

arXiv:2404.15242v1 [cs.LG] 23 Apr 2024

A Hybrid Kernel-Free Boundary Integral Method with
Operator Learning for Solving Parametric Partial
Differential Equations In Complex Domains

Shuo Ling*

Liwei Tan*

School of Mathematical Sciences,
Shanghai Jiao Tong University,
Shanghai 200240, P.R. China

School of Mathematical Sciences,
Shanghai Jiao Tong University,
Shanghai 200240, P.R. China

Wenjun Ying†

School of Mathematical Sciences, MOE-LSC and Institute of Natural Sciences,
Shanghai Jiao Tong University, Minhang,
Shanghai 200240, P.R. China.

April 24, 2024

Abstract

The Kernel-Free Boundary Integral (KFBI) method presents an iterative solution to boundary integral equations arising from elliptic partial differential equations (PDEs). This method effectively addresses elliptic PDEs on irregular domains, including the modified Helmholtz, Stokes, and elasticity equations. The rapid evolution of neural networks and deep learning has invigorated the exploration of numerical PDEs. An increasing interest is observed in deep learning approaches that seamlessly integrate mathematical principles for investigating numerical PDEs. We propose a hybrid KFBI method, integrating the foundational principles of the KFBI method with the capabilities of deep learning. This approach, within the framework of the boundary integral method, designs a network to approximate the solution operator for the corresponding integral equations by mapping the parameters, inhomogeneous terms and boundary information of PDEs to the boundary density functions, which can be regarded as the solution of the integral equations. The models are trained using data generated by the Cartesian grid-based KFBI algorithm, exhibiting robust generalization capabilities. It accurately predicts density functions across diverse boundary conditions and parameters within the same class of equations. Experimental results demonstrate that the trained model can directly infer the boundary density function with satisfactory precision, obviating the need for iterative steps in solving boundary integral equations. Furthermore, applying the inference results of the model as initial values for iterations is also reasonable; this approach can retain the inherent second-order accuracy of the KFBI method while accelerating the traditional KFBI approach by reducing about 50% iterations.

*These authors contributed equally to this work.

†Corresponding author. Email: wying@sjtu.edu.cn

1 Introduction and Related Works 3

Elliptic problems are widely applied in the fields of electrochemistry [11, 43], electromagnetism [6], computational fluid dynamics [21, 44], shape optimisation problems [19, 67] and other areas in science [7, 9, 50, 66]. Representative methods for solving specific elliptic problems numerically are the finite difference method [1, 4, 16, 25, 30, 32, 41, 51, 65], finite element method [8, 10, 24, 27, 33, 34], boundary integral method [22, 28, 57, 58]. Some numerical methods based on deep learning [14, 15, 23, 26, 35, 38, 47, 60] are becoming popular in recent years. As a competitive approach to traditional and new methods, the Kernel-Free Boundary Integral (KFBI) method [52–54] has shown its advantages. 4

The KFBI method is executed on a Cartesian grid for the resolution of general elliptic PDEs within domains of irregular shape with smooth perimeters. The KFBI method iteratively addresses the boundary integral equations and maintains symmetry and positive definiteness in the resultant discrete systems. This preservation enables the employment of effective solution strategies, including FFT-based or geometric multigrid solvers. Originating in the boundary integral method, the KFBI method not only retains the favourable conditioning of the boundary integral equation but also obviates the direct computation of Green’s function, which is notably complex in irregular domains [53, 55]. Consequently, its effectiveness is particularly notable in overcoming computational mathematics challenges. In recent years, the KFBI method has been extensively applied [12, 55, 56, 62, 64]. 5

Deep learning methodologies have been acknowledged for their transformative potential in scientific research, offering accelerated solutions that approximate or surpass traditional methods in some specific scenarios [20, 29, 31, 46]. Deep neural networks (DNNs) have been increasingly utilized for solving PDEs, circumventing the explicit discretization requirement, and learning mappings in specific spaces favorable for discovering solutions to PDEs. Simultaneously, DNN-based methods demonstrate efficacy in mitigating the curse of dimensionality and prove beneficial in addressing certain inverse problems. Various neural network architectures, loss functions, and activation functions have been explored for this purpose. For instance, the deep Galerkin method (DGM) [49] and physics-informed neural networks (PINNs) [48] employ equation residuals as loss functions within a general framework for PDE resolution. These networks are refined through stochastic gradient descent, applying spatial point random sampling within the domain. Implement of conditions is achieved by network integration [3] or loss penalization, with the latter relying on penalty coefficients as hyper-parameters. However, the fine-tuning of these coefficients is a complex process necessitating further methodical investigation. A recently proposed boundary integral network (BINet) [37] presented a convolution representation of the solutions to elliptic PDEs using Green’s functions. This approach was subsequently extended to a method for discovering general Green’s functions, which can be acquired through training neural networks [36]. 6

In the context of operator learning, classical neural networks, which are confined to mapping between finite-dimensional spaces, face limitations in learning discretization-specific solutions on two-dimensional or higher-dimensional grids. This necessitates the development of mesh-invariant neural networks and other DNN-based methods who can play the role of reducing the dimension of operator learning. Recent studies have introduced the concept of learning mesh-free, infinite-dimensional operators using neural networks [5, 35, 38, 40, 42]. These neural operators, capable of resolving a class of PDEs rather than specific instances, allow evaluations at arbitrary temporal and spatial points. As an example, the Fourier Neural Operator (FNO) is proposed in [35], utilizing Fourier transformation for network architecture design. For DeepONet proposed in [38], a network structure comprising branch and trunk nets is introduced, addressing 7

PDE parameters and spatial coordinates, respectively. Additionally, Deep Green [18] and MOD-net [61] employ neural networks to approximate Green's function, effectively representing the solution operator for nonlinear PDEs by mapping source terms or boundary values to solutions.

To a certain extent, the previously mentioned class of operator learning methods appears to lack precision assurances, with some methods exhibiting less satisfactory accuracy. This deficiency can be attributed to inadequate utilization of mathematical prior knowledge. In this work, we introduce a novel approach called hybrid kernel-free boundary integral (hybrid KFBI) method that integrates operator learning with the KFBI method. The method solves equations within the framework of boundary integral methods, featuring DNNs designed to approximate the solution operators of the corresponding boundary integral equations, which maps from the parameters, inhomogeneous terms and boundary information of PDEs to the boundary density functions. The hybrid KFBI Method has advantages including:

High Data Quality: KFBI is fundamentally a boundary integral method that transforms two-dimensional problems into one-dimensional boundary problems, achieving dimensionality reduction in the model. Furthermore, due to its high precision, the KFBI method can produce high-quality data for training process of hybrid KFBI method.

High Precision: The trained model can directly predict density functions (instead of solving the boundary integral equations iteratively) for computing solutions for the original PDEs, significantly reducing the solution time for the KFBI method while maintaining high accuracy (relative error in the order of $1E-3$ or $1E-4$). It can also be employed as an initial value, substantially decreasing the number of iterations required without compromising the accuracy relative to the KFBI method.

Strong Generalizability: Based on our meticulously designed network architecture and input-output methodology, each trained model is applicable to a broad range of equations. For instance, when elastic equations are considered, the different inhomogeneous terms, boundary conditions, and physical parameters can be inputted into the same model, which demonstrates robust generalization capabilities.

Model Dimensionality Reduction: The boundary density functions are defined on the boundary whose dimension is less by 1 than the original computational domain, which is helpful for operator learning. Lower dimension leads to fewer sampling points which will reduce the computational cost.

High Integration Capability: Our approach possesses the ability to integrate with certain other methods, such as those reliant on high-performance computing using GPUs, thereby achieving a more efficient solution efficiency.

The structure of this paper is methodically organized as follows. Initially, the fundamental theory underlying the boundary integral (BI) method and the KFBI method is introduced in Section 2. Following this, Section 3 elaborates on various aspects of the hybrid KFBI method in detail, which integrates operator learning techniques to enhance its efficacy and versatility. Subsequently, Section 4 is dedicated to presenting the numerical results obtained. The concluding Section 5 engages in a comprehensive discussion regarding the merits, limitations, and prospects of the hybrid KFBI method, encapsulating the essence of this research.

2 Preliminaries 10

In this section, we will expound upon a segment of knowledge pertaining to BI methods, serving as the foundation for both the KFBI method and the hybrid KFBI approach. Let Ω be a bounded, two-dimensional domain of irregular and complex structure, with a boundary $\Gamma = \partial\Omega$ possessing

at least C^2 continuity. The function $u(\mathbf{x}) = u(x, y)$, where $\mathbf{x} \in \mathbf{R}^2$, remains undetermined (analogous considerations apply to \mathbf{R}^d for $d > 2$). The foundational principles of the BI method are discussed for the Dirichlet boundary condition. The case with the Neumann condition is addressed in Appendix B.

As shown in Fig. 1, to solve the boundary value problems on domain Ω by the BI method, we first embed the irregular domain Ω into a larger rectangle domain \mathcal{B} and denote by $\Omega^c = \mathcal{B} \setminus \Omega$ the complement of the domain Ω in \mathcal{B} .

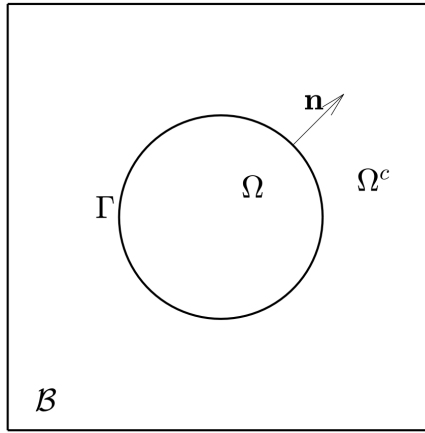


Figure 1: BI method and KFBI method computation domain

Suppose \mathcal{L} is an elliptic operator, consider the elliptic equation

$$(\mathcal{L}u)(\mathbf{x}) = f(\mathbf{x}), \quad \text{in } \Omega, \quad (1)$$

where $\mathcal{L} = \nabla \cdot \sigma(\mathbf{x}) \nabla - \kappa(\mathbf{x})$ with the diffusion tensor σ and the reaction coefficient $\kappa \geq 0$ which are at least continuously differentiable over the regular domain \mathcal{B} . Subject to pure Dirichlet boundary condition

$$u(\mathbf{x}) = g_D(\mathbf{x}), \quad \text{on } \Gamma, \quad (2)$$

where $f(\mathbf{x})$ and $g_D(\mathbf{x})$ are known functions of $\mathbf{x} = (x, y)$ with sufficient smoothness for solving the corresponding equation.

According to the standard BI method [2, 59], let $G(\mathbf{x}, \mathbf{y})$ be Green's function on the rectangle \mathcal{B} associated with the elliptic PDE (1), which satisfies for any $\mathbf{y} \in \mathcal{B}$,

$$\mathcal{L}G(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x} - \mathbf{y}), \quad \mathbf{x} \in \mathcal{B}, \quad (3)$$

$$G(\mathbf{x}, \mathbf{y}) = 0, \quad \mathbf{x} \in \partial\mathcal{B}, \quad (4)$$

where $\delta(\mathbf{x} - \mathbf{y})$ is the Dirac delta function. Let $\mathbf{n}_{\mathbf{y}}$ be the unit outward normal vector at point $\mathbf{y} \in \Gamma$, ϕ be a function defined on the boundary Γ and F be a function defined on Ω , we firstly

define the double layer boundary integral and volume integral by 21

$$(W\phi)(\mathbf{x}) := \int_{\Gamma} \frac{\partial G(\mathbf{y}, \mathbf{x})}{\partial \mathbf{n}_{\mathbf{y}}} \phi(\mathbf{y}) ds_{\mathbf{y}}, \quad \text{for } \mathbf{x} \in \Omega \cup \Omega^c, \quad (5)$$

$$(YF)(\mathbf{x}) := \int_{\Omega} G(\mathbf{y}, \mathbf{x}) F(\mathbf{y}) d\mathbf{y}, \quad \text{for } \mathbf{x} \in \Omega. \quad (6)$$

Thanks to the symbols and properties of the involved potential and volume integral, the boundary integral equation(BIE) about the density function φ for (1)-(2) can be reformulated as a Fredholm boundary integral equation of the second kind [17, 45] 23

$$\frac{1}{2}\varphi(\mathbf{x}) + (W\varphi)(\mathbf{x}) + (Yf)(\mathbf{x}) = g_D(\mathbf{x}), \quad \mathbf{x} \in \Gamma. \quad (7)$$

