DGNET: LEARNING SPATIOTEMPORAL PDES WITH DISCRETE GREEN NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spatiotemporal partial differential equations (PDEs) underpin a wide range of scientific and engineering applications, yet classical solvers are computationally costly on large or irregular domains. Neural PDE solvers have emerged as an efficient alternative, but they struggle to generalize to *unseen source terms*, which represent external forcing such as heat generation, body forces, or reaction rates. Since existing models typically mix the source with the system state, they lack a principled mechanism to capture source responses. We propose DGNet, a discrete Green network that explicitly decouples system evolution from source response. The key idea is to transform the classical Green's function—a cornerstone of PDE theory—into a graph-based discrete formulation, preserving the superposition principle in a computable update rule. To ensure fidelity on irregular meshes, we construct a hybrid operator by combining physics-based discretizations with GNN-based corrections, while a lightweight residual GNN captures dynamics beyond the operator. Across three categories of spatiotemporal PDE scenarios, DGNet consistently achieves state-of-the-art accuracy. In particular, on the most challenging setting with entirely novel source terms, DGNet maintains stable performance while existing approaches collapse.

1 INTRODUCTION

Spatiotemporal partial differential equations (PDEs) form the foundation of modeling dynamical systems across science and engineering, governing phenomena in fluid dynamics (Hirsch, 2007), weather forecasting (Lynch, 2008), molecular dynamics (Lelievre & Stoltz, 2016), and energy systems (Ríos-Mercado & Borraz-Sánchez, 2015). Accurately solving such PDEs is crucial for scientific discovery and engineering design. Classical numerical solvers (Anderson, 2002; Evans et al., 2012) can provide reliable solutions but often become computationally prohibitive for large-scale or irregular domains. This has spurred growing interest in neural PDE solvers that learn to approximate solutions more efficiently (Raissi et al., 2019; Brandstetter et al., 2022; Zeng et al., 2025). Since spatiotemporal PDEs are typically discretized over irregular meshes, they can be naturally represented as graphs, making graph-based formulations a compelling foundation for learning-based solvers.

A distinctive feature of spatiotemporal PDEs is the presence of a source term, $f(\mathbf{x}, t)$, representing external forcing applied over space and time. Examples include time-varying heat sources in conduction (Hahn & Özisik, 2012), body forces in fluid dynamics (Pope, 2001), time-dependent currents in electromagnetics (Taflove et al., 2005), spatiotemporal reaction rates in reaction–diffusion systems (Murray, 2007), and seismic excitations in elasticity (Aki & Richards, 2002). These sources are not peripheral details but often define the problem itself. Nevertheless, existing neural solvers usually entangle the source term with the system state as a single input, lacking a dedicated structure to model its effect. This design severely limits generalization: models trained on one set of sources exhibit sharp performance degradation when confronted with unseen sources, a scenario that is ubiquitous in scientific and engineering practice.

Another difficulty lies in combining physics priors with neural solvers. While physics-informed designs can improve data efficiency, naively injecting priors into flexible neural architectures is brittle on irregular meshes and complex dynamics. Green's function provides a principled foundation: it effectively "extracts" the solution u from the direct action of the operator $\mathcal{L}_{\mathbf{x}}$, recasting the PDE solution into two explicit components—the homogeneous evolution and the forced response. This

separation is precisely what enables us to treat the source term as a first-class input, rather than mixing it implicitly with the system state. However, applying continuous Green's functions directly in learning is intractable: closed-form expressions rarely exist for complex geometries, and evaluating the integrals is prohibitively expensive. Our key contribution is to make this classical theory discrete. We develop a discrete Green formulation on graphs that preserves the superposition structure in a computable update rule, thus bridging continuous PDE theory with neural solvers and providing a natural interface for embedding physics priors together with learnable corrections from GNNs.

Our contributions can be summarized as follows:

- We propose DGNet, a new paradigm for spatiotemporal PDEs that explicitly decouples system evolution from source response. This is made possible by leveraging Green's function to "extract" the solution u from the direct action of the operator $\mathcal{L}_{\mathbf{x}}$, thereby treating the source term as a first-class input rather than an implicit perturbation.
- We develop a discrete Green's function formulation on graphs, transforming a classical continuous tool into a computable update rule. This preserves the superposition principle in the discrete setting, and provides a principled interface where physics priors are embedded at the operator level and GNNs learn corrections that compensate discretization errors on irregular meshes.
- We demonstrate that DGNet achieves state-of-the-art performance across multiple PDE benchmarks, with particularly strong gains in the most challenging scenarios involving diverse and unseen source terms.

2 Related Work

Physics-informed neural networks (PINNs). PINNs (Raissi et al., 2019) represent a prior-heavy paradigm, where PDE residuals and boundary conditions are incorporated as soft constraints in the training loss. This makes them data-efficient, since only sparse observations are required. However, PINNs essentially learn a specific solution under the training setup, and thus cannot generalize to domains, parameters, or forcing terms beyond the training distribution. Variants (Yu et al., 2022; Sukumar & Srivastava, 2022; Costabal et al., 2024) have attempted to improve stability or efficiency, but the lack of out-of-distribution generalization remains a fundamental limitation.

Neural operators. Neural operator methods adopt the opposite, largely data-driven stance: they directly learn mappings between infinite-dimensional function spaces from data. Representative works include DeepONet (Lu et al., 2021) and the Fourier Neural Operator (FNO) (Li et al., 2021), along with many extensions in the spectral or kernel domains (Guibas et al., 2021; Li et al., 2023; George et al., 2024; Tran et al., 2023). These models achieve strong generalization across resolutions and physical parameters. Yet, they usually treat external forcing implicitly within the dynamics, and lack an explicit mechanism to handle varying source terms during time marching.

Graph-based PDE solvers. For spatiotemporal PDEs on irregular domains, graphs provide a natural backbone. The Graph Network Simulator (GNS) (Sanchez-Gonzalez et al., 2020b) introduced the encoder–processor–decoder framework for physical simulation. Subsequent works enhanced physical consistency by embedding conservation laws (Sanchez-Gonzalez et al., 2020a; Cranmer et al., 2020; Bishnoi et al., 2023), incorporating operator blocks (Seo et al., 2019; Horie & Mitsume, 2022; Zeng et al., 2025), or enforcing invariants such as momentum conservation (Bishnoi et al., 2024). BENO (Wang et al., 2024) further drew inspiration from Green's functions, but focused on time-independent elliptic PDEs. Despite these advances, graph-based solvers remain limited in generalizing beyond training conditions.

Generalization challenges in PDE solvers. Generalization is central to neural PDE solvers. Neural operators such as FNO (Li et al., 2021) generalize across resolutions, DeepONet (Lu et al., 2021) across parameters, and BroGNet (Bishnoi et al., 2024) across system sizes. Yet a critical gap remains: none of these approaches can generalize to unseen source terms, which frequently arise in scientific and engineering systems. Such varying sources are ubiquitous in practice, including time-varying heat sources in conduction (Hahn & Özisik, 2012), body forces in fluid dynamics (Pope, 2001), time-dependent currents in electromagnetics (Taflove et al., 2005), spatiotemporal reaction rates in reaction–diffusion systems (Murray, 2007), and seismic excitations in elasticity (Aki & Richards, 2002).

