Physics-informed neural networks for solving derivative-constrained partial differential equations

Kentaro Hoshisashi,* Carolyn E. Phelan, and Paolo Barucca
Department of Computer Science, University College London, Gower Street, London WC1E 6BT, United Kingdom

(Received 9 June 2025; accepted 14 April 2026; published 4 May 2026)

Physics-informed neural networks (PINNs) recast partial differential equations (PDEs) solving as an optimization problem in function space by minimizing a residual-based objective, yet many applications require additional derivative-based relations that are just as fundamental as the governing equations. In this paper, we present derivative-constrained PINNs (DC-PINNs), a general framework that treats constrained PDE solving as an optimization guided by a minimum objective function criterion where the physics resides in the minimum principle. DC-PINNs embed general nonlinear constraints on states and derivatives, e.g., bounds, monotonicity, convexity, incompressibility, computed efficiently via automatic differentiation, and they employ self-adaptive loss balancing to tune the influence of each objective, reducing reliance on manual hyperparameters and problem-specific architectures. DC-PINNs consistently reduce constraint violations and improve physical fidelity versus baseline PINN variants, representative hard-constraint formulations on benchmarks, including heat diffusion with bounds, financial volatilities with arbitrage-free, and fluid flow with vortices shed. Explicitly encoding derivative constraints stabilizes training and steers optimization toward physically admissible minima even when the PDE residual alone is small, providing reliable solutions of constrained PDEs grounded in energy minimum principles.

DOI: 10.1103/5bbf-p6zk

I. INTRODUCTION

Physics-informed neural networks (PINNs) [1] have recently emerged as a powerful deep learning framework to solve partial differential equations (PDEs). By embedding PDE residuals directly into the loss function of a neural network, PINNs recast the task of solving PDEs as an optimization problem in the function space, thus shifting the focus from discrete numerical schemes to a variational viewpoint that naturally connects to broader physical principles [2,3]. Unlike finite-difference and finite-element solvers, which remain highly effective but can become costly in many-query settings such as inverse problems, parameter inference, and uncertainty quantification, as well as on complex or evolving geometries that require repeated remeshing, PINNs provide a flexible formulation that can incorporate additional physical structure.

This reformulation represents a theoretical shift from viewing the PDE as a forward operator to interpreting it as the condition that the system minimizes a variational or energy principle [4]. The physics is encoded not only in the PDE but in the fact that the admissible solution minimizes an appropriate functional [5], leveraging a minimum objective function criterion. This perspective has been discussed in a broader scientific context, from minimum dissipation principles in fluid dynamics [6,7], to energy-based formulations in statistical mechanics and quantum systems [8], and to physics-guided optimization principles in complex systems [9]. In this perspective, PINNs and their extensions provide a computational framework that leverages this minimum principle by embedding the governing equations and constraints into a single objective, and training the network corresponds to searching for the physical extremum. Such constraints can be as fundamental to physics as the PDE itself, and neglecting them risks producing solutions that are mathematically feasible but physically implausible [8].

However, despite their empirical successes, PINNs still face challenges in the presence of additional equality or inequality constraints beyond the PDE itself, as pioneered in [10–13]. Such constrained PDEs are ubiquitous in real-world applications, e.g., bounds, monotonicity, convexity, incompressibility, to name a few. For instance, incompressible flow models rely on derivative consistency to link pressure gradients and velocity fields, and violations can lead to nonphysical wake dynamics. In quantitative finance, no-arbitrage requirements translate into derivative inequalities (e.g., monotonicity and convexity) that render a calibrated surface unusable if violated. Even when derivative constraints are not explicitly imposed, the adaptive balancing component introduced here is designed to reduce sensitivity to manual loss-weight tuning, although we only validate it in the derivative-constrained setting in this manuscript. Naive techniques for embedding constraints can lead to unstable

*Contact author: k.hoshisashi@ucl.ac.uk

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

training, slow convergence, and violation of constraints [14]. Although more sophisticated approaches have been proposed to better handle constraints in PINNs, conservative PINNs, and enforce equality constraints, augmented Lagrangian approaches and theory-guided neural networks show stronger performance in PDEs constrained by inequality [15–20]. However, a persistent difficulty lies in balancing competing error terms, the residual PDE, boundary conditions, and constraint penalties, which often differ in magnitude and sensitivity. As a result, these methods remain highly dependent on delicate hyperparameter tuning and problem-specific architectural choices.