The solution $u(\mathbf{x})$ to the Dirichlet BVP (1)-(2) is given by 25

$$u(\mathbf{x}) = (W\varphi)(\mathbf{x}) + (Yf)(\mathbf{x}), \quad \mathbf{x} \in \Omega, \quad (8)$$

where φ is the solution of equation (7). 27

From (7), it is convenient to denote 28

$$\widetilde{W}(\varphi)(\mathbf{x}) := \left(\frac{1}{2}\mathcal{I} + W\right)(\varphi)(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \quad (9)$$

where \mathcal{I} is the identity operator. 30

In the KFBI method, the boundary integral equation (7) can be solved by the Richardson iteration numerically. Given any initial guess $\varphi_0(\mathbf{x})$, for $k = 0, 1, 2, \dots$, perform the following steps until convergence (within a predefined tolerance) 31

$$u_k^+(\mathbf{x}_m) = \widetilde{W}\varphi_k(\mathbf{x}_m), \quad m = 1, 2, \dots, M, \quad (10)$$

$$\varphi_{k+1}(\mathbf{x}_m) = \varphi_k(\mathbf{x}_m) + 2\gamma[\widetilde{g}_D(\mathbf{x}_m) - u_k^+(\mathbf{x}_m)], \quad m = 1, 2, \dots, M, \quad (11)$$

where $\{\mathbf{x}_i\}_{i=1}^M$ are discrete points on boundary Γ and $\widetilde{g}_D(\mathbf{x}) := g_D(\mathbf{x}) - (Yf)(\mathbf{x})$ need only calculate once before Richardson iteration. Note that equation (10) is from potential theory about double potential [28] instead of an arbitrary definition. It can be shown by considering the spectral radius of \widetilde{W} that the Richardson iteration is convergent if $\gamma \in (0, 1)$. The superscript '+' in the BIE means one-sided limit from the domain Ω . More specifically, let $w(\mathbf{x})$ be an arbitrary piecewise smooth function with discontinuities only existing at the interface Γ . We denote

$$w^+(\mathbf{x}) = \lim_{\mathbf{z} \rightarrow \mathbf{x}, \mathbf{z} \in \Omega} w(\mathbf{z}). \quad (12)$$

Similarly, the restriction of $w(\mathbf{x})$ in Ω^c , $w^-(\mathbf{x})$ is defined as 35

$$w^-(\mathbf{x}) = \lim_{\mathbf{z} \rightarrow \mathbf{x}, \mathbf{z} \in \Omega^c} w(\mathbf{z}). \quad (13)$$

Once the unknown density function $\varphi(\mathbf{x})$ is obtained when the iteration (11) converges. The unknown function $u(\mathbf{x})$ can be calculated according to the formula (8). 37

The remaining task involves determining an efficient approach for computing volume integrals Yf and boundary integrals $W\varphi$ to achieve the equations (10) during iteration. The KFBI method offers a methodology that eliminates the need for an explicit expression of the Green's function. This involves transforming computation of volume and boundary integrals into solution of corresponding interface PDEs whose details are shown in Appendix A. 38

3 Methodology for Hybrid KFBI Method 39

This section introduces the hybrid KFBI method, which combines traditional KFBI methods detailed in Appendix A with deep learning techniques, emphasizing efficient solutions for multiple instances of the same type of PDEs. A specialized class of networks designed to learn the solution operator for boundary integral equations will be integrated into the KFBI framework for PDE resolution. Attention is restricted to PDEs subject to Dirichlet boundary conditions. However, it is noted that Neumann boundary conditions are addressed within a similar framework, which will not be extensively discussed here.

3.1 Solution Operator for Integral Equations 41

In the KFBI method, the boundary integral equation (7) is resolved iteratively, utilizing methods like Richardson iteration. Given that equation (7) remains non-singular for a general symmetric positive definite diffusion tensor σ and non-negative reaction coefficient k [52], the solution operator $\mathcal{S}_{\mathcal{L},\Omega}$ can be defined as mapping the modified boundary value $\widetilde{g_D}$ to the density φ . The encapsulation of the right-hand side f of equation (1) and the boundary condition g_D into $\widetilde{g_D}$ enables the learning of $\mathcal{S}_{\mathcal{L},\Omega}$, thereby facilitating the resolution of a wide range of equations with variable f and assorted boundary conditions.

Remark 1. For fixed elliptic operator \mathcal{L} and domain Ω satisfying the requirement of the KFBI method, the operator $\mathcal{S}_{\mathcal{L},\Omega}$ between the function spaces defined above is linear.

3.2 Operator Learning 44

A natural and appealing wish is to obtain the explicit or computationally convenient form of the operator $\mathcal{S}_{\mathcal{L},\Omega}$, which was nearly impossible in the past. With advanced deep learning and related technologies, models based on deep neural networks now offer the potential to realize this vision. In other words, we can use neural networks to approximate this operator and aim to obtain the best network parameters through training on data for the most effective approximation.

We provide a detailed exposition of the methodology tailored for $\Omega \in \mathbb{R}^2$, noting that analogous approaches apply to higher-dimensional contexts. Recall that in the standard KFBI method, we can discretize the larger rectangle \mathcal{B} using a rectangular grid with cell numbers $I \times J$, for which $\mathcal{B} \supset \Omega$ and $\partial\Omega \cap \partial\mathcal{B} = \emptyset$. The discretization of the density φ is achieved through periodic cubic splines, facilitating the straightforward and efficient computation of density derivatives. More precisely, we presuppose the existence of M quasi-uniformly spaced nodes on the boundary $\partial\Omega$. Note the meanings of I , J , and M will be consistently employed throughout the subsequent sections of this paper. To better integrate with the standard KFBI method, a natural and effective approach for operator learning is to set the input and output of the neural network to be the values of $\widetilde{g_D}$ and φ at the M points on the closed curve $\partial\Omega$, respectively, when using network $\mathcal{N}_{\mathcal{L},\Omega}$ to approximate the operator $\mathcal{S}_{\mathcal{L},\Omega}$. In addition, we facilitate the utilization of the standard KFBI method for the generation of multiple datasets (which will be elaborated upon in detail in subsection 3.2.2), expediently employed in the training of the neural network.

If the neural network $\mathcal{N}_{\mathcal{L},\Omega}$ can be successfully trained and possesses high approximation accuracy and strong generalization, it will significantly aid in solving equations $\mathcal{L}u = f$ in the fixed domain Ω with free inhomogeneous term f and Dirichlet-type boundary conditions $u|_{\partial\Omega} = g^D$ by KFBI method (the role of the model, or its specific application, will be discussed in detail in next section 3.3). The network $\mathcal{N}_{\mathcal{L},\Omega}$ has already been demonstrated to yield satisfactory results in subsequent numerical examples.

However, a more ambitious idea is to explore incorporating entire or partial boundary in-formation yielded by boundary curves into the network, enabling it to operate beyond the constraints of solving equations solely within a fixed region. Considering the difficulties posed by arbitrarily closed curves are incalculable, a more common approach is considering a class of parametric curves. Specifically, $\mathcal{S}_{\mathcal{L}}$ can be considered as a solution operator for equation $\frac{1}{2}\varphi + \mathcal{K}\varphi = \widetilde{g}_D$ (caused by the fixed operator \mathcal{L}) on a class of $\partial\Omega$, which is mapping $(\partial\Omega, \widetilde{g}_D)$ to the density φ . Thus, we aspire for a neural network $\mathcal{N}_{\mathcal{L}}$ that can simultaneously intake boundary position information of $\partial\Omega$ and values about the function \widetilde{g}_D , yielding output values about density φ . Denote by $\{\mathbf{x}_i\}_{i=1}^M$ the point set in the KFBI method which are used to discrete the boundary $\partial\Omega$. Recall that in the previous network $\mathcal{N}_{\mathcal{L},\Omega}$, we configured the input of this network as $(\widetilde{g}_D(\mathbf{x}_1), \widetilde{g}_D(\mathbf{x}_2), \dots, \widetilde{g}_D(\mathbf{x}_M))^T$ and the output as $(\varphi(\mathbf{x}_1), \varphi(\mathbf{x}_2), \dots, \varphi(\mathbf{x}_M))^T$. Returning to the discussion about network $\mathcal{N}_{\mathcal{L}}$, the required input should be divided into two parts. One component consists of the values of the function \widetilde{g}_D , which remains consistent with the above, while the other component should incorporate spatial location information yielded by a class of boundary, such as ellipses with different major and minor axes. Here, we propose two methods for incorporating boundary position information into the network. One approach is to directly incorporate the positional coordinates into the network, that is, setting the network's input as $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M)^T$ and $(\widetilde{g}_D(\mathbf{x}_1), \widetilde{g}_D(\mathbf{x}_2), \dots, \widetilde{g}_D(\mathbf{x}_M))^T$, where $\mathbf{x}_i = (x_{i,1}, x_{i,2})^T$ is the coordinate for all $i = 1, 2, \dots, N$. Another approach involves adding only the parameters of the boundary curve to the network's input. For instance, the ellipses with parameters of axial lengths r_a, r_b can be defined as $\partial\Omega_{\{r_a, r_b\}} := \{(x, y) : \frac{(x-x_0)^2}{r_a^2} + \frac{(y-y_0)^2}{r_b^2} = 1\}$, here x_0 and y_0 are constant numbers. Then the input of the network $\mathcal{N}_{\mathcal{L}}$ can be designed as $(r_a, r_b)^T$ and $(\widetilde{g}_D(\mathbf{x}_1), \widetilde{g}_D(\mathbf{x}_2), \dots, \widetilde{g}_D(\mathbf{x}_M))^T$. While it is true that various methods can be suggested, these two approaches have been observed to be effective and computationally efficient in subsequent numerical experiments, especially the first approach that was able to include more information.

The parametric PDEs that we aim to investigate not only manifests in the ability to represent the domain boundaries parametrically but also occasionally entails solving equations induced by a parametric operator \mathcal{L} , such as the Helmholtz equations and screened Poisson equations. In this case, we need a neural network \mathcal{N}_{Ω} to approximate the solution operator \mathcal{S}_{Ω} , which is defined as mapping $(\mathcal{L}, \widetilde{g}_D)$ to the density φ for a class of parameterized elliptic operator \mathcal{L} . Similar to the preceding discussion, we can incorporate the parameters of operator \mathcal{L} into the network's input. Specifically, if the operator \mathcal{L} has parameters k_1, k_2, \dots, k_n , we can set the input of networks \mathcal{N}_{Ω} as $(k_1, k_2, \dots, k_n)^T$ and $(\widetilde{g}_D(\mathbf{x}_1), \widetilde{g}_D(\mathbf{x}_2), \dots, \widetilde{g}_D(\mathbf{x}_M))^T$.

Remark 2. Our ultimate goal is to incorporate both the parameters of the equation operator \mathcal{L} and the parameters of the boundary $\partial\Omega$ into the network \mathcal{N} , which is used to approximate the solution operator \mathcal{S} of equation $\frac{1}{2}\varphi + \mathcal{K}\varphi = \widetilde{g}_D$ caused by the parametric operator \mathcal{L} and on a class of parametric $\partial\Omega$. Incorporating the operator's parameters \mathcal{L} and the boundary $\partial\Omega$ parameters into the network does not pose new technical challenges. However, concerns arise regarding the difficulties in generating sufficient data and potential issues related to the increased size of network parameters that might result in slower inference speeds. The subsequent numerical experimentation section will elucidate the attempts we made, while this could become a significant topic for consideration in our future exploration.

3.2.1 Network Architectures

Networks without Parameter Component The specific structure of the network $\mathcal{N}_{\mathcal{L},\Omega}$ which receives $(\widetilde{g}_D(\mathbf{x}_1), \dots, \widetilde{g}_D(\mathbf{x}_M))^T$ as input and outputs $(\varphi(\mathbf{x}_1), \dots, \varphi(\mathbf{x}_M))^T$, is outlined in this section. Note that the network must possess a linear structure by remark 1. A fully con-

nected neural network, comprising only input and output layers without or with linear activation functions, is typically utilized. A network directly mapping input to output with M^2 parameters is considered suitable. However, for large M values (e.g., $M = 1024$ when discretizing the boundary $\partial\Omega$ with 1024 points), the number of network parameters can be substantial. To address this, a multi-layered network structure is recommended for large M values. This involves mapping the M -dimensional input to a hidden layer of $M//d$ dimensions (where d is a constant such as 2, 4, or 8, and the operator $//$ signifies floor division), then mapping to another hidden layer of $M//d$ dimensions, before reaching the M -dimensional output. The total parameters in this network are calculated as $2 \times M \times (M//d) + (M//d)^2$, significantly less than M^2 for an appropriately chosen d .

Networks with Parameter Component In this part, the parameter component is incorporated into the input of the network $\mathcal{N}_{\mathcal{L}}$, \mathcal{N}_{Ω} or \mathcal{N} . As outlined in the previous subsection, this input component may extra include positional coordinates of points on the domain boundary $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M)^T$, boundary parameters such as r_a, r_b , or operator parameters like k_1, k_2, \dots, k_n . It is noted that if the quantity of parameters is insufficient, an additional preprocessing layer is introduced. This layer, a fully connected network (with or without an activation function), maps a limited number of initial input parameters to a broader set. All parameters from PDEs (after the preprocessing if it is necessary) are collectively denoted as p_1, p_2, \dots, p_N for clarity in the network structure description. Each p_i can be a scalar or a coordinate, with N representing the total number of parameters, including those from any preprocessing layer. Precise determination of these details based on the input and output shapes is crucial when constructing a specific network.