3 DGNET: DISCRETE GREEN NETWORKS WITH SOURCE SUPERPOSITION

Here, we introduce our proposed DGNet, for learning spatiotemporal PDEs with varying sources. We begin in Section 3.1 by formalizing the problem setup and highlighting the critical role of source terms. In Section 3.2, we introduce Green's function and the superposition principle, which naturally decompose the PDE solution into state evolution and source response. In Section 3.3, we derive the discrete Green formulation by discretizing both space and time, leading to a source-aware update rule. In Section 3.4, we describe our physics–neural hybrid operator that combines numerical discretization with a GNN-based correction to construct a high-fidelity solver. Finally, Section 3.5 describes the prediction and training procedure.

3.1 PROBLEM FORMULATION

We consider physical systems governed by spatiotemporal partial differential equations (PDEs) of the form

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = \mathcal{L}_{\mathbf{x}}[u(\mathbf{x},t)] + f(\mathbf{x},t), \quad (\mathbf{x},t) \in \Omega \times (0,T], \tag{1}$$

where $u(\mathbf{x},t) \in \mathbb{R}$ denotes the physical state, $\mathcal{L}_{\mathbf{x}}$ is a spatial differential operator characterizing the system dynamics, and $f(\mathbf{x},t)$ is the source term.

The evolution is uniquely determined by an initial condition (IC) and boundary conditions (BCs). The IC specifies $u(\mathbf{x},0) = u_0(\mathbf{x})$. On the boundary $\partial\Omega$, we primarily consider Dirichlet BCs $u(\mathbf{x},t) = g(\mathbf{x},t)$ for $\mathbf{x} \in \partial\Omega_D$ and Neumann BCs $\frac{\partial u(\mathbf{x},t)}{\partial \mathbf{n}} = h(\mathbf{x},t)$ for $\mathbf{x} \in \partial\Omega_N$, with $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \cap \partial\Omega_N = \emptyset$. Other BC types (e.g., Robin, periodic) are accommodated in the same framework.

In many real-world systems, the source $f(\mathbf{x},t)$ varies substantially across space and time and thus plays a decisive role in the solution behavior. However, existing learning-based solvers often assume a fixed or implicitly absorbed source, limiting generalization across forcing patterns.

3.2 GREEN'S FUNCTION REPRESENTATION

Directly solving the PDE (1) is often challenging. The Green's function method from PDE theory (Arfken et al., 2013) provides a principled way to characterize the system response. The Green's function $G(\mathbf{x},t;\mathbf{x}',\tau)$ is uniquely determined by

$$\left(\frac{\partial}{\partial t} - \mathcal{L}_{\mathbf{x}}\right) G(\mathbf{x},t;\mathbf{x}',\tau) = \delta(\mathbf{x}-\mathbf{x}')\,\delta(t-\tau), \tag{2}$$

where $\delta(\cdot)$ is the Dirac delta function. Physically, $G(\mathbf{x},t;\mathbf{x}',\tau)$ describes the influence observed at $(\mathbf{x},t)$ due to a unit-strength point source applied at $(\mathbf{x}',\tau)$.

Crucially, the Green's function acts as a mathematical device to "extract" the solution u from the direct action of the operator $\mathcal{L}_{\mathbf{x}}$. Instead of learning $\mathcal{L}_{\mathbf{x}}[u]$ as an inseparable whole, the Green representation rewrites the PDE solution in terms of a propagation kernel defined by $\mathcal{L}_{\mathbf{x}}$ and a convolution with the source term. According to the superposition principle (Boyce et al., 2017), the complete solution $u(\mathbf{x},t)$ can therefore be expressed as

$$u(\mathbf{x},t) = \underbrace{\int_{\Omega} G(\mathbf{x},t;\mathbf{x}',0)\,u_0(\mathbf{x}')\,d\mathbf{x}'}_{\text{Evolution of initial state}} + \underbrace{\int_0^t \int_{\Omega} G(\mathbf{x},t;\mathbf{x}',\tau)\,f(\mathbf{x}',\tau)\,d\mathbf{x}'d\tau}_{\text{Response to source term}}. \tag{3}$$

This representation makes two key points explicit. First, the solution naturally decomposes into the evolution of the initial condition and the accumulated response to the source term. Second, it elevates $f(\mathbf{x},t)$ to a first-class input of the model, enabling a unified treatment of diverse and time-varying forcing conditions. In the next section, we describe how this continuous formulation is discretized on graphs, leading to a discrete Green's function that forms the foundation of our learning architecture.

3.3 DISCRETIZATION

While Green's function provides an elegant continuous representation, it is rarely available in closed form on complex domains and evaluating the integrals is computationally prohibitive. We therefore develop a discrete Green formulation on graphs that preserves the superposition structure in a computable update rule. In this process, the role of the Green's function naturally emerges as a discrete propagation operator that evolves the system state and incorporates the effect of the source term at each time step.

Graph-based ODE system. Graph-based discretization is particularly suitable for spatiotemporal PDEs on irregular meshes, as it provides a flexible representation of spatial operators and naturally supports message-passing based corrections. Therefore in DGNet, we discretize the spatial domain Ω into N nodes $\{\mathbf{x}_0, \dots, \mathbf{x}_{N-1}\}$, and represent it as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the edge set constructed once using Delaunay triangulation. Each node $i \in \mathcal{V}$ corresponds to a spatial location $\mathbf{x}_i$.

At a discrete time t^k, the physical state values $\{u(\mathbf{x}_i, t^k)\}_{i=0}^{N-1}$ form a state vector

$$\mathbf{u}^k = \big(u(\mathbf{x}_0, t^k), \dots, u(\mathbf{x}_{N-1}, t^k)\big) \in \mathbb{R}^N,$$

and similarly the source term values $\{f(\mathbf{x}_i, t^k)\}_{i=0}^{N-1}$ form a vector

$$\mathbf{f}^k = \big(f(\mathbf{x}_0, t^k), \dots, f(\mathbf{x}_{N-1}, t^k)\big) \in \mathbb{R}^N.$$

When the time index k is not critical, we simply denote them by $\mathbf{u}$ and $\mathbf{f}$.

With these definitions, the continuous PDE (1) can be written in discrete form as

$$\frac{d\mathbf{u}}{dt} = \mathbf{L}\mathbf{u} + \mathbf{f}, \tag{4}$$

where $\mathbf{L} \in \mathbb{R}^{N\times N}$ is the matrix discretization of the spatial operator $\mathcal{L}_{\mathbf{x}}$. For node i, the action of $\mathbf{L}$ on the state vector $\mathbf{u}$ is $[\mathbf{L}\mathbf{u}]_i = \sum_{j=1}^{N} [\mathbf{L}]_{ij} [\mathbf{u}]_j$, where $[\cdot]_{ij}$ denotes the (i,j)-th entry of a matrix and $[\cdot]_j$ denotes the j-th component of a vector.

Time integration. We apply a midpoint (Crank–Nicolson) discretization on $[t_k, t_{k+1}]$: let $\Delta t = t_{k+1} - t_k$, the time increment is $(\mathbf{u}^{k+1} - \mathbf{u}^k)/\Delta t$, and the right-hand side is approximated by a trapezoidal average, yielding

$$\frac{\mathbf{u}^{k+1} - \mathbf{u}^k}{\Delta t} = \tfrac{1}{2}\big(\mathbf{L}\mathbf{u}^k + \mathbf{L}\mathbf{u}^{k+1}\big) + \tfrac{1}{2}\big(\mathbf{f}^k + \mathbf{f}^{k+1}\big). \tag{5}$$

Rearranging terms yields a sparse linear system:

$$\big(\mathbf{I} - \tfrac{\Delta t}{2}\mathbf{L}\big)\,\mathbf{u}^{k+1} = \big(\mathbf{I} + \tfrac{\Delta t}{2}\mathbf{L}\big)\,\mathbf{u}^k + \tfrac{\Delta t}{2}\left(\mathbf{f}^k + \mathbf{f}^{k+1}\right). \tag{6}$$

Discrete Green's function. From Eq. 6, the discrete Green's function is identified as

$$\mathbf{G}(\Delta t) = \Big(\mathbf{I} - \tfrac{\Delta t}{2}\mathbf{L}\Big)^{-1}.$$

Thus the single-step update becomes

$$\mathbf{u}^{k+1} = \mathbf{G}(\Delta t)\Big(\mathbf{I} + \tfrac{\Delta t}{2}\mathbf{L}\Big)\mathbf{u}^k + \mathbf{G}(\Delta t)\tfrac{\Delta t}{2}\big(\mathbf{f}^k + \mathbf{f}^{k+1}\big). \tag{7}$$

This discrete formulation mirrors the superposition principle (3): the next state $\mathbf{u}^{k+1}$ results from the propagation of the current state under $\mathbf{G}(\Delta t)$ together with the accumulated response to the source term within the interval Δt. A full derivation of this update scheme is provided in Appendix A.

