DEEP GENERALIZED GREEN'S FUNCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The Green's function has ubiquitous and unparalleled usage in the efficient problem-solving of partial differential equations (PDEs) and analyzing systems governed by PDEs. However, obtaining a closed-form Green's function for most PDEs on various domains is often impractical. The numerical Green's function (NGF) method seeks an approximate Green's function using traditional methods, like finite element analysis, especially for complex problems in fracture mechanics and dynamic scattering. We introduce the Deep Generalized Green's Function (DGGF), a deep-learning approach that addresses the challenges of problem-specific modeling, long time requirements, and data storage demands associated with NGF. Our method efficiently solves PDE problems using an integral format of solutions. It outperforms direct methods, such as FEM and physics-informed neural networks (PINNs). Additionally, our method relieves the training burden and scales to higher dimensions. Unlike the direct Gaussian approximation of a Dirac delta function, our method can be used to solve PDEs in higher dimensions. Because our method directly addresses the singularity, it can be used to solve different PDEs without prior knowledge. Unlike BI-GreenNet, which is limited to PDEs with known expressions of the singular part of the Green's function, our method does not require prior knowledge of the singularity. The results confirm the advantages of DGGFs and the benefits of Generalized Greens Functions as a novel and effective approach to solving PDEs without suffering from singularities.

1 INTRODUCTION

Efficiently solving partial differential equations (PDEs) is a major area of study across science and engineering. Closed-form solutions to PDEs offer the greatest efficiency, however, they have only been obtained for a small set of relatively simple problems. Consequently, PDEs are most often solved using computationally expensive numerical methods such as the finite difference method (FDM) or the finite element method (FEM). The computational burden of these methods has motivated investigation of more computationally efficient methods to solve challenging PDEs. In recent years, deep neural networks (DNNs) have shown potential as a promising alternative PDE-solving method, which can be mesh-free and data-driven. The most notable DNN-based PDE solving technique is the physics-informed neural network (PINN) (Raissi et al., 2019; Cuomo et al., 2022b; Jin et al., 2021b; Cai et al., 2021), in which a DNN represents the PDE solution function, and the PDE constraints serve as the training loss. Other methods are also proposed to integrate different PDE-solving methods with deep learning, include DeepRitz (Yu et al., 2018), Deep Galerkin Methods (Sirignano & Spiliopoulos, 2018) which solves the variant forms of the problem instead of the original strict differential form.

Despite their potential benefits, a significant drawback of DNN-based strategies lies in their training cost, which remains significant compared to FEM approaches. Data-driven methods, like Fourier Neural Operator (FNO) (Li et al., 2020), or DeepONet(Lu et al., 2021), learn an operator that maps coefficients, excitation functions, or domain shape to the solution functions, which solves wide range of problems with one pass of training. However, these data-driven methods require sufficient data from simulations or experiments, which may be unavailable in the first place.

One potential pathway to overcome the computational challenges of PDEs is the use of Green's functions (GFs), wherein our goal is to find an integral kernel - termed the GF - which then can be used to solve a given PDE problem using an integral over the domain of the PDE. The GF approach thereby avoids the need to use a mesh and invert large matrices, as is required by FEM

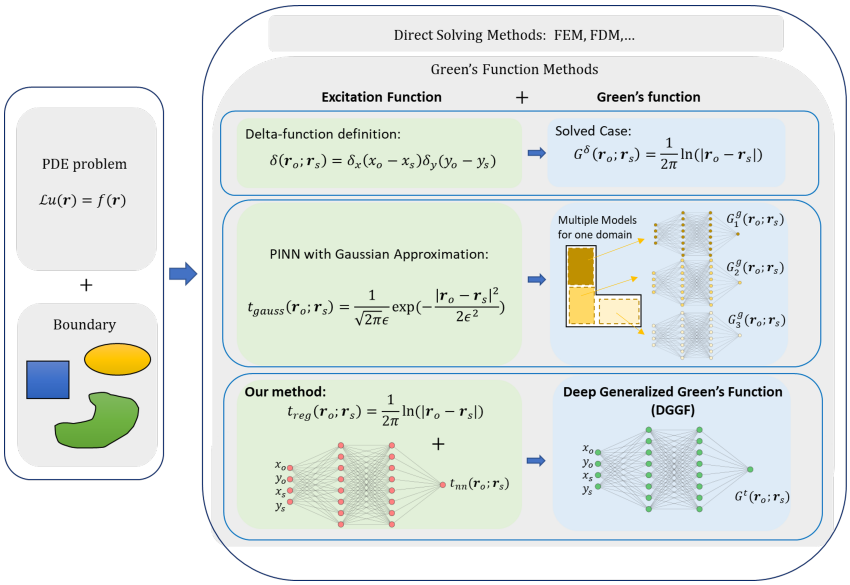

Figure 1: Schematic showing two types of PDE solvers: Direct Methods solve each PDE problem individually, and the Green's function method retrieves a kernel function first and constructs the solution in convolutional forms.

and FDM. Additionally, once a GF is computed for a given differential operator it can be re-used to efficiently solve a whole class of PDEs (e.g., those with different boundary conditions (BCs), or stimulus functions - see Sec. X). Despite the significant advantages of GFs, only a small number of differential operators have a (known) closed-form GF (e.g., two-dimensional (2-D) Poisson equation on a square), greatly limiting their use to solve practical PDE problems. Numerical methods for GFs (NGFs) have also been investigated however, these methods suffer from high computational costs, often sacrifice accuracy, or may require multiple stages of modeling.