A structure combining convolutional neural networks (without pooling layers) and fully connected networks is employed to handle the parameter component input $(p_1, p_2, \dots, p_N)^T$. The aim is to produce an intermediate element, denoted as I_1 , matching the dimensions of the input $(\widetilde{g_D}(\mathbf{x}_1), \dots, \widetilde{g_D}(\mathbf{x}_M))^T$. Simultaneously, a linear transformation, devoid of nonlinear activation functions and maintaining the shape for $(\widetilde{g_D}(\mathbf{x}_1), \dots, \widetilde{g_D}(\mathbf{x}_M))^T$, yields another intermediate element, denoted as I_2 . Element-wise multiplication is then applied between these two intermediate elements. Following this, a linear structure similar to the previous section is applied to the post-multiplication result. Note that nonlinear operations, such as the ReLU activation function, acting only on the component of parameter input, this architectural design ensures input linearity for function values when input parameters remain constant while effectively utilizing the information in the parameter component of the input. The subsequent figure 2 illustrates the specific network architecture and note that the network architecture is partly inspired by DeepONet [38].

Remark 3. The ‘parameter’ mentioned in the preceding paragraph refers to the neural network’s parameter input, denoted as $(p_1, p_2, \dots, p_N)^T$, rather than the model parameters of the neural network itself. Please avoid any confusion between them.

3.2.2 Strategies for Generating Training Data

Initially, with fixed \mathcal{L} and Ω , focus is placed on data generation for training the operator $\mathcal{N}_{\mathcal{L}, \Omega}$. For fixed operator \mathcal{L} and domain Ω , an exact solution to equation (1) can be constructed, and the KFBI method can be executed based on the inhomogeneous term f and boundary condition g_D derived from this exact solution. This process yields a pair $(\widetilde{g_D}, \varphi)$, assuming the KFBI method’s minor error is negligible. Specifically, pairs $(\widetilde{g_D}(\mathbf{x}_1), \dots, \widetilde{g_D}(\mathbf{x}_M))^T$ and $(\varphi(\mathbf{x}_1), \dots, \varphi(\mathbf{x}_M))^T$ are obtained from each exact solution. To train the linear neural network $\mathcal{N}_{\mathcal{L}, \Omega}$ effectively, more than

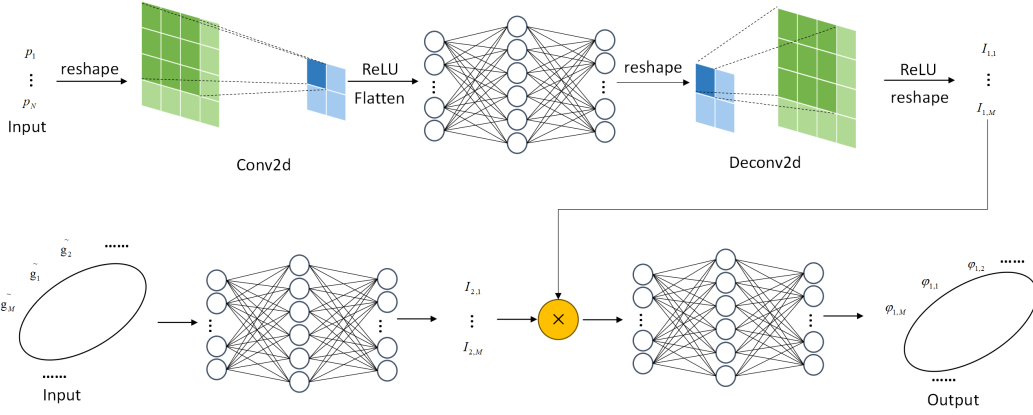


Figure 2: Network architecture schematic for networks with parameter component. In this graph, $(I_{1,1}, I_{1,2}, \dots, I_{1,M})^T = I_1$ and $(I_{2,1}, I_{2,2}, \dots, I_{2,M})^T = I_2$ are used to present the intermediate elements, \tilde{g}_i is used to present $\tilde{g}_D(\mathbf{x}_i)$ and φ_i is used to present $\varphi(\mathbf{x}_i)$ for all $i = 1, 2, \dots, M$. The schematic diagram of fully connected networks do not represent the actual size of the network model in practical use, they are for illustrative purposes only and these three blocks do not share their parameters. Also note that the ‘Flatten’ and ‘reshape’ operations are performed to obtain the correct shapes of tensors, enabling them to be properly passed to the next network block, and some terms in this diagram have the same meanings as their corresponding terms in the **PyTorch** library, such as ‘Conv2d’ represents 2D convolution and ‘ConvTransposed2d’ represents 2D transposed convolution.

M linearly independent vectors \tilde{g}_D should be added to the dataset. Consequently, constructing over M exact solutions for equation (1) to obtain corresponding sets of \tilde{g}_D and φ becomes necessary. Typically, M can be chosen as a power of 2, like 128, 256, 512, etc. In general, to get a more generalized model, we will make the size of the data set much larger than M although a training data set with M linearly independent pairs is enough due to the linearity of real operators and the networks. For instance, several thousand pairs of \tilde{g}_D and φ are enough to train the model when $M = 128$.

For training \mathcal{N}_L , \mathcal{N}_Ω or \mathcal{N} , sufficient data sets are generated in the same manner for each parameter pair (p_1, p_2, \dots, p_N) . These parameter pairs (p_1, p_2, \dots, p_N) are uniformly sampled within the given parameter space to ensure adequate data volume.

3.2.3 Loss Function and Training Process

For fixed \mathcal{L} and Ω , training the parameters $\Theta_{\mathcal{L}, \Omega}$ of operator $\mathcal{N}_{\mathcal{L}, \Omega}$ with dataset $\{(\tilde{g}_{D,i}, \varphi_i)\}_{i=1}^n$ gained from the strategy mentioned above by using this trivial loss function:

$$\text{loss}(\Theta_{\mathcal{L}, \Omega}) = \sum_{i=1}^n \|\mathcal{N}_{\mathcal{L}, \Omega}(\tilde{g}_{D,i}; \Theta_{\mathcal{L}, \Omega}) - \varphi_i\|_2^2, \quad (14)$$

where $\tilde{g}_{D,i} = (\tilde{g}_{D,i}(\mathbf{x}_1), \tilde{g}_{D,i}(\mathbf{x}_2), \dots, \tilde{g}_{D,i}(\mathbf{x}_M))^T$, $\varphi_i = (\varphi_i(\mathbf{x}_1), \varphi_i(\mathbf{x}_2), \dots, \varphi_i(\mathbf{x}_M))^T$ are the i -th pair of data and $\|\cdot\|_2$ is used to represent the 2-norm of vectors. For training \mathcal{N}_L , \mathcal{N}_Ω or \mathcal{N} , similar loss functions are utilized.

The dataset derived from the strategy above is divided into two parts for training: an 80%

training set and a 20% test set through a random split. This division aims to mitigate overfitting and underfitting risks. The training process uses the **PyTorch** deep learning framework, with **Adam** as the optimizer. An initial learning rate of 0.001 is set, and a learning rate scheduler with reduced rate 0.6 is incorporated to dynamically adjust the learning rate if the loss exceeds 1500 times without a decrease. When no further reduction in the loss function is observed, typically after not exceeding 500,000 or 800,000 epochs, the training terminates, at which point the value of the loss function is generally within the magnitude of $1\text{E-}5$ in our experiments. This training is executed on a single **NVIDIA GeForce RTX 3080** graphics card utilizing the **CUDA** platform.

3.3 Hybrid KFBI Method Based on Trained Models

This section introduces the significant role offered by the trained neural networks $\mathcal{N}_{\mathcal{L},\Omega}$ (or $\mathcal{N}_{\mathcal{L}}, \mathcal{N}_{\Omega}, \mathcal{N}$). The networks are designed to infer the density function φ . Consequently, the inference results from these neural networks can be directly utilized as density functions φ , which are then integrated into the formula (8). Solving the corresponding interface problem (21) yields the solution to the PDE (1). Alternatively, the output φ from the neural networks can be used as the initial value in the iterative KFBI method, reducing the number of iterations and saving time. These approaches are subsequently referred to as Strategies 1 and 2, respectively. Regarding the utilization of neural network inference results as initial values, there exists lots of related works, each demonstrating commendable efficacy within their respective applications, such as [39].

Remark 4. • *The two methods above have been observed to significantly reduce iteration count and time compared to the standard KFBI method. Being quite radical, the first method effectively reduces the KFBI method's iteration count to one, thereby considerably decreasing the running time, though at some cost to precision (relative to the KFBI method). However, as will be demonstrated in subsequent numerical experiments, this approach still achieves reasonably good numerical accuracy. The second method offers a balanced approach, acting as a preprocessing solution for setting the initial value in the KFBI iterative process. This method reduces a portion of the iteration count without compromising precision (relative to the KFBI method). Both methods are proven feasible and valuable through the upcoming numerical experiments. Additionally, due to the relatively simple architecture of the neural network, the time required for a single inference is negligible.*

- *The operator learning approach based on the KFBI method, as proposed in this work, effectively reduces the problem's dimensionality. When addressing equation (1) for $d=2$, the input and output of the operator $\mathcal{S}_{\mathcal{L},\Omega}$ (or $\mathcal{S}_{\mathcal{L}}, \mathcal{S}_{\Omega}$), approximated by the neural network $\mathcal{N}_{\mathcal{L},\Omega}$ (or $\mathcal{N}_{\mathcal{L}}, \mathcal{N}_{\Omega}$), are functions on a one-dimensional manifold, and thus are naturally represented by vectors with M components. This aspect of the method significantly aids in solving elliptic PDEs on Ω by learning mappings between functions defined on $\partial\Omega$, highlighting a significant advantage of the approach.*
- *The training of the neural network $\mathcal{N}_{\mathcal{L},\Omega}$ enables the solution of a wide range of equations (1), where both the non-homogeneous term f and boundary conditions g_D can vary. The applicability of neural networks $\mathcal{N}_{\mathcal{L}}, \mathcal{N}_{\Omega}$ and \mathcal{N} is further extended, demonstrating their powerful range, which stands as another distinctive feature of this method. These models are pivotal in solving PDEs, as evidenced in Section 4.*

- The proposed method can be synergistically combined with other techniques, such as multi-grid solvers and GPU acceleration, to enhance solution efficiency. 65

4 Numerical Experiments 66

This section presents numerical results for various classes of equations, including Laplace, Poisson, Stokes, modified Helmholtz, and Naiver equations. The focus is on the application of the hybrid KFBI method, detailing its operational mechanism and demonstrating its superiority in terms of numerical accuracy and iteration count. Since the neural network’s structure, loss function, and training process have been elaborated in the preceding section, emphasis here is placed on post-training neural network applications. Excluding the training phase, programming in C and C++ is utilized, with the libtorch library on the C++ platform facilitating network inference computations. The examples in section 4.1.1 are tested on a machine with a CPU ‘12th Gen Intel(R) Core(TM)i5-12600KF 3.70 GHZ’ while the other examples are tested on a machine with a CPU ‘11th Gen Inter(R) Core(TM) i5-1135G7 @ 2.40GHz’. 67

For the KFBI method implementations, the Richardson iteration is employed as equation (11) with $\gamma = 0.75$, setting the initial value to $\varphi_0 = 2g_D$ and the relative tolerance to $1E-8$, in line with research [52]. Unless otherwise specified, the parameters regarding the iteration stages mentioned above will be applied in each numerical experiment. Alphanumeric symbol definitions, such as I , J , and M , align with those in the second paragraph of section 3.2. The numerical solution error is calculated by evaluating the norm of the difference between the numerical and exact solutions at all grid points in domain Ω , with the error typically referring to absolute error unless stated otherwise. 68

4.1 Hybrid KFBI without Parameter Component 69

4.1.1 Laplace Equations 70

We focus on the post-training operations of the neural network model, excluding the training process, and this principle is consistently applied thereafter. In this context, the Laplace equation is considered with $\mathcal{L} = \Delta$ and the fixed homogeneous term $f = 0$. The boundary $\partial\Omega$ is defined by the set $\{(x, y) : x = c_x + r_a \cos(\alpha) \cos(t) - r_b \sin(\alpha) \sin(t), y = c_y + r_a \sin(\alpha) \cos(t) - r_b \cos(\alpha) \sin(t) \text{ for } t \in [0, 2\pi)\}$, where the ellipse, characterized as rotated, is defined by $c_x = 0.2, c_y = 0.4, r_a = 1.0, r_b = 0.5$, and $\alpha = \frac{\pi}{4}$. 71 The bounding box \mathcal{B} for the interface problem is established as $\mathcal{B} = [-1.2, 1.2] \times [-1.2, 1.2]$. The number of discrete points along the boundary curve M is determined as $M = \max\{I, J\}$. To demonstrate the network’s generalization capability, the exact solutions chosen for testing in the following sections were not included in the training data, a principle consistently upheld in subsequent examples. 71