As naively computing the inverse $\big(\mathbf{I} - \frac{\Delta t}{2}\mathbf{L}\big)^{-1}$ is infeasible for large systems. Instead, we solve the equivalent sparse linear system, where $\mathbf{L}$ is already sparse due to local mesh interactions. Importantly, the coefficient matrix $\big(\mathbf{I} - \frac{\Delta t}{2}\mathbf{L}\big)$ depends only on the static mesh geometry and thus remains fixed throughout rollout. We therefore adopt a "factorize once, solve many times" strategy: a single sparse LU factorization is performed before rollout, and its factors are cached and reused for all time steps via efficient forward/backward substitution. This reduces per-step cost to nearly linear in the number of nonzeros, making the discrete Green solver practical for large meshes (see Appendix C.2 for details).

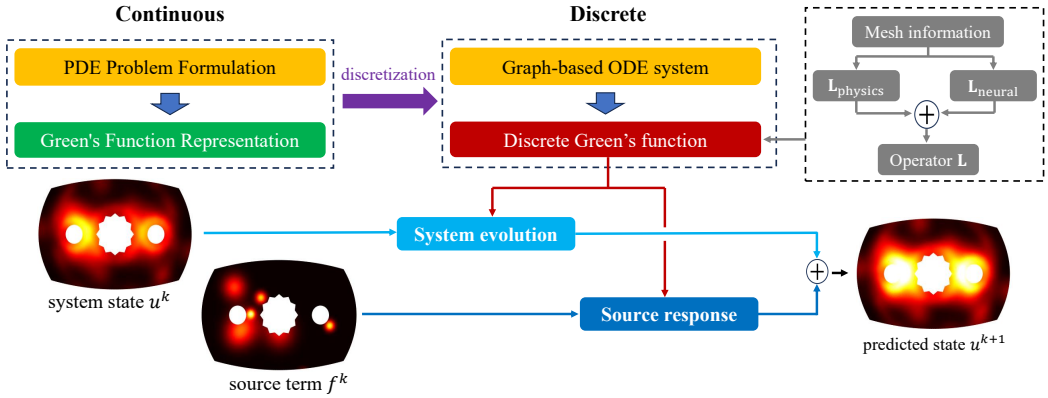

Figure 1: Overview of DGNet architecture. The model centers on a hybrid operator $\mathbf{L} = \mathbf{L}_{\text{physics}} + \mathbf{L}_{\text{neural}}$, where $\mathbf{L}_{\text{physics}}$ encodes gradient and Laplacian discretizations and $\mathbf{L}_{\text{neural}}$ is a GNN-based correction for mesh-induced errors. This operator is integrated into the discrete Green's function update (Eq. 7), which naturally combines system evolution with source-term response.

3.4 Physics–Neural Hybrid Operator

The core of our architecture lies in constructing a high-fidelity operator $\mathbf{L}$ for the discrete ODE system. Instead of relying solely on data-driven models or purely hand-crafted discretizations, we propose a hybrid approach that combines the best of both worlds:

$$\mathbf{L} = \mathbf{L}_{\text{physics}} + \mathbf{L}_{\text{neural}}. \tag{8}$$

Here, $\mathbf{L}_{\text{physics}}$ is built directly from mesh geometry using numerical discretization techniques, ensuring consistency with physical laws. Meanwhile, $\mathbf{L}_{\text{neural}}$ is a learnable correction term, parameterized by a GNN, that compensates for discretization errors on irregular meshes.

3.4.1 Physics prior operator $\mathbf{L}_{\text{physics}}$

Among the various spatial operators appearing in PDEs, the gradient and Laplacian are the most fundamental and widely used components (e.g., in diffusion, convection–diffusion, and Poisson-type equations). Accordingly, we construct $\mathbf{L}_{\text{physics}}$ directly from mesh geometry using established discretization schemes, namely the Green–Gauss theorem (Löhner, 2008) and the discrete Laplace–Beltrami operator (Meyer et al., 2003). These methods are widely used in computational physics and provide physically consistent priors on irregular meshes, ensuring that $\mathbf{L}_{\text{physics}}$ encodes reliable physics knowledge. Accordingly, $\mathbf{L}_{\text{physics}}$ is instantiated as follows:

- **Gradient operator.** $[\mathbf{L}_{\text{physics}}]_{ij} = \frac{m_{ij} l_{ij}}{2A_i}$ if $j \in N(i)$ and, $[\mathbf{L}_{\text{physics}}]_{ii} = -\sum_{k \in N(i)} \frac{m_{ik} l_{ik}}{2A_i}$, other elements are zero. Where A_i is the control volume, and m_{ij} is is the scalar projection of unit normal $\mathbf{n}_{ij}$, l_{ij} is the face length.
- **Laplacian operator.** $[\mathbf{L}_{\text{physics}}]_{ij} = \frac{w_{ij}}{d_i}$ if $j \in N(i)$ and $[\mathbf{L}_{\text{physics}}]_{ii} = -\frac{1}{d_i} \sum_{k \in N(i)} w_{ik}$, other elements are zero. With cotangent weights $w_{ij} = \frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij})$ and Voronoi area d_i.

3.4.2 Neural correction operator $\mathbf{L}_{\text{neural}}$

Although $\mathbf{L}_{\text{physics}}$ provides a principled foundation, its discretization error can be significant on coarse or skewed meshes. We therefore introduce a correction matrix $\mathbf{L}_{\text{neural}}$, learned by a GNN following the Encode–Process–Decode paradigm (Sanchez-Gonzalez et al., 2020b).

Encoder. Node features consist of the spatial coordinates $\mathbf{x}_i$ and node type C_i (interior or boundary), which are mapped to embeddings as $\mathbf{h}_i^0 = \text{NodeEncoder}([\mathbf{x}_i, C_i])$. Edge features consist of the relative displacement $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ and distance $d_{ij} = \|\mathbf{x}_j - \mathbf{x}_i\|_2$, mapped to embeddings as $\mathbf{e}_{ij} = \text{EdgeEncoder}([\mathbf{x}_{ij}, d_{ij}])$.

Processor. The processor consists of M layers of message-passing neural networks (MPNNs). At the m-th layer, we obtain

$$\mathbf{h}_i^{m+1} = \text{Update}\left(\mathbf{h}_i^m,\ \sum\nolimits_{j\in N(i)} \text{Message}(\mathbf{h}_i^m, \mathbf{h}_j^m, \mathbf{e}_{ij})\right) + \mathbf{h}_i^m. \tag{9}$$

Decoder. After M layers, for each edge (i, j), we concatenate final node embeddings and pass them through a decoder to predicts edge-level corrections:

$$[\mathbf{L}_{\text{neural}}]_{ij} = \text{Decoder}([\mathbf{h}_i^M, \mathbf{h}_j^M]). \tag{10}$$

For simplicity, we implement all neural components, including the node encoder, edge encoder, message function, update function, and decoder, as multilayer perceptrons (MLPs).