Recently, DNNs have also been investigated to learn the GF. Like other DNN-based approaches, these methods are mesh-free and can potentially learn GFs for challenging real-world problems. One major challenge for DNN-based approaches to solving PDEs is the presence of singularities, especially the Dirac delta function. To approximate a GF, the differential operator applied to the DNN must produce a singularity, which requires unbounded dense sampling near these singularities. Some recent DNN-based approaches have been proposed to overcome this problem. One is GF-Net (Teng et al., 2022), which utilizes Gaussians to approximate the Dirac delta function. By employing smooth Gaussians, GF-Net significantly relaxes the Dirac delta function singularity. However, this method represents one Green's function on a single domain with multiple networks, making it challenging to apply to larger domains. Training may be complex depending on the desired accuracy level, and approximation errors may be of concern. The BI-GreenNet method is another important technique for expressing the non-singular part of Green's functions on different domains using DNNs (Lin et al., 2023). This method takes advantage of the linearity of the Green's function and performs well when the singular part of the Green's function can be computed analytically. However, a limitation is the need for an analytic solution for the singular part.

To overcome the challenges presented by the singularity of GF here we propose and demonstrate a generalized *Green's function* method and its DNN-based representation – termed *Deep Generalized Green's function* (DGGF). The DGGF addresses the singularity issue while inheriting all the benefits of both the mesh-free neural network method and using the Green's function to solve linear PDE problems. Using a number of benchmark PDE problems, we demonstrate that the DGGF method provides stable solutions, exhibits a fast convergence rate, and achieves high accuracy.

Our contributions are summarized below:

- We construct DGGFs that are applicable to all linear PDEs, and amenable to stable and efficient numerical computation. We prove that DGGFs maintain desirable usages of the original Green's function,

- We demonstrate achieved accuracy in constructing PDE solutions using DGGFs for four different types of PDEs in various dimensions and for different boundary shapes and conditions, and
- We experimentally demonstrate that the DGGF yields faster neural network training with superior performance compared to state of the art DNN based Green's function approaches.

2 Related Works

Traditional methods include Fourier and Laplace Transform Methods, FEM and FDM. The idea of using a neural network to efficiently solve PDEs, to the best of our knowledge, dates back to the 1990s (Chen & Chen, 1995; Lagaris et al., 1998; Lee & Kang, 1990). However, the lack of computational power and auto-differentiation capabilities severely limited the efficacy of neural networks in this domain. More recently, there has been a resurgence of the application of neural networks in solving PDEs due to works on physics-informed neural networks (PINNs) (Ren et al., 2022) in various scientific fields such as fluid dynamics (Wang et al., 2020; Jin et al., 2021a), thermodynamics (Zobeiry & Humfeld, 2021), and electromagnetism (Baldan et al., 2021). A detailed discussion of PINNs and their applications can be found in (Cuomo et al., 2022a). Several recent studies have proposed PINN improvements including reweighting loss terms in the objective function (Wang et al., 2021), new loss terms (Chiu et al., 2022), and more complicated neural network structures (Ren et al., 2022). Different DNN-based PDE solving methods are listed and compared in the Table 4. Another notable method for solving PDEs is DeepRitz (Yu et al., 2018), which solves the variational form of the original strict differential form, resulting in relaxed constraints and often more robust convergence to the solution function. However, the main limitation of this approach to calculating the Green's function is the time-consuming process of evaluating a multidimensional integral at every epoch of the training process. To the best of our knowledge, there have been a limited number of studies for approximating the Green's function with DNNs. A notable approach, referred to as GF-Net and described in (Teng et al., 2022), utilizes Gaussians to approximate the Dirac delta function. By employing smooth Gaussians, this method significantly relaxes the Dirac delta function singularity. The use of Gaussian functions offers a systematic approach to achieve high accuracy by reducing the Gaussian width. However, this method represents one Green's function on a single domain with multiple networks, making it challenging to apply to larger domains. Training may also be complex depending on the desired accuracy level, and approximation errors may be of concern.

The BI-GreenNet method is another important technique for expressing the non-singular part of Green's functions on different domains using DNNs (Lin et al., 2023). This method takes advantage of the linearity of the Green's function and performs well when the singular part of the Green's function can be computed analytically. However, a limitation of this method is the need for an analytic solution for the singular part.

Various related data-driven approaches (Gin et al., 2021; Boullé et al., 2022b;a) learn the Green's function of an unknown system by incorporating it into a convolution integral and attempting to construct the solution function. These methods address a different type of problem, namely inferring the properties of an unknown system, instead of directly solving the underlying PDE. By utilizing the Green's function in a convolutional form, these approaches eliminate the need for evaluating the Dirac delta function. However, they may require fine sampling of the input function making them potentially challenging to implement in various scenarios of interest.

3 Deep generalized Green's function

3.1 Preliminaries

In this section, we first introduce a general *PDE problem* and describe the traditional Green's function method. A forward PDE problem on a domain $D \subset \mathbb{R}^n$ is meant to solve $u(\boldsymbol{x}) \in U : D \to \mathbb{R}$ with a given function $f(\boldsymbol{x}) \in F : D \to \mathbb{R}$ with a linear differential operator $\mathcal{L} : U \to F$ of order $p \in N^+$, subject to the following constraints,

$$\begin{cases} \mathcal{L}u(\boldsymbol{x}) &= f(\boldsymbol{x}), \quad \forall\, \boldsymbol{x} \in D, \\ \mathcal{B}[u(\boldsymbol{x})] &= u_b(\boldsymbol{x}), \quad \forall\, \boldsymbol{x} \in \partial D, \end{cases} \tag{1}$$

where some appropriate BCs $\mathcal{B}$ are imposed such that 1 has a unique solution. If the problem is time-dependent, the domain is extended to $D \times \mathbb{R}^+$ and suitable initial values of $u(\boldsymbol{x}, t)$ and its derivatives should also be specified. In the following derivation, we focus on time-independent cases but the extension to time-dependent problem is straightforward.