We choose the boundary conditions given by different exact solutions to solve the Laplace equation. Table 1 displays the results obtained for Laplace Equation 1, where the exact solution $u(x, y) = \exp(x) \cos(y) + \exp(y) \sin(x)$ is applied within the elliptic domain. 72

Remark 5. In table 1, the second-row records results where the model’s inference outputs are directly employed as the density function for solving the corresponding interface problem, aligning with Strategy 1 detailed in section 3.3. The last seven rows of the table compare the outcomes of the standard KFBI method with those achieved by initiating the KFBI iterative process using the model’s inference results, a method consistent with Strategy 2 in the same section. It is noted that the precision of the numerical solutions obtained through this approach closely parallels that 74

| Grid size ($I \times J$) | 128×128 | 256×256 | 512×512 | 1024×1024 |
|------------------------------------|------------------|------------------|------------------|--------------------|
| L_∞ error for Strategy 1 | 5.4E-3 | 2.2E-3 | 1.3E-3 | 1.7E-3 |
| L_∞ error for standard KFBI | 4.0E-5 | 1.4E-5 | 1.7E-6 | 1.4E-7 |
| L_∞ error for Strategy 2 | 4.0E-5 | 1.4E-5 | 1.7E-6 | 1.4E-7 |
| Iterations (standard KFBI) | 26 | 26 | 26 | 26 |
| Iterations (Strategy 2) | 11 | 12 | 12 | 11 |
| Running time of standard KFBI (s) | 0.04154 | 0.1120 | 0.4010 | 1.513 |
| Running time of Strategy 2 (s) | 0.02233 | 0.05463 | 0.1997 | 0.8390 |
| Time saved (Strategy 2) | 46% | 51% | 50% | 45% |

Table 1: Result of Laplace equation 1: comparison of accuracy and efficiency in solving Laplace equation 1 using different methods.

of the standard KFBI method, attributable to the identical relative tolerance set for the iterative processes. The subsequent formatting of the tables in section 4.1 will remain consistent.

Table 2 displays the results for Laplace Equation 2, characterized by the exact solution $u(x, y) = \sin(3x) \sinh(3y) + 0.5 \cosh(x) \cos(y)$, applied within the elliptic domain.

| Grid size ($I \times J$) | 128×128 | 256×256 | 512×512 | 1024×1024 |
|------------------------------------|------------------|------------------|------------------|--------------------|
| L_∞ error for Strategy 1 | 2.7E-3 | 1.4E-3 | 7.0E-4 | 8.6E-4 |
| L_∞ error for standard KFBI | 2.0E-4 | 5.2E-5 | 1.3E-5 | 3.3E-6 |
| L_∞ error for Strategy 2 | 2.0E-4 | 5.2E-5 | 1.3E-5 | 3.3E-6 |
| Iterations (standard KFBI) | 26 | 26 | 26 | 26 |
| Iterations (Strategy 2) | 11 | 10 | 12 | 9 |
| Running time of standard KFBI (s) | 0.04230 | 0.1144 | 0.4040 | 1.545 |
| Running time of Strategy 2 (s) | 0.02296 | 0.05020 | 0.2023 | 0.7450 |
| Time saved (Strategy 2) | 46% | 56% | 50% | 52% |

Table 2: Result of Laplace equation 2: comparison of accuracy and efficiency in solving the Laplace equation 3 using different methods.

Results for Laplace Equation 3, featuring the exact solution $u(x, y) = \sin(2.5x) \sinh(2.5y)$ and applied to the elliptic domain, are detailed in Table 3.

The experimental data reveal that adopting Strategy 1 enables the accurate resolution of the corresponding Laplace equation with the model's assistance, resulting in a substantial reduction in the time required for solving PDEs, approximately 10% of that required by the standard KFBI method. Moreover, implementing Strategy 2 significantly reduces iterative time (by about 50%) while maintaining precision comparable to the standard KFBI method.

| | | | | |
|------------------------------------|------------------|------------------|------------------|--------------------|
| Grid size ($I \times J$) | 128×128 | 256×256 | 512×512 | 1024×1024 |
| L_∞ error for Strategy 1 | 2.0E-3 | 9.6E-4 | 6.0E-4 | 1.0E-3 |
| L_∞ error for standard KFBI | 9.1E-5 | 3.7E-5 | 4.7E-6 | 1.2E-6 |
| L_∞ error for Strategy 2 | 9.1E-5 | 3.7E-5 | 4.7E-6 | 1.2E-6 |
| Iterations (standard KFBI) | 25 | 25 | 25 | 25 |
| Iterations (Strategy 2) | 11 | 11 | 14 | 12 |
| Running time of standard KFBI (s) | 0.04061 | 0.1099 | 0.3988 | 1.510 |
| Running time of Strategy 2 (s) | 0.02237 | 0.05335 | 0.2289 | 0.8460 |
| Time saved (Strategy 2) | 45% | 51% | 43% | 44% |

Table 3: Result of Laplace equation 3: comparison of accuracy and efficiency in solving Laplace equation 4 using different methods.

4.1.2 Two-Dimensional Stokes Equations

In this section, 2D Stokes equation with Dirichlet boundary condition is defined as

$$\begin{aligned} -\Delta \mathbf{u} + \nabla p &= \mathbf{f}, & \text{in } \Omega^+ \cup \Omega^-, \\ \nabla \cdot \mathbf{u} &= 0, & \text{in } \Omega^+ \cup \Omega^-, \\ \mathbf{u} &= \mathbf{g}_D, & \text{on } \partial\Omega, \end{aligned} \quad (15)$$

In the context presented, $\mathbf{u} = (u^{(1)}, u^{(2)})^T$ and p denote the unknown fluid velocity and pressure functions, respectively, while $\mathbf{f} = (f^{(1)}, f^{(2)})^T$ represents the external force function.

The application of the KFBI method to solve the Stokes equation, as elaborated in the paper by Dong et al. [13], needs to be delved into here. This equation's solution (\mathbf{u}, p) is expressible in terms of volume and boundary integrals, involving the density function φ , which solves the corresponding boundary integral equation. The right-hand side of this boundary integral equation is denoted as \mathbf{g}_D . Terms such as $\mathcal{S}_{\mathcal{L}, \Omega}$ (or $\mathcal{S}_{\mathcal{L}}, \mathcal{S}_\Omega$) and $\mathcal{N}_{\mathcal{L}, \Omega}$ (or $\mathcal{N}_{\mathcal{L}}, \mathcal{N}_\Omega$) retain their previous meanings. Notably, \mathbf{g}_D becomes a vector function $(g_D^{(1)}, g_D^{(2)})$, and a similar transformation applies to φ . However, this does not introduce complications in the network's construction and training, as a tensor of size $(M, 2)$ can be conveniently reshaped into a vector of length $2M$. For this example, the absolute tolerance for iteration is set at $1\text{E-}8$, and the initial value in the standard KFBI is $\varphi_0 = (0, 0)^T$.

The boundary $\partial\Omega = \{(x, y) : x = c_x + r_a \cos(\alpha) \cos(t) - r_b \sin(\alpha) \sin(t) \text{ and } y = c_y + r_a \sin(\alpha) \cos(t) - r_b \cos(\alpha) \sin(t) \text{ for } t \in [0, 2\pi)\}$ with $c_x = 0, c_y = 0, r_a = 1.0, r_b = 0.6$ and $\alpha = 0$ such that Ω is an ellipse. The bounding box \mathcal{B} for the interface problem is set to be $\mathcal{B} = [-1.2, 1.2] \times [-1.2, 1.2]$. The discrete number of the boundary curve M is set as $M = \max\{I, J\}$.

Table 4 displays the results for Stokes Equation 1, using the exact solution specified in equation (16), applied to the elliptic domain.

$$\begin{cases} u^{(1)} = [(x^2 + y^2) \cos(7 \arctan(\frac{y}{x})) + 3.5(1 - (x^2 + y^2)) \cos(5 \arctan(\frac{y}{x}))](x^2 + y^2)^{\frac{5}{2}} \\ u^{(2)} = -3.5(1 - (x^2 + y^2)) \sin(5 \arctan(\frac{y}{x})) (x^2 + y^2)^{\frac{5}{2}} \\ p = -14 \cos(6 \arctan(\frac{y}{x})) (x^2 + y^2)^3 \end{cases} \quad (16)$$

| Grid size ($I \times J$) | 128×128 | 256×256 | 512×512 | 1024×1024 |
|---------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|
| L_∞ error for Strategy 1 | $1.4E-3$ $1.1E-2$ $1.3E-2$ | $1.1E-4$ $1.2E-4$ $1.3E-3$ | $2.1E-4$ $2.3E-4$ $1.9E-3$ | $1.9E-4$ $1.7E-4$ $3.6E-3$ |
| L_∞ error for KFBI | $2.5E-4$ $5.2E-4$ $3.9E-3$ | $6.3E-5$ $9.4E-5$ $6.0E-4$ | $1.3E-5$ $2.4E-5$ $1.7E-4$ | $3.1E-6$ $6.2E-6$ $4.5E-5$ |
| L_∞ error for Strategy 2 | $2.5E-4$ $5.2E-4$ $3.9E-3$ | $6.3E-5$ $9.4E-5$ $6.9E-5$ | $1.3E-5$ $2.4E-5$ $1.7E-4$ | $3.1E-6$ $6.2E-6$ $4.5E-5$ |
| Iterations (KFBI) | 20 | 20 | 20 | 20 |
| Iterations (Strategy 2) | 10 | 10 | 12 | 12 |
| Running time of KFBI (s) | 1.368 | 5.404 | 23.84 | 119.3 |
| Running time of Strategy 2 (s) | 0.7433 | 2.939 | 14.08 | 68.40 |
| Time saved (Strategy 2) | 46% | 46% | 41% | 43% |

Table 4: Result of 2D Stokes equation 1: comparison of accuracy and efficiency in solving 2D Stokes equation 1 using different methods. The error vector is computed by the numerical solution $(u_{\text{numerical}}^{(1)}, u_{\text{numerical}}^{(2)}, p_{\text{numerical}})$ and the exact solution $(u_{\text{exact}}^{(1)}, u_{\text{exact}}^{(2)}, p_{\text{exact}})$.

From this table, it can be observed that our model significantly enhances the efficiency of solving the Stokes equation. When employing Strategy 1, we achieve high-precision numerical solutions. Moreover, timing experiments indicate that the time required to solve the equation on grids of size 128×128 , 256×256 , 512×512 and 1024×1024 using Strategy 1 is merely 0.1280s, 0.3862s, 1.650s and 6.103s, respectively. This leads to a substantial reduction in computation time, approximately 90%! On the other hand, when employing Strategy 2, we can reduce the running time by almost half without compromising precision compared to the standard KFBI method.

4.2 Hybrid KFBI on Parametric PDEs 99

4.2.1 Poisson Equations on Star-Shaped Domains 100

In this section, \mathcal{L} is designated as Δ , with f being freely chosen as the inhomogeneous term of the Poisson equation. The boundary defining the star-shaped domain Ω is characterized by $\partial\Omega = \{(x, y) : x = (1.0 - S_c + S_c \cos(S_m t)) \cos(t), y = (1.0 - S_c + S_c \cos(S_m t)) \sin(t) \text{ for } t \in [0, 2\pi)\}$, where $S_m \in \{3, 4, 5, 6\}$ and $S_c \in [0.05, 0.20]$ are parameters included in the input part of networks. The bounding box \mathcal{B} for the interface problem is established as $\mathcal{B} = [-1.5, 1.5] \times [-1.5, 1.5]$. The discrete number of points along the boundary curve M is determined as $M = \max\{I, J\}$, with $I = J = 256$ in this example.

Table 5 presents the results for Poisson Equation 3 with the exact solution $u(x, y) = \exp(x) \cos(y) + \exp(y) \sin(x) + \exp(0.6x + 0.8y)$, applied to star-shaped domains varying in parameters S_m, S_c .