In this study, we propose derivative-constrained PINNs (DC-PINNs), a general and robust framework for solving derivative-constrained PDEs using deep learning. Our approach seamlessly integrates the constraints into the learning process, ensuring that the solution satisfies the prescribed conditions while maintaining the benefits of PINNs. The key contributions of this study include:

(1) A flexible constraint-aware loss function that admits general nonlinear constraints and seamlessly incorporates them into the PDE residual objective.

(2) Self-adaptive loss balancing techniques that automatically tune the weightings of the objective terms, including derivatives obtained through automatic differentiation, stabilizing training across diverse problem settings.

(3) Demonstration of the approximation ability of DC-PINNs on benchmark PDEs from simple to complex, including the heat equations, structural modeling of finance, and fluid dynamics problems, showcasing improvements over PINNs approaches.

The study aims to highlight the importance of explicit constraint handling in PINNs, as relying solely on the PDE residual term may lead to solutions that violate critical physical principles. The extended approach can explore an application to several key areas:

(1) Basic PDE problems with added physical insights.

(2) Problems requiring multiple derivative constraints to avoid system failure.

(3) Complex problems with diverse internal dynamics that typically resist convergence.

We present applications including heat equations, volatility surface models in finance, and incompressible flows in fluid dynamics, comparing our framework with existing approaches. Recasting derivative-constrained PDE solving as a multi-objective optimization problem opens up the possibility of efficiently solving a wider range of physically constrained systems with improved accuracy and reliability.

Baselines and fairness. Standard PINNs do not encode derivative inequalities. We therefore include (i) $PINNs + ineq(fixed)$, which adds derivative inequality attentions with fixed global weights, and (ii) hard-constraint baselines (*hPINNs (pen.)*, *hPINNs (AL)*, *AL-PINNs*), in which inequalities are treated by homotopy or augmented Lagrangians. Compared to these, DC-PINNs uses one-sided penalties on derivative constraints but learns both category-level coefficients and sample-level scalings, which we ablate to disentangle the benefit of constraints versus adaptivity.

II. PROBLEM FORMULATION

A. Derivative-constrained PDE problem

We seek a mapping $u : \Omega \subset \mathbb{R}^n \to \mathbb{R}^d$ that is approximated by a neural network $u_\theta \equiv \phi_\theta(\boldsymbol{x})$. Let $\mathcal{D}$ denote the set of differential operators applied to u_θ, including the zeroth-order identity. Training is posed as a constrained optimization problem,

$$\hat{\theta} = \arg\min_\theta \ \mathcal{L}(\boldsymbol{x}, \mathcal{D}u_\theta) \ \text{ s.t. } \begin{cases} f(\boldsymbol{x}, \mathcal{D}u_\theta) = 0 & \boldsymbol{x} \in \Omega, \\ b(\boldsymbol{x}, \mathcal{D}u_\theta) = 0 & \boldsymbol{x} \in \partial\Omega, \\ h(\boldsymbol{x}, \mathcal{D}_h u_\theta) \leqslant 0 & \boldsymbol{x} \in \Omega, \end{cases} \tag{1}$$

where $\mathcal{L}$ is the training objective, f represents PDE residuals in the domain, b represents the boundary operator, and h represents derivative inequality constraints. Here, $\mathcal{D}_h \subseteq \mathcal{D}$ specifies the derivatives that appear in the inequalities. A problem is called *derivative-constrained* when it includes inequality conditions on derivatives of u. Typical examples are monotonicity constraints ($\nabla u \geqslant 0$), directional convexity constraints ($\mathrm{diag}(\nabla^2 u) \geqslant 0$), or slope bounds ($\|\nabla u\| \leqslant L$). This notion refers to constraints on derivatives and not to the order of the PDE itself, and reflects physical phenomena that cannot be fully expressed by the PDE alone.

