INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 21, Number 5, Pages 764–792

doi: 10.4208/ijnam2024-1031

CHOICE OF INTERIOR PENALTY COEFFICIENT FOR INTERIOR PENALTY DISCONTINUOUS GALERKIN METHOD FOR BIOT'S SYSTEM BY EMPLOYING MACHINE LEARNING

SANGHYUN LEE, TEERATORN KADEETHUM, AND HAMIDREZA M. NICK

Abstract. This paper uses neural networks and machine learning to study the optimal choice of the interior penalty parameter of the discontinuous Galerkin finite element methods for both the elliptic problems and Biot's systems. It is crucial to choose the optimal interior penalty parameter, which is not too small or too large for the stability, robustness, and efficiency of the approximated numerical solutions. Both linear regression and nonlinear artificial neural network methods are employed and compared using several numerical experiments to illustrate the capability of our proposed computational framework. This framework is integral to developing automated numerical simulation because it can automatically identify the optimal interior penalty parameter. Real-time feedback could also be implemented to update and improve model accuracy on the fly.

Key words. Discontinuous Galerkin, interior penalty, neural networks, machine learning, finite element methods.

1. Introduction

Discontinuous Galerkin (DG) finite element method, which is also known as the Interior Penalty method (IP), is one of the most popular non-conforming finite elements employed for various realistic applications, especially with discontinuous material properties [1, 2, 3, 4, 5]. The advantages of the IP-DG method include the following. First, DG preserves the local flux conservation with highly varying material properties [6, 7, 8, 9, 10]. In addition, DG can deal robustly with general partial differential equations and equations whose type changes within the computational domain, such as from advection dominated to diffusion dominated [11, 12, 13].

However, one of the disadvantages of DG is that the method's stability and accuracy depend on the interior penalty parameter in front of the jump term that needs to be chosen. In other words, the performance of the DG methods highly depends on the choice of the interior penalty parameter. For example, if the penalty parameter is too small, the stability of the scheme is not guaranteed, and the linear solver will not converge. If the penalty parameter is too large, DG schemes might converge to the continuous Galerkin finite element methods and often suffer from the linear solver. Thus, it is crucial to employ the optimal interior penalty parameter. Several studies of the lower bounds for the penalty parameter have been obtained in the past [14, 15, 16, 17, 18]. Moreover, weighted interior penalty parameters for the cases where the diffusion coefficient is discontinuous were studied in [19, 20]. Specific illustrations on the selection of the penalty parameters are shown in [21].

This paper proposes a new procedure to find the optimal interior penalty parameters for both elliptic problems and the poroelastic Biot's system. Since the choice of the optimal interior penalty parameters for multiphysics multiscale coupled problems or problems with discontinuous and heterogeneous material properties are

Received by the editors on April 17, 2023 and, accepted on January 14, 2024.
2000 *Mathematics Subject Classification.* 65N12, 65M60.

nontrivial by the traditional analytic approaches, we employ machine learning processes to predict the optimal interior penalty parameters. Many machine learning models have been a center of attention for decades because of their approximation power that could be practically applied to various applications [22, 23]. These algorithms range from classic linear regression models [24, 25], spatial interpolation techniques such as kriging [26] or maximum likelihood estimation [27], and nonlinear approximation functions such nonlinear regression [28] or deep learning [29].

Recently, deep learning has become more attractive with several advantages, including that it is scalable [30], suitable for GPU functionality [31], and requires less computational resources is less due to the mini-batch gradient descent approach [32]. Deep learning has also been successfully applied to solve partial differential equations, generally solved by classical numerical methods such as finite difference, finite volume, or finite element methods [33, 34, 35, 36, 37, 38]. Moreover, this technique has been used to assist the traditional numerical methods such as finite elements to enhance their performance [39, 40, 41, 42, 43]. Hence, this paper aims to apply this method to identify the optimal interior penalty parameters in complex problems.

The proposed procedure benefits the simple elliptic problem or Biot's equations and any multiphysics multiscale coupled problems. Besides, in cases where many simulations have to be performed with different settings, e.g., mesh size, material properties, or various interior penalty schemes, our proposed framework can automatically identify the optimal interior penalty parameter. Real-time feedback could also be implemented to update and improve model accuracy.

The paper is organized as follows. Our governing system and finite element discretizations are in Section 2 and Section 3, respectively. Details about the machine learning algorithm are discussed in Section 4. The numerical results are in Section 5; this section illustrates the effects of interior penalty parameters on both solution quality and simulation behavior. Performance between linear and nonlinear approximation functions is also compared. Finally, the conclusions follow in Section 6.

2. Mathematical Model

In this section, we briefly recapitulate the Biot system for poroelasticity that we will discuss in this paper. Let $\Omega \subset \mathbb{R}^d$ ($d \in \{1, 2, 3\}$) be the computational domain, which is bounded by the boundary, $\partial\Omega$. The time domain is denoted by $\mathbb{T} = (0, T]$ with $T > 0$. Then the coupling between the fluid flow and solid deformation can be captured by applying Biot's equation of poroelasticity, which is composed of linear momentum and mass balance equations [44].

First, the mass balance equation is given as [45]:

$$\rho\left(\phi c_f + \frac{\alpha - \phi}{K_s}\right)\frac{\partial}{\partial t}p + \rho\alpha\frac{\partial}{\partial t}\nabla\cdot\mathbf{u} - \nabla\cdot\boldsymbol{\kappa}(\nabla p - \rho\mathbf{g}) = g \text{ in } \Omega\times\mathbb{T}, \tag{1}$$

where $p(\cdot, t) : \Omega\times(0; T] \to \mathbb{R}$ is a scalar-valued fluid pressure, $\mathbf{u}(\cdot, t) : \Omega\times(0; T] \to \mathbb{R}^d$ is a vector-valued displacement, ρ is a fluid density, ϕ is an initial porosity, c_f is a fluid compressibility, $\mathbf{g}$ is a gravitational vector, g is a sink/source. Here, $\nabla\cdot\mathbf{u}$ term represents the volumetric deformation and $\boldsymbol{\kappa}$ is defined as:

$$\boldsymbol{\kappa} := \frac{\rho\mathbf{k}_m}{\mu}, \tag{2}$$

where $\mathbf{k}_m$ is a matrix permeability tensor and μ is a fluid viscosity.

The following boundary and initial conditions supplement the mass balance equation (the fluid flow problem):

$$p = p_D \text{ on } \partial\Omega_p \times \mathbb{T}, \tag{3}$$

$$-\nabla \cdot \boldsymbol{\kappa}(\nabla p - \rho \mathbf{g}) \cdot \mathbf{n} = q_D \text{ on } \partial\Omega_q \times \mathbb{T}, \tag{4}$$

$$p = p_0 \text{ in } \Omega \text{ at } t = 0, \tag{5}$$

where p_D and q_D are specified pressure and flux, respectively, and $\partial\Omega$ is decomposed to pressure and flux boundaries, $\partial\Omega_p$ and $\partial\Omega_q$, respectively.

Secondly, the linear momentum balance equation can be written as follows:

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, p) = \mathbf{f}. \tag{6}$$

For simplicity, a body force $\mathbf{f}$ is neglected in this study. Here, $\boldsymbol{\sigma}$ is total stress, which is defined as:

$$\boldsymbol{\sigma} := \boldsymbol{\sigma}(\mathbf{u}, p) = \boldsymbol{\sigma}'(\mathbf{u}) - \alpha p \mathbf{I}, \tag{7}$$

where $\mathbf{I}$ is the identity tensor and α is Biot's coefficient defined as [46]:

$$\alpha := 1 - \frac{K}{K_s}, \tag{8}$$

with the bulk modulus of a rock matrix K and the solid grains modulus K_s. In addition, $\boldsymbol{\sigma}'$ is an effective stress written as:

$$\boldsymbol{\sigma}' := \boldsymbol{\sigma}'(\mathbf{u}) = 2\mu_l \boldsymbol{\epsilon}(\mathbf{u}) - \lambda_l \nabla \cdot \mathbf{u}\mathbf{I}, \tag{9}$$

where λ_l and μ_l are Lamé constants. By assuming a small displacement, a strain is defined as:

$$\boldsymbol{\epsilon}(\mathbf{u}) := \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right). \tag{10}$$

Thus, we can write the linear momentum balance supplemented by its boundary and initial conditions as:

$$\nabla \cdot \boldsymbol{\sigma}'(\mathbf{u}) + \alpha \nabla \cdot p\mathbf{I} = \mathbf{f} \quad \text{in } \boldsymbol{\Omega} \times \mathbb{T}, \tag{11}$$

$$\mathbf{u} = \mathbf{u}_D \text{ on } \partial\Omega_u \times \mathbb{T}, \tag{12}$$

$$\boldsymbol{\sigma}' \cdot \mathbf{n} = \boldsymbol{\sigma}_D \text{ on } \partial\Omega_t \times \mathbb{T}, \tag{13}$$

$$\mathbf{u} = \mathbf{u}_0 \text{ in } \Omega \text{ at } t = 0, \tag{14}$$

where $\mathbf{u}_D$ and $\boldsymbol{\sigma}_D$ are prescribed displacement and traction at boundaries, respectively, and t is time. Here, $\partial\Omega$ can be decomposed to displacement and traction boundaries, $\partial\Omega_u$ and $\partial\Omega_t$, respectively, for the solid deformation problem.

3. Numerical Discretizations

This paper employs the discontinuous Galerkin (DG) finite element method for spatial discretization. Let $\mathcal{T}_h$ be the shape-regular (in the sense of Ciarlet) triangulation by a family of partitions of Ω into d-simplices T (triangles/squares in $d = 2$ or tetrahedra/cubes in $d = 3$). We denote by h_T the diameter of T and we set $h = \max_{T \in \mathcal{T}_h} h_T$. Also, we denote by $\mathcal{E}_h$ the set of all edges and by $\mathcal{E}_h^I$ and $\mathcal{E}_h^\partial$ the collection of all interior and boundary edges, respectively. In the following notation, we assume edges for two dimensions, but the results hold analogously for faces in a three-dimensional case. The space $H^s(\mathcal{T}_h)$ ($s \in \mathbb{R}$) is the set of element-wise H^s functions on $\mathcal{T}_h$, and $L^2(\mathcal{E}_h)$ refers to the set of functions whose traces on the elements of $\mathcal{E}_h$ are square integrable. Let $\mathbb{Q}_l(T)$ denote the space of polynomials of partial degree at most l. Throughout the paper, we use the standard notation for

Sobolev spaces and their norms. For example, let $E \subseteq \Omega$, then $\|\cdot\|_{1,E}$ and $|\cdot|_{1,E}$ denote the $H^1(E)$ norm and seminorm, respectively. For simplicity, we eliminate the subscripts on the norms if $E = \Omega$.

Since we consider the nonconforming DG methods, let

$$e = \partial T^+ \cap \partial T^-, \quad e \in \mathcal{E}_h^I,$$

where T^+ and T^- be two neighboring elements and we denote by h_e the length of the edge e. Let $\mathbf{n}^+$ and $\mathbf{n}^-$ be the outward normal unit vectors to ∂T^+ and ∂T^-, respectively ($\mathbf{n}^\pm := \mathbf{n}_{|T^\pm}$). For any given function ξ and vector function $\boldsymbol{\xi}$, defined on the triangulation $\mathcal{T}_h$, we denote $\xi^\pm$ and $\boldsymbol{\xi}^\pm$ by the restrictions of ξ and $\boldsymbol{\xi}$ to $T^\pm$, respectively.