Remark 6. In Table 5, the second-row records results where the model's inference outputs are directly used as the density function for solving the corresponding interface problem, aligning with Strategy 1 in section 3.3. The last six rows of the table compare outcomes between the

| (S_m, S_c) | (3, 0.20) | (4, 0.10) | (5, 0.05) | (6, 0.15) |
|---|-----------|-----------|-----------|-----------|
| L_∞ error for Strategy 1 | 4.2E-3 | 5.8E-3 | 3.3E-3 | 7.8E-3 |
| L_∞ error for standard KFBI | 1.2E-5 | 1.3E-5 | 1.3E-5 | 5.0E-5 |
| L_∞ error for Strategy 2 | 1.2E-5 | 1.3E-5 | 1.3E-5 | 5.1E-5 |
| Iterations (standard KFBI) | 26 | 26 | 25 | 27 |
| Iterations (Strategy 2) | 14 | 13 | 14 | 17 |
| Network inference cost (converted to iteration counts) | 0.69 | 0.67 | 0.57 | 0.63 |
| Saved iterations | 43% | 47% | 42% | 35% |

Table 5: Result of Poisson equation 1: comparison of accuracy and efficiency in solving Poisson equation 1 using different methods.

standard KFBI method and those achieved by initiating the KFBI iterative process using the model's inference results, as outlined in Strategy 2 of the same section. Notably, the penultimate row in the table represents the conversion of network inference time into an equivalent number of iterations, offering an alternative perspective on observing the saving of iteration time. The subsequent formatting of the tables in sections 4.2 will remain consistent.

The table illustrates that, despite including parameter components, the trained model retains its accuracy and robust generalization capabilities. The error associated with Strategy 1 is consistently maintained at magnitudes of 10^{-3} , and its execution time is notably reduced compared to the standard KFBI method, equivalent to a saving of about 22.6 iterations (considering that Strategy 1 entails one network inference and a single call to the interface problem solver). In the case of Strategy 2, the model demonstrates the ability to reduce approximately 45% of the iteration steps required by the standard KFBI method without compromising precision.

4.2.2 Stokes Equations on Ellipses

In this section, Stokes Equation (15) is considered. The boundary defining the elliptic domain Ω is described by $\partial\Omega = \{(x, y) : x = r_a \cos(t), y = r_b \sin(t) \text{ for } t \in [0, 2\pi)\}$, with parameters $r_a, r_b \in [0.8, 1.2]$. The coordinates on parameterized $\partial\Omega$ appear in the networks. The bounding box \mathcal{B} for the interface problem is established as $\mathcal{B} = [-1.5, 1.5] \times [-1.5, 1.5]$. The discrete number of points along the boundary curve M is set at $M = \max\{I, J\}$, with $I = J = 128$ in this example, and the absolute tolerance for iteration is fixed at $1\text{E-}8$.

Table 6 displays the results for Stokes Equation 2, using the exact solution given by equation (17), and applied to ellipses with varying parameters r_a, r_b .

$$\begin{cases} u^{(1)} = 2xy \\ u^{(2)} = 1 - (x^2 + y^2) \\ p = -4y \end{cases} \quad (17)$$

From the data presented in this table, a conclusion akin to that drawn in section 4.2.1 is derived regarding our trained model. It is observed that employing the model's direct inference results as the exact solution for the density function (namely, Strategy 1) yields sufficiently high accuracy.

| (r_a, r_b) | (0.8, 0.8) | (1.0, 1.1) | (1.1, 1.0) | (1.1, 1.1) |
|--|---|--|--|--|
| L_∞ error for Strategy 1 73 | $\begin{bmatrix} 7.6E-4 \\ 3.5E-4 \\ 9.7E-3 \end{bmatrix}$ 87 | $\begin{bmatrix} 5.4E-4 \\ 7.7E-4 \\ 5.1E-3 \end{bmatrix}$ 87 | $\begin{bmatrix} 5.0E-4 \\ 4.8E-4 \\ 5.5E-3 \end{bmatrix}$ 87 | $\begin{bmatrix} 7.6E-4 \\ 7.2E-4 \\ 6.2E-3 \end{bmatrix}$ 87 |
| L_∞ error for KFBI 73 | $\begin{bmatrix} 5.2E-5 \\ 3.5E-5 \\ 1.3E-3 \end{bmatrix}$ 87 | $\begin{bmatrix} 3.5E-5 \\ 2.8E-5 \\ 6.9E-4 \end{bmatrix}$ 116 | $\begin{bmatrix} 2.5E-5 \\ 2.4E-5 \\ 5.5E-4 \end{bmatrix}$ 116 | $\begin{bmatrix} 2.9E-5 \\ 3.2E-5 \\ 5.5E-4 \end{bmatrix}$ 116 |
| L_∞ error for Strategy 2 73 | $\begin{bmatrix} 5.2E-5 \\ 3.5E-5 \\ 1.3E-3 \end{bmatrix}$ 87 | $\begin{bmatrix} 3.5E-5 \\ 2.8E-5 \\ 6.9E-4 \end{bmatrix}$ 116 | $\begin{bmatrix} 2.5E-5 \\ 2.4E-5 \\ 5.5E-4 \end{bmatrix}$ 116 | $\begin{bmatrix} 2.9E-5 \\ 3.2E-5 \\ 5.5E-4 \end{bmatrix}$ 116 |
| Iterations (KFBI) | 26 | 26 | 26 | 26 73 |
| Iterations (Strategy 2) | 17 | 14 | 15 | 15 105 |
| Network inference cost (converted to iteration counts) 65 | 0.19 | 0.22 | 0.21 | 0.20 71 |
| Save iterations | 34% | 45% | 41% | 42% 73 |

Table 6: Result of 2D Stokes equation 2: comparison of accuracy and efficiency in solving 2D Stokes Equation 2 using different methods. The error vector is computed by the numerical solution $(u_{\text{numerical}}^{(1)}, u_{\text{numerical}}^{(2)}, p_{\text{numerical}})$ and the exact solution $(u_{\text{exact}}^{(1)}, u_{\text{exact}}^{(2)}, p_{\text{exact}})$. 68

4.2.3 Modified Helmholtz Equations 112

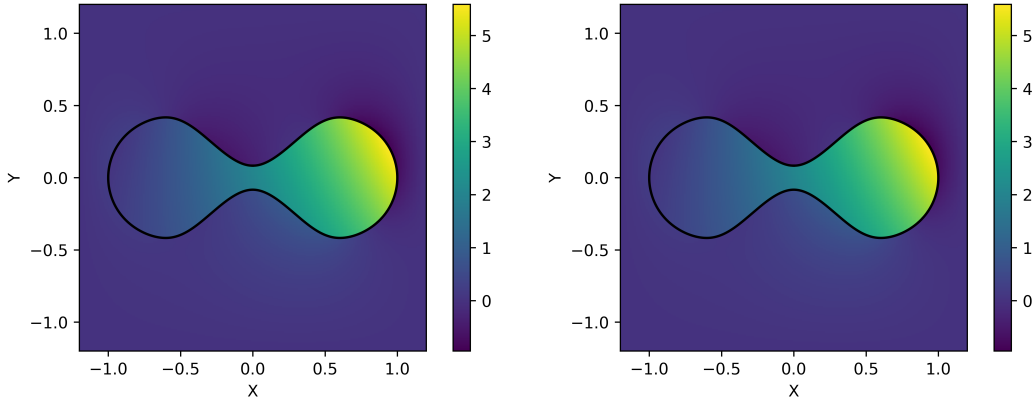
In this section, the operator \mathcal{L} is defined as $\Delta - \kappa \mathcal{I}$, where \mathcal{I} represents the identity operator, and $\kappa > 0$ serves as the coefficient for the modified Helmholtz equation. The domain Ω is designated a dumbbell-shaped area, as depicted in figure 3. Specifically, $\partial\Omega$ is a periodic C^2 cubic spline curve, passing through control points detailed in Appendix C.1. The coefficient κ , ranging between 1 and 5, is integrated into the input part of the networks. The bounding box \mathcal{B} for the interface problem is established as $\mathcal{B} = [-1.2, 1.2] \times [-1.2, 1.2]$. The number of discrete points along the boundary curve M is set at $M = 92$, with $I = J = 256$ in this example. The absolute tolerance for iteration is fixed at 1E-8, and the initial value for the standard KFBI method is set as $\varphi_0 = (0, 0)^T$. 71

Table 7 presents the results for modified Helmholtz Equation 1, featuring the exact solution $u(x, y) = \exp(x) \cos(y) + \exp(y) \sin(x) + \exp(0.6x + 0.8y)$, applied to the dumbbell-shaped domain with varying values of κ . For illustration, numerical solutions obtained through Strategy 1 and Strategy 2, using a grid size of 256×256 and a coefficient $\kappa = 3$, are depicted in figure 3. 72

| κ | 1.1 | 1.3 | 1.5 | 2 | 3 |
|--|--------|--------|--------|--------|-----------|
| L_∞ error for Strategy 1 | 1.3E-3 | 1.4E-3 | 1.3E-3 | 1.2E-3 | 1.3E-3 73 |
| L_∞ error for standard KFBI | 1.5E-4 | 1.8E-4 | 2.2E-4 | 2.8E-4 | 4.2E-4 73 |
| L_∞ error for Strategy 2 | 1.5E-4 | 1.8E-4 | 2.2E-4 | 2.8E-4 | 4.2E-4 73 |
| Iterations (standard KFBI) | 57 | 56 | 55 | 53 | 49 115 |
| Iterations (Strategy 2) | 25 | 25 | 27 | 26 | 25 115 |
| Network inference cost (converted to iteration counts) 65 | 0.18 | 0.19 | 0.16 | 0.20 | 0.19 71 |
| Save iterations | 56% | 55% | 51% | 51% | 49% 73 |

Table 7: Result of modified Helmholtz equation 1: comparison of accuracy and efficiency in solving modified Helmholtz equation 1 using different methods. 67

The table demonstrates that the model trained in this example exhibits exceptional and consistent performance. For varying κ values, Strategy 1 enables the achievement of satisfactory precision. Furthermore, the application of Strategy 2 significantly reduces the iteration time of the KFBI method. Notably, due to the domain's shape in this example, the iteration count for the standard KFBI method is considerably higher than in previous cases. However, the models outperform through both Strategy 1 and Strategy 2. Particularly with Strategy 1, corresponding to only about 1.2 iterations (involving one network inference and a single call to the interface problem solver), the method saves over 97% of iteration time compared to the standard KFBI method, which requires over 50 iterations.



Numerical solution by Strategy 1. Numerical solution by Strategy 2.

Figure 3: Numerical solutions of Modified Helmholtz equation 1 given by Strategy 1 and Strategy 2 in grid 256×256 . Note that the figures show the results given by solving the corresponding interface problem in KFBI whose interior solution is the desired results for the original PDE.

4.2.4 Modified Helmholtz Equations on Parametric Domains

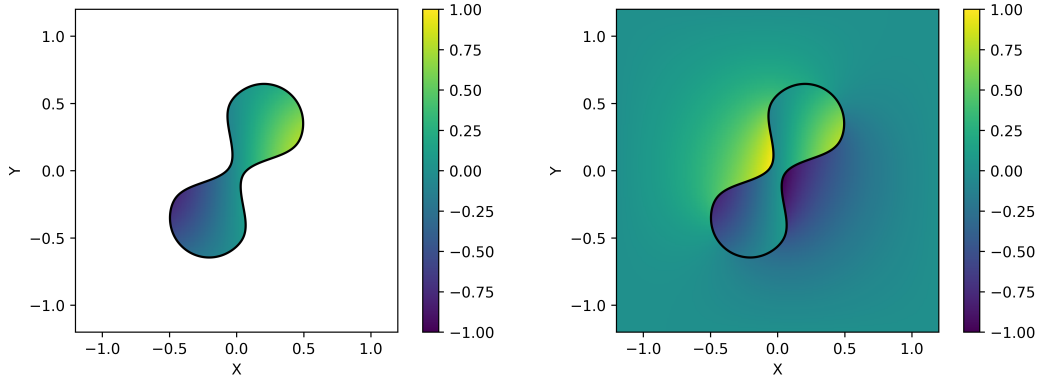
In this section, we extend the framework presented in 4.2.3 by incorporating diversity in the computational domain. Specifically, beyond varying the parameter κ within the range $[0.5, 3.0]$, the computational domain Ω can also undergo transformations involving rotation by an angle α and scaling by a factor r relative to the original domain Ω_0 . Here, the boundary of domain Ω_0 is similarly obtained through cubic spline interpolation of a set of control points (which is recorded in Appendix C.1 for details), and $\alpha \in [0, 2\pi]$, $r \in [0.6, 1]$. In this case, both κ and the coordinates of $\partial\Omega$ along with the values of $\widehat{g_D}$ at M discrete points are the input of the network model. The number of discrete points along the boundary curve M is set at $M = 46$, with $I = J = 128$ in this example. The absolute tolerance for iteration is fixed at $1\text{E-}6$, and the initial value for the standard KFBI method is set as $\varphi_0 = (0, 0)^T$. Other details correspond to the experiments in 4.2.3.