For discretization, we employ supervised data $D_0 = \{(\boldsymbol{x}_0, u_0)\} = \{(\boldsymbol{x}_i^{(0)}, u_i^{(0)})\}_{i=1}^{N_0}$, boundary points $\boldsymbol{x}_b = \{\boldsymbol{x}_i^{(b)}\}_{i=1}^{N_b}$, interior collocation points $\boldsymbol{x}_f = \{\boldsymbol{x}_i^{(f)}\}_{i=1}^{N_f}$, and inequality-enforcement points $\boldsymbol{x}_h = \{\boldsymbol{x}_i^{(h)}\}_{i=1}^{N_h}$.

This work focuses on inequality constraints involving derivatives and adopts a discretize-then-optimize strategy with gradient-based training of artificial neural networks (ANNs). The continuous problem is first discretized into a finite-dimensional nonlinear problem. The derivatives with respect to the inputs are evaluated by automatic differentiation (AD) through the typical ANN, that is, the multilayer perceptron (MLP). To ensure that all required partial derivatives exist and are well defined, the activation function must be sufficiently smooth with respect to the inputs. For PDEs that involve derivatives up to order k, the network output must be at least C^k continuous. Because training employs gradient-based optimization through these derivatives, the activation should, in practice, be C^{k+1}, which is satisfied, for example, by the hyperbolic tangent (tanh) activation function.

B. Physics-informed neural networks (PINNs)

Physics-informed neural networks (PINNs), introduced in [1], are artificial neural networks that integrate physical laws directly into the training objective via partial differential equations (PDEs). This paradigm offers a data-efficient class of universal function approximators in which the governing PDE residuals and boundary or initial conditions are treated as components of the loss function.

We consider a parametrized PDE system of the form

$$\begin{aligned} f(\boldsymbol{x}, \mathcal{D}u_\theta(\boldsymbol{x})) &= 0, \quad \boldsymbol{x} \in \Omega, \\ b(\boldsymbol{x}, \mathcal{D}u_\theta(\boldsymbol{x})) &= 0, \quad \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{2}$$

where $u_\theta \equiv \phi_\theta(\boldsymbol{x})$ denotes the neural network approximation of the true solution u. Here, $\mathcal{D}$ denotes the set of differential

operators obtained by automatic differentiation (AD) of the neural network with respect to its inputs.

The training problem is formulated as the unconstrained minimization of a total loss functional,

$$\mathcal{L} := \mathcal{L}_0 + \mathcal{L}_f + \mathcal{L}_b, \tag{3}$$

where the three terms correspond to supervised data loss, residual PDE loss in the domain, and loss of the boundary or initial condition, respectively. Explicitly, one has

$$\mathcal{L}_0 := \frac{1}{N_0} \| u_\theta(\boldsymbol{x}_0) - u_0 \|_2^2, \quad \mathcal{L}_f := \frac{1}{N_f} \| f(\boldsymbol{x}_f, \mathcal{D}u_\theta(\boldsymbol{x}_f)) \|_2^2,$$
$$\mathcal{L}_b := \frac{1}{N_b} \| b(\boldsymbol{x}_b, \mathcal{D}u_\theta(\boldsymbol{x}_b)) \|_2^2. \tag{4}$$

Here, N_0, N_f, N_b denote the cardinalities of the supervised dataset, the set of interior collocation points, and the set of boundary collocation points, respectively. Norms are evaluated over the entire vector of residuals at the respective collocation points. The observed targets u_0 contribute only to $\mathcal{L}_0$, while $\mathcal{L}_f$ and $\mathcal{L}_b$ depend solely on the PDE operator f and the boundary operator b.

C. Derivative-constrained PINNs (DC-PINNs)

We now extend the preceding framework to the case of derivative-constrained PDEs, as formulated in (1). Specifically, we consider inequality constraints on derivatives of the form

$$h(\boldsymbol{x}, \mathcal{D}_h u_\theta(\boldsymbol{x})) \leqslant 0, \quad \boldsymbol{x} \in \Omega, \tag{5}$$

where $\mathcal{D}_h \subseteq \mathcal{D}$ denotes the subset of derivatives that appear in the inequality conditions.