3.5 Prediction and Training

With the hybrid operator $\mathbf{L} = \mathbf{L}_{\text{physics}} + \mathbf{L}_{\text{neural}}$, the system state is updated by the discrete Green solver in Eq. 7, yielding predictions $\hat{\mathbf{u}}^k$ at each global time step t_k.

To further improve accuracy, we include a lightweight residual GNN correction module in the prediction path. This design follows prior physics–neural hybrid solvers (Meng & Karniadakis, 2020; Wu et al., 2024; Long et al., 2023; Zeng et al., 2025), which demonstrate that small residual modules can effectively approximate dynamics not explicitly captured by the physical operator, while preserving interpretability of the operator-based formulation.

For training, the full trajectory $(\mathbf{u}^0, \ldots, \mathbf{u}^{T-1})$ is segmented into shorter sub-sequences of length $Q \ll T$: $(\mathbf{u}^{s_0}, \mathbf{u}^{s_1}, \ldots, \mathbf{u}^{s_{Q-1}})$, where s_q denotes the global time index of the q-th step in the sub-sequence (corresponding to time point t_{s_q}). The model rolls out from $\mathbf{u}^{s_0}$ to predict $(\hat{\mathbf{u}}^{s_1}, \ldots, \hat{\mathbf{u}}^{s_{Q-1}})$. Following prior work (Brandstetter et al., 2022; Pfaff et al., 2021; Sanchez-Gonzalez et al., 2020b), we apply the pushforward trick and inject small noise into $\mathbf{u}^{s_0}$ during training to alleviate error accumulation and improve robustness. The loss is computed only on the first and last predictions of each sub-sequence:

$$\mathcal{L} = \|\hat{\mathbf{u}}^{s_1} - \mathbf{u}^{s_1}\|^2 + \|\hat{\mathbf{u}}^{s_{Q-1}} - \mathbf{u}^{s_{Q-1}}\|^2.$$

The learnable parameters are optimized with Adam using a decayed learning rate schedule.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate DGNet on three categories of spatialtemporal PDE systems: (1) *classical equations* (Allen–Cahn, Fisher–KPP, FitzHugh–Nagumo) to validate accuracy on canonical diffusion-, advection-, and reaction-driven dynamics, (2) *complex geometric domains* (contaminant transport with cylinder, sediment, and irregular obstacles) to test robustness on irregular meshes and boundary conditions, and (3) *generalization to unseen source terms* (laser heat treatment) to evaluate adaptability to novel forcing conditions. Table 1 summarizes the governing equations, physical contexts, and the meaning of the source term in each scenario. Detailed dataset parameters (domains, mesh sizes, time steps, boundary conditions, and trajectory splits) are deferred to Appendix B.

Baselines. We compare DGNet with representative neural PDE solvers: DeepONet (Lu et al., 2021), MGN (Pfaff et al., 2021), MP-PDE (Brandstetter et al., 2022), PhyMPGN (Zeng et al., 2025), and BENO (Wang et al., 2024). These cover operator-learning, graph-based, and hybrid paradigms. All baselines are tuned to have comparable numbers of parameters and training costs. Implementation details are provided in Appendix C.

Evaluation Metrics. We report Mean Squared Error (MSE) and Relative ℓ_2 Error (RNE). RNE is defined as $\|\hat{u} - u\|_2 / \|u\|_2$, where $\hat{u}$ and u denote predicted and ground-truth states. We report $\log_{10}$ MSE for readability, while the relative ranking remains consistent with raw MSE.

Table 1: Governing equations and physical contexts for experimental scenarios.

Name	Physical Scenario	Mathematical Formulation	Meaning of Source Term
Allen–Cahn	Phase separation	$\partial_t u = \epsilon^2 \nabla^2 u - (u^3 - u)$	Driving force for separation
Fisher–KPP	Population dynamics	$\partial_t u + \mathbf{c} \cdot \nabla u = \rho u(1-u)$	Logistic growth
FitzHugh–Nagumo	Excitable systems	$\partial_t u = D_u \nabla^2 u + \Gamma(-u^3 + u - v)$; $\partial_t v = D_v \nabla^2 v + \Gamma\beta(u - \alpha v)$	Excitation–recovery coupling
Contaminant Transport	Channel flow with obstacles	$\partial_t c + \mathbf{u} \cdot \nabla c = D\nabla^2 c + \rho_g c(1-c) - k_d c$	Reaction (generation/decay)
Laser Heat	Laser heating	$\rho c_p \partial_t T = k\nabla^2 T - h(T - T_{\text{amb}}) + Q(x,t)$	Moving laser heat source

4.2 Performance Comparison

Table 2 summarizes results across all experimental scenarios, covering the three categories of spatiotemporal PDE systems considered in this paper. DGNet achieves state-of-the-art performance on every task and metric. Additional extended tables and visualizations are provided in Appendix D.

Table 2: Results across scenarios from three categories of spatiotemporal PDE systems. For both MSE (in log scale) and RNE, lower values indicate better performance. The best results are highlighted in bold.

Scenario	Metric	DeepONet	MGN	MP-PDE	BENO	PhyMPGN	DGNet
Allen-Cahn	MSE	2.60e-01	2.70e-01	8.52e-01	2.52e+00	5.16e-01	**8.75-03**
	RNE	0.6686	0.6813	1.2109	2.0813	0.9420	**0.0188**
Fisher-KPP	MSE	3.05e-02	3.66e-03	9.90e-02	6.26e-02	1.50e-02	**2.59e-04**
	RNE	0.4181	0.1448	0.7530	0.5989	0.9270	**0.0238**
FitzHugh-Nagumo	MSE	2.49e-06	3.75e-05	6.464e-06	2.14e-04	1.69e-03	**1.18e-07**
	RNE	0.9745	3.4815	1.4454	8.3106	23.5696	**0.0952**
Cylinder	MSE	4.44e-02	6.38e-03	9.31e-02	6.76e-02	4.13e-01	**1.00e-04**
	RNE	0.5976	0.7154	0.8644	0.7364	1.8201	**0.0196**
Sediments	MSE	3.61e-02	5.94e-03	7.10e-03	1.07e-01	2.00e-01	**4.60e-04**
	RNE	0.4759	0.6103	0.6673	0.8180	1.1186	**0.0282**
Complex Obstacles	MSE	5.33e-02	7.79e-03	6.09e-03	7.66e-02	2.97e-01	**6.69e-05**
	RNE	0.5061	0.6120	0.5410	0.6069	1.1956	**0.0211**
Laser Heat	MSE	2.48e+03	4.98e+03	3.88e+03	1.95e+03	6.78e+03	**1.76e+01**
	RNE	0.1208	0.1711	0.1510	0.1071	0.1998	**0.0102**

4.2.1 Classical PDEs

To validate the accuracy of DGNet as a general-purpose PDE solver, we first evaluate it on three well-established classical systems: the diffusion-dominated Allen–Cahn equation, the advection-driven Fisher–KPP equation, and the coupled FitzHugh–Nagumo system representing excitable media. These tasks span distinct physical regimes (diffusion, advection–reaction, and multi-variable coupling), and are widely used as canonical benchmarks in scientific machine learning.