Table 8 presents the results for modified Helmholtz Equation 2, featuring the exact solution $u(x, y) = \sin(2.1x) \cos(1.9y)$, applied to the transformed dumbbell-shaped domain with varying values α , r and κ . For illustration, numerical solutions obtained through Strategy 1 and Strategy 2, using a grid size of 128×128 and coefficients $\alpha = \frac{\pi}{3}$, $r = 0.7$ and $\kappa = 2.8$, are depicted in figure 4.

From the above results, it is evident that our model can effectively handle parametric PDEs.

| (α, r, κ) | $(\frac{\pi}{3}, 0.7, 2.8)$ | $(\frac{\pi}{4}, 0.7, 2.8)$ | $(\frac{\pi}{6}, 0.71, 0.9)$ | $(\frac{\pi}{5}, 0.68, 1.3)$ |
|---|-----------------------------|-----------------------------|------------------------------|------------------------------|
| L_2 error for Strategy 1 | 1.1E-3 | 1.5E-3 | 4.2E-3 | 1.6E-3 |
| L_2 error for standard KFBI | 4.4E-5 | 4.8E-5 | 1.7E-5 | 2.2E-5 |
| L_2 error for Strategy 2 | 4.4E-5 | 4.8E-5 | 1.7E-5 | 2.2E-5 |
| Iterations (standard KFBI) | 50 | 49 | 54 | 54 |
| Iterations (Strategy 2) | 28 | 26 | 32 | 32 |
| Network inference cost (converted to iteration counts) | 0.86 | 0.81 | 0.83 | 0.79 |
| Save iterations | 42% | 45% | 39% | 39% |

Table 8: Result of modified Helmholtz equation 2: comparison of accuracy and efficiency in solving modified Helmholtz equation 2 using different methods.



Exact solution. Numerical solution by Strategy 1.

Figure 4: Exact solution and Numerical solution of Modified Helmholtz equation 2 given by Strategy 1 in grid 128×128 . Note that the right figure show the result given by solving the corresponding interface problem in KFBI whose interior solution is the desired result for the original PDE.

where not only the parameters of the equation can vary but also systematic variations can be applied to the computational domain. Figure 4 above further demonstrates that our Strategy 1 exhibits remarkable accuracy, while its computational speed is astonishing, consuming less than 3% of the time required by the traditional KFBI method.

4.2.5 Naiver Equations

In this section, Naiver equation with Dirichlet boundary condition is defined as

$$\begin{aligned} \nabla \cdot \boldsymbol{\sigma}(\mathbf{u}) + \mathbf{f} &= \mathbf{0}, \quad \text{in } \Omega, \\ \mathbf{u} &= \mathbf{g}_D, \quad \text{on } \partial\Omega, \end{aligned} \quad (18)$$

here $\mathbf{u} = (u^{(1)}, u^{(2)})^T$, $\mathbf{f} = (f^{(1)}, f^{(2)})^T$ represent displacement variable and external force, respectively. Stress tensor is

$$\boldsymbol{\sigma}(\mathbf{u}) = \lambda \nabla \cdot \mathbf{u} \mathbf{I} + 2\mu \boldsymbol{\epsilon}(\mathbf{u}), \quad (19)$$

here $\epsilon(\mathbf{u}) = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$ is the linear strain tensor, \mathbf{I} is the 2×2 identity matrix. Lamé coefficients λ and μ is given by Young's modulus E and Poisson's ratio ν :

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}.$$

By inserting equation (19) into equation (18), we can obtain this PDE with unknown \mathbf{u}

$$-\mu \Delta \mathbf{u} - (\lambda + \mu) \nabla (\nabla \cdot \mathbf{u}) = \mathbf{f}, \quad \text{in } \Omega. \quad (20)$$

The solution of the Naiver equation using the KFBI method, as detailed in the paper by Zhao et al. [63], needs not to be elaborated upon here; readers interested in its implementation may consult the referenced paper. It is important to note that the solution \mathbf{u} to this equation can be expressed through volume and boundary integrals, with the density function φ resolving the corresponding boundary integral equation. In this work, the right-hand side of this boundary integral equation is continued to be denoted as $\mathbf{g}_{\widetilde{\mathbf{D}}}$, derived from both $\mathbf{g}_{\mathbf{D}}$ and \mathbf{f} . Terms such as $S_{\mathcal{L},\Omega}$ and $N_{\mathcal{L},\Omega}$ retain their previously established meanings.

This section defines the domain Ω as a heart-shaped area, as illustrated in figure 5. $\partial\Omega$ is a periodic C^2 cubic spline curve, traversing control points detailed in Appendix C.2. The parameters $E \in [10^8, 10^9]$ and $\nu \in [0.35, 0.45]$ are integrated into the input part of networks, covering the typical range of Young's modulus and Poisson's ratio for most plastics. The bounding box \mathcal{B} for the interface problem is established as $\mathcal{B} = [-1.2, 1.2] \times [-1.2, 1.2]$. The number of discrete points on the boundary curve M is set at $M = \max\{I, J\}$ with $I = J = 256$. Absolute tolerance for iteration is determined to be 1E-8, and the initial value for the standard KFBI method is set as $\phi_0 = (0, 0)^T$.

Remark 7. • In practice, $\log_{10} E$ is used in the networks instead of E .

Table 9 presents the results for Naiver Equation 1, featuring exact solution $u^{(1)} = \sin x \cos y + xy$ and $u^{(2)} = \cos x \sin y + xy$, applied to the heart-shaped domain with various (E, ν) parameter values. As an example, the numerical solution obtained through Strategy 2, employing a grid size of 256×256 and coefficients $(E, \nu) = (5.5 \times 10^8, 0.4)$, is depicted in figure 5.

| (E, ν) | (3E08, 0.4) | (5.2E08, 0.35) | (5.5E08, 0.4) | (9E08, 0.36) |
|---|----------------------|----------------------|----------------------|----------------------|
| L_∞ error for Strategy 1 | $9.5E-4$ $8.5E-4$ | $5.1E-4$ $4.9E-4$ | $9.6E-4$ $8.5E-4$ | $1.8E-3$ $1.1E-3$ |
| L_∞ error for KFBI | $1.4E-4$ $1.2E-4$ | $1.0E-4$ $8.9E-5$ | $1.4E-4$ $1.2E-4$ | $1.1E-4$ $9.3E-5$ |
| L_∞ error for Strategy 2 | $1.4E-4$ $1.2E-4$ | $1.0E-4$ $8.9E-5$ | $1.4E-4$ $1.2E-4$ | $1.1E-4$ $9.3E-5$ |
| Iterations (KFBI) | 26 | 26 | 26 | 26 |
| Iterations (Strategy 2) | 15 | 14 | 15 | 17 |
| Network inference time in iteration steps | 0.079 | 0.072 | 0.081 | 0.071 |
| Save iterations | 42% | 46% | 42% | 34% |

Table 9: Result of Naiver equation 1: comparison of accuracy and efficiency in solving Naiver equation 1 using different methods. The error vector is computed by the numerical solution $(u_{\text{numerical}}^{(1)}, u_{\text{numerical}}^{(2)})$ and the exact solution $(u_{\text{exact}}^{(1)}, u_{\text{exact}}^{(2)})$.

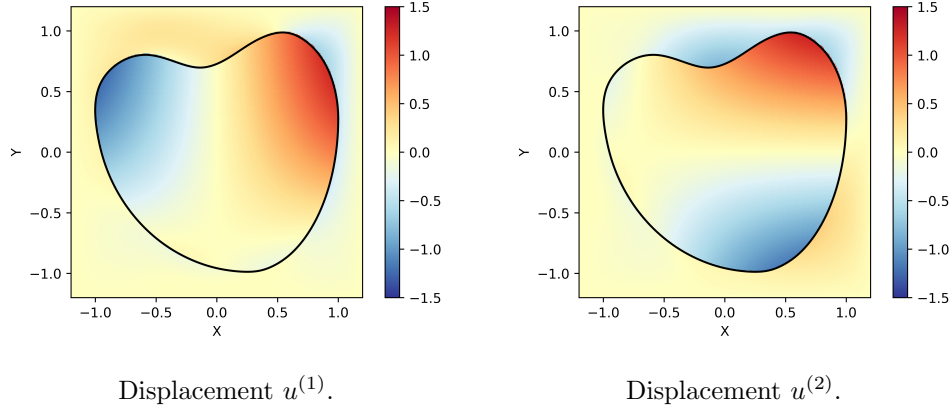


Figure 5: Numerical solution of the 2D Naiver equation 1 given by Strategy 2 in grid 256×256 . Note that the figures show the result given by solving the corresponding interface problem in KFBI whose interior solution is the desired result for the original PDE.

From the data in this table, a conclusion is derived from section 4.2.3 regarding our trained model. It becomes evident that applying direct model inference outcomes as the solution for the density function (Strategy 1) yields a sufficiently high level of accuracy.

4.2.6 Naiver Equations on Domains with Boundaries Defined by Perturbed Control Points

In this section, we continue to focus on the Naiver equations presented in 4.2.5. We allow for perturbations to be applied to certain control points, and incorporate the coordinates of these control points into the input of the model (note that $\partial\Omega$ is obtained through cubic spline interpolation of the control points). Specifically in this experience, the original domain is a heart-shaped domain whose boundary is obtained through cubic spline interpolation of 28 control points, identical to the boundary described in 4.2.5. We allow for perturbations in the x-direction of the 1st, 14th, and 27th control points, with distances of $\epsilon_1, \epsilon_{14}, \epsilon_{27}$, respectively. Here $\epsilon_i \in [-0.15, 0.15], i \in \{1, 14, 27\}$. As previously mentioned, the coordinates of these three points will be included in the model's input along with the values of $\widetilde{\mathbf{g}}_{\mathbf{D}}$ at M discrete points. The equation parameters E and ν are fixed at 1E9 and 0.45, respectively. The bounding box \mathcal{B} for the interface problem is established as $\mathcal{B} = [-1.5, 1.5] \times [-1.5, 1.5]$. The number of discrete points on the boundary curve M is set at $M = \max\{I, J\}$ with $I = J = 128$. Absolute tolerance for iteration is determined to be 1E-8, and the initial value for the standard KFBI method is set as $\phi_0 = (0, 0)^T$.

Table 10 presents the results for Naiver Equation 2, featuring exact solution $u^{(1)} = \sin x \cos y + xy$ and $u^{(2)} = \cos x \sin y + xy$, applied to the 'quasi-heart-shaped' domain with perturbed boundary with various $(\epsilon_1, \epsilon_{14}, \epsilon_{27})$ parameter values. As an example, the numerical solution obtained through Strategy 1, employing a grid size of 128×128 and perturbations $(\epsilon_1, \epsilon_{14}, \epsilon_{27}) = (+0.05, +0.05, -0.1)$, is depicted in figure 6.

In this section, we allow for perturbations to be applied to certain control points that determine the boundaries of the domain, incorporating this information as part of the model input. Experimental results demonstrate that the model under this scenario still maintains strong capabilities. Particularly noteworthy, both the error metrics and figure 6 representations illustrate the commendable accuracy achieved when employing the model and Strategy 1 to solve the

CONCLUSION 111

| $(\epsilon_1, \epsilon_{14}, \epsilon_{27})$ | $(+0.05, +0.05, -0.1)$ | $(-0.04, +0.02, 0)$ | $(+0.02, -0.01, +0.06)$ |
|--|------------------------|----------------------|-------------------------|
| L_∞ error for Strategy 1 | $3.7E-3$ $5.4E-3$ | $3.5E-3$ $3.1E-3$ | $5.1E-3$ $3.0E-3$ |
| L_∞ error for KFBI | $1.1E-3$ $1.2E-3$ | $1.1E-3$ $1.2E-3$ | $1.1E-3$ $1.2E-3$ |
| L_∞ error for Strategy 2 | $1.1E-3$ $1.2E-3$ | $1.1E-3$ $1.2E-3$ | $1.1E-3$ $1.2E-3$ |
| Iterations (KFBI) | 36 | 35 | 35 |
| Iterations (Strategy 2) | 23 | 20 | 22 |
| Network inference time in iteration steps | 0.12 | 0.11 | 0.14 |
| Save iterations | 36% | 43% | 37% |

Table 10: Result of Naiver equation 2: comparison of accuracy and efficiency in solving Naiver equation 2 using different methods. The error vector is computed by the numerical solution $(u_{\text{numerical}}^{(1)}, u_{\text{numerical}}^{(2)})$ and the exact solution $(u_{\text{exact}}^{(1)}, u_{\text{exact}}^{(2)})$ 111