To solve the problem in 1, FDM directly discretizes $\mathcal{L}$ with Euler formula on a grid while FEM transforms the original differential form to a weak form on a mesh. In contrast, the Green's function method solves an adjoint problem to 1 first. Formally, the Green's function, denoted by $G(\boldsymbol{x}, \boldsymbol{y}) : D \times D \to \mathbb{R}$, is the solution to the following adjoint problem to the original problem 1,

$$\begin{cases} \mathcal{L}^* G(\boldsymbol{x}, \boldsymbol{y}) & = \delta(\boldsymbol{y} - \boldsymbol{x}), \quad \forall\, \boldsymbol{y} \in D \\ \mathcal{B}^*[G(\boldsymbol{x}, \boldsymbol{y})] & = 0, \quad \forall\, \boldsymbol{y} \in \partial D \end{cases} \tag{2}$$

for every interior point in the domain, $\boldsymbol{x} \in intD$. The Dirac delta function δ , strictly speaking, a distribution, is infinite when the argument is zero and zero everywhere otherwise. The derivation of the adjoint problem consists of two stages: determining first the explicit form of the adjoint operator $\mathcal{L}^*$, and then the adjoint BCs $\mathcal{B}^*$; see Appendix A for more details. Once $G(\boldsymbol{x}, vy)$ in 2 is solved, the solution $u(\boldsymbol{x})$ is constructed in the following integral,

$$u(\boldsymbol{x}) = \int_D G(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{y}) d\boldsymbol{y} + \int_{\partial D} p(\boldsymbol{x}, \boldsymbol{y}) ds_{\boldsymbol{y}}, \tag{3}$$

where the second term $p(.)$ is a boundary term determined by a specific $\mathcal{B}$ in the original problem . For Dirichlet BC, i.e., $\mathcal{B}[u(\boldsymbol{x})] = u(\boldsymbol{x})$, then $p(\boldsymbol{x}, \boldsymbol{y}) = -u_b(\boldsymbol{y}) \frac{\partial G}{\partial \mathbf{n}}(\boldsymbol{x}, \boldsymbol{y})$, where $\mathbf{n}$ is the outward normal vector of the boundary (Kreyszig et al., 2008).

3.2 Generalized Green's Function

The main challenge for numerical evaluation of the Green's Function comes from the highly singular Dirac delta function. In this work, we propose to use an alternative form to replace the traditional Green's function, which is easier to evaluate numerically but mathematically equivalent in terms of the solution construction. The derivation has two stages. First, we work with functions with compact supports contained in the domain D such that any boundary terms automatically vanish. The second stage aims at imposing the correct generalized adjoint BCs, which are dependent on the specific BC type of the original problem in 1.

3.2.1 Formal Generalized Green's Function

Equation 3.1 can be viewed as a linear functional which maps f to the solution function value at some point $u(\boldsymbol{x})$. From the Riesz representation theorem, we can also formally propose a linear functional which maps the Laplacian of the f, i.e. Δf to the solution such that $\Delta f \mapsto u(\boldsymbol{x})$, and we use G^t as the new kernel to replace the traditional Green's function G. The motivation is that by taking the derivative of f, the Green's function G should be integrated in the integral 3.1 to maintain the resulting value. The high singularity of the kernel in the solution integral is then alleviated by this integration operation. Therefore, we have the following definition for the generalized Green's function.

Definition 3.1. The generalized Green's function $G^t : D \times D \to \mathbb{R}$ on a compact domain is defined to be the solution to the following generalized adjoint PDE problems:

$$\begin{cases} \mathcal{L}^* G^t(\boldsymbol{x}, \boldsymbol{y}) & = -\psi(\boldsymbol{x}, \boldsymbol{y}), \\ \mathcal{B}^*[G^t(\boldsymbol{x}, \boldsymbol{y})] & = 0. \end{cases} \tag{4}$$

where $\psi : D \times D \to \mathbb{R}$ is chosen such that the integral holds, $u(\boldsymbol{x}, \boldsymbol{y}) = \int_D G^t(\boldsymbol{x}, \boldsymbol{y}) \Delta f(\boldsymbol{y}) d\boldsymbol{y}$.

Note that a similar generalized Green's function concept is proposed in *a posteriori* error analysis of FEM, where different forms of ψ are chosen for specific usages, for instance, estimating the averaged error over a curve. However, the primary difference is that in our definition, the Laplacian of f instead of the function itself is used and the solution function u is the main target of our method.

Remark 3.1. Here, we use the Laplacian of the excitation function, namely $\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$. But this choice is not limited to the Laplacian of f. It is manifest from the following derivation that the known fundamental solution for all dimensions and the self-adjointness of the Laplacian makes the application relatively simple.

Now, we derive the correct form of the function $\psi(\boldsymbol{x}, \boldsymbol{y})$ in the definition 3.1. The input function $\psi(\boldsymbol{x}, \boldsymbol{y})$ can be first divided into the sum of two parts, a singular term $\psi_s(\boldsymbol{x}, \boldsymbol{y})$ and a regular term $\psi_r(\boldsymbol{x}, \boldsymbol{y})$, such that $\psi = \psi_s(\boldsymbol{x}, \boldsymbol{y}) + \psi_r(\boldsymbol{x}, \boldsymbol{y})$. The singular part serves as the source of the Dirac delta function with the application of the differential operator, in this case, Δ. And the regular part is used to impose the correct boundary condition of ψ. The exact boundary condition value ψ_b is elaborated in next section.

