Learning Domain-Independent Green's Function For Elliptic Partial Differential Equations

Pawan Negi[a], Maggie Cheng[a], Mahesh Krishnamurthy[b], Wenjun Ying[c], Shuwang Li[a,*]

[a]*Department of Applied Mathematics, Illinois Institute of Technology, Chicago, USA*
[b]*Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, USA*
[c]*School of Mathematical Sciences, MOE-LSC and Institute of Natural Science, Shanghai Jiao Tong University, Shanghai, China*

Abstract

Green's function characterizes a partial differential equation (PDE) and maps its solution in the entire domain as integrals. Finding the analytical form of Green's function is a non-trivial exercise, especially for a PDE defined on a complex domain or a PDE with variable coefficients. In this paper, we propose a novel boundary integral network to learn the domain-independent Green's function, referred to as BIN-G. We evaluate the Green's function in the BIN-G using a radial basis function (RBF) kernel-based neural network. We train the BIN-G by minimizing the residual of the PDE and the mean squared errors of the solutions to the boundary integral equations for prescribed test functions. By leveraging the symmetry of the Green's function and controlling refinements of the RBF kernel near the singularity of the Green function, we demonstrate that our numerical scheme enables fast training and accurate evaluation of the Green's function for PDEs with variable coefficients. The learned Green's function is independent of the domain geometries, forcing terms, and boundary conditions in the boundary integral formulation. Numerical experiments verify the desired properties of the method and the expected accuracy for the two-dimensional Poisson and Helmholtz equations with variable coefficients.

Keywords: Boundary integral method, Green's function

*Corresponding author
Email address: sli15@iit.edu (Shuwang Li)

1. Introduction

[1] Elliptic partial differential equations (PDEs) arise in many research areas, such as fluid dynamics, geophysics, electrostatics, electro-magnetics, image processing, and materials science. Green's function, also known as the fundamental solution of the PDE, enables one to write the solution of the PDE as integrals. With the help of the Green's function, computational methods based on integral formulations have been developed and show excellent accuracy and efficiency [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. Green's function also plays a significant role in the analysis of PDEs and helps to establish the well-posedness and regularity properties of the PDEs [14]. Usually, both the PDE analysis and numerical methods require an analytical form of the Green's function to establish theories and numerical schemes (e.g., quadratures).

Recently, as a variant of the classic integral methods, the kernel-free boundary integral method (KFBIM) was proposed [15, 16]. One of the salient features of the KFBIM is that it does not require an explicit form of the Green's function or special quadratures to directly evaluate integrals. The main idea behind KFBIM is to reinterpret the boundary integrals as solutions to an equivalent simple interface problems. It can be solved efficiently using Cartesian grid-based methods. Compared to the traditional finite difference or finite element methods, the KFBIM produces a well-conditioned linear systems and requires only a fixed number of iterations to converge (when an iterative method is applied). The KFBIM has been successful in numerically solving elliptic PDEs in two and three dimensions [17, 18, 19, 20]. However, one still does not gain insight into the Green's function (fundamental solution) of the PDE.

Methods for finding the Green's functions include analytically deriving formulas or computing eigenfunction expansions for PDEs defined on simple geometries or numerically solving a singular PDE (e.g., by approximating the Dirac delta function). However, when the geometry of the domain is complex, or the PDE has variable coefficients, finding the analytical form of

the Green's function is indeed a non-trivial exercise.

In recent years, neural networks (NNs) have been employed to solve partial differential equations owing to the development of automatic differentiation, see [21, 22, 23, 24, 25, 26] and many others. Using these neural networks, the boundary value or interface problem can be solved by simply adding a loss term for the interface or boundary condition [27]. Tseng et al. [28] proposed to introduce a cusp-enforced level set function as an additional feature to the NN. However, the physics-informed neural networks (PINNs) approach is known to have poor convergence for boundary value problem [29, 30, 31].

In order to overcome this issue, researchers proposed boundary integral-based neural networks. Lin et al. [30] used known Green's functions to evaluate solutions to various boundary value problems. They employed single and double-layer potentials as the loss function. Lin et al. [32] extended the approach to approximate the Green's function for the boundary value problem, provided that the Green's function for the infinite space is available. Boullé et al. [33] used the Gaussian process to generate arbitrary pairs of test and source functions, which are then used to learn the Green function as well as the homogeneous solution of the given PDE. However, the learned Green's functions were specific to a particular domain and boundary condition. Very recently, Teng et al. [34] proposed to learn Green's function for the PDEs with a given domain by approximating the Dirac delta with a Gaussian function, which allows one to readily employ the learned Green's function for this domain with any other set of boundary conditions. Peng et al. [35] replaced the Dirac delta with an alternative input for which an analytical form can be calculated for a given boundary condition on an arbitrarily shaped domain. However, one must solve the network again for the given PDE defined in a different domain.

In this work, we propose a novel boundary integral based neural network that employs a radial basis function (RBF) kernel-based neural network to learn domain-agnostic Green's function of an elliptical PDE. Since the learned Green's function is not limited to a specific domain, it can be readily employed in the integral formulations for solving a moving interface or a boundary problem in fluids, materials, and wave propagation [36, 37, 7, 38, 39, 40]. Especially, problems defined in a heterogeneous media, where PDE coefficients are spatially dependent.

We construct boundary integral neural network with a trainable Green's function referred to as BIN-G to learn solutions of an elliptic PDE. In BIN-

G, we use single and double-layer potentials to satisfy boundary conditions. We evaluate the density function in these potentials using multi-layer perceptron (MLP) networks. We train the BIN-G by minimizing the residual of the PDE such that the Green function is the solution, together with the mean-squared error in the solution using the boundary integral equation for the prescribed test functions. To gain efficiency, we exploit the symmetry property of Green's function in our network, allowing us to learn the Green's function using a 1D sample space. Furthermore, the boundary integral formulation allows one to learn the density functions using boundary samples alone, resulting in further reduction in the computational cost.

Our salient contributions are as follows:

- We construct a RBF kernel-based neural network that allows us to preset the approximation points and their support radius near the singularity. This offers faster and more accurate learning compared to the MLP networks.

- We propose loss function that includes both the loss due to the PDE and the boundary integral equations. It offers learning the domain-independent Green's function of the PDE and the domain-dependent density functions during the training process.

- We employ two distinct sample spaces over which the losses are computed. A relatively large domain on which the residual of the PDE is minimized, allows one to employ the learned domain-independent Green's function to larger domains.

- The learned Green's function can be readily applied to train new density functions on arbitrarily-shaped domains and sets of boundary conditions.

We perform numerical experiments that verify the desired properties of the methods and the expected accuracy for the two-dimensional Poisson equation and Helmholtz equation with variable coefficients.

In the next section, we present mathematical preliminaries related to the boundary integral formulation. In section 3, we present the architecture of the proposed neural network. In section 4, we discuss the training strategy to learn the domain-independent Green's function. In section 5, we discuss the generalization of the learned Green's function to different domain and

boundary conditions. In section 6, we demonstrate the capability of the neural network to learn known Green's functions followed by learning the Green's function of variable coefficient PDEs. In section 7, we summarize and discuss the outcome of the present study and future work.

2. Mathematical Preliminaries

The boundary integral method is widely used to solve elliptic boundary value problems. Consider a scalar field u defined in a domain Ω with boundary $\partial\Omega$, an elliptic partial differential equation takes the form

$$\mathscr{L}_{\mathbf{x}} u(\mathbf{x}) = f(\mathbf{x}),\ \forall\ \mathbf{x} \in \Omega, \tag{1}$$

where $\mathscr{L}_{\mathbf{x}}$ is an elliptic differential operator and $f(\mathbf{x})$ is a source (forcing) function. We use the notation $\mathscr{L}_{\mathbf{x}}$ to specify the variable on which the operation is being performed. The above equation may be subjected to a Dirichlet boundary condition

$$u(\mathbf{x}) = g^D(\mathbf{x}),\ \forall\ \mathbf{x} \in \partial\Omega, \tag{2}$$

or a Neumann boundary condition

$$\frac{\partial u(\mathbf{x})}{\partial \nu} = g^N(\mathbf{x}),\ \forall\ \mathbf{x} \in \partial\Omega, \tag{3}$$

where ν is the outward normal to the boundary. The boundary may also be subjected to Dirichlet on the part of the boundary and Neumann on the rest of the boundary. A problem defined using eq. (1) and eq. (2) is known as an interior Dirichlet problem. Similarly, a problem defined using eq. (1) and eq. (3) is known as an interior Neumann problem. A similar kind of problem may be defined for exterior domains and interfaces [41].