Next, we define the weighted average operator $\{\cdot\}_{\delta_e}$ as follows: for $\zeta \in L^2(\mathcal{T}_h)$ and $\boldsymbol{\tau} \in L^2(\mathcal{T}_h)^d$,

$$\{\zeta\}_{\delta e} = \delta_e \zeta^+ + (1-\delta_e)\,\zeta^-, \quad \text{and} \quad \{\boldsymbol{\tau}\}_{\delta e} = \delta_e \boldsymbol{\tau}^+ + (1-\delta_e)\,\boldsymbol{\tau}^-, \quad \text{on } e \in \mathcal{E}_h^I, \tag{15}$$

where δ_e is calculated by [47, 19].

$$\delta_e := \frac{\kappa_e^-}{\kappa_e^+ + \kappa_e^-}. \tag{16}$$

Here,

$$\kappa_e^+ := \left(\mathbf{n}^+\right)^T \cdot \boldsymbol{\kappa}^+ \cdot \mathbf{n}^+, \quad \text{and } \kappa_e^- := \left(\mathbf{n}^-\right)^T \cdot \boldsymbol{\kappa}^- \cdot \mathbf{n}^-, \tag{17}$$

where κ_e is a harmonic average of κ_e^+ and κ_e^- read as:

$$\kappa_e := \frac{2\kappa_e^+ \kappa_e^-}{\left(\kappa_e^+ + \kappa_e^-\right)}. \tag{18}$$

On the other hand, for $e \in \mathcal{E}_h^\partial$, we set $\{\zeta\}_{\delta_e} := \zeta$ and $\{\boldsymbol{\tau}\}_{\delta_e} := \boldsymbol{\tau}$. The jump across the interior edge will be defined as

$$[\![\zeta]\!] = \zeta^+ \mathbf{n}^+ + \zeta^- \mathbf{n}^- \quad \text{and} \quad [\![\boldsymbol{\tau}]\!] = \boldsymbol{\tau}^+ \cdot \mathbf{n}^+ + \boldsymbol{\tau}^- \cdot \mathbf{n}^- \quad \text{on } e \in \mathcal{E}_h^I.$$

For $e \in \mathcal{E}_h^\partial$, we let $[\![\zeta]\!] := \zeta \mathbf{n}$ and $[\![\boldsymbol{\tau}]\!] := \boldsymbol{\tau} \cdot \mathbf{n}$.

Finally, we introduce the finite element space for the discontinuous Galerkin method, which is the space of piecewise discontinuous polynomials of degree k by

$$V_{h,k}^{\mathsf{DG}}(\mathcal{T}_h) := \left\{\psi \in L^2(\Omega) |\ \psi_{|_T} \in \mathbb{Q}_k(T),\ \forall T \in \mathcal{T}_h\right\}. \tag{19}$$

Moreover, we use the notation:

$$(v,w)_{\mathcal{T}_h} := \sum_{T \in \mathcal{T}_h} \int_T v\, w dx, \quad \forall\, v, w \in L^2(\mathcal{T}_h),$$

$$\langle v,w\rangle_{\mathcal{E}_h} := \sum_{e \in \mathcal{E}_h} \int_e v\, w\, d\gamma, \quad \forall\, v, w \in L^2(\mathcal{E}_h).$$

3.1. Pressure problem. First, we introduce the backward Euler DG approximation to (1). We define a partition of the time interval $0 =: t^0 < t^1 < \cdots < t^N := \mathbb{T}$ and denote the uniform time step size by $\delta t := t^n - t^{n-1}$. The DG finite element space approximation of the pressure $p(\mathbf{x}, t)$ is denoted by $P(\mathbf{x}, t) \in V_{h,k}^{\mathsf{DG}}$. Let $P^n := P(\mathbf{x}, t^n)$ for $0 \le n \le N$. We set a given initial condition for the pressure as P^0 and assume the displacement at time t, $\mathbf{u}(\cdot, t)$ is given. For simplicity, the terms gravity and source/sink are neglected. Then, the time-stepping algorithm reads as follows: Given P^{n-1},

$$\text{Find } P^n \in V_{h,k}^{\mathsf{DG}} \text{ such that } \mathcal{S}_\theta(P^n, w; \mathbf{u}) = \mathcal{F}_\theta(w), \quad \forall\, w \in V_{h,k}^{\mathsf{DG}}, \tag{20}$$

where $\mathcal{S}_\theta$ and $\mathcal{F}_\theta$ are the bilinear form and linear functional as defined by

(21)
$$\begin{aligned}\mathcal{S}_\theta(v,w;\mathbf{u}) :=& \frac{\rho}{\delta t}\left(\phi c_f + \frac{\alpha-\phi}{K_s}\right)(v,w)_{\mathcal{T}_h} + (\boldsymbol{\kappa}\nabla v, \nabla w)_{\mathcal{T}_h} \\ & - \left\langle \{\boldsymbol{\kappa}\nabla v\}_{\delta_e}, [\![w]\!]\right\rangle_{\mathcal{E}_h^1} + \rho\alpha(\frac{\partial}{\partial t}\nabla\cdot\mathbf{u}, w)_{\mathcal{T}_h} - \theta\left\langle [\![v]\!], \{\boldsymbol{\kappa}\nabla w\}_{\delta_e}\right\rangle_{\mathcal{E}_h^1} \\ & + \beta(k)\langle h_e^{-1}\kappa_e [\![v]\!], [\![w]\!]\rangle_{\mathcal{E}_h^1}, \quad \forall v, w \in V_{h,k}^{\mathsf{DG}},\end{aligned}$$

and

$$\begin{aligned}\mathcal{F}_\theta(w) :=& \frac{1}{\delta t}(P^{n-1}, w)_{\mathcal{T}_h} - \langle q_D, [\![w]\!]\rangle_{\mathcal{E}_h^{N,\partial}} - \theta\left\langle p_D, \{\boldsymbol{\kappa}\nabla w\}_{\delta_e}\right\rangle_{\mathcal{E}_h^{D,\partial}} \\ & + \beta(k)\langle h_e^{-1}\kappa_e p_D, [\![w]\!]\rangle_{\mathcal{E}_h^{D,\partial}}, \quad \forall w \in V_{h,k}^{\mathsf{DG}}.\end{aligned} \tag{22}$$

The choice of θ leads to different DG algorithms. For example, i) $\theta = 1$ for SIPG(β)$-k$ methods [48, 49], which later has been extended to the advection-diffusion problems in [50, 51], ii) $\theta = -1$ for NIPG(β)$-k$ methods [52], and iii) $\theta = 0$ for IIPG(β)$-k$ method [53].

The interior penalty parameter, $\beta(k)$, is a function of polynomial degree approximation, k. Here, h_e is a characteristic length of the edge $e \in \mathcal{E}_h$ calculated as:

$$h_e := \frac{\text{meas}\,(T^+) + \text{meas}\,(T^-)}{2\,\text{meas}(e)}, \tag{23}$$

where meas(.) represents a measurement operator, measuring length, area, or volume. Several analyses for the choice of the interior penalty parameter, β, are shown in [14, 15, 16, 17, 18] and this β is the quantity that we investigate in this paper. The study could be applicable not only for DG but also for other interior penalty methods, such as enriched Galerkin methods [54, 55, 56, 57].

3.2. Displacement problem. For the displacement $\mathbf{u}$, we employ the classical continuous Galerkin (CG)finite element methods for the spatial discretizations as in [55, 58] where the function space is defined as

$$W_{h,k}^{\mathsf{CG}}(\mathcal{T}_h) := \left\{\boldsymbol{\psi}_u \in \mathbb{C}^0(\Omega;\mathbb{R}^d) : \boldsymbol{\psi}_u|_T \in \mathbb{Q}_k(T;\mathbb{R}^d), \forall T \in \mathcal{T}_h\right\}, \tag{24}$$

where $\mathbb{C}^0(\Omega;\mathbb{R}^d)$ denotes the space of vector-valued piecewise continuous polynomials, $\mathbb{Q}_k(T;\mathbb{R}^d)$ is the space of polynomials of degree at most k over each element T.

The CG finite element space approximation of the displacement $\mathbf{u}(\mathbf{x},t)$ is denoted by $\mathbf{U}(\mathbf{x},t) \in W_{h,k}^{\mathsf{CG}}$. Let $\mathbf{U}^n := \mathbf{U}(\mathbf{x},t^n)$ for $0 \le n \le N$. We set a given initial condition for the displacement as $\mathbf{U}^0$, and the pressure at time t, P^n is given from the previous section. Then, the time-stepping algorithm reads as follows: Given P^n,

$$\text{Find } \mathbf{U}^n \in W_{h,k}^{\mathsf{CG}} \text{ such that } \mathcal{A}(\mathbf{U}^n, \mathbf{w}; P^n) = \mathcal{D}(\mathbf{w}), \quad \forall\, \mathbf{w} \in W_{h,k}^{\mathsf{CG}}, \tag{25}$$

where $\mathcal{A}$ and $\mathcal{D}$ are the bilinear form and linear functional as defined as

(26)
$$\mathcal{A}(\mathbf{v},\mathbf{w};P^n) := \sum_{T\in\mathcal{T}_h}\int_T \boldsymbol{\sigma}'(\mathbf{v}) : \epsilon(\mathbf{w})\, dV + \sum_{T\in\mathcal{T}_h}\int_T \alpha\nabla P^n \nabla\mathbf{w}\, dV, \quad \forall\, \mathbf{v},\mathbf{w} \in W_{h,k}^{\mathsf{CG}},$$

and

$$\mathcal{D}(\mathbf{w}) := \sum_{T\in\mathcal{T}_h}\int_T \mathbf{f}\mathbf{w}\,dV + \sum_{e\in\mathcal{E}_h^N}\int_e \boldsymbol{\sigma}_D\mathbf{w}\,dS, \quad \forall\,\mathbf{w}\in W_{h,k}^{\mathsf{CG}}. \tag{27}$$

3.3. Poroelasticity problem. Finally, the resulting variational formulation for solving the Biot's poroelasticity system reads as follows: Given $(\mathbf{U}^{n-1}, P^{n-1}) \in W_{h,k}^{\mathsf{CG}} \times V_{h,k}^{\mathsf{DG}}$ with $0 \le n \le N-1$, find $(\mathbf{U}^n, P^n) \in W_{h,k}^{\mathsf{CG}} \times V_{h,k}^{\mathsf{DG}}$ such that

$$\mathcal{S}_\theta(P^n, w; \mathbf{U}^n) = \mathcal{F}_\theta(w), \quad \forall\, w \in V_{h,k}^{\mathsf{DG}},$$
$$\mathcal{A}(\mathbf{U}^n, \mathbf{w}; P^n) = \mathcal{D}(\mathbf{w}), \quad \forall\,\mathbf{w}\in W_{h,k}^{\mathsf{CG}}.$$

Numerical analyses for the existence and uniqueness of the system and extensions to consider different applications by utilizing the presented mix of continuous and discontinuous Galerkin methods are discussed in [59, 60, 61, 62] and references cited therein.