We then have the following lemma:

Lemma 3.1. The function ψ should have the property of $-\Delta\psi(\boldsymbol{x}, \boldsymbol{y}) = \delta(\boldsymbol{x}, \boldsymbol{y})$. And the two parts of the correct form of ψ should respectively satisfy:

$$-\Delta\psi_s(\boldsymbol{x}, \boldsymbol{y}) = \delta(\boldsymbol{x} - \boldsymbol{y}), \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \qquad \text{(Fundamental Solution)} \tag{5}$$

$$-\Delta\psi_r(\boldsymbol{x}, \boldsymbol{y}) = 0, \quad \psi_r(\boldsymbol{x}, \boldsymbol{y})|_{\partial D} = \psi_b(\boldsymbol{x}, \boldsymbol{y})|_{\partial D}, \qquad \text{(Boundary Value Problem).} \tag{6}$$

The singular part $\psi_s(\boldsymbol{x}, \boldsymbol{y})$ is the fundamental solution of the Laplace equation and is known in closed-form:

$$\psi_s(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} -\frac{1}{2\pi} \ln(|\boldsymbol{x} - \boldsymbol{y}|), & n = 2 \\ \frac{1}{4\pi} \frac{1}{|\boldsymbol{x}-\boldsymbol{y}|}, & n = 3 \\ -\frac{1}{4\pi^2} \frac{1}{|\boldsymbol{x}-\boldsymbol{y}|^2}, & n = 4, \end{cases} \tag{7}$$

where n is the dimension of the domain D. Then the problem reduces to solving for $\psi_r(\boldsymbol{x}, \boldsymbol{y})$ which is a standard PDE boundary value problem that can be easily solved.

Next we prove that the alternative excitation function ψ in 3.1 make the solution integral in the definition of the generalized Green's function 3.1 hold.

Proof. The solution function u can be expressed as the convolution of the Dirac delta function with itself. Refer to 5 and $\delta = -\Delta\psi$ gives,

$$u(\boldsymbol{x}) = (u, \delta) = \int_D u(\boldsymbol{y})\delta(\boldsymbol{x} - \boldsymbol{y})d\boldsymbol{y} = (u, -\Delta\psi). \tag{8}$$

The function ψ is linked to the generalized Green's function $L^*G^t = -\psi$ by the definition in 3.1. Next, we interchange Δ and $\mathcal{L}^*$ by the Schwarz's theorem and use the bilinear identity of $\mathcal{L}$. It yields,

$$(u, -\Delta\psi) = (u, \Delta L^* G^t) = (u, L^* \Delta G^t) = (Lu, \Delta G^t). \tag{9}$$

We use the bilinear identity of Δ again. Note that it is self-adjoint, $\Delta^* = \Delta$, which gives

$$(Lu, \Delta G^t) = (\Delta Lu, G^t). \tag{10}$$

Referring to the original problem in 1 we have $\mathcal{L}u = f$, arriving at the formal integral of using the generalized Green's function,

$$u(\boldsymbol{x}) = (\Delta f, G^t) = \int_D \Delta_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) G^t(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y}, \tag{11}$$

where the subscript of Δ denotes the variable that it operates on.

□

3.2.2 Generalized Adjoint Boundary Conditions

In previous section, the formal generalized Green's function is derived and proved assuming all related functions have compact support contained in the domain D. This condition implies that any boundary terms vanish in the bilinear identity, i.e. $(v, Lu) - (L^*v) =$ boundary terms, which involves the function values of u, v or their directional derivative on the boundary.

Here, we derive the suitable generalized BCs and write down the complete solution integral using generalized Green's function. In the proof above, 3.2.1 and 3.2.1 all have boundary terms if related functions are not vanishing on the boundary. The first boundary term in 3.2.1 is

$$(u, L^*\Delta G^t) - (Lu, \Delta G^t) = \int_{\partial D} p(u, \Delta G^t) d\mathbf{s}, \tag{12}$$

where $p(\cdot, \cdot)$ denotes the boundary terms corresponding to the specific form of $\mathcal{L}$. Depending on the BC type in the original problem 1, the correct BC on ΔG^t can be imposed. For example, if the original problem is a Dirichlet boundary type, namely $u = u_b(\boldsymbol{x}), \forall \boldsymbol{x} \in \partial D$, then the corresponding BC of G^t should be of the same type and homogeneous, i.e.

$$\Delta G^t(\boldsymbol{x}, \boldsymbol{y}) = 0, \forall \boldsymbol{x} \in \partial D, \forall \boldsymbol{y} \in intD. \tag{13}$$

By posing the correct BC, the unknown term in the boundary integral, such as derivatives of u, would be multiplied by zero. If the reduced boundary term is denoted as p°, for the Dirichlet BC, we have

$$(u, L^*\Delta G^t) = (Lu, \Delta G^t) + \int_{\partial D} p^\circ(u_b, \Delta G^t) d\mathbf{s}. \tag{14}$$

Neumann BC or other boundary condition types can also be derived similarly. The second boundary term in 3.2.1 comes from the known Δ and the exact form is

$$(f, \Delta G^t) - (\Delta f, G^t) = \int_{\partial D} [f \partial_{\mathbf{n}} G^t - G^t \partial_{\mathbf{n}} f] d\mathbf{s}, \tag{15}$$

where $\partial_{\mathbf{n}}$ denotes the directional derivatives along the outward normal of the boundary.