The fundamental solution $G(\mathbf{x}, \mathbf{y})$ of the PDE in eq. (1) satisfies

$$\mathscr{L}_{\mathbf{x}} G(\mathbf{x}, \mathbf{y}) = -\delta(\mathbf{x}, \mathbf{y}),\ \forall\ \mathbf{x} \in \Omega, \tag{4}$$

where $\mathbf{y}$ is the center of the function G. The fundamental solution is the Green's function on an infinite domain such that it satisfies the Sommerfeld radiation condition

$$\left(\frac{\mathbf{x}}{|\mathbf{x}|}, \nabla u(\mathbf{x})\right) - i\kappa u(\mathbf{x}) = o\left(\frac{1}{|\mathbf{x}|}\right), |\mathbf{x}| \to \infty, \tag{5}$$

where $\kappa = 0$ in the present case. The solution of eq. (1) with homogeneous Dirichlet boundary condition is given by

$$u(\mathbf{x}) = -\int_{\Omega} f(\mathbf{y})G(\mathbf{x},\mathbf{y})d\mathbf{y}, \tag{6}$$

where $G(\mathbf{x},\mathbf{y})$ also satisfies the homogeneous boundary condition i.e $G(\mathbf{x},\mathbf{y}) = 0,\ \forall\ \mathbf{x} \in \partial\Omega$ along with eq. (4). In the case of the non-homogeneous boundary conditions, we use the following definitions and theorems from the potential theory [41].

Definition 1. *Let g be a continuous function defined on $\partial\Omega$. The single layer potential*

$$\bar{u}(\mathbf{x}) := -\int_{\partial\Omega} g(\mathbf{y})G(\mathbf{x},\mathbf{y})dS(\mathbf{y}). \tag{7}$$

Similarly, the double-layer potential

$$\bar{\bar{u}}(\mathbf{x}) := -\int_{\partial\Omega} h(\mathbf{y})\frac{\partial G}{\partial \nu_y}(\mathbf{x},\mathbf{y})dS(\mathbf{y}), \tag{8}$$

where h is a continuous function defined on $\partial\Omega$, ν_y is the outward normal to the boundary at $\mathbf{y}$.

Theorem 1. *Given a PDE of the form $\mathscr{L}_{\mathbf{x}}u(\mathbf{x}) = f(\mathbf{x}),\ \forall\ \mathbf{x} \in \Omega$ with a non-homogenous boundary condition, the solution of the interior Dirichlet problem is given by*

$$u(\mathbf{x}) = \int_{\Omega} f(\mathbf{y})G(\mathbf{x},\mathbf{y})d\mathbf{y} + \bar{\bar{u}} \tag{9}$$

with

$$\lim_{\mathbf{x}\to\mathbf{x}_o^-} u(\mathbf{x}) = -\frac{1}{2}h(\mathbf{x_o}) + u(\mathbf{x_o}),\ \forall\ \mathbf{x}_o \in \partial\Omega. \tag{10}$$

where $\mathbf{x}_o^-$ mean converging in the interior of Ω. Similarly, for the interior Neumann problem, the solution is given by

$$u(\mathbf{x}) = \int_{\Omega} f(\mathbf{y})G(\mathbf{x},\mathbf{y})d\mathbf{y} + \bar{u} \tag{11}$$

with

$$\lim_{\mathbf{x}\to\mathbf{x}_o} u(\mathbf{x}) = u(\mathbf{x_o})\ \forall\ \mathbf{x}_o \in \partial\Omega. \tag{12}$$

We note that the $G(\mathbf{x}, \mathbf{y})$ in theorem 1 satisfies the homogenous boundary condition on $\partial\Omega$ along with eq. (4). However, one can also evaluate the solution using both the first layer and second layer potential

$$u(\mathbf{x}) = -\int_{\Omega} f(\mathbf{y}) G(\mathbf{x}, \mathbf{y}) d\mathbf{y} + \bar{\bar{u}} - \bar{u}, \tag{13}$$

where G is a domain independent Green's function that does not require to satisfy homogeneous boundary conditions. Furthermore, at the boundary, we use the condition in eq. (10) irrespective of the type of the boundary condition.

Traditional integral approaches require the analytical form of the Green's function to design quadratures. However, it is challenging to calculate an explicit expression for the Green's function, especially when the PDEs are defined in a complex domain or have variable coefficients. To circumvent this constraint, a kernel-free boundary integral methods have been proposed [15, 16] and are widely used to solve variable coefficient PDEs. However, this method does not produce any information about Green's function of the PDE. In the next section, we develop a neural network-based approach to tackle the problem.

3. Boundary integral network architecture

Recently, Lin et al. [30] proposed Boundary integral Network (BINet), where a known domain independent Green's function of the PDE is used, and the density function h or g in the definition 1 is learned using neural networks. In our method, we do not use the analytical Green's function and use three distinct networks to learn $G(\mathbf{x}, \mathbf{y})$, $h(\mathbf{x})$, and $g(\mathbf{x})$ functions using test functions. We call the proposed **B**oundary **I**ntegral **N**etwrok with unknown **G**reen's function as BIN-G. We use $G(\mathbf{x}, \mathbf{y}) \equiv G(\mathbf{x}, \mathbf{x}_c)$, where $\mathbf{x}_c$ is the center of the Green's function in the following discussion.

We consider an interior Neumann boundary value problem as shown in eq. (1) with $f(\mathbf{x}) = 0$ to simply the network architecture. In fig. 1, we show a typical network that evaluates the single-layer potential, which is the solution to the present problem. It takes the position as an input $\mathbf{x}$ and produces the value of the field $u(\mathbf{x})$ as an output. We generate the coordinates $\{\mathbf{x}^b = (x^b, y^b) | \mathbf{x}^b \in \partial\Omega\}$ by discretizing the boundary of interest using N_b points. The various layers involved in the network are shown as follows:

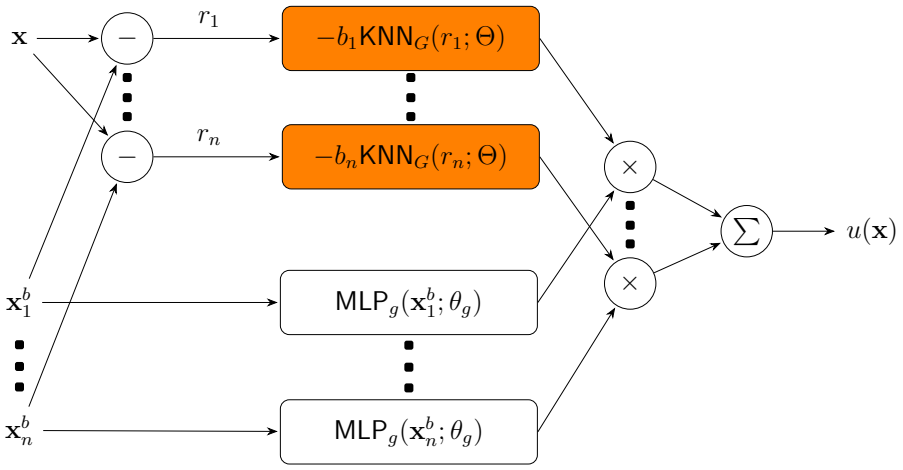


Figure 1: Network architecture of the BIN-G. Using the distance r_i between $\mathbf{x}$ and $\mathbf{x}_i^b$ enforcing the symmetry of the Green's function. The block highlighted in orange evaluates a domain-independent Green's function learned using a KNN.

1. The input are coordinates $\mathbf{x} \in \Omega$ at which one desire to obtained the solution.
2. The distance is calculated from all the boundary points $\{\mathbf{x}_1^b, \ldots, \mathbf{x}_n^b\}$ as an intermediate step in r. We perform this step to ensure the symmetry of the Green's function while training. One can ensure symmetry by employing a loss function that minimizes the interchanges of argument of the Green's function as well. Therefore, removing this step only slows down learning and requires additional terms in the loss function.
3. The boundary points are fixed. We input the boundary points to the multi-layer perceptron (MLP) network denoted by MLP_g to evaluate the density function g in the definition 1.
4. From the distance values r, the Green's function value is calculated using a kernel-based neural network (KNN) denoted by KNN_G discussed in the next section.
5. In the next two steps, numerical quadrature is performed using the evaluated Green's and density function. We use a three-point linear element and a four-point triangular element to perform numerical integration on the boundary and the volume, respectively.