In order to enforce such inequality constraints within a gradient-based training scheme, we formulate a one-sided penalty functional. For each inequality operator h, we define the violation loss as

$$\mathcal{L}_h := \frac{1}{N_h} \| [\, h(\boldsymbol{x}_h, \mathcal{D}_h u_\theta(\boldsymbol{x}_h))]_+ \|_2^2, \tag{6}$$

where $[\cdot]_+ := \max\{0, \cdot\}$ denotes the standard hinge operator. It yields zero gradients inside the feasible region and a linear signal once violations occur, which we found to be numerically stable when combined with AD on input derivatives. Near $h = 0$ we smooth the derivative at the origin (C^1) to avoid optimizer jitter. We also tested smooth hinges $\mathrm{softplus}_\delta(h) = \delta \log(1 + e^{h/\delta})$ and squared hinges; while comparable in accuracy, they required tighter tuning of global weights to prevent slow drift along nearly-feasible manifolds. The hinge thus strikes a balance between strictness and simplicity for derivative inequalities.

The total loss in the DC-PINN framework generalizes the standard PINN loss by incorporating both inequality constraints and weighting coefficients:

$$\mathcal{L} := \lambda_0 \hat{\mathcal{L}}_0 + \lambda_f \hat{\mathcal{L}}_f + \lambda_b \hat{\mathcal{L}}_b + \lambda_h \hat{\mathcal{L}}_h, \tag{7}$$

where $\lambda_0, \lambda_f, \lambda_b, \lambda_h \geqslant 0$ are the scalar coefficients that balance the relative importance of each category of constraints. The modified (hatted) loss terms are defined as

$$\hat{\mathcal{L}}_0 := \frac{1}{N_0} \| m_0(u_\theta(\boldsymbol{x}_0) - u_0) \|_2^2,$$
$$\hat{\mathcal{L}}_f := \frac{1}{N_f} \| m_f f(\boldsymbol{x}_f, \mathcal{D}u_\theta(\boldsymbol{x}_f)) \|_2^2,$$
$$\hat{\mathcal{L}}_b := \frac{1}{N_b} \| m_b b(\boldsymbol{x}_b, \mathcal{D}u_\theta(\boldsymbol{x}_b)) \|_2^2,$$
$$\hat{\mathcal{L}}_h := \frac{1}{N_h} \| m_h [\, h(\boldsymbol{x}_h, \mathcal{D}_h u_\theta(\boldsymbol{x}_h))]_+ \|_2^2. \tag{8}$$

Factors m_0, m_f, m_b, m_h are the weights of diagonal matrices applied to each residual component before computing the Euclidean norm. Their role is to provide appropriate scaling to residuals corresponding to different physical quantities, thereby preventing imbalance due to differing magnitudes, and to allow selective emphasis on certain state variables or differential operators in multi-output neural networks. In contrast, the coefficients λ in (7) operate at the categorical level, balancing supervised data, PDE residuals, boundary conditions, and inequality constraints as a whole.

Often, we face several conditions for inequality constraints simultaneously. In that case, we introduce additional terms of the form $\lambda_h \hat{\mathcal{L}}_h$ for each constraint, leading to a collection of penalties corresponding to multiple inequalities. The set of all such K_h conditions can be denoted as

$$\mathcal{H}u_\theta := \{\, h_k(\boldsymbol{x}_{h_k}, \mathcal{D}_{h_k} u_\theta(\boldsymbol{x}_{h_k}))\}_{k=1}^{K_h}, \quad \boldsymbol{x} \in \Omega, \tag{9}$$

which may involve multiple, possibly derivative-based, inequality constraints.

DC-PINNs generalize standard PINNs by simultaneously enforcing PDE dynamics, boundary or initial conditions, and derivative inequality constraints. The framework allows solutions that are consistent with both the governing PDE system and additional derivative-based structural conditions that reflect physical or financial principles, but also introduces additional difficulty as it effectively becomes a multi-objective optimization problem where the different categories of losses must be balanced to reach a feasible and accurate solution.