Table 2 reports the quantitative results. DGNet consistently outperforms all baselines by a large margin across all equations, achieving up to one to two orders of magnitude lower mean squared error (MSE). This demonstrates that the proposed discrete Green formulation, combined with physics-informed operators, yields substantial improvements in precision. Figure 2 provides qualitative comparisons. For the Allen–Cahn and Fisher–KPP equations, DGNet's predictions closely match the ground truth, whereas other models exhibit noticeable deviations, with MGN performing relatively better than the rest. The FitzHugh–Nagumo system is particularly challenging due to the formation of nonlinear spiral waves. Here, only DGNet successfully reproduces the propagation of spiral structures over long horizons, while baselines either dissipate the patterns or introduce severe distortions. Interestingly, BENO—although designed for steady-state PDEs—captures some ripple-like features, likely because its architecture explicitly incorporates boundary conditions. These results

confirm that DGNet is capable of accurately solving diverse PDEs with very different underlying dynamics, highlighting its versatility and reliability as a spatiotemporal solver.

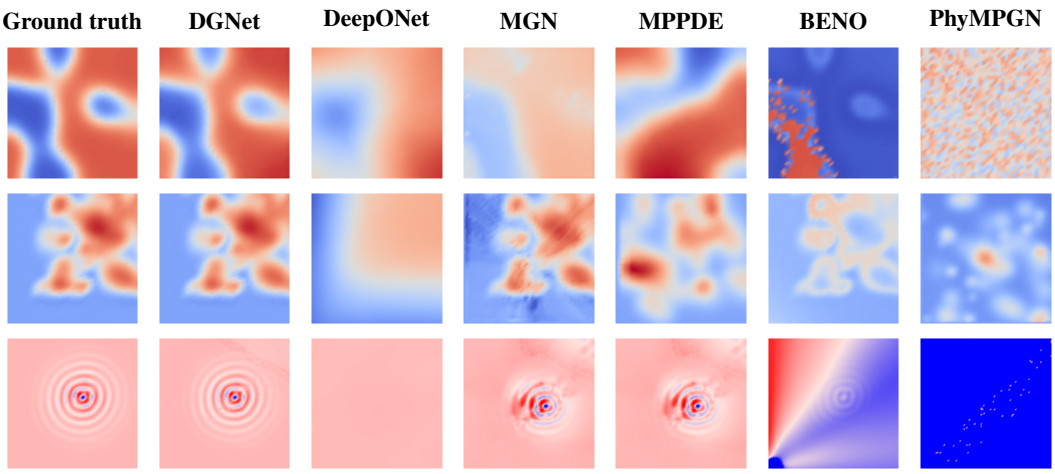

Figure 2: Visualization of prediction results on classical PDE scenarios. Rows from top to bottom correspond to the Allen–Cahn, FitzHugh–Nagumo, and Fisher–KPP equations, respectively.

4.2.2 Complex Geometric Domains

We next examine the ability of DGNet to handle PDEs defined on irregular meshes with complex boundaries. These scenarios simulate contaminant transport in a channel flow, with obstacles of varying shapes that induce rich flow structures. The three cases considered are: (i) a circular cylinder, (ii) sediment deposits on the walls, and (iii) a combination of elliptical obstacles and airfoil-shaped wall structures. Such settings are challenging due to intricate boundary conditions and highly nonuniform meshes.

Table 2 shows that DGNet achieves the lowest errors across all three domains, substantially outperforming baseline models. Figure 3 provides qualitative comparisons. In the cylinder scenario, DGNet accurately captures the Kármán vortex street in the wake of the obstacle, while other models exhibit distorted or dissipated vortical patterns. In the more complex sediment and obstacle cases, DGNet's predictions remain visually indistinguishable from ground truth, successfully reproducing contaminant filaments stretched and folded by the flow. By contrast, baseline methods either blur these fine-scale structures or fail to track their spatial location. These results demonstrate that the discrete Green formulation, combined with physics-informed operators, generalizes robustly to irregular geometries and nontrivial boundary conditions.

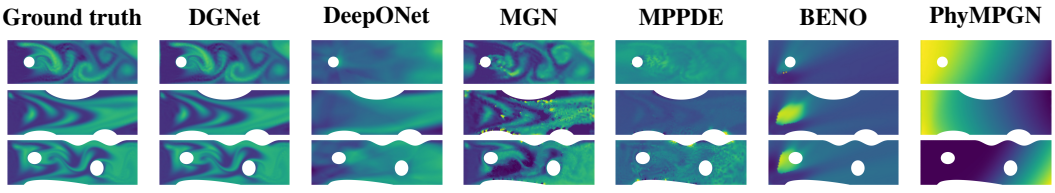

Figure 3: Visualization of prediction results on scenarios with complex geometric domains. Rows from top to bottom correspond to the cylinder, sediment, and complex obstacle cases, respectively.

4.2.3 Generalization to Unseen Source Terms

A key motivation for DGNet is its ability to generalize beyond source terms encountered during training. To test this, we consider a laser heat treatment task governed by the transient heat equation, where the source term $f(\mathbf{x}, t)$ corresponds to the spatiotemporally varying power density of moving laser beams. During training, the model is exposed to trajectories generated from a subset of source patterns. At test time, it must adapt to entirely novel laser paths, such as spline and Lissajous curves, that were never seen during training. This setup directly evaluates the zero-shot generalization capability to new forcing conditions.

Quantitative results in Table 2 show that most baseline models suffer severe degradation when confronted with unseen source terms: their errors increase by several orders of magnitude compared to the training distribution. In contrast, DGNet maintains stable accuracy, with almost no performance drop relative to its results on seen source terms. Figure 4 further highlights this difference. DGNet accurately reproduces the spatiotemporal evolution of the temperature field, including the movement and spread of high-temperature regions, even under entirely novel laser trajectories. By comparison, baseline predictions collapse: the high-temperature regions become misplaced or overly diffused, failing to follow the true laser paths. These experiments confirm that DGNet uniquely addresses a critical limitation of existing neural PDE solvers, achieving robust generalization to unseen source terms—a scenario ubiquitous in real-world scientific and engineering applications.

Figure 4: Visualization of generalization performance on the laser heat scenario with unseen sources.

4.3 Ablation Study

We compare DGNet against four variants on the complex obstacle scenario: (A) **w/o $\mathbf{L}_{\text{physics}}$**: removes the physics prior operator; (B) **w/o $\mathbf{L}_{\text{neural}}$**: removes the neural correction operator; (C) **w/o Residual GNN**: removes the residual GNN while retaining the discrete Green solver; (D) **w/o Green**: replaces the discrete Green solver with a generic end-to-end GNN that only consumes operator features. The results are shown in Figure 5. Among all variants, **w/o Green** suffers the largest performance drop, highlighting the central role of the discrete Green solver as the physical backbone. Removing $\mathbf{L}_{\text{physics}}$ degrades performance more severely than removing $\mathbf{L}_{\text{neural}}$, demonstrating that the physics prior provides essential structural knowledge while the neural correction mainly fine-tunes discretization errors. Finally, the decline in the **w/o Residual GNN** variant confirms the residual GNN's effectiveness in capturing additional dynamics and improving overall accuracy.

