}	1.95e+0	2^{-5}	2^{-10}	1.95e+0	2^{-6}	2^{-12}	1.95e+0
-1	2^{-4}	2^{-8}	2.25e+1	2^{-5}	2^{-10}	2.74e+2	2^{-6}	2^{-12}	2.99e+2
0	2^{-4}	2^{-8}	1.95e+0	2^{-5}	2^{-10}	1.95e+0	2^{-6}	2^{-12}	1.95e+0
ω	2^{-4}	2^{-8}	3.06e+0	2^{-5}	2^{-10}	3.05e+0	2^{-6}	2^{-12}	3.05e+0
u^{DL}	2^{-4}	2^{-8}	3.27e-3	2^{-5}	2^{-10}	1.98e-4	2^{-6}	2^{-12}	1.23e-5

Next, we show that the initial guess by the deep learning approach enables Newton's method to converge to a solution with the precision of finite element methods, i.e., the numerical convergence order in terms of h defined by

$$\text{order} := \log_2 \left| \frac{e_k^h}{e_k^{h/2}} \right|$$

is 2 as proved by Theorem 4.1. For the purpose of convenience, we choose $h = 2^{-\ell}$ for different integers ℓ's and let $e_{h^2} = h^2$ as the theoretical precision of finite element methods. We repeatedly apply the same deep learning method as in Table 5 to generate different initial guesses u_0^h for different sizes h. Let $N_{\max} = 15$ and $\epsilon = 0.01 \times h^2$ in Algorithm 1. Table 8 summarizes the performance of Algorithm 1 and numerical results verify the accuracy and the convergence order of the Int-Deep framework, even though the number of epochs in the training of deep learning is $O(100)$ and the initial error is very large.

Table 8: The performance of Int-Deep in Algorithm 1 for Example 5.4 in 1D with different mesh sizes h.

h	e_{h^2}	# Epoch	$\|e^{DL}\|_{0,\infty,h}$	#K	$\|e^h\|_{0,\infty,h}$	order
2^{-7}	6.10e-5	200	2.97e-1	5	1.17e-3	-
2^{-8}	1.53e-5	200	2.97e-1	5	2.92e-4	2.00
2^{-9}	3.81e-6	200	2.97e-1	5	7.29e-5	2.00
2^{-10}	9.54e-7	200	2.97e-1	5	1.82e-5	2.00
2^{-11}	2.38e-7	200	2.97e-1	5	4.56e-6	2.00
2^{-12}	5.96e-8	200	2.97e-1	5	1.14e-6	2.00
2^{-13}	1.49e-8	200	2.97e-1	5	2.85e-7	2.00
2^{-14}	3.73e-9	200	2.97e-1	5	7.12e-8	2.00

Journal Pre-proof

Finally, we show that the performance of the Int-Deep in Algorithm 1 is independent of the size of DNNs, which is supported by the numerical results in Table 9.

Table 9: The performance of Int-Deep in Algorithm 1 for Example 5.4 in 1D with different DNN width N and depth L when $h = \frac{1}{1024}$ and #Epoch = 400.

L \ N	10			30			50		
	$\|e^{DL}\|_{0,\infty,h}$	$\|e^{h}\|_{0,\infty,h}$	#K	$\|e^{DL}\|_{0,\infty,h}$	$\|e^{h}\|_{0,\infty,h}$	#K	$\|e^{DL}\|_{0,\infty,h}$	$\|e^{h}\|_{0,\infty,h}$	#K
2	3.72e+0	1.82e-5	8	2.27e-1	1.82e-5	5	5.19e-2	1.82e-5	4
4	1.05e-1	1.82e-5	4	1.63e-2	1.82e-5	3	1.51e-2	1.82e-5	3
6	9.84e-2	1.82e-5	4	8.48e-3	1.82e-5	3	5.34e-3	1.82e-5	3

Case $d = 2$.

Again, we show that the initial guess by the deep learning approach enables Newton's method to converge to a solution with the precision of finite element methods, i.e., the numerical convergence order in terms of h is almost 2 as proved by Theorem 4.1. We repeatedly apply the variational formula in (3.6) with $\gamma = 500$ and the deep learning method to generate different initial guesses u_0^h for different sizes h. Let $N_{\max} = 15$ and $\epsilon = 0.01 \times h^2$ in Algorithm 1. Let $e_{h^2} = |h^2 \log h|$. Table 10 summarizes the performance of Algorithm 1 and numerical results verify the accuracy and the convergence order of the Int-Deep framework, even though the number of epochs in deep learning is $O(100)$ and the initial error is very large.

Table 10: The performance of Int-Deep in Algorithm 1 for Example 5.4 in 2D with different mesh sizes h.

h	e_{h^2}	# Epoch	$\|e^{DL}\|_{0,\infty,h}$	#K	$\|e^{h}\|_{0,\infty,h}$	order
2^{-4}	1.08e-2	200	2.19e+0	5	2.03e-1	-
2^{-5}	3.38e-3	200	2.19e+0	5	5.92e-2	1.78
2^{-6}	1.02e-3	200	2.20e+0	5	1.55e-2	1.94
2^{-7}	2.96e-4	200	2.20e+0	6	3.92e-3	1.98
2^{-8}	8.46e-5	200	2.20e+0	6	9.83e-4	2.00

Finally, we show that the performance of the Int-Deep in Algorithm 1 is independent of the size of DNNs, which is supported by the numerical results in Table 11.

Table 11: The performance of Int-Deep in Algorithm 1 for Example 5.4 in 2D with different DNN width N and depth L when $h = \frac{1}{128}$ and #Epoch = 400.

L \ N	10			30			50		
	$\|e^{DL}\|_{0,\infty,h}$	$\|e^{h}\|_{0,\infty,h}$	#K	$\|e^{DL}\|_{0,\infty,h}$	$\|e^{h}\|_{0,\infty,h}$	#K	$\|e^{DL}\|_{0,\infty,h}$	$\|e^{h}\|_{0,\infty,h}$	#K
2	3.15e+0	3.92e-3	6	2.91e+0	3.92e-3	6	3.32e+0	3.92e-3	6
4	4.06e+0	3.92e-3	6	2.53e+0	3.92e-3	5	7.59e-1	3.92e-3	4
6	3.54e+0	3.92e-3	6	1.20e+0	3.92e-3	4	1.05e+0	3.92e-3	4

5.2.2 Linear eigenvalue problems

Here, we demonstrate the efficiency of the Int-Deep framework in Algorithm 2 (deep learning combined with the power method) and verify Theorem 4.2.