The mathematical expression for the network shown in fig. 1 is

$$u(\mathbf{x}) = -\sum_{i}^{N_b} b_i \ \mathrm{MLP}_g(\mathbf{x}_i^b; \theta_g) \ \mathrm{KNN}_G(||\mathbf{x} - \mathbf{x}_i^b||; \Theta), \tag{14}$$

where b_i are the integration weights, $\mathbf{x}_i^b \in \partial\Omega$ and θ_g and Θ are the learnable parameters.

In the case of the Dirichlet boundary value problem, we use automatic differentiation to evaluate the gradient of the Green's function in the orange block. The mathematical expression of this kind of network is given by

$$u(\mathbf{x}) = -\sum_{i}^{N_b} c_i \ \mathrm{MLP}_h(\mathbf{x}_i^b; \theta_h) \ \mathbf{n}_i \cdot \nabla \mathrm{KNN}_G(||\mathbf{x} - \mathbf{x}_i^b||; \Theta), \tag{15}$$

where c_i are the integration weights, θ_h are learnable parameters, n_i are the outward normal at $\mathbf{x}_i^b \in \partial\Omega$, and the gradient is evaluated using automatic differentiation.

Similarly, in the presence of a non-zero forcing function $f(\mathbf{x})$, we add the volume integration $\int_\Omega G(\mathbf{x}, \mathbf{x}_c) f(\mathbf{x}_c) d\mathbf{x}_c$ after computing boundary integral. We discretize the volume using N_i quadrature points. The mathematical expression of this kind of network is given by

$$\begin{aligned} u(\mathbf{x}) = &\sum_{i}^{N_i} a_i f(\mathbf{x}_i) \mathrm{KNN}_G(||\mathbf{x} - \mathbf{x}_i||; \Theta) \\ &- \sum_{i}^{N_b} b_i \ \mathrm{MLP}_h(\mathbf{x}_i^b; \theta_h) \ \mathbf{n}_i \cdot \nabla \mathrm{KNN}_G(||\mathbf{x} - \mathbf{x}_i^b||; \Theta) \\ &- \sum_{i}^{N_b} c_i \mathrm{MLP}_g(\mathbf{x}_i^b; \theta_g) \mathrm{KNN}_G(||\mathbf{x} - \mathbf{x}_i^b||; \Theta), \end{aligned} \tag{16}$$

where a_i are the integration weights, $\mathbf{x}_i \in \Omega$. Since we focus on learning a domain agnostic Green's function, we use the architecture defined in eq. (16) following eq. (13) for all our test cases. In the next section, we discuss the Green's function network architecture.

3.0.1. Green's function network architecture

In order to evaluate Green's function for the elliptic operator $\mathscr{L}_\mathbf{x}$, we use a radial basis function (RBF) kernel-based neural network (KNN) [42, 43].

Any scalar function f defined in a domain Ω can be approximated using a RBF kernel K as

$$f(\mathbf{x}_i) = \sum_j w_j K(\mathbf{x}_i; \zeta_j, \lambda_j), \tag{17}$$

where w_j is the integration weights, ζ_j is the kernel center, and λ_j is the scaling parameter. Therefore, we can approximate Green's function as $G(\mathbf{x}, \mathbf{x}_c; \Theta) = \text{KNN}_G(r_i; \Theta)$, where

$$\text{KNN}_G(r_i; \Theta) = \sum_j w_j K(r_i; \zeta_j, \lambda_j), \tag{18}$$

where $r_i = ||\mathbf{x}_i - \mathbf{x}_c||$, and $\Theta = \{w_1, w_2, ..., w_n, \zeta_1, \zeta_2, ..., \zeta_n, \lambda_1, \lambda_2, .., \lambda_n\}$ are the learnable parameters. Equation (18) ensures that $G(\mathbf{x}, \mathbf{x}_c; \Theta) = G(\mathbf{x}_c, \mathbf{x}; \Theta)$. We note that n is a hyper-parameter denoting the number of approximating points (or particles). In fig. 2, we show the architecture of the proposed network. Since we convert all coordinates into the distances r from the center of the Green's function, we use a one-dimensional kernel

$$K(r_i; \zeta_i, \lambda_i) = \exp\left(-0.5\left(\frac{r_i - \zeta_i}{\lambda_i}\right)^2\right). \tag{19}$$

Since, the parameters λ and ζ has numerical origin, we can create a set of values such that a high slope function near singularity can be learn fast. We create n particles, exponential spaced in $[0, 3.0]$ with a linear increase of support radius in $[0.001, 0.2]$. The particles near zero are closely spaced with the lowest support radius. In the next section, we discuss the training process of all the networks in detail.

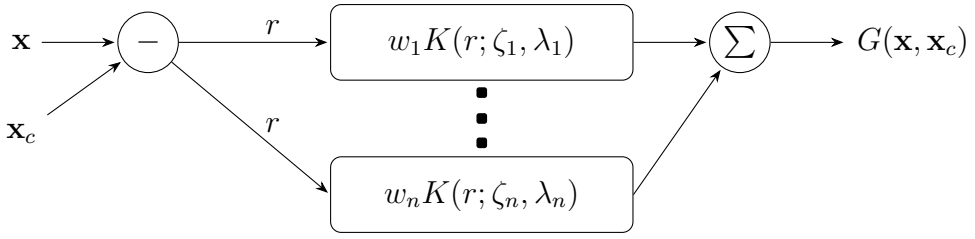


Figure 2: KNN-based architecture to evaluate domain-independent Green's function.

4. BIN-G training methodology

In this section, we discuss the method used to train BIN-G to learn the domain-independent Green's function for a given elliptic PDE. As discussed in section 2, the Green's function satisfies eq. (4). Therefore, it is logical to minimize the following loss,

$$\min_{\Theta} ||\mathscr{L}_{\mathbf{x}} G(\mathbf{x}, \mathbf{x}_c; \Theta) + \tilde{\delta}(\mathbf{x} - \mathbf{x}_c)||_{\Omega}^2, \tag{20}$$

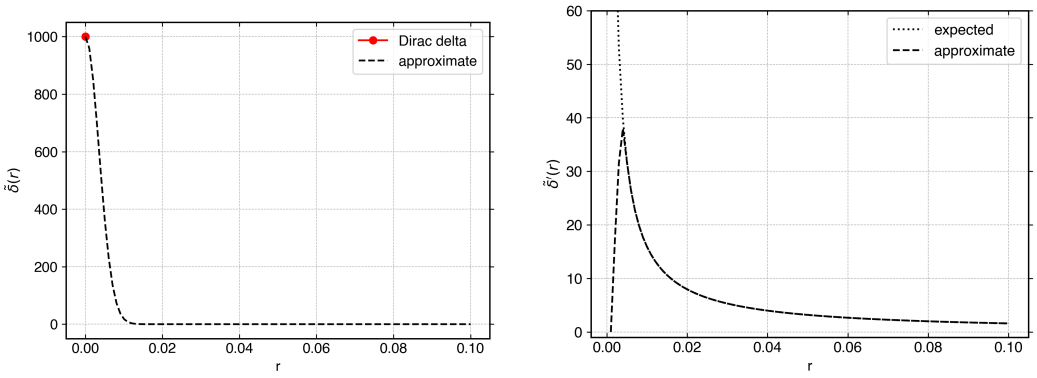
where $\tilde{\delta}$ is an approximation of the Dirac delta function. Teng et al. [34] proposed to use a Gaussian approximation for the Dirac delta and train the NN to learn the solution like a traditional MLP network. However, Lin et al. [30] showed that traditional MLPs are unable to converge for the Helmholtz equation. In other words, the presence of Gaussian approximation prevents the network from learning the desired slope near the singularity. In this paper, we mitigate this issue by learning the slopes near singularity using boundary integral formulation.

4.1. Green's function training scheme

We note that the NN cannot learn an infinite slope. Furthermore, the slope of the learned Green's function will be zero at the singularity due to continuity assured by the NNs. In fig. 3, we show a schematic plot for the Dirac delta approximation and the derivative of the Green's function. Therefore, it is very difficult to find an approximation of the Dirac delta such that it does not influence the Green's function, has a zero slope at the singularity, and has a very large slope close to the singularity. In order to remedy this problem, we satisfy the PDE away from singularity on a 1D domain for a large length. Additionally, we solve a boundary value problem for two distinct test functions on a finite domain.