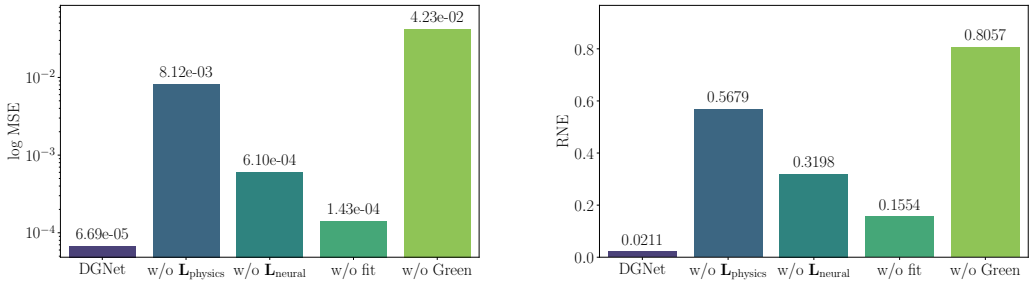

Figure 5: Ablation study on the complex obstacle scenario.

5 Conclusion

In this work, we introduced DGNet, a discrete Green network that tackles two longstanding challenges of neural PDE solvers: robust handling of irregular geometries and generalization to unseen source terms. By discretizing Green's function on graphs, DGNet transforms a classical continuous tool into a practical solver that preserves the superposition principle. Coupled with physics-based operators and GNN corrections, this framework provides a principled interface between physical fidelity and learnable flexibility. Extensive experiments across three categories of spatiotemporal PDE systems demonstrated that DGNet consistently achieves state-of-the-art accuracy, and uniquely maintains stable performance under entirely novel source terms, where existing baselines collapse. Looking ahead, several open challenges remain, including extending the method to strongly nonlinear PDEs, scaling to very large systems, and generalizing from two- to three-dimensional domains. We view these as promising directions for advancing discrete Green methods and broadening their impact in scientific machine learning.

REFERENCES

Keiiti Aki and Paul G Richards. *Quantitative seismology*. 2002.

John D Anderson. *Computational fluid dynamics: the basics with applications*. McGraw-Hill New York, 2002.

George B. Arfken, Hans J. Weber, and Frank E. Harris. *Mathematical Methods for Physicists*. Academic Press, 7th edition, 2013.

Suresh Bishnoi, Ravinder Bhattoo, Sayan Ranu, and N. M. Anoop Krishnan. Enhancing the inductive biases of graph neural ODE for modeling physical systems. In *International Conference on Learning Representations*, 2023.

Suresh Bishnoi, Jayadeva, Sayan Ranu, and N M Anoop Krishnan. BroGNet: Momentum-conserving graph neural stochastic differential equation for learning brownian dynamics. In *International Conference on Learning Representations*, 2024.

William E. Boyce, Richard C. DiPrima, and Douglas B. Meade. *Elementary Differential Equations and Boundary Value Problems*. Wiley, 11th edition, 2017.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

Francisco Sahli Costabal, Simone Pezzuto, and Paris Perdikaris. Δ-PINNs: Physics-informed neural networks on complex geometries. *Engineering Applications of Artificial Intelligence*, 127: 107324, 2024.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Gwynne Evans, Jonathan Blackledge, and Peter Yardley. *Numerical methods for partial differential equations*. Springer Science & Business Media, 2012.

Robert Joseph George, Jiawei Zhao, Jean Kossaifi, Zongyi Li, and Anima Anandkumar. Incremental spatial and spectral learning of neural operators for solving large-scale PDEs. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Efficient token mixing for transformers via adaptive fourier neural operators. In *International Conference on Learning Representations*, 2021.

David W Hahn and M Necati Özisik. *Heat conduction*. John Wiley & Sons, 2012.

Charles Hirsch. *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*. Elsevier, 2007.

Masanobu Horie and Naoto Mitsume. Physics-embedded neural networks: Graph neural PDE solvers with mixed boundary conditions. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23218–23229, 2022.

Tony Lelievre and Gabriel Stoltz. Partial differential equations and stochastic methods in molecular dynamics. *Acta Numerica*, 25:681–880, 2016.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

Rainald Löhner. *Applied computational fluid dynamics techniques: an introduction based on finite element methods*. John Wiley & Sons, 2008.

Cheng Long, Siyuan Liu, and Zhiping Wang. PGODE: Towards high-quality system dynamics modeling. In *AAAI Conference on Artificial Intelligence*, volume 37, pp. 9706–9714, 2023.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Learning nonlinear operators via DeepONet. *Nature Machine Intelligence*, 3(9):764–774, 2021.

Peter Lynch. The origins of computer weather prediction and climate modeling. *Journal of Computational Physics*, 227(7):3431–3444, 2008.

Xuhui Meng and George E Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. *Journal of Computational Physics*, 401:109028, 2020.

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, pp. 35–57. Springer, 2003.

James D Murray. *Mathematical biology: I. An introduction*, volume 17. Springer Science & Business Media, 2007.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Stephen B Pope. Turbulent flows. *Measurement Science and Technology*, 12(11):2020–2021, 2001.

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.

Roger Z Ríos-Mercado and Conrado Borraz-Sánchez. Optimization problems in natural gas transportation systems: A state-of-the-art review. *Applied Energy*, 147:536–555, 2015.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ODE integrators. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020a.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020b.

Sungyong Seo, Chuizheng Meng, and Yan Liu. Physics-aware difference graph networks for sparsely-observed dynamics. In *International Conference on Learning Representations*, 2019.

Natarajan Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022.

Allen Taflove, Susan C Hagness, and Melinda Piket-May. Computational electromagnetics: the finite-difference time-domain method. *The Electrical Engineering Handbook*, 3(629-670):15, 2005.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. In *International Conference on Learning Representations*, 2023.

Haixin Wang, Jiaxin Li, Anubhav Dwivedi, Kentaro Hara, and Tailin Wu. BENO: Boundary-embedded neural operators for elliptic PDEs. In *International Conference on Learning Representations*, 2024.

Hao Wu, Huiyuan Wang, Kun Wang, Weiyan Wang, Yangyu Tao, Chong Chen, Xian-Sheng Hua, Xiao Luo, et al. Prometheus: Out-of-distribution fluid dynamics modeling with disentangled graph ODE. In *International Conference on Machine Learning*, 2024.

Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

Bocheng Zeng, Qi Wang, Mengtao Yan, Yang Liu, Ruizhi Chengze, Yi Zhang, Hongsheng Liu, Zidong Wang, and Hao Sun. PhyMPGN: Physics-encoded message passing graph network for spatiotemporal PDE systems. In *International Conference on Learning Representations*, 2025.

A Derivation of the Discrete Green Solver

For completeness, we outline the derivation of the discrete Green solver and its relation to implicit time integration. Unlike explicit propagation matrices, which are dense and costly to construct, our approach realizes the Green operator implicitly through sparse linear solves.

A.1 Explicit vs. Implicit Green Formulation

The discrete propagation matrix $\mathbf{G}(\Delta t)$ formally maps the state $\mathbf{u}^k$ to $\mathbf{u}^{k+1}$ and serves as the discrete analogue of the continuous Green's function:

$$\mathbf{u}^{k+1} = \mathbf{G}(\Delta t)\mathbf{u}^k.$$

Each column of $\mathbf{G}(\Delta t)$ corresponds to the system response from a unit impulse at one node, while each row represents how contributions from all nodes aggregate to update a given node. However, explicitly forming $\mathbf{G}(\Delta t)$ requires $O(N^2)$ storage and computation, which is infeasible for large meshes. We therefore derive an implicit realization based on stable time integration.