4. Machine Learning Algorithm

In this section, we present the details of the two machine learning algorithms employed in this paper to seek the effect and optimal choice of the interior penalty parameter. First, the linear approximation algorithm, which is so-called linear regression or logistic regression, is shown depending on the output type. Then, the nonlinear approximation algorithm, the artificial neural network (ANN) with a deep learning algorithm, is described. See Figure 1 for the detailed outline. Then, the next section compares the performance between linear and nonlinear approximation algorithms for each given problem to find the optimal penalty parameters. The two

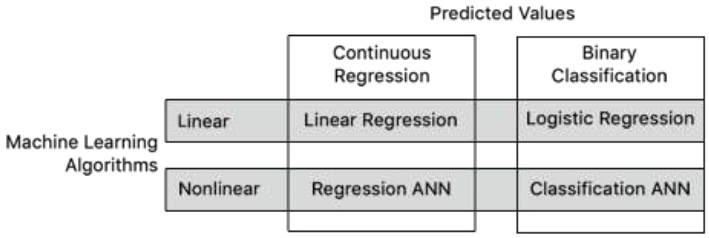

FIGURE 1. Illustration of the methods used in this study.

different algorithms (linear/nonlinear) will provide two types of predicted values; one is a continuous predicted value referred to as a continuous regression model, and the second is a binary predicted value referred to as a binary classification model. Finding the optimal penalty parameter could be solved by continuous (regression model) and binary (classification model) predictions.

First, we discuss the loss functions for each predicted value, continuous regression, and binary classification models. For the predicted value by employing the continuous regression model, we use mean squared error (MSE) as a loss function, which is defined as

$$\text{MSE} := \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2, \tag{28}$$

where n represents a number of data points, Y_i is a true or observed values at index i, and $\hat{Y}_i$ is a predicted value at index i.

To compare the performance of the linear regression and nonlinear ANN algorithms, by using the continuous regression predicted values, we define R^2 and the explained variance score (EVS). Here, R^2 is

$$R^2 := 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}, \tag{29}$$

where SS_{res} is a residual sum of squares read as:

$$SS_{\text{res}} := \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2, \tag{30}$$

and SS_{tot} is a total sum of squares defined as:

$$SS_{\text{tot}} := \sum_{i=1}^{n} \left(Y_i - \bar{Y}\right)^2, \tag{31}$$

where $\bar{(\cdot)}$ is an arithmetic average operator and $\bar{Y}$ is the arithmetic average of Y_i. Next, EVS is defined as

$$\text{EVS} := 1 - \frac{\text{Var}\{Y - \hat{Y}\}}{\text{Var}\{Y\}}, \tag{32}$$

where $\text{Var}\{\cdot\}$ is a variance operator. Note that if $\overline{Y - \hat{Y}} = 0$, then $R^2 = \text{EVS}$ or we have unbiased estimator [63].

For the predicted value by employing the binary classification models, we use binary cross entropy (BCE) as the loss function, which is defined as follows:

$$\text{BCE} := -\frac{1}{n} \sum_{i=1}^{n} \left(Y_i \cdot \log\left(\mathbb{P}\left(Y_i\right)\right) + \left(1 - Y_i\right) \cdot \log\left(1 - \mathbb{P}\left(Y_i\right)\right)\right), \tag{33}$$

where $\mathbb{P}\left(\cdot\right)$ is a probability function. Then, we use the accuracy function (ACC) to compare the results from logistic regression and classification ANN, and it is defined as

$$\text{ACC} := \frac{\sum \text{ True positive } + \sum \text{ True negative}}{n}, \tag{34}$$

where 'True positive' and 'True negative' represent cases where the prediction agrees with the observation. See Figure 2 for more details.

Moreover, we employ different optimization algorithms to minimize each loss function for linear and nonlinear algorithms, and we describe these in the following sections.

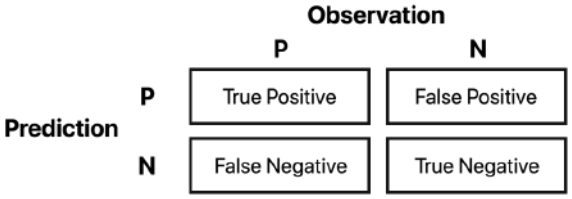

FIGURE 2. Illustration of a confusion matrix, where **P** denotes positive and **N** denotes negative.

4.1. Linear approximation algorithm. Two types of linear approximation algorithms are used in this work: (i) (multivariate) linear regression and (ii) (multivariate) logistic regression. These two models produce continuous and binary predictive values, respectively, and can consider any number of input values. The main idea of these models is to map the linear relationship between multiple independent variables (input) and one dependent variable (output). As discussed previously, equations (28) and (33) are used as the loss function for multivariate linear and logistic regressions, respectively. To minimize these functions, we follow the classical stochastic gradient descent (SGD) [64] solver to minimize equation (28), and limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [65] solver to minimize equation (33).

Finally, we split our available data points into two parts: training set and test set using the Scikit-learn package, an open-source software machine learning library for the Python programming language [64]. We use the same training set to train both linear and nonlinear approximation functions. The test set compares performances between the linear and nonlinear approximation algorithms. The splitting ratio of the data sets used in this paper is 0.8 of the total available data for the training set and 0.1 of the total available data for the test set. We note that we only utilize 90% of the available data set for training and test set to be consistent with the number of the data sets for the nonlinear algorithm. The nonlinear algorithm requires 10% of the available data set for the validation set. Each variable input is transformed into a numeric variable, and each continuous data is normalized by its mean and variance using the prepossessing library of the Scikit-learn package [64].

4.2. Nonlinear approximation algorithm. Similar to the previous section, we have two types of nonlinear approximation algorithms used in this work: (i) regression ANN model and (ii) classification ANN model. These models map the nonlinear relationship between input (features) to output by using nonlinear activation functions such as Sigmoid, Tanh, or rectified linear unit (ReLU) functions [66, 67, 29]. The number of hidden layers also plays an important role in defining whether the neural network has deep (number of hidden layers is greater than one) or shallow (number of hidden layers is one) architecture. [68]. Moreover, the shallow and deep neural networks, called Wide and Deep Learning, can be combined to optimize the performance and generalization [69]. Figure 3 presents the neural network architecture used in this study. The number of output nodes is always one, but the number of input nodes is determined by the nature of each problem, which will be discussed later. Hyperparameters [29] are determined by the number of hidden layers (N_{hl}) and the number of neurons (N_n).

The artificial neural network used in this study is built on the TensorFlow platform with Keras wrapper [70, 71]. The ReLU is employed as the activation function for each neuron in each hidden layer for both regression and classification ANN. The output layer of the classification ANN is subjected to the Sigmoid activation function, while the output layer of the regression ANN is not subjected to any activation functions since the output values are continuous.

To minimize the loss functions, equations (28) (for the regression ANN model) and (33) (for the classification ANN model), the mini-batch gradient descent method is used with a batch size of 10 [32, 72]. This method is effective since (i) it requires less memory and becomes more effective when the size of data is significant, and (ii) it helps to prevent gradient-decent optimizations trapped in the local minimum [73, 74]. Then, adaptive moment estimation (ADAM) [75] solver is employed.

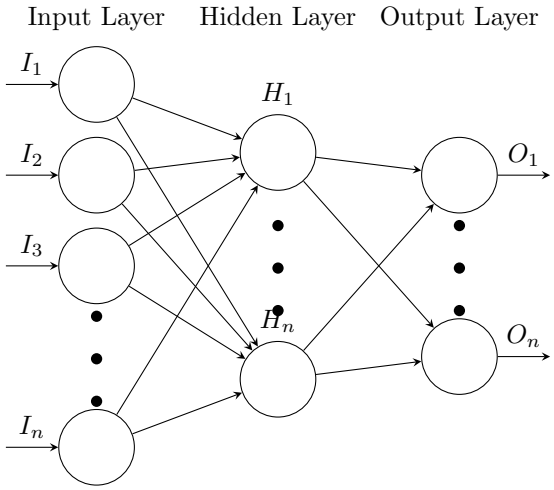

FIGURE 3. The general neural network architecture used in this study. The input layer contains up to i input nodes, and the output layer is composed of $1, ..., k$ output nodes. The number of hidden layers we denote N_{hl}, and each hidden layer is composed of N_n neurons.

Similar to the linear approximation algorithms, we split our available data points into three parts (i) training set, (ii) validation set, and (iii) test set using Scikit-learn package [64]. We note that the same training and test set data points are applied for both linear and nonlinear algorithms, but the validation set is only for the nonlinear algorithms. The validation set is used to tune hyperparameters. The splitting ratio used in this paper is 0.8 of the total available data for the training set, 0.1 of the total available data for the validation set, and 0.1 of the total available data for the test set. Some studies suggest that it may be possible to use only training and test sets and neglect the validation set when we have limited data [76]. In that case, hyperparameters are tuned by the test set, where the test set is also used to compare the performance. We do not prefer this case since the test set should be kept separated for the last process, and the ANN algorithms should have no prior knowledge before being tested [77, 78].

5. Numerical results

In this section, we present several numerical experiments to illustrate the capability of our proposed algorithm and computational framework. All the numerical experiments are computed by the program built by employing Scikit [64] for linear algorithms and TensorFlow with Keras wrapper [70, 71] for nonlinear algorithms. Moreover, the presented results are also computed by the JMP platform (SAS) [24] to verify our results. The continuous regression for the predicted values is mainly used for Section 5.1 to find the optimal penalty parameter for the elliptic problem in each case. Thus, the performance of the linear and nonlinear regression ANN are compared in this section. For Section 5.2, the binary classification for the predicted values is used to find the optimal penalty parameter for Biot's equation in each case. Here, the performance of the linear logistic regression and nonlinear classification ANN methods are compared.

5.1. Effect and optimal choice of interior penalty parameter for elliptic equations. First, we study the effect and optimal choice of the interior penalty parameter β by considering the simplified elliptic equation of the flow problem (1). We obtain the following simplified elliptic equation by assuming that the pressure does not depend on the time and $\mathbf{u}$ ($\alpha = 0$).

$$-\nabla \cdot (\boldsymbol{\kappa} \nabla p) = g \text{ in } \Omega. \tag{35}$$

We note that gravity ($\mathbf{g}$) is neglected for simplicity, and g is the source/sink term. In each of the following problems, we compare the performance of the linear regression and nonlinear regression ANN, where the predicted values are continuous.

5.1.1. The effect of a polynomial degree approximation (k). Before we employ the machine learning algorithm presented in Section 4, we investigate the effect of the polynomial degree approximation on the penalty parameter. We illustrate the effect of a polynomial degree approximation on the choice of optimal β using different linear solvers (direct or iterative solvers) and discretization schemes. Here, different discretization schemes indicate the options for choosing IIPG ($\theta = 0$) or SIPG ($\theta = 1$).

For this case, we set the exact solution in $\Omega = [0, 1]^2$ as

$$p(x, y) := \sin(x + y), \tag{36}$$

and κ ($\boldsymbol{\kappa} := \kappa \boldsymbol{I}$) has different values in a range of $[1.0 \times 10^{-18}, 1.0]$. Furthermore, homogeneous boundary conditions are applied to all boundaries. In particular, we study the five different k values (1, 2, 3, 4, and 5), and each case is tested by a different combination of linear solvers and θ. The detailed algorithm is presented in Algorithm 1.