D. Multi-objective optimization

Selecting weights and constraint-sensitive losses is delicate because large penalties improve constraint satisfaction but can make optimization ill-conditioned [18]. To improve robustness, we employ two automated balancing schemes inspired by [21,22]. The first amplifies the gradients per category to strengthen the derivative-based inequality constraints. The second mitigates multiscale imbalance across the categorized losses in (8), stabilizing the magnitudes of the gradient over epochs given the sparsity induced by (6).

a. Individual loss balancing. Following a self-adaptive weighting strategy, we attach trainable multiplicative weights m to each categorized loss and update them via gradient ascent in tandem with the network parameters θ:

$$m_\chi^{(j)}(k+1) = m_\chi^{(j)}(k) + \eta_m \nabla_{m_\chi^{(j)}} \hat{\mathcal{L}}_\chi(k), \quad \chi \in \{0, b, f, h\}. \tag{10}$$

This update increases the influence of losses with larger violations.

b. Categorized loss balancing. In parallel, we update the weights per category λ using the mean absolute gradient magnitude with respect to θ:

$$\lambda_\chi(k+1) = \begin{cases} 1, & \text{if} \overline{|\nabla_\theta \hat{\mathcal{L}}_\chi(k)|} = 0, \\ \lambda_\chi(k) + \sum_{\chi'} \overline{|\nabla_\theta \hat{\mathcal{L}}_{\chi'}(k)|} \big/ \overline{|\nabla_\theta \hat{\mathcal{L}}_\chi(k)|}, & \text{otherwise}, \end{cases} \tag{11}$$

where $\overline{|\cdot|}$ denotes the mean absolute value in the element. We use absolute values, rather than squared ones, to retain sensitivity to rare but severe inequality violations, since most of the elements of $\nabla_\theta \mathcal{L}_h$ are typically zero.

Both mechanisms adjust weights at user-specified intervals based on relative gradient scales, promoting balanced contributions of all loss terms, including unstable inequality losses, during training [21,22]. The sensitivity to η_m and p_m is reported in Appendix C.

E. Hard-constraint formulations for derivative-constrained PDEs

We next consider an alternative strategy to DC-PINNs for handling additional constraints, namely a hard-constraint approach. Several practically relevant realizations are based on the augmented Lagrangian methodology [18,23] and the classical constrained optimization theory [24,25]. We treat these approaches as fair baselines because they represent the three most standard families of constrained optimization in the PINN literature: (i) exact (architectural) enforcement when a closed-form transform exists, (ii) penalty continuation, and (iii) (augmented) Lagrangian updates with dual variables. All target the same feasibility objective as DC-PINNs but differ in how they manage optimization stiffness and weight selection, which complements the multi-objective discussion in Sec. II D.

a. Architectural approach (hPINN). Pointwise box constraints for boundary conditions (b) can be enforced exactly by output transforms in hPINN formulations [18],

$$\psi_\theta^{(hard)} = u_{\min} + (u_{\max} - u_{\min})\psi(\boldsymbol{x}). \tag{12}$$

This approach is effective for variables whose admissible range is known *a priori*. However, it is generally impractical for derivative-based inequalities such as $u_x \leqslant U$. We include this baseline to represent the strongest form of hard enforcement available when admissible transforms exist, and to make explicit its limited applicability to derivative inequalities.

b. Penalty method (pen.). We penalize violations of derivative inequalities via $[h_\theta]_+$ and employ an outer loop that monotonically increases the weight $\lambda > 0$ by scaling it by a factor $\beta > 1$ at outer iteration i:

$$\mathcal{L}^{(\text{pen.})} = \mathcal{L}_0 + \lambda^{(i)}\mathcal{L}_f + \lambda^{(i)}\mathcal{L}_h, \quad \lambda^{(i+1)} = \beta\lambda^{(i)} \ (\beta > 1). \tag{13}$$

This is the hard-constraint analog of a soft hinge penalty, hardened by outer iterations. This baseline is widely used because it is simple to implement, but it can introduce stiff optimization dynamics as $\lambda^{(i)}$ grows, making comparisons with adaptive weighting particularly informative.