Journal Pre-proof

Example 5.5. *Consider the following problem*

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

where $\Omega = (0,1)^d$ *for* $d = 1$ *and* 2. *The smallest eigenvalue is* $d\pi^2$. *The corresponding eigenfunction is* $u(x) = \sin(\pi(x-1))$ *for* $d = 1$, *and* $u(x,y) = \sin(\pi(x-1))\sin(\pi(y-1))$ *for* $d = 2$.

Case $d = 1$.

We show that the initial guess by the deep learning approach enables the power method to quickly converge to a solution with the precision of finite element methods, i.e., the numerical convergence order in terms of h is 2 as proved by Theorem 4.2. We repeatedly apply the variational formula in (3.15) (with $x_0 = 0.5$ and $\gamma = 100$) and the deep learning method to generate different initial guesses u_0^h for different sizes h. Let $N_{\max} = 10$ and $\epsilon = h^2$ in Algorithm 2. Table 12 summarizes the performance of Algorithm 2 and numerical results verify the accuracy and the convergence order of the Int-Deep framework, even though the number of epochs in deep learning is $O(100)$.

Table 12: The performance of Int-Deep in Algorithm 2 for Example 5.5 in 1D with different mesh sizes h.

h	#Epoch	$\|e^{DL}\|_0$	#K	$\|\lambda - \lambda^h\|$	eigenvalue order	$\|e^h\|_{0,}$	eigenfunction order
2^{-5}	300	1.13e-2	3	7.93e-3	-	8.08e-4	-
2^{-6}	300	1.14e-2	4	1.98e-3	2.00	2.02e-4	2.00
2^{-7}	300	1.14e-2	5	4.95e-4	2.00	5.05e-5	2.00
2^{-8}	300	1.14e-2	6	1.24e-4	2.00	1.26e-5	2.00
2^{-9}	300	1.14e-2	7	3.04e-5	2.03	3.17e-6	2.00

Next, we show that the performance of Int-Deep in Algorithm 2 is independent of the size of DNNs, which is supported by the numerical results in Table 13.

Table 13: The performance of Int-Deep in Algorithm 2 for Example 5.5 in 1D with different DNN width N and depth L when $h = \frac{1}{512}$ and #Epoch = 400.

N / L	10			30			50		
	$\|\lambda - \lambda^h\|$	$\|e^h\|_0$	#K	$\|\lambda - \lambda^h\|$	$\|e^h\|_0$	#K	$\|\lambda - \lambda^h\|$	$\|e^h\|_0$	#K
2	3.06e-5	3.30e-6	7	2.61e-5	3.40e-6	8	3.67e-5	3.47e-6	8
4	3.10e-5	3.26e-6	6	3.09e-5	3.17e-6	6	3.24e-5	3.19e-6	8
6	4.25e-5	4.42e-6	8	2.98e-5	3.24e-6	7	3.09e-5	3.30e-6	6

Case $d = 2$.

Again, we show that the initial guess by the deep learning approach enables the power method to quickly converge to a solution with the precision of finite element methods, i.e., the numerical convergence order in terms of h is 2 as proved by Theorem 4.2. We repeatedly apply the variational formula in (3.15) (with $\boldsymbol{x}_0 = (0.5, 0.5)$ and $\gamma = 100$) and the deep learning method to generate different initial guesses u_0^h for different sizes h. Let $N_{\max} = 10$ and $\epsilon = h^2$ in Algorithm 2. Table 14 summarizes the performance of Algorithm 2 and numerical results verify the accuracy and the convergence order of the Int-Deep framework, even though the number of epochs in deep learning is $O(100)$.

Journal Pre-proof

Table 14: The performance of Int-Deep in Algorithm 2 for Example 5.5 in 2D with different mesh sizes h.

h	#Epoch	$\|e^{DL}\|_0$	#K	$\|\lambda-\lambda^h\|$	eigenvalue order	$\|e^h\|_0$	eigenfunction order
2^{-4}	300	5.56e-2	4	1.91e-1	-	6.63e-3	-
2^{-5}	300	5.65e-2	5	4.76e-2	2.00	1.70e-3	1.96
2^{-6}	300	5.68e-2	7	1.19e-2	2.00	4.19e-4	2.02
2^{-7}	300	5.69e-2	8	2.97e-3	2.00	1.07e-4	1.97
2^{-8}	300	5.69e-2	10	7.43e-4	2.00	2.62e-5	2.03

Next, we show that the performance of Int-Deep in Algorithm 2 is independent of the size of DNNs, which is supported by the numerical results in Table 13.

Table 15: The performance of Int-Deep in Algorithm 2 for Example 5.5 in 2D with different DNN width N and depth L when $h=\frac{1}{128}$ and #Epoch = 400.

N / L	10			30			50		
	$\|\lambda-\lambda^h\|$	$\|e^h\|_0$	#K	$\|\lambda-\lambda^h\|$	$\|e^h\|_0$	#K	$\|\lambda-\lambda^h\|$	$\|e^h\|_0$	#K
2	2.97e-3	1.09e-4	9	2.97e-3	1.08e-4	8	2.97e-3	1.08e-4	8
4	2.97e-3	1.07e-4	7	2.97e-3	1.07e-4	7	2.97e-3	1.09e-4	7
6	2.97e-3	1.07e-4	6	2.97e-3	1.11e-4	7	2.97e-3	1.05e-4	8

5.2.3 Nonlinear eigenvalue problems

As used in [24], denote by res^{DL} and res^h the residuals of u^{DL} and u^h corresponding to the matrix form of (4.11), respectively. In this example, we denote $\|\cdot\|_2$ as the vector L_2-norm in Euclidian space. Then we investigate the accuracy of Ini-Deep framework in Algorithm 3 through the case of the nonlinear Schrödinger equation.

Example 5.6. *Consider the following problem*

$$\begin{cases} -\Delta u + V(x)u + 10u^3 = \lambda u & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \\ \|u\|_0 = 1, \end{cases}$$

where $\Omega=(0,1)^d$ for $d=1,2$, $V(x)$ is a potential with two Gaussian wells on $(0,1)$ for $d=1$, and with four Gaussian wells on $(0,1)^2$ for $d=2$.

Case $d=1$

We first show that the initial guess by the deep learning approach enables Newton's method to quickly converge to the target solution. We repeatedly apply the variational formula in (3.19) (with $x_0=0.5$ and $\gamma=100$) and the deep learning method to generate different initial guesses u_0^h for different sizes h. Let $N_{\max}=10$ and $\epsilon=h^2$ in Algorithm 3. Table 16 summarizes the performance of Algorithm 3.