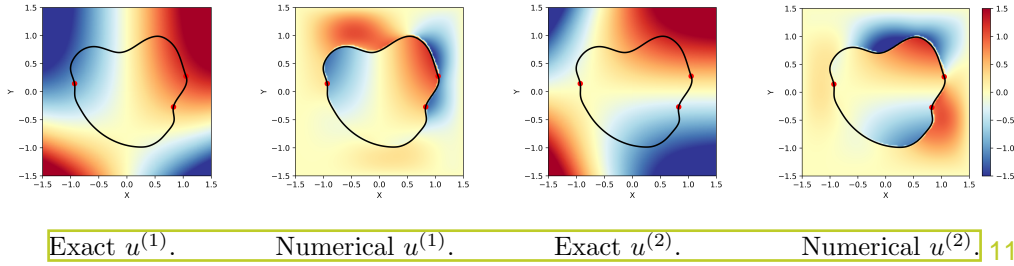


Figure 6: Exact solution and numerical solution of the 2D Naiver equation 2 given by Strategy 1 in grid 128×128 . Note that the figures show the result given by solving the corresponding interface problem in KFBI whose interior solution is the desired result for the original PDE. 137

equations. 67

5 Conclusion 111

In this study, a hybrid Kernel-Free Boundary Integral Method, integrated by KFBI method and operator learning, was rigorously examined for solving parametric partial differential equations (PDEs) in complex domains. This novel approach demonstrated significant advancements in computational efficiency and accuracy. 157

Key findings include the model's robust generalization capabilities, allowing for accurate predictions across various boundary conditions and parameters within the same class of equations. The integration of neural network-trained models with the framework of boundary integral method significantly accelerates the computational process, reducing traditional KFBI method iterations by approximately 50% times while retaining its inherent second-order accuracy. Moreover, the study highlighted the flexibility of the hybrid method in dealing with a range of equations in complex domains. The method's ability to transform two-dimensional problems into one-dimensional boundary problems, coupled with its independence from Green's functions, positioned it as a highly efficient computational tool. 158

However, challenges may appear related to generating sufficient data for training and potential issues with the size of networks for more complicated problems, which may potentially affecting inference speeds. These areas, identified for future exploration, underscore the necessity for continuous optimization of the method. To further align the hybrid KFBI Method with engineering applications, it is proposed that its scope be expanded to encompass time-dependent PDEs such as the Schrödinger, Navier-Stokes, and Maxwell equations. Additionally, the development of a three-dimensional variant of the hybrid KFBI Method is identified as a pressing objective. This expansion and enhancement are anticipated to significantly broaden the method's applicability and efficacy in complex engineering scenarios.

In conclusion, the hybrid KFBI Method, augmented with deep learning, presents a powerful tool for operator learning and solving PDEs in complex domains, offering significant improvements in computational efficiency especially under the situation of solving abundant PDEs of the same class. Its potential applications span across various fields, necessitating further research to harness its capabilities fully.

Acknowledgments

This work is financially supported by the Strategic Priority Research Program of Chinese Academy of Sciences(Grant No. XDA25010405). It is also partially supported by the National Key R&D Program of China, Project Number 2020YFA0712000, the National Natural Science Foundation of China (Grant No. DMS-11771290) and the Science Challenge Project of China (Grant No. TZ2016002). Additionally, it is supported by the Fundamental Research Funds for the Central Universities. In addition, we are profoundly grateful to Professor Zhi-Qin John Xu for his unwavering guidance and unwavering support throughout our research journey. His profound expertise and invaluable insights have not only shaped the trajectory of this work but have also inspired us to reach new heights in our academic pursuits.

References

- [1] K.P. Bube A. Wiegmann. The explicit-jump immersed interface method: finite difference methods for pdes with piecewise smooth solutions. *SIAM Journal on Numerical Analysis*, 37(3):827–862, 2000. 240
- [2] M. H. Aliabadi and P. H. Wen. *Boundary element methods in engineering and sciences*, volume 4. World Scientific, 2011. 241
- [3] Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018. 242
- [4] P.A. Berthelsen. A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions. *Journal of Computational Physics*, 197(1):364–386, 2004. 243
- [5] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B. Kovachki, and Andrew M. Stuart. Model Reduction And Neural Networks For Parametric PDEs. *The SMAI Journal of computational mathematics*, 7:121–157, 2021. 244
- [6] Yingbin Chai, Kangye Huang, Shangpan Wang, Zhichao Xiang, and Guanjin Zhang. The extrinsic enriched finite element method with appropriate enrichment functions for the helmholtz equation. *Mathematics*, 11:1664, 03 2023. 245
- [7] Roman Chapko and Rainer Kress. Rothe’s method for the heat equation and boundary integral equations. *Journal of Integral Equations and Applications*, 9(1):47 – 69, 1997. Cited by: 43; All Open Access, Bronze Open Access. 246
- [8] Long Chen, Huayi Wei, and Min Wen. An interface-fitted mesh generator and virtual element methods for elliptic interface problems. *Journal of Computational Physics*, 334, 01 2017. 247
- [9] Hongwei Cheng, Jingfang Huang, and Terry Jo Leiterman. An adaptive fast solver for the modified helmholtz equation in two dimensions. *Journal of Computational Physics*, 211(2):616–637, 2006. 248
- [10] C.-C. CHU, I. G. GRAHAM, and T.-Y. HOU. A new multiscale finite element method for high-contrast elliptic interface problems. *Mathematics of Computation*, 79(272):1915–1955, 2010. 249
- [11] Jie Ding, Zhongming Wang, and Shenggao Zhou. Positivity preserving finite difference methods for poisson–nernst–planck equations with steric interactions: Application to slit-shaped nanopore conductance. *Journal of Computational Physics*, 397:108864, 2019. 250
- [12] Haixia Dong, Shuwang Li, Wenjun Ying, and Zhongshu Zhao. Kernel-free boundary integral method for two-phase stokes equations with discontinuous viscosity on staggered grids, 2023. 251
- [13] Haixia Dong, Zhongshu Zhao, Shuwang Li, Wenjun Ying, and Jiwei Zhang. Second order convergence of a modified mac scheme for stokes interface problems, 2023. 252
- [14] Weinan Ee and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6, 09 2017. 253

REFERENCES

- [15] Chen Fan and Zhiyue Zhang. Decoupling numerical method based on deep neural network for nonlinear degenerate interface problems, 2023. 254
- [16] Ronald P Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of computational physics*, 152(2):457–492, 1999. 255
- [17] W.L. Wendland G.C. Hsiao. *Boundary integral equations*. Springer, 2008. 256
- [18] Craig R Gin, Daniel E Shea, Steven L Brunton, and J Nathan Kutz. Deepgreen: deep learning of green’s functions for nonlinear boundary value problems. *Scientific reports*, 11(1):1–14, 2021. 257
- [19] Wei Gong, Jiajie Li, and Shengfeng Zhu. Improved discrete boundary type shape gradients for pde-constrained shape optimization. *SIAM Journal on Scientific Computing*, 44(4):A2464–A2505, 2022. 258
- [20] Daniel Greenfeld, Meirav Galun, Ronen Basri, Irad Yavneh, and Ron Kimmel. Learning to optimize multigrid pde solvers. In *International Conference on Machine Learning*, pages 2415–2423. PMLR, 2019. 259
- [21] Leslie Greengard and Mary Catherine Kropinski. An integral equation approach to the incompressible navier-stokes equations in two dimensions. *SIAM Journal on Scientific Computing*, 20(1):318 – 336, 1998. Cited by: 37; All Open Access, Green Open Access. 260
- [22] Leslie Greengard and Vladimir Rokhlin. A new version of the fast multipole method for the laplace equation in three dimensions. *Acta Numerica*, 6:229–269, 1997. 261
- [23] Cuiyu He, Xiaozhe Hu, and Lin Mu. A mesh-free method using piecewise deep neural network for elliptic interface problems. *Journal of Computational and Applied Mathematics*, 412:114358, 2022. 262
- [24] Songming Hou and Xu-Dong Liu. A numerical method for solving variable coefficient elliptic equation with interfaces. *Journal of Computational Physics*, 202(2):411–445, 2005. 263
- [25] Songming Hou, Liquan Wang, and Wei Wang. A numerical method for solving the elliptic interface problems with multi-domains and triple junction points. *Journal of Computational Mathematics*, pages 504–516, 2012. 264
- [26] Wei-Fan Hu, Te-Sheng Lin, and Ming-Chih Lai. A discontinuity capturing shallow neural network for elliptic interface problems. *Journal of Computational Physics*, 469:111576, 2022. 265
- [27] Peiqi Huang, Haijun Wu, and Yuanming Xiao. An unfitted interface penalty finite element method for elliptic interface problems. *Computer Methods in Applied Mechanics and Engineering*, 323:439–460, 2017. 266
- [28] M. A. Jaswon. Integral equation methods in potential theory. i. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 275(1360):23–32, 1963. 267
- [29] Chiyu “Max” Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, Karthik Kashinath, Mustafa Mustafa, Hamdi A. Tchelepi, Philip Marcus, Mr Prabhat, and Anima Anandkumar. Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2020. 268

REFERENCES

- [30] Hans Johansen and Phillip Colella. A cartesian grid embedded boundary method for poisson's equation on irregular domains. *Journal of Computational Physics*, 147(1):60–85, 1998.
- [31] Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.
- [32] Randall J LeVeque and Zhilin Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis*, 31(4):1019–1044, 1994.
- [33] Zhilin Li, Yanping Lin, and Xiaohui Wu. New cartesian grid methods for interface problems using the finite element formulation. *Numerische Mathematik*, 96, 03 1999.
- [34] Zhilin Li, Wei-Cheng Wang, I-Liang Chern, and Ming-Chih Lai. New formulations for interface problems in polar coordinates. *SIAM Journal on Scientific Computing*, 25(1):224–245, 2003.
- [35] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.
- [36] Guochang Lin, Fukai Chen, Pipi Hu, Xiang Chen, Junqing Chen, Jun Wang, and Zuoqiang Shi. Bi-greenet: learning green's functions by boundary integral network. *Communications in Mathematics and Statistics*, 11(1):103–129, 2023.
- [37] Guochang Lin, Pipi Hu, Fukai Chen, Xiang Chen, Junqing Chen, Jun Wang, and Zuoqiang Shi. Binet: Learn to solve partial differential equations with boundary integral networks. *CSIAM Transactions on Applied Mathematics*, 4(2):275–305, 2023.
- [38] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 03 2021.
- [39] Kevin Luna, Katherine Klymko, and Johannes P. Blaschke. Accelerating gmres with deep learning in real-time, 2021.
- [40] Nicholas H. Nelsen and Andrew M. Stuart. The random feature model for input-output maps between banach spaces. *SIAM Journal on Scientific Computing*, 43(5):A3212–A3243, 2021.
- [41] H. Johansen P. McCorquodale, P. Colella. A cartesian grid embedded boundary method for the heat equation on irregular domains. *Journal of Computational Physics*, 173(2):620–635, 2001.
- [42] Ravi G Patel, Nathaniel A Trask, Mitchell A Wood, and Eric C Cyr. A physics-informed operator regression framework for extracting data-driven continuum models. *Computer Methods in Applied Mechanics and Engineering*, 373:113500, 2021.
- [43] Yiran Qian, Cheng Wang, and Shenggao Zhou. A positive and energy stable numerical scheme for the poisson–nernst–planck–cahn–hilliard equations with steric interactions. *Journal of Computational Physics*, 426:109908, 2021.