In the fig. 4, we show different sample spaces used in the training of the Green's function. In the sample space shown in blue, we obtain the loss due to the PDE. Since the Green's function learned are radially symmetric, we can also obtain samples from a 1D domain shown in black color. Therefore, we obtain samples from a one-dimensional domain $\Omega_{1D} := [0, 3]$ to evaluate the residue of the PDE. However, we ignore samples in $S_\alpha = \{x | x - x_c < \alpha, x \in \Omega_{1D}\}$, where α is a hyper-parameter set to 0.01. We assume $x_c = 0.0$ in the 1D domain. The loss due to the PDE

$$\mathcal{L}_{PDE} = \left|\mathscr{L}_{\mathbf{x}} \mathrm{KNN}_G(\mathbf{x}, \mathbf{x}_c; \Theta)\right|^2, \tag{21}$$



(a) Dirac delta function

(b) Derivative of a typical Green's function

Figure 3: Comparison of the approximation of the Dirac delta and the expected derivative of the Green's function at the singularity.

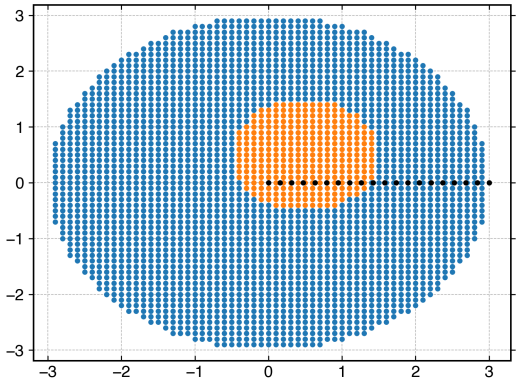


Figure 4: The sample space used in the training of the domain independent Green's function network. The domain used to compute the loss due to the PDE is a 1D domain shown in black, representing the domain in blue for a 2D PDE (due to symmetry of the Green's function). The orange domain is used to compute loss due to boundary integral equations.

where $\mathbf{x} \in \Omega_{1D} - S_\alpha$, Θ are the learnable parameters for the KNN. The function $|\bullet|$ is a discrete L_2 norm given by

$$\sqrt{\sum_i^N \frac{(\bullet)^2}{N}}, \tag{22}$$

where N is the number of samples.

In a 2D domain, much smaller than the blue domain, we evaluate the solution using the BIN-G as discussed in section 3. We use the test functions $\phi_1(\mathbf{x} \equiv (x, y)) = \sin(2\pi x)\sin(2\pi y)$ and $\phi_2(\mathbf{x} \equiv (x, y)) = \exp(-(x^2 + 2y^2 + 1))$ in the domain Ω in orange in fig. 4. The boundary condition and the forcing function can be obtained for both the test function using the domain information and the PDE. We employ a circular domain with radius 0.5 centered at $(0.5, 0.5)$. We sample N random points from the interior of the Ω and compute the mean squared error for prescribed test functions. The loss due to test function ϕ_1

$$\mathcal{L}_{BI_1} = \left|(u(\mathbf{x}; \theta, \Theta) - \phi_1(\mathbf{x})\right|^2, \tag{23}$$

and due to test function ϕ_2

$$\mathcal{L}_{BI_2} = \left|(u(\mathbf{x}; \theta, \Theta) - \phi_2(\mathbf{x})\right|^2, \tag{24}$$

where $u(\mathbf{x}_i)$ is evaluated using the BIN-G, $\theta = \{\theta_g, \theta_h\}$ and Θ are the learnable parameters for the MLP and KNN, respectively.

Combining all the component, the total loss used to train the BIN-G network is given by

$$\mathcal{L}(\theta, \Theta) = \beta\mathcal{L}_{PDE} + \gamma(\mathcal{L}_{BI_1} + \mathcal{L}_{BI_2}) + \eta\mathcal{L}_N, \tag{25}$$

where $\beta = 1.0$, $\gamma = [0.1, 1.0]$, and $\eta = [0.01, 0.1]$. We define

$$\mathcal{L}_N = |\mathrm{KNN}_G(\mathbf{x}, \mathbf{x}_c; \Theta)|^2 + |\mathrm{MLP}_h(\mathbf{x}_b; \theta_h)|^2 + |\mathrm{MLP}_g(\mathbf{x}_b; \theta_g)|^2, \tag{26}$$

a loss normalization of the function magnitudes. It ensures that the magnitude of the functions is minimal. This cannot be achieved by the usual L_2 normalization where the loss due to the magnitude of the parameter squared is minimized due to the presence of the KNN in the network, which assumes a large value of the parameters near the singularity. We note that the contribution of $\mathcal{L}_N$ is very low compared to other loss functions.

We train the NNs using Adam optimizer with a learning rate of 10^{-4} for the 10^5 epochs unless stated otherwise. Since the learned Green's function is radially symmetric, the 1D domain Ω_{1D} captures a larger domain with a much smaller number of samples. This reduces the training time significantly. We note that the Green's function training simultaneously trains the density functions h and g as well. However, one can readily employ the learned Green's function for any other domain and boundary condition using the method described in the next section.

5. Generalization to other domain and boundary conditions

In this section, we use the learned Green's function to obtain the solution to a problem with the same governing PDE but different domain and boundary conditions. We note that the Green's function is not in the exact analytical form, but rather parameterized by a neural network. Since it is a free-space Green's function, it can be employed to different domain and boundary conditions using eq. (13).

In order to learn the PDE solution for a new domain and boundary conditions, we only need to learn the density functions MLP_h and MLP_g in eq. (16), which are parameterized by multilayer perceptron neural networks. In the following, we describe the training of neural networks (MLP_h and MLP_g) for a interior Dirichlet problem. We use theorem 1 to obtain the loss function

$$\mathcal{L}(\theta) = |g^D(\mathbf{x}) - (u(\mathbf{x}, \theta) + 0.5h(\mathbf{x}, \theta))|^2, \ \forall \ \mathbf{x} \in \partial\Omega, \tag{27}$$

where $\mathbf{x}$ is the input, g^D is the given Dirichlet boundary condition and θ is the set of trainable parameters in the MLP in Figure 1.

In algorithm 1, we show the pseudo-code for the training procedure using learned Green's function. We need samples from the boundary only. In the `load_nn_network`, we load a trained KNN network which evaluated the GF for the given coordinates and center. In `initialize_h_network`, and `initialize_g_network`, we initialize the MLP_h and MLP_g network with uniformly random values. The `while` loop is the traditional training process where `initialize_grad` resets the gradient values to zero, and the `test_function(x, y)` are solution on the boundary for an interior Dirichlet problem. The `eval_sol` is the BIN-G network that utilizes the learned GF network and evaluates the PDE solution. We compute the loss in eq. (27)

in the variable `loss`, and use the Adams optimizer in `optimizer_step` as was done in the learning of the Green's function.

Algorithm 1: Pseudo-code to learn density function employing a leaned Green's function.

Input: $\{\{(x, y)|x, y \in \partial\Omega\}$
Result: trained KNN_G network

```
n_train = 10^5;
KNN_G = load_knn_network();
MLP_h = initialize_h_network();
MLP_g = initialize_g_network();
while i < n_train do
    initialize_grad();
    u = eval_sol(x, y, KNN_G, MLP_h, MLP_g);
    u_on_boundary = u + MLP_h(x, y)/2;
    loss = mean_squared(u_on_boundary - test_function(x, y));
    optimizer_step(loss, MLP_h, MLP_g, ...);
    i++;
end
```

The proposed method is a straightforward application of boundary integral formulation. However, one possible future endeavor is to express the boundary using a parameterized curve following the method proposed by Mezzadri et al. [44], and train the parameterized network using samples from different domain shapes, resulting from different curve parameters.

6. Results and discussion

In this section, we demonstrate the applicability of the proposed method to solve various elliptic partial differential equations. We first learn the Green's function for the Laplace equation. We then learn the Green's function for the Helmholtz equation for different k values. Finally, we learn the Green's function of a variable coefficient elliptic PDE. For all the test cases, we readily employ the trained Green's function network to train the density functions MLP_g and MLP_h to learn the solution on a different domain and boundary conditions. In order to verify the learned GF, we use a new test function

$$u(\mathbf{x} \equiv (x, y)) = \exp(-x)\cos(y) + \exp(-y)\sin(x), \tag{28}$$

on a square-shaped domain of unit length for all the test cases unless stated otherwise. We use the test function in eq. (28) to obtain the forcing function and boundary condition and then use that as prescribed data to the network to evaluate the test function. All the NN present in this work are implemented using the open source `pytorch` [45] package.

For all test cases, we consider the KNN_G network discussed in section 3.0.1 with 400 approximation points, which results in 1200 parameters to be trained. On the testing data, for all the results, we report relative L_2 error

$$L_2 = \left(\frac{\sum_i^N (f(\mathbf{x}_i) - f_o(\mathbf{x}_i))^2}{\sum_i^N (f_o(\mathbf{x}_i))^2} \right)^{\frac{1}{2}}, \tag{29}$$

where N is the number of test points, f_o is the known test function and f is the value obtained using the BIN-G. We run all the simulations on a "Apple M2 Ultra Chip". It takes 5.7 hours to train the Green's function for a variable coefficient PDE. However, for the constant coefficient PDEs since we use a $1D$ sample space, it learns the GF in 3 hours. Furthermore, for all the cases it takes approximately 9.5 minutes to retrain the density functions using the learned GF.