Journal Pre-proof

Table 16: The performance of Int-Deep in Algorithm 3 for Example 5.6 in 1D with different mesh sizes h.

h	#Epoch	$\|res^{DL}\|_2$	λ^{DL}	#K	$\|res^h\|_2$	λ^h
2^{-5}	300	6.10e-1	23.71	2	2.41e-6	23.60
2^{-6}	300	4.72e-1	23.69	3	1.13e-10	23.58
2^{-7}	300	3.46e-1	23.69	3	1.54e-11	23.58
2^{-8}	300	2.49e-1	23.69	3	2.57e-12	23.58
2^{-9}	300	1.77e-1	23.69	3	1.51e-12	23.58

Next, we show that the performance of Int-Deep in Algorithm 3 is independent of the size of DNNs, which is supported by the numerical results in Table 17.

Table 17: The performance of Int-Deep in Algorithm 3 for Example 5.6 in 1D with different DNN width N and depth L when $h = \frac{1}{512}$ and #Epoch = 400.

N / L	10			30			50		
	λ^h	$\|res^h\|_2$	#K	λ^h	$\|res^h\|_2$	#K	λ^h	$\|res^h\|_2$	#K
2	23.58	1.33e-12	5	23.58	1.44e-12	4	23.58	1.50e-12	3
4	23.58	1.53e-12	3	23.58	2.14e-12	3	23.58	1.48e-12	3
6	23.58	1.44e-12	3	23.58	1.49e-12	3	23.58	1.34e-12	3

Case $d = 2$.

Again, we show that the initial guess by the deep learning approach enables Newton's method to quickly converge to the target solution. We repeatedly apply the variational formula in (3.19) (with $\boldsymbol{x}_0 = (0.5, 0.5)$ and $\gamma = 100$) and the deep learning method to generate different initial guesses u_0^h for different sizes h. Let $N_{\max} = 10$ and $\epsilon = h^2$ in Algorithm 3. Table 18 summarizes the performance of Algorithm 3.

Table 18: The performance of Int-Deep in Algorithm 3 for Example 5.6 in 2D with different mesh sizes h.

h	#Epoch	$\|res^{DL}\|_2$	λ^{DL}	#K	$\|res^h\|_2$	λ^h
2^{-4}	300	4.07e-1	39.74	2	9.06e-6	39.46
2^{-5}	300	2.46e-1	39.49	2	2.13e-6	39.19
2^{-6}	300	1.35e-1	39.43	3	1.43e-10	39.12
2^{-7}	300	7.04e-2	39.42	3	1.53e-11	39.10

Next, we show that the performance of Int-Deep in Algorithm 3 is independent of the size of DNNs, which is supported by the numerical results in Table 19.

Table 19: The performance of Int-Deep in Algorithm 3 for Example 5.6 in 2D with different DNN width N and depth L when $h = \frac{1}{128}$ and #Epoch = 400.

N / L	10			30			50		
	λ^h	$\|res^h\|_2$	#K	λ^h	$\|res^h\|_2$	#K	λ^h	$\|res^h\|_2$	#K
2	39.10	1.60e-12	3	39.10	7.88e-13	4	39.10	4.26e-13	3
4	39.10	3.28e-10	3	39.10	1.30e-11	3	39.10	6.33e-12	3
6	39.10	3.77e-11	3	39.10	8.30e-11	3	39.10	2.10e-10	3

Journal Pre-proof

6 Conclusion

This paper proposed the Int-Deep framework from a new point of view for designing highly efficient solvers of low-dimensional nonlinear PDEs with a finite element accuracy leveraging both the advantages of traditional algorithms and deep learning approaches. The Int-Deep framework consists of two phases. In the first phase, an approximate solution to the given nonlinear PDE is obtained via deep learning approaches using DNNs of size $O(1)$ and $O(100)$ iterations. In the second phase, the approximate solution provided by deep learning can serve as a good initial guess such that traditional iterative methods converge in $O(\log(\frac{1}{\epsilon}))$ iterations to the ϵ precision of finite element methods. The Int-Deep framework outperforms existing purely deep learning-based methods or traditional iterative methods. The code can be shared per request.

In particular, based on variational principles, we propose new methods to formulate the problem of solving nonlinear PDEs into an unconstrained minimization problem of an expectation over a function space parametrized via DNNs, which can be solved efficiently via batch stochastic gradient descent (SGD) methods due to the special form of expectation. Unlike previous methods in which the form of expectation is only derived for nonlinear PDEs related to variational equations, our proposed method can also handle variational inequalities and eigenvalue problems, providing a unified variational framework for a wider range of nonlinear problems. With the good initialization given by deep learning, we hope to reduce the difficulty of designing an efficient traditional iterative algorithm for variational inequalities. We will leave this as future work.

Acknowledgments. J. H. was partially supported by NSFC (Grant No.11571237). H. Y. was partially supported by the National Science Foundation under the grant award 1945029. We would like to thank reviewers for their valuable comments to improve the manuscript.

References

[1] R.A. Adams. *Sobolev Spaces*. Academic Press, New York-London, 1975.

[2] A. Ambrosetti and A. Malchiodi. *Nonlinear Analysis and Semilinear Elliptic Problems*. Cambridge University Press, Cambridge, 2007.

[3] J.R. Anglin and W. Ketterle. Bose -Einstein Condensation of Atomic Gases. *Nature*, 416:211–218, 2002.

[4] I. Babuška and J. Osborn. Eigenvalue Problems. In *Handbook of Numerical Analysis*. North-Holland, Amsterdam, 1991.

[5] I. Babuška and J. E. Osborn. Finite Element-Galerkin Approximation of the Eigenvalues and Eigenvectors of Selfadjoint Problems. *Math. Comp.*, 52:275–297, 1989.

[6] A.R. Barron. Universal Approximation Bounds for Superpositions of a Sigmoidal Function. *IEEE Trans. Inform. Theory*, 39:930–945, 1993.

[7] S. Bartels. *Numerical Methods for Nonlinear Partial Differential Equations*. Springer, Cham, 2015.

[8] C. Basdevant, M. Deville, P. Haldenwang, J.M. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, and A.T Patera. Spectral and Finite Difference Solutions of the Burgers Equation. *Comput. & Fluids*, 14:23 – 41, 1986.

Journal Pre-proof

[9] J. Berg and K. Nyström. A Unified Deep Artificial Neural Network Approach to Partial Differential Equations in Complex Geometries. *Neurocomputing*, 317:28 – 41, 2018.

[10] P. B. Bochev and M. D. Gunzburger. *Least-squares Finite Element Methods*. Springer, New York, 2009.

[11] S.C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*. Springer, New York, 1994.