Algorithm 1: Investigation procedure for the elliptic problem

```
Initialize the data set of κ that used in the investigation
for i < n_κ, where n_κ is the size of the specified data set κ, do
  Assign κ := κ[i] I, β := β_0, where β_0 = 100.00, and h := h_0, where
  h_0 = 6.25 × 10^−2
  while error convergence rate is optimal do
    Update β := 0.99 × β {Except the first loop}
    for j < n_h, where n_h = 6 do
      Solve (35) and compute the error
      Calculate error convergence rate
      Update h := 0.5 × h
    end for
  end while
  return β, this β is the smallest, which the optimal error convergence rate
  can be observed.
end for
```

The main idea of this algorithm is that we reduce the β values (1% by each test) until the optimal error convergence rate is no longer guaranteed. Thus, in other words, we focus on finding the smallest β that ensures the optimal convergence rate. Here, the optimal convergence rate is obtained by six computation cycles on uniform triangular meshes, where the mesh size h is divided by two for each cycle. The behavior of the $H^1(\Omega)$ semi-norm errors for the approximated solution versus the mesh size h is checked.

The results presented in Table 1 show that the lowest (optimal) β values for SIPG, ($\theta = 1$) for each case, are higher than those for IIPG ($\theta = 0$). Besides, the smallest β values increase as k increases. However, the choice of linear solver, either direct or iterative solver, did not influence the results. These computations are implemented by using FEniCS [79], and the direct solver used in this problem is the lower-upper decomposition (LU). In contrast, the conjugate gradient (CG) method with the algebraic multigrid methods (AMG) method preconditioner [80] are employed for an iterative scheme.

TABLE 1. The lowest β value that provides the optimal error convergence rate solution with different k, θ, and linear solver.

k	SIPG ($\theta = 1$)		IIPG ($\theta = 0$)	
	direct solver	iterative solver	direct solver	iterative solver
1	1.11	1.11	0.83	0.83
2	2.80	2.80	2.74	2.74
3	5.79	5.79	5.68	5.68
4	9.99	9.99	9.79	9.79
5	14.97	14.97	14.67	14.67

5.1.2. Effect of interior penalty parameter for linear solvers and optimal choice by employing machine learning algorithms. However, it is observed that the choice of β values significantly impacts the number of iterations for the linear solver. Figures 4 and 5 illustrate the number of iterations of the linear solver for SIPG and IIPG, respectively. In the beginning, the number of iterations decreases when β decreases, but when β approaches zero, the number of iterations increases dramatically. Subsequently, the solver becomes unstable and doesn't converge to the solution.

Thus, we confirmed that the linear solvers' choice of β is essential. To be precise, if β is too large, the iteration number is high, but also, the too-small value of β can cause a high number of iterations and, more importantly, non-convergence.

To find the optimal β for the iterative solver, we need to consider β, which requires a minimum of the linear solver iteration, provides stable solutions, and ensures optimal error convergence rate. Hence, we first identify the parameters that impact the number of iterations (dependent variables) by employing the chi-squared test [64]. Table 2 illustrates the test results, and p-values for each variable are presented. We note that θ, β, h, and k have a p-value of less than 0.025; therefore, we include these variables as independent variables for further predictive model development. The κ, however, does not affect the results since κ is included in a coefficient of the penalty term as shown in (21).

Subsequently, from the results in Table 2, we employ the linear and nonlinear machine learning algorithms that were presented in Section 4 to find the optimal choice of β, which ensures both the minimum iteration number for the linear solver and optimal convergence rate (stability). To elaborate, we want to find a range of β that could guarantee stability, see Table 1, while utilizing the minimum number of iterations. Figures 4 and 5 present the number of iterations with different β within various range of κ. The number of iterations highly depends on the linear solver with different κ values in the lower-order cases.

In this problem, we have a total of 182,881 data sets (all the values we plot on Figures 4 and 5). As discussed in both sections 4.1 and 4.2, the data sets are

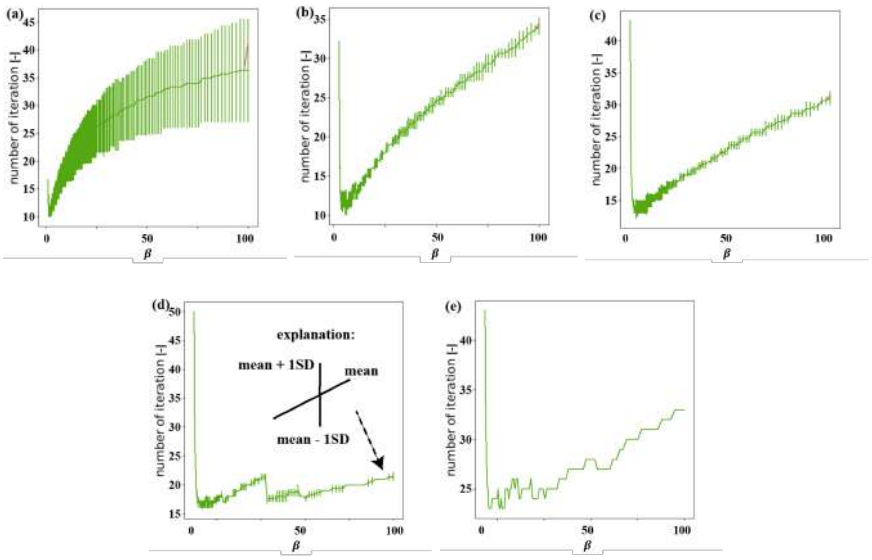

FIGURE 4. Number of linear iterative solver of SIPG for (**a**) $k = 1$, (**b**) $k = 2$, (**c**) $k = 3$, (**d**) $k = 4$, and (**e**) $k = 5$. Note that each line represents a different value of κ, and the error bar shows the mean and standard deviation ($\pm$ 1 SD) of each number of iterations; see (**d**) for an explanation. We observe that each β variability arises because we conducted runs with various values of κ for each β. The linear solver significantly influences the number of iterations, especially in lower k cases, where different κ values play a crucial role. However, the impact of κ becomes less prominent as k increases.

TABLE 2. p-value results for each explanatory variable for the elliptic equation.

Variable	p-value
θ	≈ 0.00
κ	≈ 1.00
β	≈ 0.00
h	≈ 0.00
k	≈ 0.00

split by training, validation, and test sets using the splitting ratio $[0.8, 0.1, 0.1]$. Thus, the number of training sets, validation sets, and test sets are $0.8 \times 182,881$, $0.1 \times 182,881$, and $0.1 \times 1828,81$, respectively. We use the training set to train the linear and nonlinear machine learning algorithms. The validation set is for tuning the hyperparameters for the nonlinear ANN models, and the test set is for comparing performances between the linear and nonlinear algorithms.

First, we begin with the linear algorithm by building the multi-variable regression [64] as follows:

$$\text{number of iteration} = \gamma_0 + \gamma_1 \times \theta + \gamma_2 \times \beta + \gamma_3 \times h + \gamma_4 \times k, \tag{37}$$

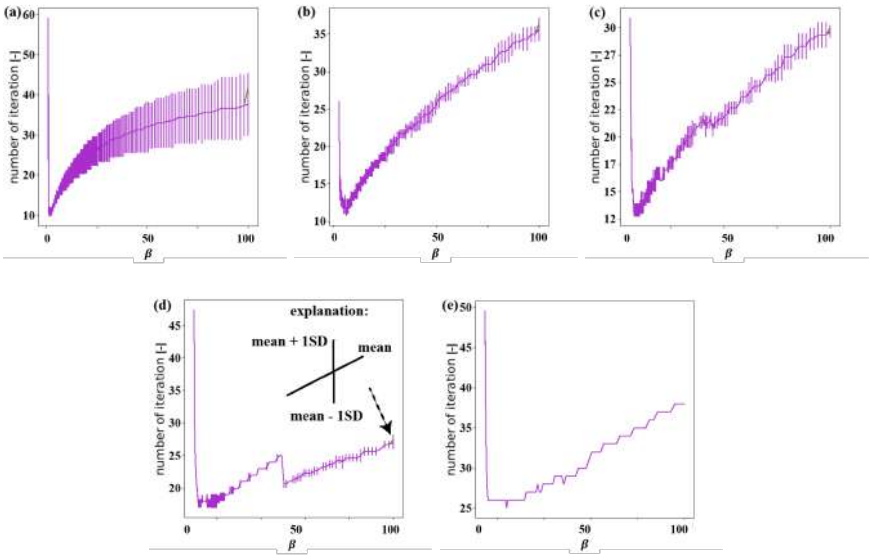

FIGURE 5. Number of linear iterative solver of IIPG for (**a**) $k = 1$, (**b**) $k = 2$, (**c**) $k = 3$, (**d**) $k = 4$, and (**e**) $k = 5$. Note that each line represents a different value of κ, and the error bar shows the mean and standard deviation ($\pm$ 1 SD) of each number of iterations; see (**d**) for an explanation. We observe that each β variability arises because we conducted runs with various values of κ for each β. The linear solver significantly influences the number of iterations, especially in lower k cases, where different κ values play a crucial role. However, the impact of κ becomes less prominent as k increases.

where $\gamma_0 = 16.93$, $\gamma_1 = -1.11$, $\gamma_2 = 0.21$, $\gamma_3 = -9.85$, and $\gamma_4 = 0.02$. These parameters provide the minimum value ($\leq 1 \times 10^{-4}$) of MSE value (28). Then, we obtain the r^2 and explained variance score (EVS) as

$$r^2 = 0.60 \quad \text{and} \quad \text{EVS} = 0.60, \tag{38}$$

by (29) and (32) as expalined in section 4.

Secondly, to compare the above linear algorithm with the nonlinear ANN algorithm, we construct the nonlinear ANN algorithm by using four inputs (θ, β, h, and k and one output (number of iteration) as presented in Figure 6. For simplicity, we assume each hidden layer has the same number of neurons, and the Rectified Linear Unit (ReLU) is used as an activation function for each hidden layer neuron. ADAM [75] is used to minimize the loss function, which is MSE (28) in this case.

Table 3 illustrates that the MSE of the validation set is generally decreased as N_{hl} and N_n are increased. Since we observe that the neural network performance is not significantly improved when $N_{hl} > 2$ and $N_n > 80$, which shows the sign of overfitting, we choose $N_{hl} = 2$ and $N_n = 80$ for the test set. Then, the final results for the nonlinear ANN algorithm give

$$r^2 = 0.98 \quad \text{and} \quad \text{EVS} = 0.98. \tag{39}$$

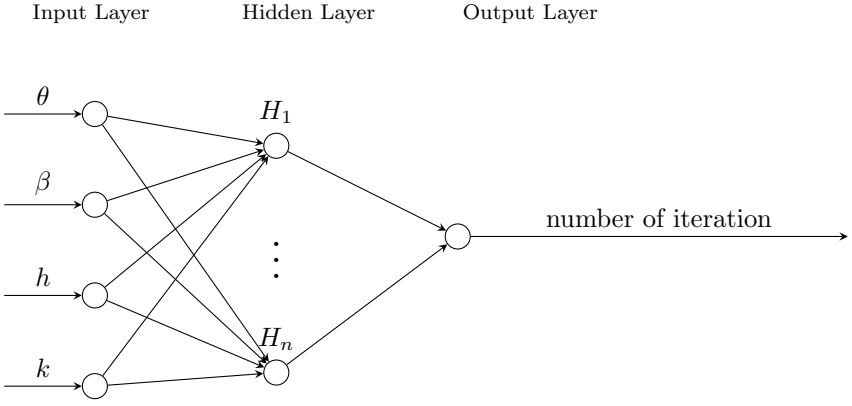

FIGURE 6. Neural network architecture used for the elliptic problem with the exact continuous solution. The number of hidden layers, N_{hl}, and the number of neurons for each hidden layer, N_n, are used as the sensitivity analysis parameters. H_1 and H_n represent the numbering of each neuron in each hidden layer.