6.1. Laplace equation

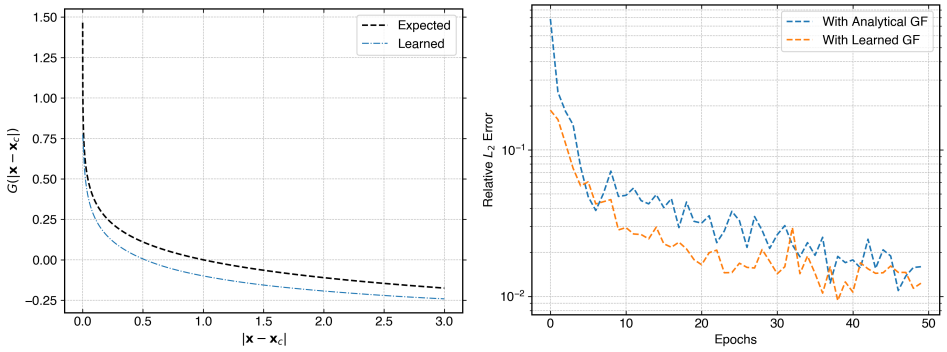
In this section, we first solve the Laplace equation. The fundamental solution of the Laplace equation is given by $G(\mathbf{x}, \mathbf{x}_c) = 1/2\pi \ln(|\mathbf{x} - \mathbf{x}_c|)$. We consider a test function in eq. (28) such that

$$\nabla^2 u(\mathbf{x}) = f(\mathbf{x}), \tag{30}$$

where $f(x) = 0$. In fig. 5a, we plot the learned Green's function against the analytical result. The learned Green's function is slightly shifted. However, the shift does not affect the solution in the case of the Laplace equation. We employ the learned Green's function to train the density function networks MLP_h and MLP_g for the test function in eq. (28). In fig. 5b, we show the L_2 error with the number of epochs showing convergence when an analytical and a learned Green's function is employed in the BIN-G. The error in the test data shows a similar trend for a learned Green's function compared to an analytical Green's function.

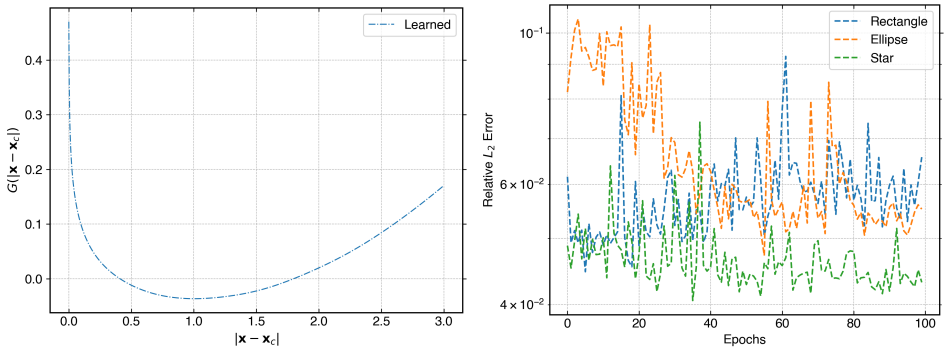
We next consider a Laplace equation with variable coefficient

$$\nabla \cdot (\sigma(\mathbf{x}) \nabla u(\mathbf{x})) = f(\mathbf{x}), \tag{31}$$



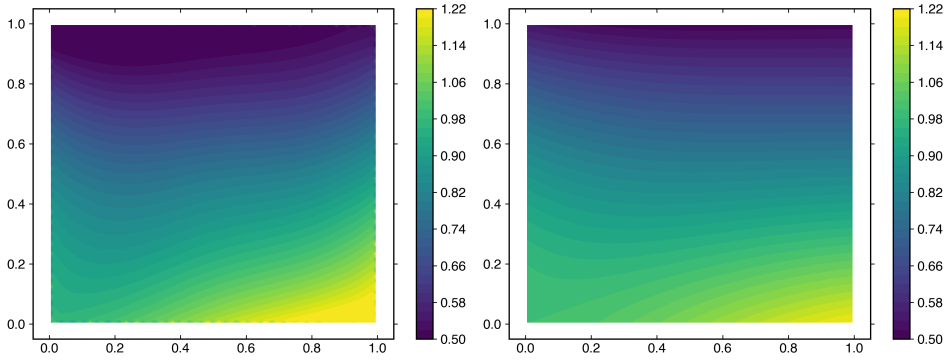
(a) Green's function (b) L_2 error on test data

Figure 5: Comparison of the learned and analytical Green's functions and convergence of L_2 error in the test data while training density function using learned and analytical Green's function of Laplace equation.



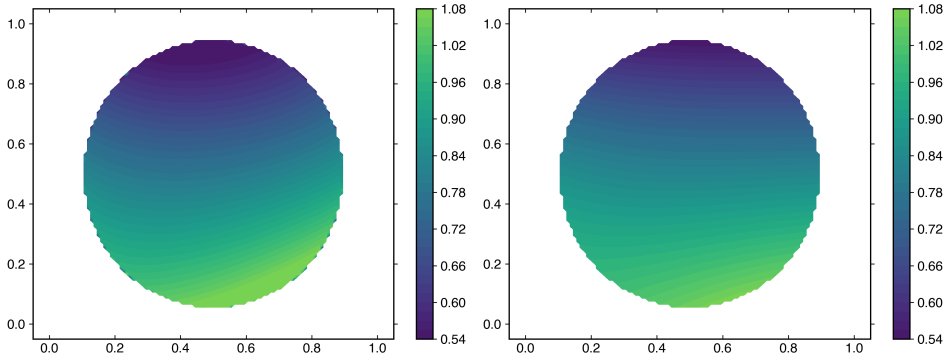
(a) Green's function (b) L_2 error on test data

Figure 6: Learned Green's function and the L_2 error on test data while training using learned Green's function on different domain shapes for the Laplace equation with variable coefficient.



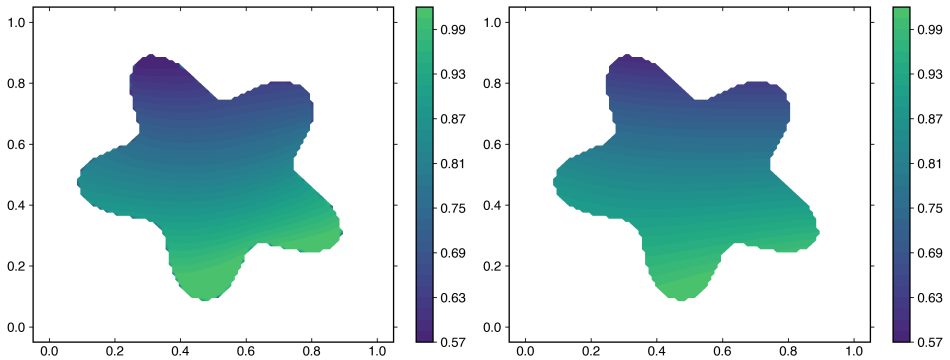
(a) Learned distribution

(b) Exact distribution



(c) Learned distribution

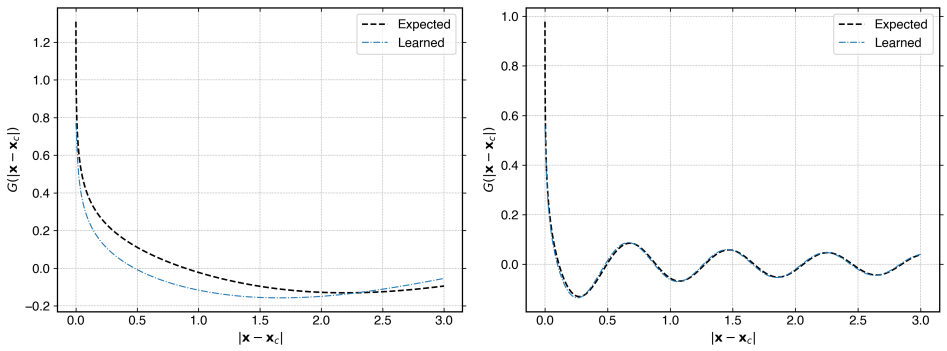
(d) Exact distribution



(e) Learned distribution

(f) Exact distribution

Figure 7: The contour of the learned and expected field using BIN-G for the Laplace equation with variable coefficient.