[12] E. Cancès, R. Chakir, L. He, and Y. Maday. Two-grid Methods for a Class of Nonlinear Elliptic Eigenvalue Problems. *IMA J. Numer. Anal.*, 38:605–645, 2018.

[13] E. Cancès, R. Chakir, and Y. Maday. Numerical Analysis of Nonlinear Eigenvalue Problems. *J. Sci. Comput.*, 45:90–117, 2010.

[14] G. Carleo and M. Troyer. Solving the Quantum Many-body Problem with Artificial Neural Networks. *Science*, 355:602–606, 2017.

[15] A. J. Chorin. Numerical Solution of the Navier-Stokes Equations. *Math. Comp.*, 22:745–762, 1968.

[16] P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, Amsterdam-New York-Oxford, 1978.

[17] M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based Approximations for Solving Partial Differential Equations. *Comm. Numer. Methods Engrg.*, 10:195–201, 1994.

[18] G. Duvaut and J. -L. Lions. *Inequalities in Mechanics and Physics*. Springer, Berlin-New York, 1976.

[19] W. E and B. Yu. The Deep Ritz Method: a Deep Learning-based Numerical Algorithm for Solving Variational Problems. *Commun. Math. Stat.*, 6:1–12, 2018.

[20] Y. Fan, J. Feliu-Fabà, L. Lin, L. Ying, and L. Zepeda-Núñez. A Multiscale Neural Network Based on Hierarchical Nested Bases. *Res. Math. Sci.*, 6:21–28, 2019.

[21] Y. Fan, L. Lin, L. Ying, and L. Zepeda-Núñez. A Multiscale Neural Network Based on Hierarchical Matrices. *Multiscale Model. Simul.*, 17:1189–1213, 2019.

[22] Y. Fan, C. Orozco Bohorquez, and L. Ying. BCR-Net: a Neural Network Based on the Nonstandard Wavelet Form. *J. Comput. Phys.*, 384:1–15, 2019.

[23] A. L. Fetter. Vortices in an Imperfect Bose Gas. I. The Condensate. *Phys. Rev. (2)*, 138:A429–A437, 1965.

[24] W. Gao, C. Yang, and J. Meza. Solving a Class of Nonlinear Eigenvalue Problems by Newton's Method. 2009.

[25] R. Glowinski. *Numerical Methods for Nonlinear Variational Problems*. Springer, New York, 1984.

[26] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, 2016.

[27] Y. Gu, H. Yang, and C. Zhou. SelectNet: Self-paced Learning for High-dimensional Partial Differential Equations. *arXiv e-prints*, page arXiv:2001.04860, 2020.

Journal Pre-proof

[28] J. Han, A. Jentzen, and W. E. Solving High-dimensional Partial Differential Equations Using Deep Learning. *Proc. Natl. Acad. Sci. USA*, 115:8505–8510, 2018.

[29] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[30] P. Hohenberg and W. Kohn. Inhomogeneous Electron Gas. *Phys. Rev. (2)*, 136:B864–B871, 1964.

[31] J.M. Hyman and B. Nicolaenko. The Kuramoto-Sivashinsky Equation: a Bridge between PDE's and Dynamical Systems. *Phys. D*, 18:113 – 126, 1986.

[32] Y. Khoo, J. Lu, and L. Ying. Solving Parametric PDE Problems with Artificial Neural Networks. *arXiv e-prints*, page arXiv:1707.03351, 2017.

[33] Y. Khoo, J. Lu, and L. Ying. Solving for High-dimensional Committor Functions Using Artificial Neural Networks. *Res. Math. Sci.*, 6:1–13, 2019.

[34] Y. Khoo and L. Ying. SwitchNet: a Neural Network Model for Forward and Inverse Scattering Problems. *SIAM J. Sci. Comput.*, 41:A3182–A3201, 2019.

[35] D. P. Kingma and J. Ba. Adam: a Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, 2014.

[36] W. Kohn and L. J. Sham. Self-consistent Equations Including Exchange and Correlation effects. *Phys. Rev.(2)*, 140:A1133–A1138, 1965.

[37] V. Kůrková. Kolmogorov's Theorem and Multilayer Neural Networks. *Neural Networks*, 5:501–506, 1992.

[38] I.E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial Neural Networks for Solving Ordinary and Partial Differential Equations. *IEEE Trans. Neural Networks*, 9:987–1000, 1998.

[39] J. Lu, Z. Shen, H. Yang, and S. Zhang. Deep Network Approximation for Smooth Functions. *arXiv e-prints*, page arXiv:2001.03040, 2020.

[40] K.S. McFall and J. R. Mahan. Artificial Neural Network Method for Solution of Boundary Value Problems with Exact Satisfaction of Arbitrary Boundary Conditions. *IEEE Trans. Neural Networks*, 20:1221–1233, 2009.

[41] A. J. Meade and A. A. Fernández. Solution of Nonlinear Ordinary Differential Equations by Feedforward Neural Networks. *Math. Comput. Modelling*, 20:19 – 44, 1994.

[42] R. M. Miura. The KortewegšcdeVries Equation: a Survey of Results. *SIAM Rev.*, 18:412–459, 1976.

[43] H. Montanelli and Q. Du. New Error Bounds for Deep ReLU Networks Using Sparse Grids. *SIAM J. Math. Data Sci.*, 1:78–92, 2019.

[44] H. Montanelli and H. Yang. Error Bounds for Deep ReLU Networks Using the Kolmogorov–Arnold Superposition Theorem. *arXiv:1906.11945*, 2019.

Journal Pre-proof

[45] H. Montanelli, H. Yang, and Q. Du. Deep ReLU Networks Overcome the Curse of Dimensionality for Bandlimited Functions. *arXiv:1903.00735*, 2019.

[46] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer, Berlin, 1994.

[47] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed Neural Networks: a Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *J. Comput. Phys.*, 378:686 – 707, 2019.

[48] K. Rudd and S. Ferrari. A Constrained Integration (CINT) Approach to Solving Partial Differential Equations Using Artificial Neural Networks. *Neurocomputing*, 155:277 – 285, 2015.

[49] K. Shao, J. Chen, Z. Zhao, and D. H. Zhang. Communication: Fitting Potential Energy Surfaces with Fundamental Invariant Neural Network. *J. Chem. Phys.*, 145:071101, 2016.

[50] R. Shekari Beidokhti and A. Malek. Solving Initial-boundary Value Problems for Systems of Partial Differential Equations Using Neural Networks and Optimization Techniques. *J. Franklin Inst.*, 346:898 – 913, 2009.

[51] Z. Shen, H. Yang, and S. Zhang. Deep Network Approximation Characterized by Number of Neurons. *arXiv e-prints*, page arXiv:1906.05497, 2019.