TABLE 3. Elliptic equation with the exact continuous solution: Mean squared error (MSE) values of the validation set for different number of hidden layers N_{hl} and different number of neurons per layer N_n.

N_{hl} \ N_n	10	20	40	80
2	4.27	1.41	1.20	0.95
4	2.21	1.43	2.13	1.58
8	3.60	1.74	1.60	0.98

By comparing (38) and (39), we note that the results from the nonlinear ANN (39) illustrate the significant improvement of the prediction performance as the r^2 and EVS are improved from the linear algorithm (multi-variable linear regression) (38) significantly.

5.1.3. The effect of the continuity of the solutions and a heterogeneous coefficient. Next, we investigate the effect on the choice of optimal β by the continuity of the solutions, heterogeneity of $\boldsymbol{\kappa}$, and θ values. In most realistic scenarios, these material parameters are discontinuous, and the interior penalty scheme will provide an accurate solution with locally conservative flux [81, 54].

For the continuous solution, we take the same exact solution (36) as used in section 5.1.1. However, in this example, we choose the heterogeneous coefficient by setting:

$$\boldsymbol{\kappa} := \kappa \sin(x+y)\boldsymbol{I}. \tag{40}$$

Next, for the discontinuous solution, we set the exact solution in $\Omega = [0,1]^1$ as:

$$p = \begin{cases} 2.0x\dfrac{\kappa_1}{\kappa_0+\kappa_1} & \text{if } x \le 0.5, \\ \dfrac{(2x-1)\,\kappa_0+\kappa_1}{\kappa_0+\kappa_1} & \text{if } x > 0.5, \end{cases} \tag{41}$$

where κ_1 and κ_2 represent multiplied coefficients for the $0 \le x \le 0.5$ and $1.0 \ge x > 0.5$ subdomains, respectively. Here, $\kappa_1 \neq \kappa_2$, and κ_1 and κ_2 have the same range of $[1.0, 1.0 \times 10^{-18}]$. Then $\boldsymbol{\kappa}$ for the discontinuous solution is

$$\boldsymbol{\kappa} := \begin{cases} \kappa_1 \boldsymbol{I} & \text{if } 0.0 \le x \le 0.5, \\ \kappa_2 \boldsymbol{I} & \text{if } 1.0 \ge x > 0.5. \end{cases} \tag{42}$$

Subsequently, the boundary conditions are applied as follows:

$$p = \begin{cases} 0.0 & \text{at } x = 0.0, \\ 1.0 & \text{at } x = 1.0. \end{cases} \tag{43}$$

To find the optimal β, which provides the optimal error convergence rate, we employ the Algorithm 1. In this case, we set $k = 1$ but vary the choice of linear solver, θ, and the exact solutions (continuous/discontinuous). The optimal β results are shown in Table 4. The results of SIPG illustrate the similarity between the continuous and discontinuous solutions. The results of IIPG, however, show a discrepancy as to the lowest β values that provide the optimal convergence rate solution, which are different between the continuous and discontinuous solutions. The type of solver, direct and iterative solvers, does not influence the results.

TABLE 4. The lowest β value that provides the optimal convergence rate solution with a different type of exact solution (continuous or discontinuous), θ, and linear solver. Note that $\boldsymbol{\kappa}$ is heterogeneous, and $k = 1$.

exact solution	SIPG		IIPG	
	direct solver	iterative solver	direct solver	iterative solver
continuous (36)	1.11	1.11	0.83	0.83
discontinuous (41)	1.11	1.11	0.89	0.89

5.1.4. Effect of interior penalty parameter for linear solvers and optimal choice by employing machine learning algorithms. Similar to the results for the continuous solution presented in the previous section 5.1.2, the choice of β influences the number of linear iterative solvers significantly, as illustrated in Figure 7. In short, when β is increased, the number of iterations increases, while the number of iterations increases sharply before the solution becomes unstable.

To predict the optimal β for the iterative solver, we employ a similar approach that was used for the continuous solution in section 5.1.2. First, we evaluate each independent variable using the chi-squared test. In this example, we have a total of 57,835 data points. The result for the chi-squared test is provided in Table 5. It is observed that θ, β, and h have a p-value of less than 0.025. Hence, we include these variables as independent variables for further predictive model development. Then, the developed multi-variable regression [64] reads:

$$\text{number of iteration} = \alpha + \gamma_1 \times \theta + \gamma_2 \times \beta + \gamma_3 \times h, \tag{44}$$

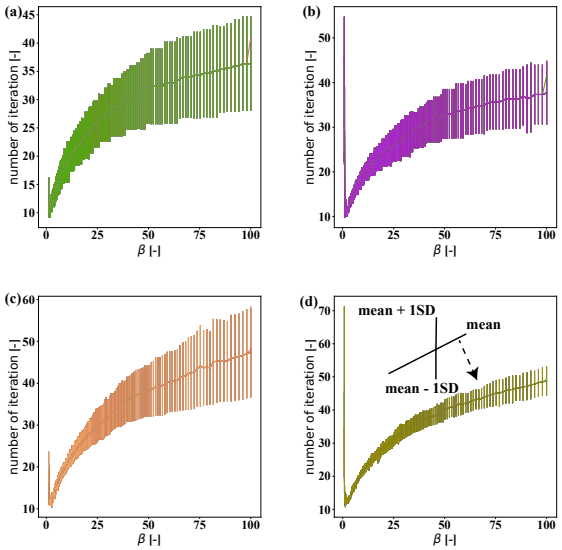

FIGURE 7. Number of iterations for linear iterative solver with heterogeneous $\boldsymbol{\kappa}$. **(a)** and **(b)** are the results for the continuous solution (36) by using SIPG and IIPG, respectivley. **(c)** and **(d)** are the results for the discontinuous solution (41) by using SIPG and IIPG, respectivley. Note that each line represents a different value of κ for the continuous solution, and κ_1 and κ_2 for the discontinuous solution. The error bar shows the mean and standard deviation ($\pm$ 1 SD) of each number of iterations; see (**d**) for an explanation. We observe that the variability in each β arises because we conducted runs with various values of κ for each β. The linear solver significantly influences the number of iterations, especially in lower k cases, where different κ values play a crucial role. However, the impact of κ becomes less prominent as k increases.

TABLE 5. Elliptic equation with the exact discontinuous solution: p-value results for each explanatory variable.

Variable	p-value
θ	≈ 0.00
κ_0	≈ 1.00
κ_1	≈ 1.00
β	≈ 0.00
h	≈ 0.00

where $\alpha = 19.59$, $\gamma_1 = -0.86$, $\gamma_2 = 0.42$, and $\gamma_3 = -19.51$. Similar to the previous equation (37), these parameters provide the minimum value ($\leq 1 \times 10^{-4}$) of MSE value (28). Other processes, including the splitting technique and optimization solvers, are the same as those utilized in the previous model. Finally, the r^2 and EVS obtained for this method is

$$r^2 = 0.84, \quad \text{and} \quad \text{EVS} = 0.84. \tag{45}$$

Next, to compare the above results by the nonlinear ANN algorithm, we construct the ANN model using three inputs (θ, β, and h) and one output (number of iterations) as shown in Figure 8. The number of hidden layers (N_{hl}) and the number of neurons (N_n) are used for tuning the hyperparameters. ReLU is used as an activation function for each hidden layer neuron. ADAM and MSE (28) are employed for the minimization method and loss function, respectively.

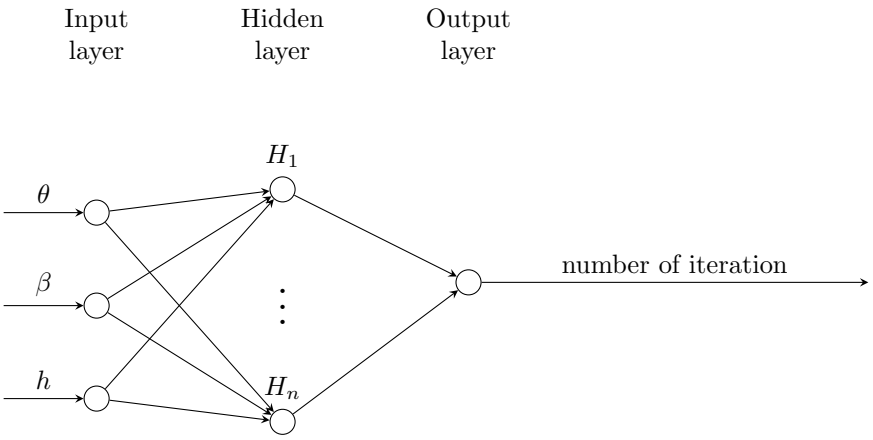

FIGURE 8. Neural network architecture used for the elliptic problem with the exact discontinuous solution. The number of hidden layers, N_{hl}, and the number of neurons for each hidden layer, N_n, are used as the sensitivity analysis parameters. H_1 and H_n represent the numbering of each neuron in each hidden layer.

Table 6 presents that the MSE of the validation set is decreased as N_{hl} and N_n are increased until $N_{hl} = 8$ and $N_n = 40$. Hence, we select $N_{hl} = 8$ and $N_n = 40$ for the test set. Then, we obtain the following final results

$$r^2 = 0.98, \quad \text{and} \quad \text{EVS} = 0.98. \tag{46}$$

The above results from the nonlinear ANN algorithms outperform the linear multivariable regression (45). From the results of the section 5.1, we can infer that the performance of the nonlinear approximation algorithm is better than the linear one; as a result, the relationship between the number of iterations and its dependent variables is nonlinear.

TABLE 6. Elliptic equation with the exact discontinuous solution: Mean squared error of the validation set for different number of hidden layers N_{hl} and different number of neurons per layer N_n.

N_{hl} \ N_n	10	20	40	80
2	5.49	1.57	3.16	1.67
4	1.93	1.53	1.71	1.63
8	1.44	2.34	1.43	1.56

5.2. Effect and optimal choice of interior penalty parameter for Biot's equations. In this example, we aim to investigate the effect of the interior penalty β on the solution quality of Biot's equations, where the elliptic flow equation is coupled with the solid mechanics as described in Section 3.2. However, employing DG approximation for the flow equation eliminates any spurious oscillations that are observed when the continuous Galerkin (CG) is used (especially at material interfaces where a large conductivity ($\boldsymbol{\kappa}$) contrast is located) as presented in [55, 58], the quality of DG solutions may be influenced by the choice of β. Thus, we employ the machine learning algorithm to find the optimal choice of β to avoid any instabilities upon the given physical and numerical parameters. In the following problems, we compare the performance of the linear logistic regression and nonlinear classification ANN, where the predicted values are binaries.