(a) k=1 (b) k=8

Figure 8: Learned Green's function compared with the analytical Green's function for the Helmholtz equation.

where $\sigma(\mathbf{x}) = 1.5 + 0.5(\sin(x) + \cos(y))$. We train the Green's function network using the same methodology and use the learned Green's function to learn density functions for new domain shapes and test function in eq. (28). We consider 3 shapes of the domain viz. rectangular, circular, and star shape. In fig. 6, we plot the learned Green's function and the convergence of the L_2 error on test samples from the domains. The L_2 error is within 6%. In case of a rectangular domain the errors are higher due to corner that have a discontinuity in the boundary normals. In fig. 7, we plot the test function learned and exact distribution. The learned contour are close to the exact distribution.

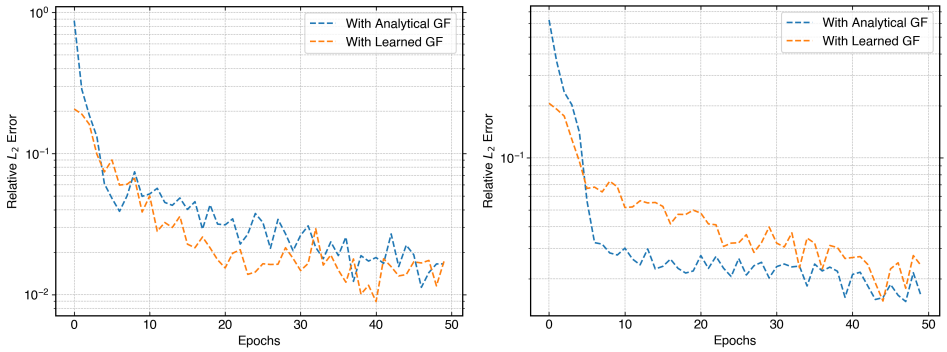
6.2. Helmholtz equation

In order to show the capability and ensure the correctness of the proposed method, we solve the Helmholtz equation for real values. We first learn the Green's function for the Helmholtz equation

$$\nabla^2 u(\mathbf{x}) + k^2 u(\mathbf{x}) = f(\mathbf{x}), \tag{32}$$

where k is the eigenvalue. We choose $k = \{1, 8\}$. We first train the network to learn the Green's function for the PDE and then employ it to learn the solution using BIN-G for the test function in eq. (28).

In fig. 8, we show the comparison of the learned Green's function with respect to the analytical form. The learned Green's function shows a close match with the analytical form for $k = 8$ compared to $k = 1$. In fig. 9, we show the comparison of the relative L_2 error in the test data while training the



(a) k=1 (b) k=8

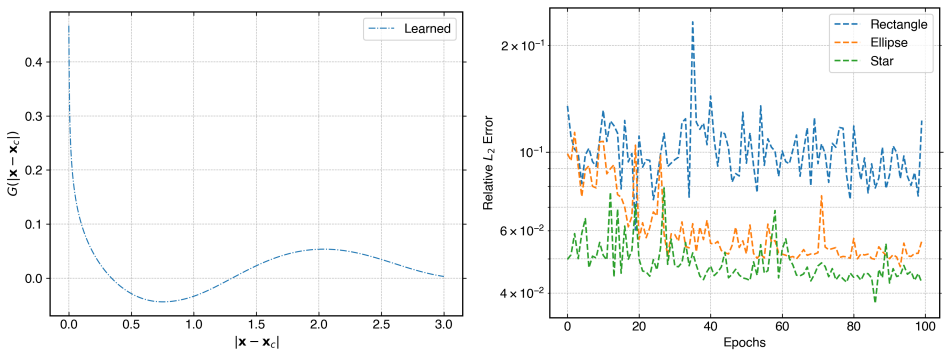
Figure 9: L_2 error on test data using BIN-G with learned Green's function and analytical Green's function for the Helmholtz equation.

density function using an analytical Green's function and the learned Green's function. The learned Green's function shows a trend close to the error using an analytical Green's function in the BIN-G. In the case of $k = 1$, the gap between the learned Green's function and the actual Green's function is high. However, the L_2 error while learning density function is close to when an analytical Green's function is used. This shows that the lower eigensolution are more likely to converge to wrong solution. We suggest to use a larger domain for lower eigenvalue PDEs.

We also consider a variable coefficient Helmholtz equation with constant eigenvalue given by

$$\nabla \cdot (\sigma(\mathbf{x})\nabla u(\mathbf{x})) + k^2 u(\mathbf{x}) = f(\mathbf{x}), \tag{33}$$

where we set $k = 4$. We learn the Green's function for the above PDE using the present method and employ it to learn the solution for the test function as done in previous test cases. However, we consider different domain shapes viz. rectangular, ellipse, and star. In fig. 10, we plot the learned Green's function and the relative L_2 error in the test data while training for different domains. As observed in the case of the Laplace equation, the errors in the case of the rectangular domain is high due to corners. The error in the case of Elliptical and star shaped domain are within 5%. These errors can be improved by using a better quadrature, larger domain, and more number of source-test function pairs. In fig. 11, we plot the learned solution and exact distribution for different domains. The higher error in the case of the rectangular domain is visible by higher values in the distribution. However,



(a) Green's function (b) L_2 error on test data

Figure 10: Learned Green's function and the L_2 error on test data while training using learned Green's function on different domain shapes for the Helmholtz equation with variable coefficient and $k = 4$.

in the case of the other two domains the contours are close to the expected distribution.

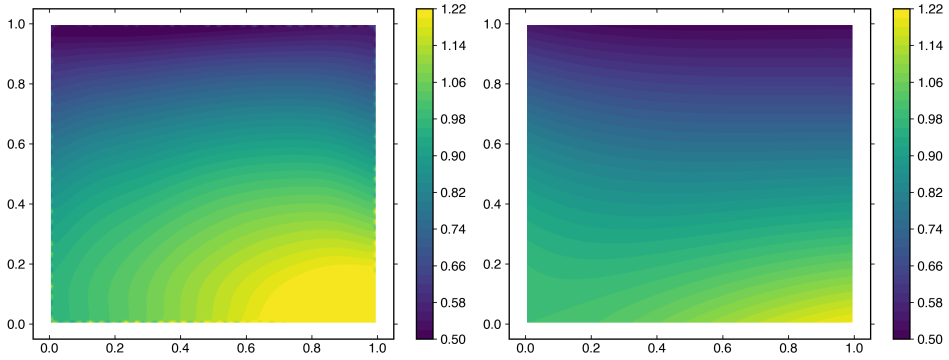
6.3. PDE with variable coefficient

In this section, we solve a variable coefficient elliptic PDE [16] given by

$$\nabla \cdot (\sigma(\mathbf{x})\nabla u(\mathbf{x})) - \kappa(\mathbf{x})u(\mathbf{x}) = f(\mathbf{x}), \tag{34}$$

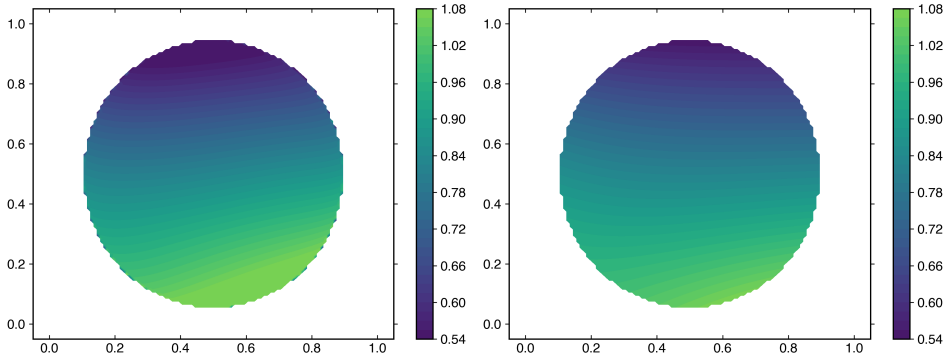
where $\sigma(x, y) = 1.5 + 0.5(\sin(x) + \cos(y))$, and $\kappa(x, y) = 20 + \exp(1.5x + 1.8y)$. We note that an analytical form of Green's function for variable coefficient PDEs are not known. We learn the Green's function for the PDE in eq. (34) as done in previous test case. We use the learned Green's function to solve PDE for an arbitrary test function on different domain shapes.