[52] Z. Shen, H. Yang, and S. Zhang. Nonlinear Approximation via Compositions. *Neural Networks*, 119:74 – 84, 2019.

[53] J. Sirignano and K. Spiliopoulos. DGM: a Deep Learning Algorithm for Solving Partial Differential Equations. *J. Comput. Phys.*, 375:1339 – 1364, 2018.

[54] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis(Third Edition)*. Springer, New York, 2002.

[55] M. Sun, X. Yan, and R. J. Sclabassi. Solving Partial Differential Equations in Real-time Using Artificial Neural Network Signal Processing as an Alternative to Finite-element Analysis. In *International Conference on Neural Networks and Signal Processing, 2003.*, volume 1, pages 381–384, China, Nanjing, 2003.

[56] W. Tang, T. Shan, X. Dang, F. M. Li, M.and Yang, S. Xu, and J. Wu. Study on a Poisson's Equation Solver Based on Deep Learning Technique. In *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pages 1–3, 2017.

[57] C. Xie, X. Zhu, D. R. Yarkony, and H. Guo. Permutation Invariant Polynomial Neural Network Approach to Fitting Potential Energy Surfaces. IV. Coupled Diabatic Potential Energy Matrices. *J. Chem. Phys.*, 149:144107, 2018.

[58] J. Xu. A Novel Two-Grid Method for Semilinear Elliptic Equations. *SIAM J. Sci. Comput.*, 15:231–237, 1994.

[59] J. Xu. Two-grid Discretization Techniques for Linear and Nonlinear PDEs. *SIAM J. Numer. Anal.*, 33:1759–1777, 1996.

[60] J Xu and A Zhou. A Two-grid Discretization Scheme for Eigenvalue Problems. *Math. Comp.*, 70:17–25, 2001.

Journal Pre-proof

[61] D. Yarotsky. Error Bounds for Approximations with Deep ReLU Networks. *Neural Networks*, 94:103–114, 2017.

[62] D. Yarotsky. Optimal Approximation of Continuous Functions by very Deep ReLU Networks. In *31st Annual Conference on Learning Theory*, volume 75, pages 1–11. 2018.

[63] Y. Zang, G. Bao, X. Ye, and H. Zhou. Weak Adversarial Networks for High-dimensional Partial Differential Equations. *arXiv e-prints*, page arXiv:1907.08272, 2019.

[64] L. Zhang, J. Han, H. Wang, W. A. Saidi, R. Car, and W. E. End-to-end Symmetry Preserving Inter-atomic Potential Energy Model for Finite and Extended Systems. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4441–4451, USA, 2018. Curran Associates Inc.

[65] A. Zhou. An Analysis of Finite-dimensional Approximations for the Ground State Solution of Bose-Einstein Condensates. *Nonlinearity*, 17:541–550, 2004.

[66] C. Zhu and Q. Lin. *The Hyperconvergence Theory of Finite Elements*. Hunan Science and Technology Publishing House, Changsha, 1989.

[67] O. C. Zienkiewicz, R. L. Taylor, and J.Z. Zhu. *Finite Element Method(Six Edition)*. Elsevier, Amsterdam, 2005.

Journal Pre-proof

Journal Pre-proof

Appendix

A Proof of Theorem 4.1

Making use of the assumption A2 in Section 4.1, we immediately have the following result (see [58]).

Lemma A.1. *Let $u \in W^{2,\infty}(\Omega)$ be a solution of (4.1). There exist two positive constant C_1^u and C_2^u such that if a function v satisfies that $\|v-u\|_{0,\infty} \le C_1^u$, then*

$$C_2^u \|\phi\|_1 \le \sup_{\chi \in V} \frac{a_v(\phi,\chi)}{\|\chi\|_1}, \quad \phi \in V.$$

The mathematical analysis for the finite element method (4.5) is very technical and has been established in [59]. In the following two lemmas, we collect some results which will be used frequently later on.

Lemma A.2. *Let $u \in W^{2,\infty}(\Omega)$ be a solution of (4.2). Then there exists a constant $h_0 > 0$ and a positive constant C_3^u independent of h such that if $h < h_0$ and a function v satisfies $\|v-u\|_{0,\infty} \le C_1^u$, then*

$$C_3^u \|\phi\|_1 \le \sup_{\chi \in V_h} \frac{a_v(\phi,\chi)}{\|\chi\|_1}, \quad \phi \in V_h.$$

This lemma can be derived by using Lemma A.1 and the arguments for proving Lemma 2.2 in [59].

Lemma A.3. *Let $u \in W^{2,\infty}(\Omega)$ be a solution of (4.2). Then there exists a positive constant $h_1 < h_0$ such that if $h < h_1$, the finite element method (4.5) has exactly one solution u^h satisfying*

$$\|u - u^h\|_1 \le C_4^u$$

for a positive constant C_4^u independent of h. Moreover,

$$\lim_{h \to 0^+} \|u - u^h\|_1 = 0.$$

Lemma A.4. *Let $u \in W^{2,\infty}(\Omega)$ be a solution of (4.2) and u^h be the finite element method of (4.5). Then, there exists a positive constant $h_2 < h_1$ such that if $h < h_2$,*

$$\begin{aligned} &\|u - u^h\|_{0,\infty} \lesssim h^2 && \text{when } d = 1, \\ &\|u - u^h\|_{0,\infty} \lesssim h^2 |\log(h)| && \text{when } d = 2. \end{aligned}$$

Proof. The estimate for $d = 2$ is given in [59]. We will derive the estimate for $d = 1$ following some ideas in [59]. To this end, we first introduce an elliptic projection operator P_h such that if $u \in V$, then $P_h u \in V_h$ satisfies

$$(\nabla(P_h u), \nabla\chi) = (\nabla u, \nabla\chi), \quad \chi \in V_h. \tag{A.1}$$

It is shown in [66] that

$$\|u - P_h u\|_{0,\infty} \lesssim \|u - P_h u\|_1 \lesssim h^2 |u|_{2,\infty}. \tag{A.2}$$

On the other hand, the finite element solution u^h satisfies

$$(\nabla u^h, \nabla\chi) + (f(u^h), \chi) = 0, \quad \chi \in V_h.$$

Journal Pre-proof

Recalling the relation (A.1) and using (4.2) gives

$$(\nabla(P_h u), \nabla\chi) = (\nabla u, \nabla\chi) = -(f(u), \chi), \quad \chi \in V_h.$$