Finally, we arrive at the complete formula of the solution integral for the generalized Green's function,

$$u(\boldsymbol{x}) = \int_D \Delta_{\boldsymbol{y}} f(\boldsymbol{x}, v\boldsymbol{y}) G^t(\boldsymbol{x}, v\boldsymbol{y}) d\boldsymbol{y} + \int_{\partial D} [p^\circ(u_b, \Delta G^t) + f \partial_{\mathbf{n}} G^t - G^t \partial_{\mathbf{n}} f] d\mathbf{s}. \tag{16}$$

3.3 MODEL

In this section, we propose a unified paradigm for solving PDEs with neural networks and the generalized Green's function. Our method involves three steps which generate one auxiliary neural network and one primary neural network for the generalized Green's function. Following the concept of physics-informed neural networks, we represent the solution function with parameterized neural networks while using the partial differential equations and BCs to define the loss function which is minimized during training. The three steps are

- Step 1: Specify the suitable boundary condition for ψ. Solve for the alternative input function ψ.
- Step 2: Solve the generalized Green's function G^t.
- Step 3: Construct the solution using the generalized Green's function neural network.

It should be noted that steps 1 and 2 are only executed once for a fixed PDE operator and domain D but the trained neural network can be reused in step 3. We elaborate on each step in the following sections.

3.3.1 SOLVING THE ALTERNATIVE INPUT FUNCTION ψ

As stated in Lemma 3.1, the regular part $\psi_r(\boldsymbol{x}, \boldsymbol{y})$ is the solution to the boundary value problem defined with $\mathcal{L}_{\boldsymbol{x}} \psi_r(\boldsymbol{x}, \boldsymbol{y}) = 0, \psi_r(\boldsymbol{x}, \boldsymbol{y}) = g(\boldsymbol{x}, \boldsymbol{y})$ where the boundary values are determined by the analytic function of $\psi_s(\boldsymbol{x}, \boldsymbol{y})$. We use a neural network $\hat{\psi}_{r,\theta}(\boldsymbol{x}, \boldsymbol{y})$ to represent the solution $\psi_r(\boldsymbol{x}, \boldsymbol{y})$ to the boundary value problems. For each training iteration, a batch of random interior point pairs of $(\boldsymbol{x}_i \in D^\circ, \boldsymbol{y}_i \in D^\circ), i = 1, 2, ..., N'_{dm}$ and a batch of random boundary point pairs $(\boldsymbol{x}_j \in \partial D, \boldsymbol{y}_j \in D^\circ), j = 1, 2, ..., N'_{bd}$ are independently sampled using the Latin hypercube

sampling method. These sampled points across the domain or on the boundary are used to evaluate the residual loss or boundary loss, respectively. The total loss function at each iteration is,

$$L_\theta = \frac{\lambda_{res}}{N'_{dm}} \sum_i^{N'_{dm}} L_{res} + \sum_j^{N'_{bd}} \lambda_{bd} L_{bd} \tag{17}$$

$$= \frac{\lambda_{res}}{N'_{dm}} \sum_i^{N_{dm}} |\mathcal{L}\hat{\psi}_\theta(\boldsymbol{x}_i, \boldsymbol{y}_i)|^2 + \frac{\lambda_{bd}}{N'_{bd}} \sum_j^{N_{bd}} |\hat{\psi}_{r,\theta}(\boldsymbol{x}_j, \boldsymbol{y}_j) - g(\boldsymbol{x}_j, \boldsymbol{y}_j)|^2 \tag{18}$$

$\lambda_{res}, \lambda_{db}$ are the weights to balance the magnitude between the PDE residual loss and the boundary loss. The partial derivatives in $\mathcal{L}$ are all computed using the AutoGrad package in PyTorch. The stochastic gradient descent (SGD) method is used to minimize the loss and results in an accurate neural network representation of $\hat{\psi}_{r,\theta}(\boldsymbol{x}, \boldsymbol{y})$.

3.3.2 Solving for the generalized Green's function G^t.

With the help of the auxiliary neural network model $\hat{\psi}_{r,\theta}(\boldsymbol{x}, \boldsymbol{y})$, the alternative excitation function $t(\boldsymbol{x}, \boldsymbol{y})$ can be readily evaluated at every point across the domain, i.e. $t(\boldsymbol{x}, \boldsymbol{y}) = \psi_s(\boldsymbol{x}, \boldsymbol{y}) + \hat{\psi}_{r,\theta}(\boldsymbol{x}, \boldsymbol{y})$. Now we use the same framework as above to train the generalized Green's function $\hat{G}^t_\phi$. For each training iteration, a batch of random interior point pairs of $\boldsymbol{x}_i \in D^\circ, \boldsymbol{y}_i \in D, i = 1, 2, ..., N_{dm}$ and a batch of random boundary point pairs $(\boldsymbol{x}_j \in \partial D, \boldsymbol{y}_j \in D^\circ), j = 1, 2, ..., N_{bd}$ are independently sampled using again the Latin hypercube sampling method. For this problem, the residual loss takes the alternative excitation function t and sets the boundary values are zero, which leads to the definition of the total loss as,

$$L_\phi = \frac{\lambda_{res}}{N_{dm}} \sum_i^{N_{dm}} L_{res} + \frac{\lambda_{res}}{N_{bd}} \sum_j^{N_{bd}} \lambda_{bd} L_{bd} \tag{19}$$

$$= \frac{\lambda_{res}}{N_{dm}} \sum_i^{N_{dm}} |\mathcal{L}\hat{G}^t_\phi(\boldsymbol{x}_i, \boldsymbol{y}_i) - (\psi_s + \hat{\psi}_{r,\theta})(\boldsymbol{x}_i, \boldsymbol{y}_i)|^2 + \frac{\lambda_{res}}{N_{bd}} \sum_j^{N_{bd}} |\hat{G}^t_\phi(\boldsymbol{x}_j, \boldsymbol{y}_j)|^2. \tag{20}$$

Once the generalized Green's function network finishes training, it can be used in equation (15) to determine the solution function for any excitation function f with,

$$u(\boldsymbol{x}_0) = \sum_j w_j \hat{G}^t_\phi(\boldsymbol{x}_0, \boldsymbol{y}_j) \Delta f(\boldsymbol{y}_j). \tag{21}$$

where $\boldsymbol{y}_{i,j} = 1, 2, ..., n$ are the quadrature points for fast evaluation of the multi-dimensional integration, and w_j are corresponding quadrature weights. It should be noted that the solution function values at multiple $\boldsymbol{x}_0$ can be computed in parallel.