In the computational domain $\Omega = [0,1]^1$, the geometry and boundary conditions are shown in Figure 9a. Here, $\boldsymbol{\kappa}$ is defined as:

$$\boldsymbol{\kappa} := \begin{cases} \kappa_1 \boldsymbol{I} & \text{if } 0.0 \leq x \leq 0.5, \\ \kappa_2 \boldsymbol{I} & \text{if } 1.0 \geq x > 0.5, \end{cases} \tag{47}$$

and we define the ratio between κ_2 and κ_1 as

$$\kappa_{mult} := \frac{\kappa_2}{\kappa_1}. \tag{48}$$

5.2.1. Effect and optimal choice of interior penalty parameter for Biot's system. In this section, we study the optimal choice of interior penalty parameter β on the solution for Biot's system. The physical parameters are set as $\mu = 10^{-6}$ kPa.s, $\rho = 1000$ kg/m^3, $K = 1000$ kPa, $K_s \approx \infty$ kPa, which leads to $\alpha \approx 1$, and $v = 0.25$. In addition, we note that $\kappa_1 = 10^{-12}$m^2 and $\kappa_2 = 10^{-16}$m^2, and Lamé coefficients λ_l and μ_l are calculated by the following equations:

$$\lambda_l = \frac{3Kv}{1+v}, \quad \text{and} \quad \mu_l = \frac{3K(1-2v)}{2(1+v)}. \tag{49}$$

The numerical parameters are given as $h = 0.05\ m$ and $\Delta t^n = 1.0$ sec and the boundary conditions are set to $\boldsymbol{\sigma_D} = [0,1]$ kPa and $p_D = 0$ Pa. In addition, LU direct solver and SIPG ($\theta = 1$) are used to solve the discretized system.

To motivate our work, for example, the numerical simulation results by comparing $\beta = 1.1$ and $\beta = 50.0$ are presented in Figure 9b. Figure 9b illustrates that the choice of β can lead to different results in pressure solution, i.e., in the case of $\beta = 1.1$, the pressure solution exhibits no spurious pressure oscillations. In contrast, the oscillations appear when $\beta = 50.0$. Note that when β is too small, the solution may also become unstable, as illustrated in the previous section for the elliptic problem and discussed in [55] for Biot's equations. Thus, seeking the optimal penalty coefficient is crucial to obtain an accurate solution.

Next, we investigate the optimal choice of β by varying the physical parameters using the procedure illustrated in Algorithm 2. The ranges of the input values for all the test cases are given as; $\kappa_1 = [1.0 \times 10^{-14}, 1.0 \times 10^{-8}]$, $\kappa_2 = [1.0 \times 10^{-17}, 1.0 \times 10^{-14}]$, $\kappa_{mult} = [1.0 \times 10^{-8}, 1.0 \times 10^{1}]$, $\beta = [4.5 \times 10^{-2}, 2.0 \times 10^{-8}]$, and $h = [7.8 \times 10^{-3}, 6.25 \times 10^{-2}]$.

To determine the quality of the numerical solution, if the approximated solution is stable and smooth with no spurious pressure oscillations, we denote it as 'good.' On the other hand, if we observe any spurious pressure oscillations from the non-stable approximated solution, we denote it as 'bad.' Thus, in this case, we utilize

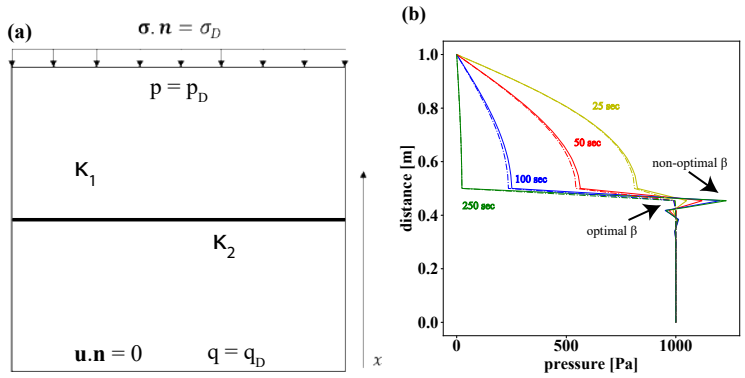

FIGURE 9. (a) Geometry and boundary conditions used in Biot's equations study and (b) pressure results for an example of the effect of β on the solution quality.

the bool type variable BOOL_QUALITY for binary classification in which 1 indicates 'good' and 0 indicates 'bad.'

Algorithm 2: Investigation procedure for Biot's equations

```
Initialize sets of each variable; BOOL_QUALITY, κ, κ_mult, h, and β
for i < n_κ1, where n_κ1 is the size of the specified κ1 list do
  Assign κ1 := κ1 [i] I
  for j < n_κmult, where n_κmult is the size of the specified κ_mult list do
    Assign κ2 := κ_mult [j] × κ1
    for k < n_h, where n_h is the size of the specified h list do
      Assign h := h [k]
      for l < n_β, where n_β is the size of the specified β list do
        Assign β := β [l]
        Solve the coupled Biot's system: (20) and (25).
        if linear solver converges then
          if spurious nonphysical oscillation is detected then
            BOOL_QUALITY = 0
          else
            BOOL_QUALITY = 1
          end if
        else
          BOOL_QUALITY = 0
        end if
      end for
    end for
  end for
end for
```

Like the previous sections, we begin with the chi-squared test to find the statistically significant explanatory variables. In total, we have 14,141 cases (data points) with 3,927 'good'(BOOL_QUALITY = 1) solutions and 10,214 'bad'(BOOL_QUALITY = 0) solutions. The chi-squared test result is presented in Table 7, and it shows that all variables, κ_1, κ_2, κ_{mult}, and h, have p-value less than 0.025. Therefore,

these variables are included as independent variables to develop the following predictive models. As discussed in both sections 4.1 and 4.2, the data sets are split by

TABLE 7. Biot's equations: p-value results for each explanatory variable.

Variable	p-value
κ_1	≈ 0.00
κ_2	≈ 0.00
κ_{mult}	≈ 0.00
β	≈ 0.00
h	≈ 0.01

training, validation, and test sets using the splitting ratio $[0.8, 0.1, 0.1]$. We employ the training set to train the linear and nonlinear machine learning algorithms. The validation set is for tuning the hyper-parameters for the nonlinear ANN models, and the test set is for comparing performances between the linear and nonlinear algorithms.

First, the multi-variable logistic regression [64] is defined as:

(50)
$$\log\left(\frac{\mathbb{P}(\mathbb{O}=1)}{1-(\mathbb{P}(\mathbb{O}=1))}\right) = \gamma_0 + \gamma_1 \times \boldsymbol{\kappa}_1 + \gamma_2 \times \boldsymbol{\kappa}_2 + \gamma_3 \times \kappa_{mult} + \gamma_4 \times \beta + \gamma_5 \times h,$$

where

(51)
$$\mathbb{O} = \begin{cases} 1, & \text{if } \mathbb{P}(\mathbb{O}=1) \geq 0.5 \\ 0, & \text{if } \mathbb{P}(\mathbb{O}=1) < 0.5. \end{cases}$$

Here, $\gamma_0 = -1.19$, $\gamma_1 = -0.06$, $\gamma_2 = 0.68$, $\gamma_3 = 5.55$, $\gamma_4 = -5.49$, and $\gamma_5 = 0.32$. These parameters provide the minimum value ($\leq 1 \times 10^{-4}$) of BCE value (33).

After applying the algorithm explained in the section 4 and 4.1, the computed accuracy (34) of the logistic regression model is

$$\text{ACC} = 0.80,$$

and the confusion matrix is presented in Table 8. In table 8, we observe that the number of 'false positives' is much higher than that of 'false negatives,' which may result in the bad solution obtained from the finite element model. When our model creates a 'false positive,' we expect the simulation results to be stable and contain no oscillation; however, the solution quality is bad.

TABLE 8. Biot's equations: Confusion matrix of the logistic regression for the test set.

total test set = 1415		Test set values	
		Good (1)	Bad (0)
Predicted values	Good (1)	172	210
	Bad (0)	74	959

Secondly, we develop the classification ANN model utilizing five inputs ($\boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2$, κ_{mult}, β, and h) and one output (BOOL_QUALITY)as shown in Figure 10 for this problem. Table 9 illustrates the result for hyperparameters tuning, and it illustrates that the ANN predictive performance is improved as N_{hl} and N_n are increased up until $N_{hl} = 4$ and $N_n = 80$. Hence, we use $N_{hl} = 4$ and $N_n = 80$ to set the

TABLE 9. Biot's equations: Accuracy of the validation set for different number of hidden layers N_{hl} and different number of neurons per layer N_n.

N_{hl} \ N_n	10	20	40	80	120
2	0.89	0.93	0.93	0.93	0.92
4	0.89	0.93	0.93	0.93	0.93
8	0.88	0.92	0.93	0.93	0.93
16	0.73	0.73	0.73	0.27	0.27
32	0.73	0.73	0.73	0.73	0.73

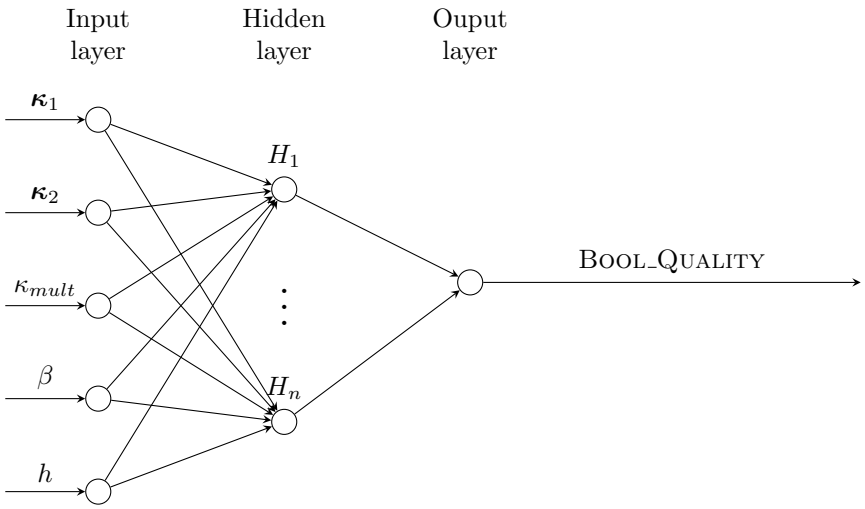

FIGURE 10. Neural network architecture used for Biot's equation. The number of hidden layers, N_{hl}, and the number of neurons for each hidden layer, N_n, are used as the sensitivity analysis parameters. H_1 and H_n represent the numbering of each neuron in each hidden layer.

hyperparameters, and we compare the nonlinear classification ANN and the logistic regression models' performance. Here, ReLU is used as an activation function for each neuron of the hidden layer, and the Sigmoid activation function is used for the output layer. ADAM and BCE (33) are employed for the minimization method and loss function, respectively.

Finally, the computed accuracy value (34) of the nonlinear classification ANN using the above test set is

$$\text{ACC} = 0.93.$$

We note that this value is much higher than that of the logistic regression model.

Furthermore, the number of 'false positive' cases presented in Table 10 is much lower than that of the linear logistic regression algorithm. This characteristic helps to prevent the finite element model from producing bad-quality simulation results, as discussed previously.

TABLE 10. Biot's equations: Confusion matrix of the artificial neural network (ANN) for the test set.

total test set = 1415		Test set values	
		Good (1)	Bad (0)
Predicted values	Good (1)	367	15
	Bad (0)	72	961

We note that one can leverage input features, including parameters like κ and/or k, or any other pertinent parameters inherent in our system, to deduce optimal values for β. This capability stems from the inherent flexibility of neural networks as versatile non-linear mapping tools. In this context, we have demonstrated this proficiency by showcasing the neural network's capacity to infer two distinct types of quantities. Firstly, it can predict the number of iterations, a continuous variable, emphasizing the network's adaptability to capture nuanced relationships. Additionally, it excels in predicting a boolean quantity, representing solution quality as either good or bad, thereby underscoring its effectiveness in handling discrete variables associated with solution evaluation.