Hence, subtracting the last two equations and using Taylor's expansion yield

$$a_{P_h u}((u^h - P_h u), \chi) = -\frac{1}{2}(f''(\zeta)(u^h - P_h u)^2, \chi) - (f(P_h u) - f(u), \chi), \quad \chi \in V_h,$$

where $\zeta = \nu(x)u^h(x) + (1-\nu(x))(P_h u)(x)$ for some $\nu(x) \in (0,1)$. We have by the estimate (A.2) that there exists a positive constant $\bar{h} < h_1$ such that if $h < \bar{h}$, $\|P_h u - u\|_{0,\infty} \le C_1^u$. In this case, it follows from Lemma A.2 and the last equation that

$$\begin{aligned}
\|u^h - P_h u\|_1 &\lesssim \sup_{\chi \in V_h} \frac{a_{P_h u}((u^h - P_h u), \chi)}{\|\chi\|_1} \\
&\lesssim \|u^h - P_h u\|_{0,\infty}^2 + \|f(u) - f(P_h u)\|_{0,\infty} \\
&\lesssim \|u^h - P_h u\|_1^2 + \|u - P_h u\|_{0,\infty} \\
&\le C\|u^h - P_h u\|_1^2 + C\|u - P_h u\|_{0,\infty}, \qquad \text{(A.3)}
\end{aligned}$$

where $C > 0$ is a generic constant. On the other hand, it follows from Lemma A.3 and the estimate (A.2) that

$$\|u^h - P_h u\|_1 \le \|u - u^h\|_1 + \|u - P_h u\|_1 \to 0 \quad \text{as } h \to 0^+.$$

Hence, there exists a positive constant $h_2 < \bar{h}$ such that if $h < h_2$, then $C\|u^h - P_h u\|_1 < 1/2$, which combined with (A.3) readily implies

$$\|u^h - P_h u\|_1 \lesssim \|u - P_h u\|_{0,\infty}.$$

Therefore, by the Sobolev embedding theorem and the estimate (A.2),

$$\begin{aligned}
\|u - u^h\|_{0,\infty} &\le \|u - P_h u\|_{0,\infty} + \|P_h u - u^h\|_{0,\infty} \\
&\lesssim \|u - P_h u\|_{0,\infty} + \|P_h u - u^h\|_1 \lesssim \|u - P_h u\|_{0,\infty} \lesssim h^2 |u|_{2,\infty},
\end{aligned}$$

as required. □

Remark 1. *As shown in [11], we require to make certain strong regularity assumption on the solution to problem* (4.2) *for* $d = 3$*, so as to derive the maximum norm estimate for the related finite element method. To avoid too technical treatment, we skip the further discussion in this case.*

The following result plays an important role in the convergence analysis of Int-Deep for solving the finite element method (4.5), which will be introduced later on.

Lemma A.5. *Let* $u \in W^{2,\infty}(\Omega)$ *be a solution of* (4.2)*. Let* u^h *be an approximation of* u *obtained by the finite element method* (4.5)*. Assume that* $h < h_2$ *with* h_2 *the same as given in Lemma A.4. For any* $v \in V_h \cap B(u)$*, define a mapping* $Tv = v + w$*, where* $w \in V_h$ *is uniquely determined by*

$$(\nabla w, \nabla\chi) + (f'(v)w, \chi) = -(\nabla v, \nabla\chi) - (f(v), \chi), \quad \chi \in V_h. \qquad \text{(A.4)}$$

Then there holds

$$\|u^h - Tv\|_1 \lesssim \|u^h - v\|_{0,p}^2 \lesssim \|u^h - v\|_1^2.$$

Journal Pre-proof

Proof. Recalling the definition (4.5), we know

$$(\nabla u^h, \nabla \chi) + (f(u^h), \chi) = 0, \quad \chi \in V_h,$$

Subtracting the above equation from (A.4) and reorganizing terms, we find

$$(\nabla E_h, \nabla \chi) + (f(u^h) - f(v) - f'(v)w, \chi) = 0, \quad \chi \in V_h,$$

where $E_h = u^h - Tv$. Furthermore, use Taylor's expansion to get

$$a_v(E_h, \chi) = -\frac{1}{2}(f''(\zeta)(u^h - v)^2, \chi), \quad \chi \in V_h,$$

where $\zeta = (1-\nu(x))v(x) + \nu(x)u^h(x)$ with $\nu(x) \in (0,1)$. Since $h < h_2$ and $v \in B(u)$, from Lemma A.3 it follows that $\|\zeta_h\|_{0,\infty}$ is uniformly bounded with respect to h. Hence, by the Hölder inequality and the Sobolev embedding theorem, for $p > 2$ and any $\chi \in V_h$,

$$\begin{aligned} a_v(E_h, \chi) &\lesssim \int_\Omega (u^h - v)^2 |\chi| \, \mathrm{d}x \\ &\lesssim \|(u^h - v)^2\|_{0,p/2} \|\chi\|_{0,p/p-2} \lesssim \|u^h - v\|_{0,p}^2 \|\chi\|_1, \end{aligned}$$

where the generic constant is independent of h but depends on p. The combination of the last estimate with Lemma A.2 immediately implies

$$\|E_h\|_1 \lesssim \|u^h - v\|_{0,p}^2 \lesssim \|u^h - v\|_1^2,$$

as required. □

Now we are ready to prove Theorem 4.1.

Proof. Denote $E_k^h = u^h - u_k^h$. Applying Lemma A.5 gives rise to

$$\|E_{k+1}^h\|_1 \lesssim \|E_k^h\|_{0,p}^2 \lesssim \|E_k^h\|_1^2. \tag{A.5}$$

When $d = 1$, By the Sobolev embedding theorem and (A.5),

$$\|E_{k+1}^h\|_{0,\infty} \lesssim \|E_{k+1}^h\|_1 \lesssim \|E_k^h\|_{0,p}^2 \lesssim \|E_k^h\|_{0,\infty}^2.$$

This means there exists a positive constant c_1 such that $\|E_k^h\|_{0,\infty} \le c_1 \|E_{k-1}^h\|_{0,\infty}^2$. Hence,

$$c_1 \|E_k^h\|_{0,\infty} \le (c_1 \|E_{k-1}^h\|_{0,\infty})^2 \le (c_1 \|E_0^h\|_{0,\infty})^{2^k}.$$