4 Numerical experiments

We demonstrate performance of the DGGF in solving PDE problems compared to several modern state-of-the-art baseline methods for different problem types. These methods are compared in the Table 2. We include different domains, including a square (SQ), a circle (CR), and two B-spline curve enclosed loops (B1 & B2), and two 3D boundaries including a cube with 1/8 corner cut (CC), and an ellipsoid (EP), along with four different PDE types, listed in Table 4. The exact domain shapes are illustrated in Fig. 4.

Specifically, we demonstrate that DGGF has higher accuracy compared to GF-Net and constructs faster results compared to PINN on several classes of widely-studied PDEs. We present results of different dimensions and boundary shapes in our experiments. For each type of PDE problem, we are interested in the performance of DGGF compared to the following baseline models:

- Method (I): Gaussian Approximation of the Dirac delta function (GF-Net),

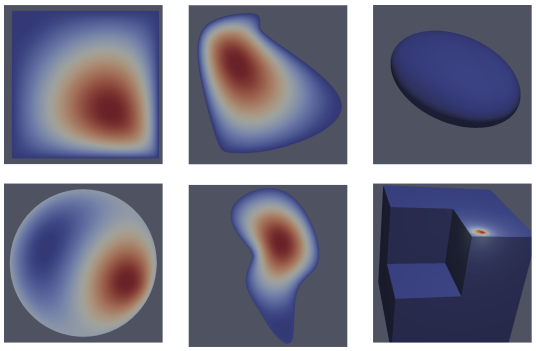

Figure 2: Different types of domain boundaries explored in the experiments. The color map indicates the generalized Green's function of a given fixed source location for the Poisson equation.

Table 1: PDE problems explored in the experiments. Boundary shapes include square (SQ), circular (CR), two Jordan curves B-spline 1 (B1) and B-spline 2 (B2), corner cut cube (CC), and an ellipsoid (EP).

CLASS	EXPRESSION	BOUNDARY SHAPE
POISSON	$\sum_{x,y,z} \partial^2_{x,y,z}$	SQ, CR, B1, B2, CC, EP
HELMHOLTZ	$\sum_{x,y,z} \partial^2_{x,y,z} + k^2$	SQ, CR, B1, B2, CC, EP
HEAT(INITIAL VALUE PROBLEM)	$\sum_{x,y,z} \partial^2_{x,y,z} - \partial_t$	SQ, CR, B1, B2, CC, EP
KLEIN-GORDON	$\sum_{x,y,z} \partial^2_{x,y,z} - \partial^2_t - k^2$	SQ, CR, B1, B2, CC, EP

- Method (II): Physics-informed neural network (PINN),
- Method (III): Numeric Green's Function (NGF),

We use the FEM solver FEniCSx (Alnaes et al., 2015; Scroggs et al., 2022b;a; Logg et al., 2012a; Logg & Wells, 2010; Logg et al., 2012b; Kirby & Logg, 2006; Logg et al., 2012c; Ølgaard & Wells, 2010; Alnaes et al., 2014; Kirby, 2012) to obtain the solution close to the ground truth for all the experiments using very fine meshes.

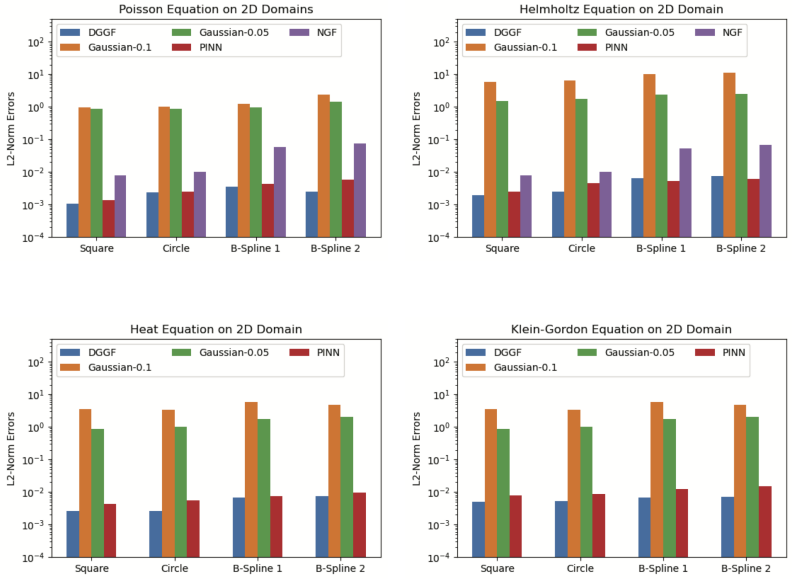

Figure 3: Results of 2-D experiments with different PDEs and boundaries showing the mean L2 norm error, including the DGGF (our approach shown as the blue colors), Gaussian (0.05 green colors) and (0.1 shown as the orange colors), PINN (red), and NGF (purple).