c. Augmented Lagrangian method (AL). Derivative inequalities can also be handled via an augmented Lagrangian loop. The augmented objective with multipliers $\mu_f \in \mathbb{R}^{N_f}$, $\mu_h \in \mathbb{R}^{N_h}$, and $\mu_h \geqslant 0$ is

$$\mathcal{L}^{(\text{AL})} = \mathcal{L}_0 + \mu_f^\top f_\theta + \lambda^{(i)}\mathcal{L}_f + \mu_h^\top [h_\theta]_+ + \lambda^{(i)}\mathcal{L}_h,$$
$$\lambda^{(i+1)} = \beta\,\lambda^{(i)} \ (\beta > 1), \tag{14}$$

with projected dual-ascent updates and a multiplicatively increased penalty parameter,

$$\mu_f^{(i+1)} = \mu_f^{(i)} + \lambda^{(i)} f_\theta^{(i+1)}, \quad \mu_h^{(i+1)} = \mu_h^{(i)} + \lambda^{(i)}\big[h_\theta^{(i+1)}\big]_+. \tag{15}$$

Thus, hPINNs can extend by augmenting the treatment of inequalities while retaining hard architectural enforcement [hPINN (AL)]. In addition, all constraints can be unified into a single AL loop (AL-PINNs). We use AL-based baselines because they are a principled alternative to fixed penalty weights and are commonly adopted in constrained PINNs; however, their outer-loop updates can be sensitive to stopping criteria and can prioritize feasibility at the expense of data fit under a fixed compute budget.

To the best of our knowledge, no prior PINN study has focused on a fully unified hard-constraints formulation that simultaneously addresses multiple equality and inequality constraints. Beyond prior work on equality constraints [18,23], our formulation integrates multiple inequalities, tightening feasibility coupling, but also increasing the ill conditioning of the optimization landscape and, consequently, the risk of convergence difficulties.

III. EXPERIMENTAL DESIGN

A. Neural network setting and training configuration

In the experiments, the network architecture ψ is a fully connected feedforward neural network with hyperbolic tangent activation functions to ensure smooth differentiability. The network parameters are initialized using the Glorot scheme [26], and the optimization is performed using Adam [27] with regularization of the weight decay, following previous work [1,22]. Training data consist of uniformly sampled points for initial and boundary conditions, while square mesh grids are used for PDE residuals and derivative inequality constraints. The accuracy of the model is evaluated by computing the errors between predictions and reference solutions on dense mesh grids. Detailed analytical formulations, network architectures, and training settings for each problem are provided in Appendixes D, E 2, and F 2.

B. Baselines and fairness

Standard PINNs [1] do not encode derivative-based inequality constraints. We therefore add *PINNs-Ineq.*, which augments PINNs with a hinge penalty $\mathcal{L}_h$ while fixing

additional weights ($\lambda_\chi \equiv 1$, $m_\chi \equiv I$). *DC-PINNs* generalize this learning per component coefficients λ_χ and per sample scale m_χ (Sec. II C). We also report ablations: *no* λ ($\lambda_\chi = 1$), *no* m ($m_\chi = I$), and *static* (both initialized and then frozen). For hard-constraint baselines (Sec. II E): *hPINN (pen.)* embeds boundary conditions (hard-b) and enforces inequalities through a penalty homotopy; *hPINN (AL)* again uses hard-b with an augmented Lagrangian for inequalities; *AL-PINNs* applies an augmented Lagrangian to both equalities and inequalities without hard-b. All baselines follow the effective hyperparameters of the original paper [18]. The explored ranges and final settings appear in Appendix C.

a. Metrics. Accuracy is measured by root mean square error (RMSE). For each component $\chi \in \{0, b, f, h\}$ on the dense validation grid $\{\boldsymbol{x}_i\}_{i=1}^{N_\chi^{\text{val}}}$ with task-specific error $e_\chi(\boldsymbol{x}_i)$ (data, boundary, PDE residual, or derivative error), we report

$$\mathrm{E}_\chi = \mathrm{RMSE}_\chi = \left(\frac{1}{N_\chi^{\text{val}}} \Sigma_{i=1}^{N_\chi^{\text{val}}} e_\chi(\boldsymbol{x}_i)^2 \right)^{1/2}. \tag{16}$$

b. Environment. All models involving differentiable operators were implemented using JAX and Flax [28,29], which enable efficient and exact computation of derivatives via automatic differentiation. Training was carried out in Google Colab [30] using TPUs (v6e-1) with XLA compilation and 173 GB of high-bandwidth memory. On average, each training run required approximately 5 (Heat) to 100 (Navie-Stokes) seconds for runtime and 2–3 minutes for compiling on a TPU v6e-1 instance, with peak memory usage close to 120 GB during the largest experiments.