On the other hand, it follows from Lemma A.4 and error estimates for the interpolation operator I_h (cf. [11, 16]) that

$$\begin{aligned} \|E_0^h\|_{0,\infty} &= \|u^h - u_0^h\|_{0,\infty} \\ &\le \|u^h - u\|_{0,\infty} + \|u - I_h u\|_{0,\infty} + \|I_h u - I_h u^{DL}\|_{0,\infty} \\ &\le c_0(h^2 + \delta), \end{aligned}$$

where $c_0 > 0$ is a generic constant. Let $\beta_1 = c_1 c_0 (h^2 + \delta)$. Then

$$\|E_k^h\|_{0,\infty} \le \beta_1^{2^k} / c_1. \tag{A.6}$$

Journal Pre-proof

When $d = 2$, for any given number $p > 2$, we have by the Sobolev embedding theorem and (A.5) that

$$\|E_{k+1}^h\|_{0,p} \lesssim \|E_{k+1}^h\|_1^2 \lesssim \|E_k^h\|_{0,p}^2,$$

which implies $c_2\|E_k^h\|_{0,p} \leq (c_2\|E_0^h\|_{0,p})^{2^k}$ for a generic positive constant c_2. In addition, by Lemma A.4 and error estimates for I_h,

$$\begin{aligned}\|E_0^h\|_{0,p} &= \|u^h - u_0^h\|_{0,p}\\ &\leq \|u^h - u\|_{0,p} + \|u - I_h u\|_{0,p} + \|I_h u - u_0^h\|_{0,p}\\ &\lesssim \|u^h - u\|_{0,\infty} + \|u - I_h u\|_{0,\infty} + \|I_h u - u_0^h\|_{0,\infty}\\ &\leq c_3(h^2|\log h| + \delta),\end{aligned}$$

where $c_3 > 0$ is a generic constant independent of h and δ but depending on p. Let $\beta_2 = c_2c_3(h^2|\log h| + \delta)$. Then

$$\|E_k^h\|_{0,p} \leq \beta_2^{2^k}/c_2,$$

and further by the inverse inequality for finite elements,

$$\|E_k^h\|_{0,\infty} \lesssim h^{-2/p}\|E_k^h\|_{0,p} \lesssim h^{-2/p}\beta_2^{2^k}. \tag{A.7}$$

Now, we have by (A.6), (A.7) and Lemma A.4 that

$$\|u - u_k^h\|_{0,\infty} \leq \|u - u^h\|_{0,\infty} + \|u^h - u_k^h\|_{0,\infty} \lesssim h^2 + \beta_1^{2^k} \quad \text{for } d = 1,$$

and

$$\|u - u_k^h\|_{0,\infty} \leq \|u - u^h\|_{0,\infty} + \|u^h - u_k^h\|_{0,\infty} \lesssim h^2|\log h| + h^{-2/p}\beta_2^{2^k} \quad \text{for } d = 2.$$

The proof is complete.

□

B Proof of Theorem 4.2

The proof of Theorem 4.2 is rather involved and requires the following elementary but nontrivial result as a key bridge to produce the optimal convergence analysis.

Lemma B.1. *Let $\{a_k\}$ be a sequence satisfying $a_{k+1} \leq a_k^2 + b$ for $k = 0, 1, 2, \cdots$. If $0 \leq a_0 < 1/2$ and $0 < b < 1/4$, then*

$$a_k \leq a_0^{2^k} + \left(2 + \frac{1}{1 - 2a_0}\right) b. \tag{B.1}$$

Proof. First of all, construct an auxiliary sequence $\{A_k\}$ generated by

$$A_{k+1} = A_k^2 + b, \quad k = 0, 1, 2, \cdots; \quad A_0 = a_0. \tag{B.2}$$

It is easy to check $a_k \leq A_k$ for all nonnegative integer k. The recursive relation (B.2) is a fixed point iteration corresponding to the following fixed point equation

$$A = A^2 + b \Rightarrow A^2 - A + b = 0, \tag{B.3}$$

which has exactly two fixed points

$$\alpha_0 = \frac{2b}{1 + \sqrt{1 - 4b}} \quad \text{and} \quad \alpha_1 = \frac{1 + \sqrt{1 - 4b}}{2}.$$

Journal Pre-proof

Next, let us study the boundedness of the sequence $\{A_k\}$. When $A_0 \le \alpha_0$, we have by mathematical induction that $A_k \le \alpha_0$ for $k = 0, 1, 2, \cdots$. In fact, if $k = 0$ the statement is true. Now we assume it holds for $k = m$ with m an any given natural number, i.e. $A_m \le \alpha_0$. Then, observing that α_0 satisfies the equation (B.3), we immediately know

$$A_{m+1} = A_m^2 + b \le \alpha_0^2 + b = \alpha_0.$$

So the statement is really true by the principle of mathematical induction. On the other hand, it is easy to check by a direct computation that $\alpha_0 \le 2b$. Hence, if $A_0 \le \alpha_0$, there holds

$$a_k \le A_k \le \alpha_0 \le 2b. \tag{B.4}$$

When $\alpha_0 \le A_0 \le \alpha_1$, we use mathematical induction again to know $\alpha_0 \le A_k \le \alpha_1$ for any nonnegative integer k. Let us now consider a function $f(x)$ defined by

$$f(x) = x + \frac{b}{x}, \quad x \in [\alpha_0, \alpha_1].$$

Clearly, it is convex over $[\alpha_0, \alpha_1]$ and hence must take the maximum value over the interval at one of the two end points. However, recalling the definitions of α_0 and α_1, we easily know

$$f(\alpha_0) = f(\alpha_1) = 1,$$

which implies $f(x) \le 1$ for $x \in [\alpha_0, \alpha_1]$. Thus, since $A_k \in [\alpha_0, \alpha_1]$, we find

$$\frac{A_{k+1}}{A_k} = \frac{A_k^2 + b}{A_k} = f(A_k) \le 1;$$

in other words, $\{A_k\}$ is a decreasing sequence. Therefore, the result $A_k \le a_0$ also holds in this case. To sum up, we find that if $a_0 < 1/2 < \alpha_1$, $A_k \le a_0$ for all nonnegative integer k.

To further our analysis, we require to construct another auxiliary sequence generated by

$$B_{k+1} = B_k^2, \quad k = 0, 1, 2, \cdots; \quad B_0 = a_0. \tag{B.5}$$

It is evident that $B_k = B_0^{2^k} = a_0^{2^k}$ and $B_k \le A_k$. Denote $d_k = A_k - B_k \ge 0$. Recalling the definitions (B.2) and (B.5), and using the fact that $A_k \le a_0$ obtained above, we find

$$d_{k+1} = A_k^2 + b - B_k^2 = d_k(A_k + B_k) + b \le 2a_0 d_k + b,$$

which readily yields

$$d_k \le (2a_0)^k d_0 + \frac{1 - (2a_0)^k}{1 - 2a_0} b \le \frac{1}{1 - 2a_0} b$$

by noting that $d_0 = 0$. Therefore,

$$A_k = B_k + d_k \le B_0^{2^k} + \frac{1 - (2a_0)^k}{1 - 2a_0} b = a_0^{2^k} + \frac{1 - (2a_0)^k}{1 - 2a_0} b$$

and

$$a_k \le A_k \le a_0^{2^k} + \frac{1}{1 - 2a_0} b. \tag{B.6}$$

We then obtain the required estimate by combining (B.4) and (B.6). □

Now we are ready to present the proof of Theorem 4.2.