A.2 Crank–Nicolson Derivation

Starting from the semi-discretized ODE system

$$\frac{d\mathbf{u}}{dt} = \mathbf{L}\mathbf{u} + \mathbf{f},$$

we apply the Crank–Nicolson scheme for stability and second-order accuracy. The time derivative is approximated by a central difference,

$$\frac{d\mathbf{u}}{dt} \approx \frac{\mathbf{u}^{k+1} - \mathbf{u}^k}{\Delta t},$$

while $\mathbf{L}\mathbf{u}$ and $\mathbf{f}$ are approximated by their averages at t^k and t^{k+1}. This yields

$$\frac{\mathbf{u}^{k+1} - \mathbf{u}^k}{\Delta t} = \tfrac{1}{2}\big(\mathbf{L}\mathbf{u}^k + \mathbf{L}\mathbf{u}^{k+1}\big) + \tfrac{1}{2}\big(\mathbf{f}^k + \mathbf{f}^{k+1}\big).$$

Rearranging terms gives the sparse linear system

$$\Big(\mathbf{I} - \tfrac{\Delta t}{2}\mathbf{L}\Big)\mathbf{u}^{k+1} = \Big(\mathbf{I} + \tfrac{\Delta t}{2}\mathbf{L}\Big)\mathbf{u}^k + \tfrac{\Delta t}{2}\big(\mathbf{f}^k + \mathbf{f}^{k+1}\big),$$

which is equivalent to applying the discrete Green's function

$$\mathbf{G}(\Delta t) = \Big(\mathbf{I} - \tfrac{\Delta t}{2}\mathbf{L}\Big)^{-1}.$$

A.3 Efficient Implementation

In practice, we do not compute the inverse explicitly. Since $\mathbf{L}$ is sparse and depends only on the static mesh geometry, the coefficient matrix $\big(\mathbf{I} - \frac{\Delta t}{2}\mathbf{L}\big)$ is also sparse and remains fixed during rollout. We therefore adopt a "factorize once, solve many times" strategy: a single sparse LU factorization is performed before rollout, and forward/backward substitution reuses the factors at each time step. This reduces the per-step cost to nearly linear in the number of nonzeros, making the discrete Green solver scalable to large systems.

B Dataset Details

We evaluate DGNet on three categories of PDE systems: (1) classical equations (Allen–Cahn, Fisher–KPP, FitzHugh–Nagumo), (2) contaminant transport in complex geometric domains, and (3) laser heat treatment with varying source terms. High-fidelity simulation data are generated with the FEniCSx finite element library on unstructured meshes and small time steps. Detailed parameters are reported in Table 3. Below we outline the dataset settings.

B.1 CLASSICAL PDES

Allen–Cahn. Phase separation with $\epsilon = 0.04$ on $[0,1]^2$, $\Delta t = 0.005$, 1000 steps. Mesh: perturbed 35×35 grid ($\sim$1296 nodes). Initial condition: random noise in $[-0.5, 0.5]$. Boundary condition: homogeneous Neumann. Trajectories: 20 (10 train / 10 test).

Fisher–KPP. Advection–reaction equation with $\rho = 1.0$, $\mathbf{c} = [0.1, 0.12]$, domain $[0,1]^2$. $\Delta t = 0.00125$, 1600 steps. Mesh: perturbed 40×40 grid ($\sim$1681 nodes). Initial condition: Gaussian blobs. Boundary: inflow Dirichlet, outflow natural. Trajectories: 20 (10 train / 10 test).

FitzHugh–Nagumo. Two-component excitable system with parameters $D_u = 1/L_s^2$, $D_v = 6.181/L_s^2$, $\Gamma = 9.657$, $\alpha = 0.5$, $\beta = 2.1$, $L_s = 16\pi$. Domain $[0,1]^2$, $\Delta t = 0.005$, 1200 steps. Mesh: perturbed 99×99 grid ($\sim$10,000 nodes). Initial: circular perturbation. Boundary: inflow Dirichlet ($u = 0, v = 0$), others Neumann. Trajectories: 10 (7 train / 3 test).

B.2 COMPLEX GEOMETRIC DOMAINS

All scenarios solve coupled Navier–Stokes and reaction–advection–diffusion equations. Fluid density $\rho = 1.0$, viscosity $\mu = 5 \times 10^{-3}$, contaminant $D = 2 \times 10^{-3}$, $\rho_g = 0.1$, $k_d = 0.025$. Simulation: $T = 100$s, $\Delta t = 0.05$s, 2000 steps. Trajectories: 20 (15 train / 5 test).

Cylinder. Rectangular channel $[0, 24] \times [0, 8]$, obstacle: central circle ($r = 1.0$), $Re = 240$.

Sediments. Same channel, three asymmetric smooth bumps attached to top/bottom walls.

Complex Obstacles. Same channel, two ellipses + NACA airfoil-shaped bumps/grooves.

B.3 LASER HEAT TREATMENT (VARYING SOURCE TERMS)

Transient heat conduction with $\rho = 7850$, $c_p = 450$, $k = 50$, $h_{\text{conv}} = 25$. Domain: complex plate with gear + circular holes. Boundary: convective, initial $T = 298.15$K. Heat source $Q(x,t)$: 10 moving lasers with random paths (orbit, spline, Lissajous, etc.). Simulation: $T = 60$s, $\Delta t = 0.5$s, 120 steps. Trajectories: 40 (20 train / 20 test).

Table 3: Simulation parameters of all datasets used in experiments.

Dataset	Domain	Nodes (approx.)	Δt	Steps	Traj. (train/test)
		Classical PDEs			
Allen–Cahn	$[0,1]^2$	1,296	0.005	1,000	10 / 10
Fisher–KPP	$[0,1]^2$	1,681	0.00125	1,600	10 / 10
FitzHugh–Nagumo	$[0,1]^2$	10,000	0.005	1,200	7 / 3
		Complex Geometric Domains			
Cylinder	$[0,24] \times [0,8]$	2,275	0.05	2,000	15 / 5
Sediments	$[0,24] \times [0,8]$	5,758	0.05	2,000	15 / 5
Complex Obstacles	$[0,24] \times [0,8]$	8,841	0.05	2,000	15 / 5
		Varying Source Term			
Laser Heat	Complex Plate	6,072	0.5	120	20 / 20

C EXPERIMENTAL SETUP

All experiments are conducted on a workstation with four NVIDIA RTX 4090 (24GB) GPUs. Training typically completes within hours to one day depending on dataset size. Ground-truth PDE data are generated using FEniCSx (dolfinx 0.9.0). DGNet is implemented in PyTorch 2.5.1 with CUDA 12.4 and Python 3.13.5.

C.1 Training Hyperparameters

All models are trained with Adam and a step LR scheduler. We adopt trajectory segmentation for efficiency, and apply the loss only to the first and last predictions of each segment to balance single-step accuracy and long-term stability. Table 4 lists the hyperparameters for each experimental scenario (the three contaminant transport cases share identical settings).

Table 4: Training hyperparameters for each experimental scenario.

Scenario	Batch Size	Initial LR	Epochs	Decay Step	Decay Rate	Sub-seq. Length	Train/Test Trajs.
Allen–Cahn	36	1e-3	600	200	0.1	20	10/10
Fisher–KPP	24	5e-4	3000	600	0.1	10	10/10
FitzHugh–Nagumo	10	1e-3	2000	500	0.1	10	7/3
Cylinder	6	5e-4	1200	300	0.2	6	15/5
Sediments	6	5e-4	1200	300	0.2	6	15/5
Complex Obstacles	6	5e-4	1200	300	0.2	6	15/5
Laser Heat	8	5e-4	400	200	0.1	8	20/20

C.2 Implementation Details

To solve the sparse linear system in Eq. (9), $\mathbf{A}\mathbf{u}_{\text{inter}}^{k+1} = \mathbf{b}$, we employ a factorize-once strategy with GPU-accelerated sparse algebra.