Table 2: Different Methods for Solving Green's Functions

Method	Excitation Form	Problem Form	Application Form
Theoretical Greens	$\delta(\boldsymbol{x}, \boldsymbol{y})$	$\mathcal{L}G(\boldsymbol{x}, \boldsymbol{y}) = \delta(\boldsymbol{x}, \boldsymbol{y})$	$G * f$
DGGF	H^t+b.t.	$\mathcal{L}G^{\circ}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = t(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})$	$G^{\circ} * \Delta f$ + b.t.
GF-Net	Gaussian	$\mathcal{L}\hat{G}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = exp(-x^2/\sigma^2)$	$\hat{G} * f$
BI-GreenNet	$-G^{th}$	$\mathcal{L}G^{bd}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = -G^{th}$	$(G^{th} + G^{bd}) * f$

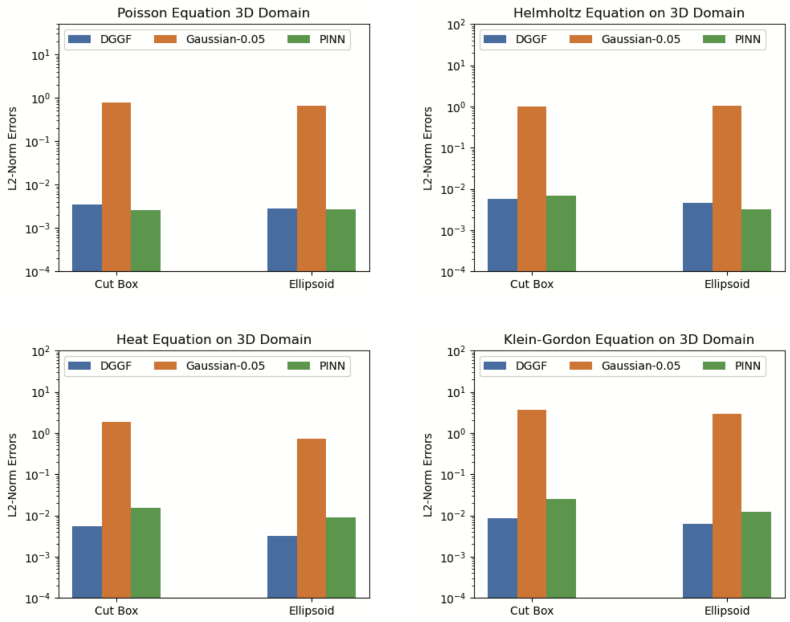

Figure 4: Results of 3D experiments with different PDEs and boundaries showing the mean L2 norm error, including the DGGF (our approach – blue colors), Gaussian (0.05 orange), and PINN (green).

4.1 Model Training

For our method, the PINN, and the GF-Net, we use the exact same DNN structure and training procedures throughout for all experiments for fair comparison. The DNN comprises 8 hidden layers with Rectified Linear Units (ReLU) activation functions, and 100 neurons in each layer. Note that the GF-Net method prescribes using multiple networks, instead of a single network, to represent the fast-varying Green's function on the whole domain. We tested this multi-network approach on simple 2-D domain problems with 16 networks and results are included in the Appendix. The complexity of arbitrary segmentation of irregular domains hinders the implementation of this method on all the cases explored in this study. The hyperparameters involved in the training of the models are N_{dm}, N_{bd}, N'_{dm}, N'_{bd},λ_{res}, λ_{bd}. Please see Sec. 3.3 for interpretation of these hyperparameters. For simplicity, we always set $\lambda_{res} = 1$ and search $\lambda_{bd} \in [1, 5, 50]$, $N_{dm} \in [50000, 100000, 150000, 200000]$, $N_{bd} \in [25000, 50000, 100000, 75000, 150000, 200000]$. The lists of values for N'_{dm} and N'_{bd} are the same as that of N_{dm} and N_{bd}. We use the learning rate of 1×10^{-3} with ADAM (Kingma & Ba, 2014) and zero regularization. GF-Net requires one additional hyperparameter, which is the width of the Gaussian used to approximate the Dirac delta function. For this, we tested two values $\epsilon \in \{.05, 0.1\}$, for every problem considered below.

4.2 Results: Accuracy

A comparison of the accuracy between DGGF and baseline methods are present in Figure 4. It can be observed that the DGGF outperforms GF-Net for both choices of the Gaussian width explored with the single network representation. In particular, the DGGF realizes at least three orders of magnitude smaller errors than that of the GF-Net, and is comparable to the PINN method. These results indicate

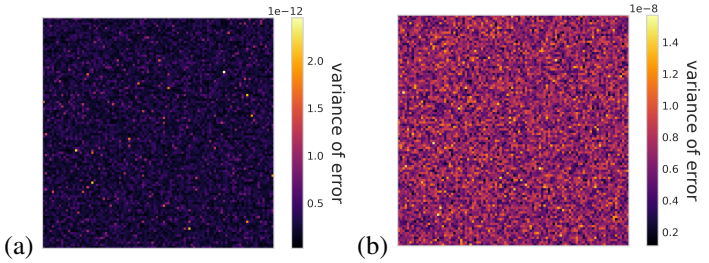

Figure 5: Stability of DGGF. Here we perform only Step 2 (G^t) in (a) and Step 1 + Step 2 (b). The variance of accuracies across points in the domain (due to random initialization of neural networks) is used to measure the stability of DGGF. Shown in (a) and (b) are variances of accuracies for each point in the 2-D Box domain.

that by transforming the Green's function to the generalized Green's function, the neural network model can learn a better kernel to construct the solution functions. Despite comparable accuracy, our method is 20,000 times faster than the PINN in the solution function construction step. Use of parallel convolution in our DGGF method takes 0.08 s for 2-D domains, compared to a 30 min training time for the PINN method on any single 2-D problem.

4.3 Ablation Study: Stability

We use 2-D Poisson problem with Box boundary to study the stability of the DGGF, i.e., the variance of error caused by different neural network random initializations. We study two scenarios: (1) only training the networks in step 2 of DGGF and (2) training networks in both step 1 and step 2. Specifically, in both scenarios, we use a network of 8 layers, all with 100 neurons each. We set $N_{dm} = 100000$, $N_{bd} = 300000$ and $\lambda_{bd} = 5$. Then we train 30 neural networks with different initializations. The variances of their accuracies for estimating Green's Function at various points in domain are then computed. As can be seen in Figure 5, the variance of accuracy across the domain is uniformly small. In particular, the variance of only training the network for G^t is to the order of 10^{-11}, demonstrating the stability of DGGF.