5.2.2. Application of the optimal choice obtained from the trained model with different settings of the material parameters. In this final section, we test the optimal choice of β obtained from the trained ANN model in the previous section with the different settings of the material parameters. Thus, the objective is to illustrate the capabilities of our computational framework by applying the trained model (see Figure 10) to different settings, which is often required in realistic scenarios with the uncertainty of the data.

We derive an optimal β from a model previously trained in Section 5.2.1. This exemplifies the potential of extending the trained model to diverse settings. To expound on this, our input features remain consistent with those employed in Section 5.2.1, encompassing κ_1, κ_2, the ratio between κ_1 and κ_2, β, and h. The output of interest is the Boolean Quality, denoting whether the solutions are categorized as good or bad.

In essence, determining an optimal β hinges on the resultant Bool Quality; if a specific β yields a Bool Quality of 'good,' we classify that particular β as optimal in this context. This approach allows us to identify and emphasize the effectiveness of certain β values in producing desirable outcomes, contributing to a nuanced understanding of the model's performance across

Here, we assume that we have a new and different heterogeneity in the permeability value $\boldsymbol{\kappa_2}$ and the Biot's coefficient α, and our trained ANN model has no prior knowledge. The geometry of the domain and the boundary conditions used in this example are shown in Figure 11. In this example, $\boldsymbol{\kappa_2}$ values are $[9.36 \times 10^{-14}, 9.57 \times 10^{-15}]$ and $\alpha = [0.4, 0.6, 0.8, 1.0]$. Other parameters are similar to those of the previous example.

The results (pressure values) are shown in Figures 12 and 13 for $t = 20$ and $t = 120$, respectively. Each time has a total of eight cases (with the choice of α and $\boldsymbol{\kappa_2}$). Using the optimal β values from the trained ANN model from the previous section with different heterogeneity still provides good predictions for the pressure values without any spurious oscillations. We note that some oscillations in these figures are due to interpolation from unstructured mesh to structured mesh for plotting purposes.

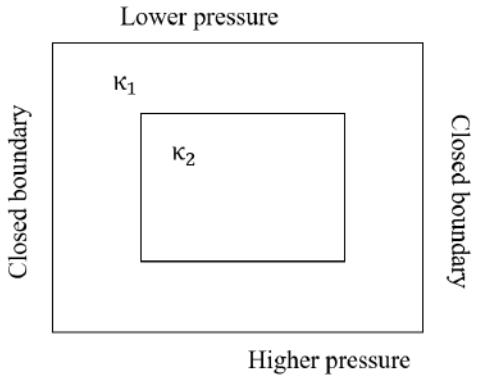

FIGURE 11. Geometry used to test an application of the optimal choice obtained from the trained model with different settings of the material parameters.

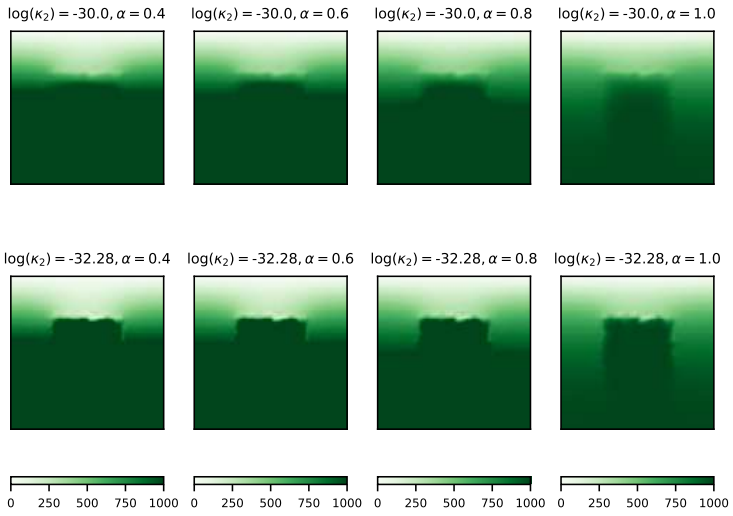

FIGURE 12. Pressure values for the eight different cases with different $\boldsymbol{\kappa_2}$ and α values at $t = 20$s.

When confronted with applying the framework to a problem devoid of a known true solution, two avenues exist to explore this challenge. Notably, both approaches necessitate users to construct a training set for the initial training of NN models.

The first approach aligns with our existing methodology, involving the utilization of input featuresnamely, κ_1, κ_2, the ratio between κ_1 and κ_2, β, and h to predict Bool Quality. This maintains consistency with our established framework,

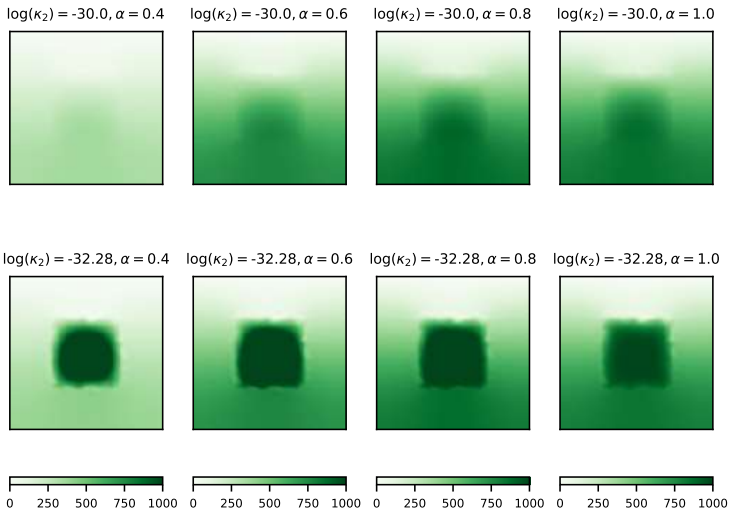

FIGURE 13. Pressure values for the eight different cases with different $\boldsymbol{\kappa_2}$ and α values at $t = 120$s.

wherein the NN model is trained to anticipate the qualitative outcome of solutions, categorized as either 'good' or 'bad.'

Conversely, the second approach deviates by employing input features that exclude β but instead focuses on predicting β such that the finite element method (FEM) solution remains stable, devoid of spurious oscillations. This alternative strategy underscores the framework's flexibility, showcasing its adaptability to different problem settings and objectives. By shifting the focus to the stability of the FEM solution, users can gain insights into the behavior of β values that contribute to stable outcomes, even in the absence of a ground truth solution for direct comparison.

We aim to emphasize that beyond utilizing β as an input feature, we can harness the broader spectrum of input features available, including parameters such as κ and/or k, or any other relevant parameters inherent in our system. This expanded set of inputs empowers us to deduce optimal values for β, showcasing the remarkable adaptability of neural networks as versatile non-linear mapping tools.

The inherent flexibility of neural networks is manifested in their ability to seamlessly incorporate various input parameters, allowing for a more comprehensive exploration of the solution space. In our current study, we have illustrated this adaptability by exemplifying the neural network's proficiency in inferring two distinct types of quantities. Firstly, it can predict the number of iterations, a continuous variable. This highlights the network's adaptability and underscores its capacity to capture nuanced relationships within the data.

Furthermore, the neural network excels in predicting a boolean quantity, discerning solution quality as either good or bad. This success underscores its effectiveness in handling discrete variables associated with solution evaluation. The framework

thus lays a solid foundation for further extension, demonstrating its applicability to a wide range of problems. By leveraging a diverse set of input features, our approach provides a robust and flexible solution, paving the way for addressing complex challenges across various domains.

6. Conclusions

This paper presents the effect of choosing the interior penalty parameter of the discontinuous Galerkin finite element methods for the elliptic problems and Biots systems. The optimal choice of the interior penalty parameter results in stable solutions, an optimum error convergence rate, and fewer iterations for the iterative solver, eliminating any spurious numerical oscillation in the approximated solutions. We propose nonlinear approximation algorithms, regression, and classification to predict the optimal choice of the interior penalty parameter. These nonlinear approximation algorithms outperform the classic linear approximation algorithms. Our proposed framework can benefit sensitivity analysis, uncertainty quantification, or data assimilation modeling, where many simulations have to be performed with different settings, e.g., mesh size, material properties, or different interior penalty schemes. Moreover, it can be extended to any multiscale multiphysics problems.

Acknowledgments

SL is supported by National Science Foundation under Grant No. NSF DMS-2208402. TK and HM have received funding from the Danish Hydrocarbon Research and Technology Centre under the Advanced Water Flooding program. We acknowledge developers and contributors of TensorFlow [70], Keras [71], Scikit-learn [64], FEniCS [79], and Multiphenics [82] libraries.

References

[1] J. Nitsche, Über ein variationsprinzip zur lösung von dirichlet-problemen bei verwendung von teilräumen, die keinen randbedingungen unterworfen sind, in: Abhandlungen aus dem mathematischen Seminar der Universität Hamburg, Vol. 36, Springer, 1971, pp. 9–15.

[2] J. Douglas, T. Dupont, Interior penalty procedures for elliptic and parabolic galerkin methods, in: Computing methods in applied sciences, Springer, 1976, pp. 207–216.

[3] M. F. Wheeler, An elliptic collocation-finite element method with interior penalties, SIAM Journal on Numerical Analysis 15 (1) (1978) 152–161.

[4] P. Percell, M. F. Wheeler, A local residual finite element procedure for elliptic equations, SIAM Journal on Numerical Analysis 15 (4) (1978) 705–714.

[5] D. N. Arnold, An interior penalty finite element method with discontinuous elements, SIAM journal on Numerical Analysis 19 (4) (1982) 742–760.

[6] B. Rivière, M. F. Wheeler, Coupling locally conservative methods for single phase flow, Computational Geosciences 6 (3) 269–284.

[7] B. Rivière, M. F. Wheeler, A discontinuous galerkin method applied to nonlinear parabolic equations, in: Discontinuous Galerkin methods, Springer, 2000, pp. 231–244.

[8] B. Rivière, M. F. Wheeler, Discontinuous galerkin methods for flow and transport problems in porous media, Communications in Numerical Methods in Engineering 18 (1) (2002) 63–68.

[9] B. Cockburn, C. Dawson, Some extensions of the local discontinuous galerkin method for convection-diffusion equations in multidimensions, Mathematics of Finite Elements and Applications, 1999.

[10] B. Cockburn, C.-W. Shu, The local discontinuous galerkin method for time-dependent convection-diffusion systems, SIAM Journal on Numerical Analysis 35 (6) (1998) 2440–2463.

[11] S. Sun, M. F. Wheeler, Discontinuous galerkin methods for coupled flow and reactive transport problems, Applied Numerical Mathematics 52 (2) (2005) 273–298.

[12] S. Sun, M. F. Wheeler, Anisotropic and dynamic mesh adaptation for discontinuous galerkin methods applied to reactive transport, Computer Methods in Applied Mechanics and Engineering 195 (25-28) (2006) 3382 – 3405.