In fig. 12a, we plot the learned Green's function. We plot the L_2 error in the test points using the learned Green's function for different domain shapes as done in previous test cases. Since the κ value is variable it involves lower eigenvalues as well. Therefore, the errors are higher compared to other variable coefficient test cases. In the case of the rectangular domain, the errors are higher due to corners compared to other domain shapes. In fig. 13, we plot the learned solution with the exact distribution. Similar to other test case, the rectangular domain shows a different contour line and magnitude. However, in case of smooth domains the contours are close to the exact distribution.



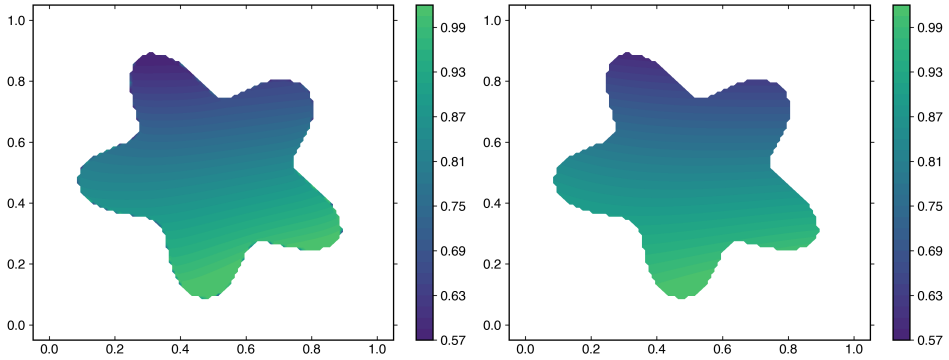
(a) Learned distribution

(b) Exact distribution



(c) Learned distribution

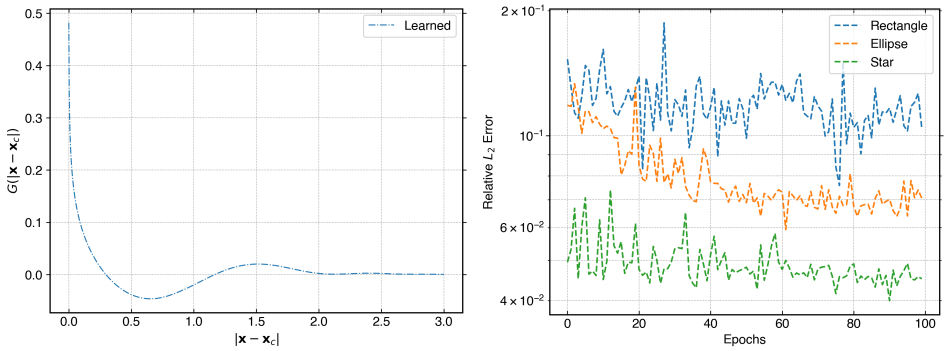
(d) Exact distribution



(e) Learned distribution

(f) Exact distribution

Figure 11: The contour of the learned field using BIN-G and the error distribution for the Helmholtz equation with variable coefficient and $k = 4$.



(a) Green's function (b) L_1 error on test data

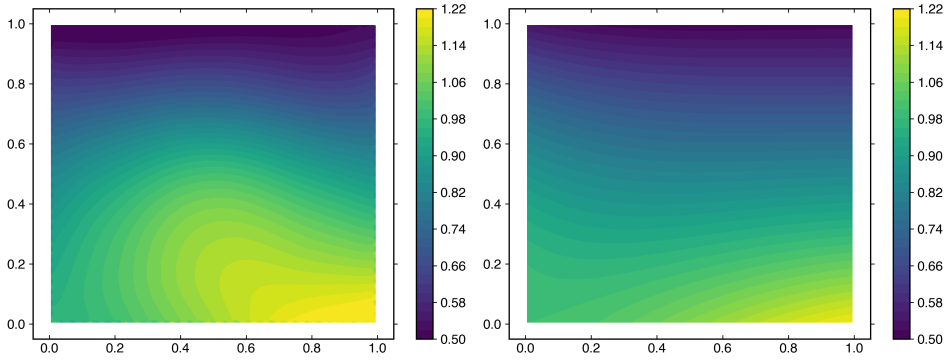
Figure 12: Learned Green's function and the L_2 error on test data while training using learned Green's function on different domain shapes for the elliptical equation with variable σ and κ.

7. Conclusions

In this paper, we propose a novel boundary integral network with unknown Green's function referred to as BIN-G. In the BIN-G, we employ a radial basis function (RBF) kernel-based neural network to evaluate the Green's function, and multi-layer perceptron networks to evaluate the density functions. The parameters of the RBF-based Green's function network viz. particle position and smoothing length are closely related to the sampling space. Therefore, a careful initialization of the particle position and smoothing length offer faster learning near the singularity of the Green's function. We use the spherical symmetry feature of the domain-independent Green's function in our network enabling us to learn Green's function using a one-dimensional sample space.

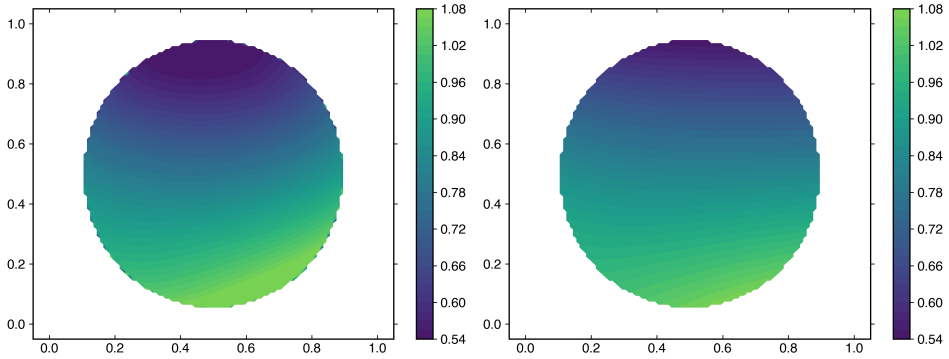
We train the neural network by simultaneously minimizing the residual of the PDE and the mean-squared error of the solution using the boundary integral equations. We use a much larger sample space on which PDE residual is minimized, compared to the domain on which solution to prescribed test functions is learned using the BIN-G. Therefore, the learned Green's function is defined for a very large space and is not constrained by the homogeneous boundary condition. Hence, it can be readily employed for any domain shape, boundary condition, and forcing function.

The efficacy of the proposed method to solve PDEs, for which the ground truth Green's functions are known, has been demonstrated. We use the



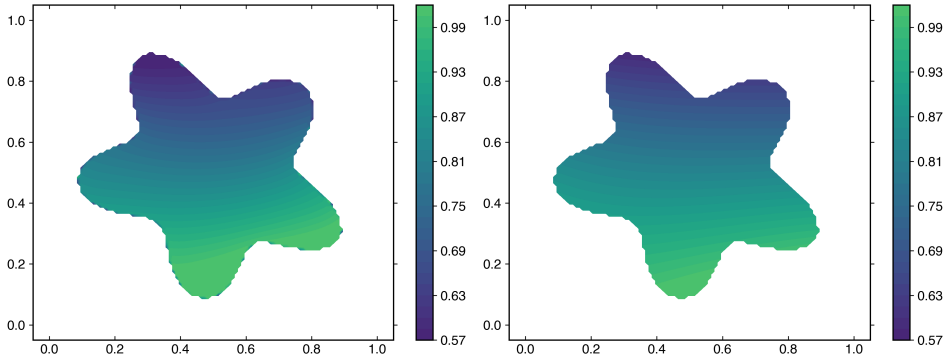
(a) Learned distribution

(b) Exact distribution



(c) Learned distribution

(d) Exact distribution



(e) Learned distribution

(f) Exact distribution

Figure 13: The contour of the learned field using BIN-G and the error distribution for the Helmholtz equation with variable variable σ and κ.

Laplace and Helmholtz equations to validate our method. We apply the method to learn domain-independent Green's function of variable coefficient PDEs, for which analytical forms of Green's function are not available. We demonstrate the applicability of the proposed network by solving a different problem on different domain shapes using the learned Green's function.

It is possible to enhance the obtained accuracy through the implementation of an adaptively sampled domain, strategically placing refined samples in proximity to the boundary. This approach not only refines training around singularities but also mitigates quadrature errors. However, these modifications give rise to convergence challenges, which can be addressed in the future. The proposed method can be extended to solve interface boundary problems as well. In the future, we would like to extend the proposed method to solve moving interface boundary problems.

8. Acknowledgements

S. L and M. K acknowledge the support from the National Science Foundation, ECCS-1927432. S. L also acknowledges the partial support from the National Science Foundation, grants DMS-1720420 and DMS-2309798. M. C thanks NSF-1936873 and 2027725 for partial supports.