5 Conclusions

To harness the full potential of Green's function reusability, we introduce the concept of deep generalized Green's function. By circumventing the singularity associated with the Dirac delta function while preserving theoretical accuracy, our method offers a practical solution. Empirical tests conducted on a comprehensive selection of partial differential equations (PDEs) have confirmed the effectiveness of our approach. Notably, DGGF exhibits lower computational resource requirements, faster convergence, and, crucially, enables efficient reuse for solving PDEs of the same type.

REPRODUCIBILITY STATEMENT

We have submitted the code of our implementation of DGGF as supplementary material. The experimental details are elaborated in Section 4.

REFERENCES

M. S. Alnaes, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*, 40, 2014. doi: 10.1145/2566630.

M. S. Alnaes, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. The FEniCS project version 1.5. *Archive of Numerical Software*, 3, 2015. doi: 10.11588/ans.2015.100.20553.

Marco Baldan, Giacomo Baldan, and Bernard Nacke. Solving 1d non-linear magneto quasi-static maxwell's equations using neural networks. *IET Science, Measurement and Technology*, 15, 03 2021. doi: 10.1049/smt2.12022.

Nicolas Boullé, Christopher J Earls, and Alex Townsend. Data-driven discovery of green's functions with human-understandable deep learning. *Scientific reports*, 12(1):4824, 2022a.

Nicolas Boullé, Christopher J Earls, and Alex Townsend. Data-driven discovery of green's functions with human-understandable deep learning. *Scientific reports*, 12(1):1–9, 2022b.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12): 1727–1738, 2021.

Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995. doi: 10.1109/72.392253.

Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, may 2022. doi: 10.1016/j.cma.2022.114909. URL https://doi.org/10.1016%2Fj.cma.2022.114909.

Salvatore Cuomo, Vincenzo Schiano di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics-informed neural networks: Where we are and what's next, 2022a. URL https://arxiv.org/abs/2201.05624.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022b.

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.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. NSFnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, feb 2021a. doi: 10.1016/j.jcp.2020.109951. URL https://doi.org/10.1016%2Fj.jcp.2020.109951.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021b.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

R. C. Kirby. FIAT: numerical construction of finite element basis functions. In A. Logg, K.-A. Mardal, and G. N. Wells (eds.), *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 13. Springer, 2012.

R. C. Kirby and A. Logg. A compiler for variational forms. *ACM Transactions on Mathematical Software*, 32, 2006. doi: 10.1145/1163641.1163644.

Erwin Kreyszig, K Stroud, and G Stephenson. Advanced engineering mathematics. *Integration*, 9 (4), 2008.

I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998. doi: 10.1109/72.712178. URL https://doi.org/10.1109%2F72.712178.

Hyuk Lee and In Seok Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91:110–131, 11 1990. doi: 10.1016/0021-9991(90)90007-N.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Guochang Lin, Fukai Chen, Pipi Hu, Xiang Chen, Junqing Chen, Jun Wang, and Zuoqiang Shi. Bi-greennet: learning green's functions by boundary integral network. *Communications in Mathematics and Statistics*, 11(1):103–129, 2023.

A. Logg and G. N. Wells. DOLFIN: automated finite element computing. *ACM Transactions on Mathematical Software*, 37, 2010. doi: 10.1145/1731022.1731030.

A. Logg, K.-A. Mardal, G. N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012a. doi: 10.1007/978-3-642-23099-8.

A. Logg, G. N. Wells, and J. Hake. DOLFIN: a C++/Python finite element library. In A. Logg, K.-A. Mardal, and G. N. Wells (eds.), *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 10. Springer, 2012b.

A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. FFC: the FEniCS form compiler. In A. Logg, K.-A. Mardal, and G. N. Wells (eds.), *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 11. Springer, 2012c.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

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.

Pu Ren, Chengping Rao, Yang Liu, Jian-Xun Wang, and Hao Sun. PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. *Computer Methods in Applied Mechanics and Engineering*, 389:114399, feb 2022. doi: 10.1016/j.cma.2021.114399. URL https://doi.org/10.1016%2Fj.cma.2021.114399.

M. W. Scroggs, I. A. Baratta, C. N. Richardson, and G. N. Wells. Basix: a runtime finite element basis evaluation library. *Journal of Open Source Software*, 7(73):3982, 2022a. doi: 10.21105/joss.03982.

M. W. Scroggs, J. S. Dokken, C. N. Richardson, and G. N. Wells. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Transactions on Mathematical Software*, 2022b. doi: 10.1145/3524456. To appear.

Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Yuankai Teng, Xiaoping Zhang, Zhu Wang, and Lili Ju. Learning green's functions of linear reaction-diffusion equations with application to fast numerical solver. In *Mathematical and Scientific Machine Learning*, pp. 1–16. PMLR, 2022.

Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, pp. 1457–1466, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403198. URL https://doi.org/10.1145/3394486.3403198.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43 (5):A3055–A3081, 2021. doi: 10.1137/20M1318043. URL https://doi.org/10.1137/20M1318043.

Bing Yu et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

Navid Zobeiry and Keith D. Humfeld. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence*, 101:104232, 2021. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2021.104232. URL https://www.sciencedirect.com/science/article/pii/S0952197621000798.

K. B. Ølgaard and G. N. Wells. Optimisations for quadrature representations of finite element tensors through automated code generation. *ACM Transactions on Mathematical Software*, 37, 2010. doi: 10.1145/1644001.1644009.