[13] I. Babuška, C. E. Baumann, J. T. Oden, A discontinuous hp finite element method for diffusion problems: 1-d analysis, Computers & Mathematics with Applications 37 (9) (1999) 103–122.
[14] M. Ainsworth, A posteriori error estimation for discontinuous galerkin finite element approximation, SIAM Journal on Numerical Analysis 45 (4) (2007) 1777–1798.
[15] M. Ainsworth, R. Rankin, Fully computable error bounds for discontinuous galerkin finite element approximations on meshes with an arbitrary number of levels of hanging nodes, SIAM Journal on Numerical Analysis 47 (6) (2010) 4112–4141.
[16] M. Ainsworth, R. Rankin, Constant free error bounds for nonuniform order discontinuous galerkin finite-element approximation on locally refined meshes with hanging nodes, IMA Journal of Numerical Analysis 31 (1) (2009) 254–280.
[17] Y. Epshteyn, B. Rivière, Estimation of penalty parameters for symmetric interior penalty galerkin methods, Journal of Computational and Applied Mathematics 206 (2) (2007) 843–872.
[18] K. Shahbazi, An explicit expression for the penalty parameter of the interior penalty method, Journal of Computational Physics 205 (2) (2005) 401–407.
[19] A. Ern, A. F. Stephansen, A posteriori energy-norm error estimates for advection-diffusion equations approximated by weighted interior penalty methods, Journal of Computational Mathematics (2008) 488–510.
[20] A. Ern, A. F. Stephansen, P. Zunino, A discontinuous galerkin method with weighted averages for advection–diffusion equations with locally small and anisotropic diffusivity, IMA Journal of Numerical Analysis 29 (2) (2009) 235–256.
[21] M. Ainsworth, R. Rankin, A note on the selection of the penalty parameter for discontinuous galerkin finite element schemes, Numerical Methods for Partial Differential Equations 28 (3) (2012) 1099–1104.
[22] M. W. Libbrecht, W. S. Noble, Machine learning applications in genetics and genomics, Nature Reviews Genetics 16 (6) (2015) 321.
[23] H. Brink, J. Richards, M. Fetherolf, Real-world machine learning, Manning Publications Co., 2016.
[24] L. Thomas, Jmp start statistics: a guide to statistics and data analysis using jmp and jmp in software, Biometrics 55 (4) (1999) 1319.
[25] G. A. Seber, A. J. Lee, Linear regression analysis, Vol. 329, John Wiley & Sons, 2012.
[26] N. Cressie, Spatial prediction and ordinary kriging, Mathematical Geology 20 (4) (1988) 405–421.
[27] I. J. Myung, Tutorial on maximum likelihood estimation, Journal of Mathematical Psychology 47 (1) (2003) 90–100.
[28] J. Y. Park, P. C. Phillips, Nonlinear regressions with integrated time series, Econometrica 69 (1) (2001) 117–161.
[29] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.
[30] T. Chilimbi, Y. Suzue, J. Apacible, K. Kalyanaraman, Project adam: Building an efficient and scalable deep learning training system, in: 11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14), 2014, pp. 571–582.
[31] H. Cui, H. Zhang, G. R. Ganger, P. B. Gibbons, E. P. Xing, Geeps: Scalable deep learning on distributed gpus with a gpu-specialized parameter server, in: Proceedings of the Eleventh European Conference on Computer Systems, ACM, 2016, p. 4.
[32] G. Hinton, N. Srivastava, K. Swersky, Neural networks for machine learning lecture 6a overview of mini-batch gradient descent, Cited on 14 (2012) 8.
[33] J.-X. Wang, J.-L. Wu, H. Xiao, Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data, Physical Review Fluids 2 (3) (2017) 034603.
[34] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.
[35] L. Swiler, M. Gulian, A. Frankel, C. Safta, J. Jakeman, A survey of constrained Gaussian process regression: approaches and implementation challenges, Journal of Machine Learning for Modeling and Computing 1 (2) (2020) 119–156.
[36] S. Lee, T. Kadeethum, Physics-informed neural networks for solving coupled flow and transport system., in: AAAI Spring Symposium: MLPS, 2021.

[37] S. Goswami, K. Kontolati, M. Shields, G. Karniadakis, Deep transfer learning for partial differential equations under conditional shift with deeponet, arXiv preprint arXiv:2204.09810 (2022).
[38] V. Oommen, K. Shukla, S. Goswami, R. Dingreville, G. Karniadakis, Learning two-phase microstructure evolution using neural operators and autoencoder architectures, arXiv preprint arXiv:2204.07230 (2022).
[39] A. Oishi, G. Yagawa, Computational mechanics enhanced by deep learning, Computer Methods in Applied Mechanics and Engineering 327 (2017) 327–351.
[40] P. Antonietti, M. Caldana, L. Dede, Accelerating algebraic multigrid methods via artificial neural networks, arXiv preprint arXiv:2111.01629 (2021).
[41] V. Silva, P. Salinas, M. Jackson, C. Pain, Machine learning acceleration for nonlinear solvers applied to multiphase porous media flow, Computer Methods in Applied Mechanics and Engineering 384 (2021) 113989.
[42] P. Antonietti, E. Manuzzi, Refinement of polygonal grids using convolutional neural networks with applications to polygonal discontinuous galerkin and virtual element methods, Journal of Computational Physics 452 (2022) 110900.
[43] T. Kadeethum, D. O'Malley, F. Ballarin, I. Ang, J. Fuhg, N. Bouklas, V. Silva, P. Salinas, C. Heaney, C. Pain, S. Lee, H. Viswanathan, H. Yoon, Enhancing high-fidelity nonlinear solver with reduced order model, Scientific Report 12 (1) (2022c) 20229.
[44] M. Biot, General theory of three-dimensional consolidation, Journal of Applied Physics 12 (2) (1941) 155–164.
[45] O. Coussy, Poromechanics, John Wiley & Sons, 2004.
[46] J. Jaeger, N. Cook, R. Zimmerman, Fundamentals of Rock Mechanics, 4th Edition, Wiley-Blackwell, 2010.
[47] A. Ern, A. F. Stephansen, P. Zunino, A discontinuous Galerkin method with weighted averages for advection-diffusion equations with locally small and anisotropic diffusivity, IMA Journal of Numerical Analysis 29 (2) (2009) 235–256.
[48] R. Stenberg, Mortaring by a method of J. A. Nitsche, in: Computational mechanics (Buenos Aires, 1998), Centro Internac. Métodos Numér. Ing., Barcelona, 1998, pp. CD–ROM file.
[49] M. Dryja, On discontinuous Galerkin methods for elliptic problems with discontinuous coefficients, Computational Methods in Applied Mathematics 3 (1) (2003) 76–85.
[50] E. Burman, P. Zunino, A domain decomposition method based on weighted interior penalties for advection-diffusion-reaction problems, SIAM Journal on Numerical Analysis 44 (4) (2006) 1612–1638.
[51] D. A. Di Pietro, A. Ern, J.-L. Guermond, Discontinuous Galerkin methods for anisotropic semidefinite diffusion with advection, SIAM Journal on Numerical Analysis 46 (2) (2008) 805–831.
[52] P. Houston, C. Schwab, E. Süli, Discontinuous *hp*-finite element methods for advection-diffusion-reaction problems, SIAM Journal on Numerical Analysis 39 (6) (2002) 2133–2163 (electronic).
[53] C. Dawson, S. Sun, M. F. Wheeler, Compatible algorithms for coupled flow and transport, Computed Methods Applied Mechanics and Engineering. 193 (23-26) (2004) 2565–2580.
[54] S. Lee, Y.-J. Lee, M. F. Wheeler, A locally conservative enriched galerkin approximation and efficient solver for elliptic and parabolic problems, SIAM Journal on Scientific Computing 38 (3) (2016) A1404–A1429.
[55] J. Choo, S. Lee, Enriched Galerkin finite elements for coupled poromechanics with local mass conservation, Computer Methods in Applied Mechanics and Engineering 341 (2018) 311–332.
[56] S. Lee, S.-Y. Yi, Locking-free and locally-conservative enriched galerkin method for poroelasticity, Journal of Scientific Computing 94 (1) (2023) 26.
[57] S.-Y. Yi, S. Lee, L. Zikatanov, Locking-free enriched galerkin method for linear elasticity, SIAM Journal on Numerical Analysis 60 (1) (2022) 52–75.
[58] T. Kadeethum, H. Nick, S. Lee, C. Richardson, S. Salimzadeh, F. Ballarin, A Novel Enriched Galerkin Method for Modelling Coupled Flow and Mechanical Deformation in Heterogeneous Porous Media, in: 53rd US Rock Mechanics/Geomechanics Symposium, American Rock Mechanics Association, New York, NY, USA, 2019.
[59] Y. Chen, Y. Luo, M. Feng, Analysis of a discontinuous galerkin method for the biots consolidation problem, Applied Mathematics and Computation 219 (17) (2013) 9043–9056.
[60] P. J. Phillips, M. F. Wheeler, A coupling of mixed and continuous galerkin finite element methods for poroelasticity i: the continuous in time case, Computational Geosciences 11 (2007) 131–144.

[61] T. Kadeethum, S. Lee, H. Nick, Finite element solvers for biots poroelasticity equations in porous media, Mathematical Geosciences 52 (2020) 977–1015.
[62] T. Kadeethum, H. M. Nick, S. Lee, F. Ballarin, Enriched galerkin discretization for modeling poroelasticity and permeability alteration in heterogeneous porous media, Journal of Computational Physics 427 (2021) 110030.
[63] R. E. Walpole, R. H. Myers, S. L. Myers, K. Ye, Probability and statistics for engineers and scientists, Vol. 5, Macmillan New York, 1993.
[64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.
[65] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python, [Online; accessed <today>] (2001–). URL `http://www.scipy.org/`
[66] L. Deng, D. Yu, et al., Deep learning: methods and applications, Foundations and Trends® in Signal Processing 7 (3–4) (2014) 197–387.
[67] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436.
[68] H. N. Mhaskar, T. Poggio, Deep vs. shallow networks: An approximation theory perspective, Analysis and Applications 14 (06) (2016) 829–848.
[69] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al., Wide & deep learning for recommender systems, in: Proceedings of the 1st workshop on deep learning for recommender systems, ACM, 2016, pp. 7–10.
[70] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015). URL `https://www.tensorflow.org/`
[71] F. Chollet, et al., Keras, `https://keras.io` (2015).
[72] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747 (2016).
[73] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.
[74] R. Ge, F. Huang, C. Jin, Y. Yuan, Escaping from saddle pointsonline stochastic gradient for tensor decomposition, in: Conference on Learning Theory, 2015, pp. 797–842.
[75] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
[76] G. James, D. Witten, T. Hastie, R. Tibshirani, An introduction to statistical learning, Vol. 112, Springer, 2013.
[77] M. Kuhn, K. Johnson, Applied predictive modeling, Vol. 26, Springer, 2013.
[78] S. J. Russell, P. Norvig, Artificial intelligence: a modern approach, Malaysia; Pearson Education Limited,, 2016.
[79] M. S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, G. N. Wells, The FEniCS Project Version 1.5, Archive of Numerical Software 3 (100) (2015).
[80] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, D. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Users Manual, Tech. Rep. ANL-95/11 - Revision 3.10, Argonne National Laboratory (2018).
[81] G. Scovazzi, M. Wheeler, A. Mikelic, S. Lee, Analytical and variational numerical methods for unstable miscible displacement flows in porous media, Journal of Computational Physics 335 (2017) 444–496.
[82] F. Ballarin, G. Rozza, Multiphenics-easy prototyping of multiphysics problems in FEniCS (2019).

Department of Mathematics, Florida State University
E-mail: lee@math.fsu.edu

Sandia National Laboratories, New Mexico, USA
E-mail: tkadeet@sandia.gov

Danish Offshore Technology Centre, Technical University of Denmark
E-mail: hamid@dtu.dk