References

1. Wrobel, L.. The Boundary Element Method, Volume 1: Applications in Thermo-Fluids and Acoustics. The Boundary Element Method; Wiley; 2002. ISBN 9780471720393. URL: `https://books.google.com/books?id=kxdEu2PJ6qwC`.

2. Crouch, S.L., Starfield, A.M., Rizzo, F.. Boundary element methods in solid mechanics. George Allen & Unwin, London; 1983.

3. Strain, J.. A boundary integral approach to unstable solidification. *Journal of computational physics* 1989;85(2):342–389.

4. Sidi, A., Israeli, M.. Quadrature methods for periodic singular and weakly singular fredholm integral equations. *Journal of Scientific Computing* 1988;3:201–231.

5. Hou, T.Y., Zhang, P.. Convergence of a boundary integral method for 3-d water waves. *Discrete and Continuous Dynamical Systems Series B* 2002;2(1):1–34.

6. Zhao, X., Ma, W., Wang, K.. Simulating laser-fluid coupling and laser-induced cavitation using embedded boundary and level set methods. *Journal of Computational Physics* 2023;472:111656.

7. Kress, R.. On the numerical solution of a hypersingular integral equation in scattering theory. *Journal of computational and applied mathematics* 1995;61(3):345–360.

8. Greenbaum, A., Greengard, L., McFadden, G.. Laplace's equation and the dirichlet-neumann map in multiply connected domains. *Journal of Computational Physics* 1993;105(2):267–278.

9. Mogilevskaya, S., Crouch, S.L.. A galerkin boundary integral method for multiple circular elastic inclusions. *International Journal for Numerical Methods in Engineering* 2001;52(10):1069–1106.

10. Geng, W., Krasny, R.. A treecode-accelerated boundary integral poisson–boltzmann solver for electrostatics of solvated biomolecules. *Journal of Computational Physics* 2013;247:62–78.

11. Greengard, L., Gueyffier, D., Martinsson, P.G., Rokhlin, V.. Fast direct solvers for integral equations in complex three-dimensional domains. *Acta Numerica* 2009;18:243–275.

12. Beale, J.T., Lai, M.C.. A method for computing nearly singular integrals. *SIAM Journal on Numerical Analysis* 2001;38(6):1902–1925.

13. Duffy, D.G.. Green's functions with applications. cRc press; 2015.

14. Evans, L.C.. Partial differential equations, ams. *Graduate Studies in Mathematics* 2010;19:749.

15. Ying, W., Henriquez, C.S.. A kernel-free boundary integral method for elliptic boundary value problems. *Journal of Computational Physics* 2007;227(2):1046–1074. doi:10.1016/j.jcp.2007.08.021.

16. Cao, Y., Xie, Y., Krishnamurthy, M., Li, S., Ying, W.. A kernel-free boundary integral method for elliptic PDEs on a doubly connected domain. *Journal of Engineering Mathematics* 2022;136(1):2. doi:10.1007/s10665-022-10233-8.

17. Xie, Y., Li, S., Ying, W.. A fourth-order kernel-free boundary integral method for interface problems. *Communications in Computational Physics* 2023;33(3):764–794.

18. Dong, H., Li, S., Ying, W., Zhao, Z.. Kernel-free boundary integral method for two-phase stokes equations with discontinuous viscosity on staggered grids. *Journal of Computational Physics* 2023;492:112379. URL: https://www.sciencedirect.com/science/article/pii/S0021999123004746. doi:https://doi.org/10.1016/j.jcp.2023.112379.

19. Zhao, Z., Dong, H., Ying, W.. Kernel free boundary integral method for 3d incompressible flow and linear elasticity equations on irregular domains. *Computer Methods in Applied Mechanics and Engineering* 2023;414:116163.

20. Zhou, H., Yang, J., Ying, W.. A kernel-free boundary integral method for the nonlinear poisson-boltzmann equation. *Journal of Computational Physics* 2023;:112423.

21. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research* 2018;18:1–43.

22. Lei, H., Wu, L., Weinan, E.. Machine-learning-based non-newtonian fluid model with molecular fidelity. *Physical Review E* 2020;102(4):043309.

23. Lei, H., Xie, P., Zhang, L., et al. Machine learning-assisted multi-scale modeling. *Journal of Mathematical Physics* 2023;64(7).

24. Qiu, C., Yan, J.. Cell-average based neural network method for hyperbolic and parabolic partial differential equations. *arXiv preprint arXiv:210700813* 2021;.

25. Zhao, X.E., Hao, W., Hu, B.. Two neural-network-based methods for solving elliptic obstacle problems. *Chaos, Solitons & Fractals* 2022;161:112313.

26. Zhao, X.E., Hao, W., Hu, B.. Convergence analysis of neural networks for solving a free boundary problem. *Computers & Mathematics with Applications* 2021;93:144–155.

27. Karniadakis, G.E., Jagtap, A.D.. Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Nonlinear Partial Differential Equations. *Communications in Computational Physics* 2020;28(5):2002–2041. doi:10.4208/cicp.OA-2020-0164.

28. Tseng, Y.H., Lin, T.S., Hu, W.F., Lai, M.C.. A cusp-capturing pinn for elliptic interface problems. *Journal of Computational Physics* 2023;491:112359.

29. Sheng, H., Yang, C.. PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries. *Journal of Computational Physics* 2021;428:110085. doi:10.1016/j.jcp.2020.110085.

30. Lin, G., Hu, P., Chen, F., Chen, X., Chen, J., Wang, J., Shi, Z.. Binet: Learn to solve partial differential equations with boundary integral networks. *CSIAM Transactions on Applied Mathematics* 2023;4(2):275–305. doi:https://doi.org/10.4208/csiam-am.SO-2022-0014.

31. Sukumar, N., Srivastava, A.. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering* 2022;389:114333. doi:10.1016/j.cma.2021.114333. arXiv:2104.08426.

32. Lin, G., Chen, F., Hu, P., Chen, X., Chen, J., Wang, J., Shi, Z.. BI-GreenNet: learning green's functions by boundary integral network. *Communications in Mathematics and Statistics* 2023;11(1):103–129.

33. Boullé, N., Earls, C.J., Townsend, A.. Data-driven discovery of Green's functions with human-understandable deep learning. *Scientific Reports* 2022;12(1):4824. doi:10.1038/s41598-022-08745-5.

34. Teng, Y., Zhang, X., Wang, Z., Ju, L.. Learning green's functions of linear reaction-diffusion equations with application to fast numerical solver. In: *Mathematical and Scientific Machine Learning*. PMLR; 2022:1–16.

35. Peng, R., Dong, J., Malof, J., Padilla, W.J., Tarokh, V.. Deep Generalized Green's Functions. 2023. arXiv:2306.02925.

36. Zhao, M., Ying, W., Lowengrub, J., Li, S.. An efficient adaptive rescaling scheme for computing moving interface problems. *Communications in Computational Physics* 2017;21(3):679–691.

37. Liu, K., Marple, G.R., Allard, J., Li, S., Veerapaneni, S., Lowengrub, J.. Dynamics of a multicomponent vesicle in shear flow. *Soft matter* 2017;13(19):3521–3531.

38. Xie, Y., Li, S., Ying, W.. A fourth-order cartesian grid method for multiple acoustic scattering on closely packed obstacles. *Journal of Computational and Applied Mathematics* 2022;406:113885.

39. Pham, K., Turian, E., Liu, K., Li, S., Lowengrub, J.. Nonlinear studies of tumor morphological stability using a two-fluid flow model. *Journal of mathematical biology* 2018;77:671–709.

40. Feng, H., Barua, A., Li, S., Li, X.. A parallel adaptive treecode algorithm for evolution of elastically stressed solids. *Communications in Computational Physics* 2014;15(2):365–387.

41. Kellogg, O.D.. Foundations of potential theory; vol. 31. Courier Corporation; 1953.

42. E, W., Ma, C., Wu, L.. Machine Learning from a Continuous Viewpoint. *Science China Mathematics* 2020;63(11):2233–2266. doi:10.1007/s11425-020-1773-8. arXiv:1912.12777.

43. Ramabathiran, A.A., Ramachandran, P.. SPINN: Sparse, Physics-based, and Interpretable Neural Networks for PDEs. *Journal of Computational Physics* 2021;doi:https://doi.org/10.1016/j.jcp.2021.110600.

44. Mezzadri, F., Gasick, J., Qian, X.. A Framework for Physics-Informed Deep Learning Over Freeform Domains. *Computer-Aided Design* 2023;160:103520. doi:10.1016/j.cad.2023.103520.

45. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.. Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., eds. *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc.; 2019:8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.