Pre-computation. The system matrix $\mathbf{A} = (\mathbf{I} - \frac{\Delta t}{2}\mathbf{L})$ depends only on mesh geometry and Δt, so it is built once before training. It is converted to a CuPy sparse matrix and factorized using `cupy.sparse.linalg.splu`, producing LU factors cached as `A_lu`.

Forward pass. At each step, the cached LU factors are used to efficiently solve $\mathbf{A}\mathbf{u}_{\text{inter}}^{k+1} = \mathbf{b}$ by forward/backward substitution, avoiding explicit matrix inversion and enabling fast long-term rollouts.

Backward pass. We implement a custom `torch.autograd.Function` with an adjoint formulation. Gradients are obtained by solving $\mathbf{A}^T\mathbf{g}_b = \mathbf{g}_{u,\text{inter}}$ using the cached LU factors, ensuring differentiability and efficiency.

D More Experimental Results

D.1 Visualization of Learned Operator $\mathbf{L}$

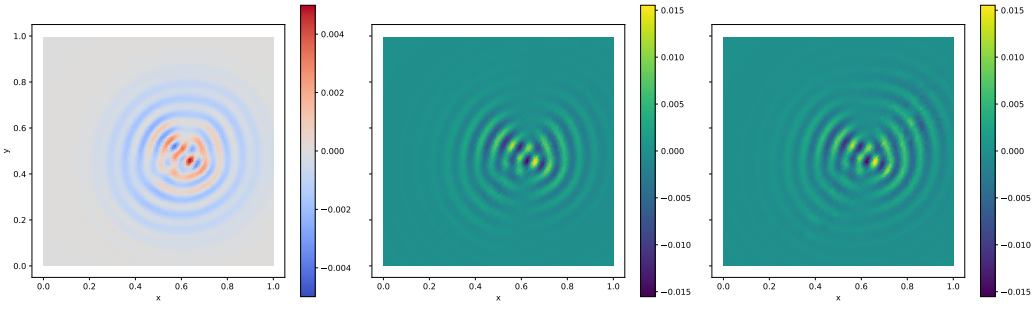

Figure 6: Comparison of the learned operator with the reference. Left: input physical field. Center: reference operator computed by finite difference. Right: operator predicted by DGNet.

Figure 6 compares the operator predicted by our model (right) with the reference computed by a finite difference scheme (center), given the same input physical field (left). The learned operator closely matches the reference in both structural patterns and numerical scale, demonstrating that DGNet accurately captures operator-level dynamics.

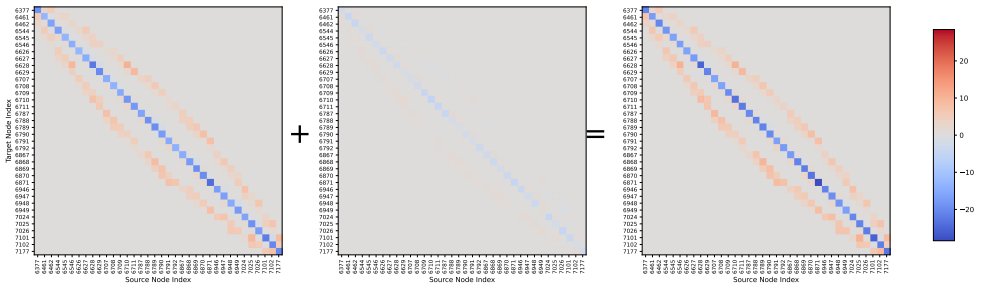

Figure 7: Heatmap visualization of operator matrices. Left: $\mathbf{L}_{\text{physics}}$. Middle: $\mathbf{L}_{\text{neural}}$. Right: final corrected operator $\mathbf{L}$.

Figure 7 shows heatmaps of $\mathbf{L}_{\text{physics}}$ (left), $\mathbf{L}_{\text{neural}}$ (middle), and the final corrected operator $\mathbf{L}$ (right). The magnitude of entries in $\mathbf{L}_{\text{neural}}$ is significantly smaller than in $\mathbf{L}_{\text{physics}}$, indicating that the neural component mainly provides fine-grained corrections on top of the physical prior.

D.2 Full Performance Comparison

Table 5 presents the complete performance comparison with standard deviations.

Table 5: Results across experimental scenarios from three categories of spatiotemporal PDE systems. For both MSE (in log scale) and RNE, lower values indicate better performance. The best results are highlighted in bold. Compared to Table 2, this table additionally reports the standard deviations.

Scenario	Metric	DeepONet	MGN	MP-PDE	BENO	PhyMPGN	DGNet
Allen-Cahn	MSE	2.60e-01 (1.2e-02)	2.70e-01 (1.5e-02)	8.52e-01 (3.1e-02)	2.52e+00 (1.1e-01)	5.16e-01 (2.4e-02)	**8.75e-03 (5.1e-04)**
	RNE	0.6686 (0.021)	0.6813 (0.025)	1.2109 (0.042)	2.0813 (0.051)	0.9420 (0.033)	**0.0188 (0.0005)**
Fisher-KPP	MSE	3.05e-02 (1.1e-03)	3.66e-03 (1.5e-04)	9.90e-02 (2.1e-03)	6.26e-02 (1.8e-03)	1.50e-02 (0.9e-03)	**2.59e-04 (1.2e-05)**
	RNE	0.4181 (0.015)	0.1448 (0.008)	0.7530 (0.021)	0.5989 (0.019)	0.9270 (0.028)	**0.0238 (0.0009)**
FitzHugh-Nagumo	MSE	2.49e-06 (1.3e-07)	3.75e-05 (1.8e-06)	6.46e-06 (2.1e-07)	2.14e-04 (1.1e-05)	1.69e-03 (8.2e-05)	**1.18e-07 (5.5e-09)**
	RNE	0.9745 (0.031)	3.4815 (0.055)	1.4454 (0.048)	8.3106 (0.121)	23.5696 (0.523)	**0.0952 (0.0021)**
Cylinder Flow	MSE	4.44e-02 (1.5e-03)	6.38e-03 (2.1e-04)	9.31e-02 (2.5e-03)	6.76e-02 (1.9e-03)	4.13e-01 (1.4e-02)	**1.00e-04 (5.2e-06)**
	RNE	0.5976 (0.018)	0.7154 (0.023)	0.8644 (0.029)	0.7364 (0.021)	1.8201 (0.045)	**0.0196 (0.0006)**
Sediments	MSE	3.61e-02 (1.2e-03)	5.94e-03 (1.8e-04)	7.10e-03 (2.2e-04)	1.07e-01 (5.3e-03)	2.00e-01 (9.1e-03)	**4.60e-04 (1.5e-05)**
	RNE	0.4759 (0.016)	0.6103 (0.019)	0.6673 (0.021)	0.8180 (0.025)	1.1186 (0.034)	**0.0282 (0.0008)**
Complex Obstacles	MSE	5.33e-02 (1.7e-03)	7.79e-03 (2.5e-04)	6.09e-03 (1.9e-04)	7.66e-02 (2.4e-03)	2.97e-01 (1.1e-02)	**6.69e-05 (2.1e-06)**
	RNE	0.5061 (0.015)	0.6120 (0.018)	0.5410 (0.016)	0.6069 (0.019)	1.1956 (0.037)	**0.0211 (0.0005)**
Laser Heat	MSE	2.48e+03 (1.2e+02)	4.98e+03 (1.5e+02)	3.88e+03 (1.4e+02)	1.95e+03 (1.1e+02)	6.78e+03 (2.1e+02)	**1.76e+01 (8.5e-01)**
	RNE	0.1208 (0.005)	0.1711 (0.008)	0.1510 (0.007)	0.1071 (0.004)	0.1998 (0.009)	**0.0102 (0.0003)**