Journal Pre-proof

Proof. For any $\phi \in V_h$, we know

$$\begin{aligned} a(P_h u - u_{k+1}^h, \chi) &= a(u, \chi) - a(u_{k+1}^h, \chi) = \lambda(u, \chi) - \lambda_k^h(u_k^h, \chi) \\ &= (\lambda - \lambda_k^h)(u, \chi) + \lambda_k^h(u - u_k^h, \chi). \end{aligned}$$

Choosing $\chi = P_h u - u_{k+1}^h$, we have by the coerciveness of $a(\cdot, \cdot)$ that

$$\begin{aligned} \alpha_0 \|P_h u - u_{k+1}^h\|_1^2 &\le a(P_h u - u_{k+1}^h, P_h u - u_{k+1}^h) \\ &\le |\lambda - \lambda_k^h|(u, P_h u - u_{k+1}^h) + |\lambda_k^h|(u - u_k^h, P_h u - u_{k+1}^h) \\ &\lesssim \left(|\lambda - \lambda_k^h| \|u\|_{-1} + |\lambda_k^h - \lambda + \lambda| \|u - u_k^h\|_{-1}\right) \|P_h u - u_{k+1}^h\|_1, \end{aligned}$$

which combined with the assumption $\|u_k^h\|_0 \le \varepsilon_1$ implies

$$\begin{aligned} \|P_h u - u_{k+1}^h\|_1 &\lesssim |\lambda - \lambda_k^h| \|u\|_0 + \left(|\lambda - \lambda_k^h| + |\lambda|\right) \|u - u_k^h\|_0 \\ &\lesssim |\lambda - \lambda_k^h| + |\lambda - \lambda_k^h| \left(\|u\|_0 + \|u_k^h\|_0\right) + \|u - u_k^h\|_0 \\ &\lesssim |\lambda - \lambda_k^h| + \|u - u_k^h\|_0. \end{aligned}$$

Hence, by the error estimate for P_h and the triangle inequality,

$$\|u - u_{k+1}^h\|_1 \le \|u - P_h u\|_1 + \|P_h u - u_{k+1}^h\|_1 \lesssim h + |\lambda - \lambda_k^h| + \|u - u_k^h\|_0. \tag{B.7}$$

Using the duality argument to the equation determining u_{k+1}^h, we deduce from (B.7) and the Cauchy inequality that

$$\begin{aligned} \|u - u_{k+1}^h\|_0 &\lesssim h \|u - u_{k+1}^h\|_1 \lesssim h^2 + h \left(|\lambda - \lambda_k^h| + \|u - u_k^h\|_0\right) \\ &\lesssim h^2 + \left(|\lambda - \lambda_k^h| + \|u - u_k^h\|_0\right)^2. \end{aligned} \tag{B.8}$$

By Lemma 4.1 and assumption $\|u_{k+1}^h\|_0 \ge \varepsilon_0 > 0$,

$$|\lambda - \lambda_{k+1}^h| = \left||u_{k+1}^h - u|_1^2 - \lambda \|u_{k+1}^h - u\|_0^2\right| / \|u_{k+1}^h\|_0^2 \lesssim \|u_{k+1}^h - u\|_1^2.$$

Inserting this into (B.7) gives

$$|\lambda - \lambda_{k+1}^h| \lesssim \|u - u_{k+1}^h\|_1^2 \lesssim \left(h + |\lambda - \lambda_k^h| + \|u - u_k^h\|_0\right)^2 \lesssim h^2 + \left(|\lambda - \lambda_k^h| + \|u - u_k^h\|_0\right)^2,$$

which combined with (B.8) implies

$$|\lambda - \lambda_{k+1}^h| + \|u - u_{k+1}^h\|_0 \lesssim h^2 + \left(|\lambda - \lambda_k^h| + \|u - u_k^h\|_0\right)^2. \tag{B.9}$$

Let $e_k = |\lambda - \lambda_k^h| + \|u - u_k^h\|_0$. Then the estimate (B.9) can be expressed as

$$e_{k+1} \le \bar{c}_4 h^2 + c_4 e_k^2,$$

where c_4 and $\bar{c}_4$ are two positive generic constants. The above inequality can expressed as

$$c_4 e_{k+1} \le c_5 h^2 + (c_4 e_k)^2, \tag{B.10}$$

Journal Pre-proof

where $c_5 = c_4\bar{c}_4$.

On the other hand, by assumption and using error estimates of I_h, we know there exists a constant $\tilde{h}_1 > 0$ such that if $h < \tilde{h}_1$, there holds

$$e_0 = |\lambda - \lambda_0^h| + \|u - u_0^h\|_0 \le |\lambda - \lambda_0^h| + \|u - I_h u\|_0 + \|I_h u - u_0^h\|_0 \le c_6(\tilde{\delta} + h^2),$$

where $c_6 > 0$ is a generic constant. For clarity, we want to point out all the generic constants given above are independent of the finite element mesh size h.

Next, we choose two positive constants $\tilde{h}_0$ and $\tilde{\delta}_0$ such that

$$\tilde{h}_0 < \tilde{h}_1, \quad c_4c_6(\tilde{\delta_0} + \tilde{h_0}^2) < 1/2 \quad \text{and} \quad c_5\tilde{h_0}^2 < 1/4. \tag{B.11}$$

For the recursive estimate (B.10), we set $a_k = c_4e_k$ and $b = c_5h^2$. Then, if $h < \tilde{h}_0$ and $\tilde{\delta} < \tilde{\delta}_0$, the conditions (B.11) hold, or equivalently the assumptions in Lemma B.1 hold, so it follows from (B.1) that

$$c_4e_k \le (c_4e_0)^{2^k} + c_5\left(2 + \frac{1}{1 - 2c_4e_0}\right)h^2 = (c_4e_0)^{2^k} + c_5\left(3 + \frac{2c_4e_0}{1 - 2c_4e_0}\right)h^2,$$

which immediately yields

$$|\lambda - \lambda_k^h| + \|u - u_k^h\|_0 \lesssim \beta_3^{2^k} + h^2.$$

The proof is complete. □

Journal Pre-proof

Journal Pre-proof

Declaration of interests

☒ 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.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Journal Pre-proof