REFERENCES

- [44] Luigi Quartapelle. Numerical solution of the incompressible navier-stokes equations. In *International series of numerical mathematics*, volume 113, 1993. 283
- [45] V. Kozlov R. Kress, V. Maz'ya. *Linear integral equations*, volume 82. Springer, 1989. 284
- [46] 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. *Journal of Computational Physics*, 378:686–707, 2019. 285
- [47] Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019. 286
- [48] Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019. 286
- [49] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018. 288
- [50] Huafei Sun and David L. Darmofal. An adaptive simplex cut-cell method for high-order discontinuous galerkin discretizations of elliptic interface problems and conjugate heat transfer problems. *Journal of Computational Physics*, 278:445–468, 2014. 289
- [51] Edward H Twizell, Abba B Gumel, and MA Arigu. Second-order, l 0-stable methods for the heat equation with time-dependent boundary conditions. *Advances in Computational Mathematics*, 6(1):333–352, 1996. 290
- [52] C.S. Henriquez W. Ying. A kernel-free boundary integral method for elliptic boundary value problems. *Journal of computational physics*, 227(2):1046–1074, 2007. 291
- [53] W.C. Wang W. Ying. A kernel-free boundary integral method for implicitly defined surfaces. *Journal of Computational Physics*, 252:606–624, 2013. 292
- [54] W.C. Wang W. Ying. A kernel-free boundary integral method for variable coefficients elliptic pdes. *Communications in Computational Physics*, 15(4):1108–1140, 2014. 293
- [55] W. Ying Y. Xie. A fourth-order kernel-free boundary integral method for the modified helmholtz equation. *Journal of Scientific Computing*, 78(3):1632–1658, 2019. 294
- [56] W. Ying Y. Xie, Z. Huang. A cartesian grid based tailored finite point method for reaction-diffusion equation on complex domains. *Computers & Mathematics with Applications*, 97:298–313, 2021. 295
- [57] Lexing Ying, George Biros, and Denis Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, 196(2):591–626. 2004. 296
- [58] Lexing Ying, George Biros, and Denis Zorin. A high-order 3d boundary integral equation solver for elliptic pdes in smooth domains. *Journal of Computational Physics*, 219(1):247–275, 2006. 297
- [59] D. Yu. *Natural boundary integral method and its applications*, volume 539. Springer Science & Business Media, 2002. 298

REFERENCES

- [60] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, jun 2020. 299
- [61] Lulu Zhang, Tao Luo, Yaoyu Zhang, Weinan E, Zhi-Qin John Xu, and Zheng Ma. Mod-net: A machine learning approach via model-operator-data network for solving pdes. *Communications in Computational Physics*, 32(2):299–335, 2022. 300
- [62] Zhongshu Zhao, Haixia Dong, and Wenjun Ying. Kernel free boundary integral method for 3d incompressible flow and linear elasticity equations on irregular domains. *Computer Methods in Applied Mechanics and Engineering*, 414:116163, 2023. 301
- [63] Zhongshu Zhao, Haixia Dong, and Wenjun Ying. Kernel free boundary integral method for 3d incompressible flow and linear elasticity equations on irregular domains. *Computer Methods in Applied Mechanics and Engineering*, 414:116163, 2023. 301
- [64] Han Zhou, Minsheng Huang, and Wenjun Ying. Adi schemes for heat equations with irregular boundaries and interfaces in 3d with applications, 2023. 303
- [65] Y.C. Zhou, S. Zhao, and G.W. We M. Feig. High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. *Journal of Computational Physics*, 213(1):1–30, 2006. 304
- [66] Y.C. Zhou, Shan Zhao, Michael Feig, and G.W. Wei. High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. *Journal of Computational Physics*, 213(1):1–30, 2006. 304
- [67] Shengfeng Zhu, Qingbiao Wu, and Chunxiao Liu. Shape and topology optimization for elliptic boundary value problems using a piecewise constant level set method. *Applied Numerical Mathematics*, 61(6):752–767, 2011. 306

A Introduction to KFBI Method 162

In the BI method, Green's function's explicit expression is a prerequisite, yet its exact form depends on the equation's specifics and the integration region's geometry. Despite attempts to substitute Green's function with a neural network [36], this approach has yet to attain high accuracy levels and struggles with variable coefficients. The KFBI method [52] circumvents the need for explicitly expressing Green's function. Here, integrals in (5)-(6) are treated as equivalent interface problems, facilitating the implementation of iterative steps (10)-(11). Specifically, the double layer boundary integral $(W\varphi)(\mathbf{x})$ and volume integral $(Yf)(\mathbf{x})$ are reformulated as the solution $v(\mathbf{x})$ of this interface problem, with pertinent terms detailed in table 11. 163

$$\begin{aligned} \mathcal{L}v(\mathbf{x}) &= \mathcal{F}(\mathbf{x}), \quad \mathbf{x} \text{ in } \Omega \cup \Omega^c, \\ [v(\mathbf{x})] &= \Phi(\mathbf{x}), \quad \mathbf{x} \text{ on } \Gamma, \\ [\partial_{\mathbf{n}}v(\mathbf{x})] &= 0, \quad \mathbf{x} \text{ on } \Gamma, \\ v(\mathbf{x}) &= 0, \quad \mathbf{x} \text{ on } \partial\mathcal{B}. \end{aligned} \quad (21) \quad 233$$

| Integral | \mathcal{F} | Φ |
|------------|--|------------------|
| $W\varphi$ | $\mathcal{F} = 0$ | $\Phi = \varphi$ |
| Yf | $\mathcal{F} = \tilde{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{in } \Omega \\ 0 & \text{in } \Omega^c \end{cases}$ | $\Phi = 0$ |

Table 11: The relationship between boundary integral or volume integral and the interface problem with specific terms. 163

In equation (21), the terms $[v(\mathbf{x})]$ and $[\partial_{\mathbf{n}}v(\mathbf{x})]$ denote the jumps in the unknown $[v(\mathbf{x})] = v^+(\mathbf{x}) - v^-(\mathbf{x})$ and its normal derivatives $[\partial_{\mathbf{n}}v(\mathbf{x})] = \partial_{\mathbf{n}}v^+(\mathbf{x}) - \partial_{\mathbf{n}}v^-(\mathbf{x})$, respectively. The function $\tilde{f}(\mathbf{x})$ represents the zero extension of the given function $f(\mathbf{x})$. Under addressing the interface problem (21), the computation of integrals for iterative processes and the solution (8) is contingent upon the numerical solution of (1) (2) within Ω . The solver for this interface problem, developed by Ying and not the primary focus of this paper, is detailed in the literature[52]. 166

B The KFBI Method for Elliptic PDEs with Neumann Boundary Conditions 167

The BI method and KFBI method for elliptic PDEs with Dirichlet BVP has been introduced in Section 2 and Appendix A. In this section, the elliptic equation (1) subject to Neumann boundary condition is considered as: 168

$$\partial_{\mathbf{n}}u(\mathbf{x}) = g_N(\mathbf{x}). \quad (22) \quad 233$$

Similarly to the Dirichlet boundary condition, the single layer potential is defined as 170

$$(S\psi)(\mathbf{x}) := \int_{\Gamma} G(\mathbf{y}, \mathbf{x})\psi(\mathbf{y})ds_{\mathbf{y}} \quad \text{for } \mathbf{x} \in \Omega \cup \Omega^c. \quad (23) \quad 233$$

Owing to detonations defined above, the BIEs for (1) and (22) also can be reformulated as a Fredholm boundary integral equation of the second kind[17, 45], which follows 172

$$\frac{1}{2}\psi(\mathbf{x}) - \partial_{\mathbf{n}}(S\psi)(\mathbf{x}) + \partial_{\mathbf{n}}(Yf)(\mathbf{x}) = g_N(\mathbf{x}). \quad (24) \quad 233$$

The solution $u(\mathbf{x})$ to Neumann BVP (1) and (22) is given by 174

$$u(\mathbf{x}) = (Yf)(\mathbf{x}) - (S\psi)(\mathbf{x}), \quad x \in \Omega. \quad (25) \quad 233$$

Numerically, the boundary integral equation (24) can be solved by simply iteration: given the artificial initial guess $\psi_0(\mathbf{x})$, for $k = 0, 1, 2, \dots$, do as follows: 176

$$\partial_{\mathbf{n}} u_k^+(\mathbf{x}_m) = \frac{1}{2} \psi(\mathbf{x}_m) - \partial_{\mathbf{n}}(S\psi)(\mathbf{x}_m), \quad m = 1, 2, \dots, M, \quad (26)$$

$$\psi_{k+1}(\mathbf{x}_m) = \psi_k(\mathbf{x}_m) + \gamma[\hat{g}_N(\mathbf{x}_m) - \partial_{\mathbf{n}} u_k^+(\mathbf{x}_m)], \quad m = 1, 2, \dots, M, \quad (27) \quad 177$$

where $\{\mathbf{x}_i\}_{i=1}^M$ are control points on boundary Γ and $\hat{g}_N(\mathbf{x}_m) := g_N(\mathbf{x}_m) - \partial_{\mathbf{n}}(Yf)(\mathbf{x}_m)$, for $\mathbf{x}_m \in \Gamma$. Suppose $w(\mathbf{x})$ is an arbitrary piecewise smooth function with derivative discontinuities on the interface Γ : 240

$$\partial_{\mathbf{n}} w^+(\mathbf{x}) = \lim_{z \rightarrow x, z \in \Omega} \partial_{\mathbf{n}} w(\mathbf{z}), \quad (28) \quad 233$$

$\partial_{\mathbf{n}} w^-(\mathbf{x})$ can be defined in the same way. 227

As for Neumann BVP, the single layer boundary integral (23) can be considered as a solution to the following interface problem: 232

$$\begin{aligned} \mathcal{L}v(\mathbf{x}) &= 0, & \text{for } \mathbf{x} \in \Omega \cup \Omega^c, & 192 \\ [v(\mathbf{x})] &= 0, & \text{for } \mathbf{x} \in \Gamma, & 192 \\ [\partial_{\mathbf{n}} v(\mathbf{x})] &= \psi(\mathbf{x}), & \text{for } \mathbf{x} \in \Gamma, & 192 \\ v(\mathbf{x}) &= 0, & \text{for } \mathbf{x} \in \partial\mathcal{B}, & 192 \end{aligned} \quad (29) \quad 233$$

The KFBI method is distinguished by its conversion of integral (23) into the resolution of the interface problem (29). 202

C Record for Control Points 235

C.1 Control Points for Dumbbell-Shaped Domain 236

The control points for the dumbbell-shaped domain in section 4.2.3 are given as following: 237

$\{0.000\text{e}+00, 8.333\text{e}-02\}$, $\{-1.944\text{e}-01, 1.698\text{e}-01\}$, $\{-3.889\text{e}-01, 3.302\text{e}-01\}$,
 $\{-5.833\text{e}-01, 4.167\text{e}-01\}$, $\{-7.917\text{e}-01, 3.608\text{e}-01\}$, $\{-9.442\text{e}-01, 2.083\text{e}-01\}$,
 $\{-1.000\text{e}+00, 0.000\text{e}+00\}$, $\{-9.442\text{e}-01, -2.083\text{e}-01\}$, $\{-7.917\text{e}-01, -3.608\text{e}-01\}$,
 $\{-5.833\text{e}-01, -4.167\text{e}-01\}$, $\{-3.889\text{e}-01, -3.302\text{e}-01\}$, $\{-1.944\text{e}-01, -1.698\text{e}-01\}$,
 $\{0.000\text{e}+00, -8.333\text{e}-02\}$, $\{1.944\text{e}-01, -1.698\text{e}-01\}$, $\{3.889\text{e}-01, -3.302\text{e}-01\}$,
 $\{5.833\text{e}-01, -4.167\text{e}-01\}$, $\{7.917\text{e}-01, -3.608\text{e}-01\}$, $\{9.442\text{e}-01, -2.083\text{e}-01\}$,
 $\{1.000\text{e}+00, 0.000\text{e}+00\}$, $\{9.442\text{e}-01, 2.083\text{e}-01\}$, $\{7.917\text{e}-01, 3.608\text{e}-01\}$,
 $\{5.833\text{e}-01, 4.167\text{e}-01\}$, $\{3.889\text{e}-01, 3.302\text{e}-01\}$, $\{1.944\text{e}-01, 1.698\text{e}-01\}$.

C.2 Control Points for Heart-Shaped Domain 236

The control points for the heart-shaped domain in section 4.2.5 are given as following: 237

$\{1.000\text{e}+00, 2.763\text{e}-01\}$, $\{9.639\text{e}-01, 5.482\text{e}-01\}$, $\{8.613\text{e}-01, 7.787\text{e}-01\}$,
 $\{7.076\text{e}-01, 9.328\text{e}-01\}$, $\{5.263\text{e}-01, 9.868\text{e}-01\}$, $\{3.070\text{e}-01, 9.118\text{e}-01\}$,
 $\{8.772\text{e}-02, 7.724\text{e}-01\}$, $\{-1.316\text{e}-01, 6.974\text{e}-01\}$, $\{-3.553\text{e}-01, 7.500\text{e}-01\}$,
 $\{-5.789\text{e}-01, 8.026\text{e}-01\}$, $\{-7.895\text{e}-01, 7.462\text{e}-01\}$, $\{-9.436\text{e}-01, 5.921\text{e}-01\}$,
 $\{-1.000\text{e}+00, 3.816\text{e}-01\}$, $\{-9.808\text{e}-01, 1.440\text{e}-01\}$, $\{-9.238\text{e}-01, -8.645\text{e}-02\}$,

$\{-8.308\text{e-}01, - 3.026\text{e-}01\}$, $\{-7.045\text{e-}01, - 4.980\text{e-}01\}$, $\{-5.488\text{e-}01, - 6.667\text{e-}01\}$,
 $\{-3.684\text{e-}01, - 8.035\text{e-}01\}$, $\{-1.689\text{e-}01, - 9.043\text{e-}01\}$, $\{4.381\text{e-}02, - 9.661\text{e-}01\}$,
 $\{2.632\text{e-}01, - 9.868\text{e-}01\}$, $\{4.271\text{e-}01, - 9.552\text{e-}01\}$, $\{5.829\text{e-}01, - 8.618\text{e-}01\}$,
 $\{7.226\text{e-}01, - 7.113\text{e-}01\}$, $\{8.392\text{e-}01, - 5.113\text{e-}01\}$, $\{9.270\text{e-}01, - 2.717\text{e-}01\}$,
 $\{9.815\text{e-}01, - 4.763\text{e-}03\}$.