IV. NUMERICAL EXPERIMENTS

To demonstrate the effectiveness of the proposed framework, we performed numerical experiments on benchmark problems in heat diffusion, volatility-surface calibration in quantitative finance, and incompressible flow with complex geometries and nonlinear constraints. We compare our approach with existing PINN-based methods and include derivative profile comparisons of trained networks.

A. One-dimensional heat equation in thermodynamics

The heat equation is a classic parabolic PDE [31] and a standard testbed to evaluate the robustness of PINNs. Consider an infinitesimally thin steel beam heated at its center. The heat in the center spreads along the beam while the ends are held at zero temperature, so the temperature decays to zero as $t \to \infty$. The problem in $\boldsymbol{x} = (x, t)$ is

$$f(x,t) = \frac{\partial u}{\partial t} - \lambda \frac{\partial^2 u}{\partial x^2}, \tag{17}$$

$$\text{s.t.} \quad u(x,0) = \sin(\pi x), \quad u(0,t) = u(1,t) = 0,$$

$$\frac{\partial^2 u}{\partial x^2} \leqslant 0, \quad \frac{\partial u}{\partial t} \leqslant 0, \tag{18}$$

for $x, t \in [0, 1]$. The analytical solution is $u(x,t) = e^{-\lambda \pi^2 t} \sin(\pi x)$. The derivative constraints in (18) reflect that the second spatial derivative and the time derivative are nonpositive throughout the domain: the initially peaked profile remains concave in space while decaying in time, and set $\lambda = 0.1$. Figure 1 shows that DC-PINNs reproduce the smooth diffusive profile of the analytical solution with minimal artifacts. Standard PINNs exhibit noisy oscillations and boundary bias, and adding fixed inequality terms only partly suppresses them. hPINN and AL-PINNs appear oversmoothed, losing temporal variation through excessive constraint enforcement. In contrast, DC-PINNs maintain both spatial smoothness and temporal decay, yielding a near-zero residual map that confirms faithful satisfaction of $\partial_t u = \partial_{xx} u$ across the domain.

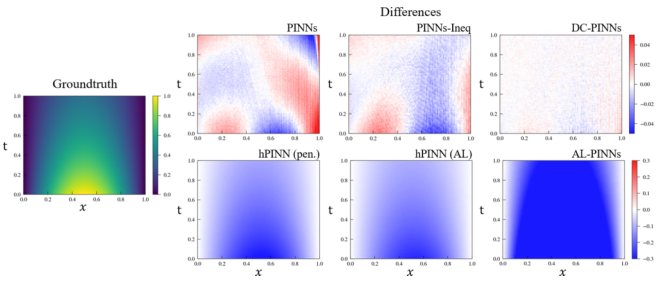


FIG. 1. Predicted solutions for the one-dimensional heat equation. Left: ground-truth magnitude. Right: model–ground-truth difference with a diverging scale.

Table I indicates that DC-PINNs achieve the best RMSE under the initial condition, the residual interior PDE, and the boundary (except for the hard-boundary treatments), while simultaneously attaining lower derivative RMSEs ($E_{h_{xx}}$, E_{h_t}). By contrast, AL-based variants drive inequality-violation metrics to zero but exhibit larger derivative errors, showing that exact feasibility alone does not guarantee accurate curvature or temporal dynamics. The qualitative second-derivative profiles in Fig. 2 corroborate this: DC-PINNs track the analytic curvature closely across space–time without spurious boundary oscillations. In general, DC-PINNs balance feasibility and accuracy, whereas purely hard treatments tend to overconstrain the solution away from the data and PDE signals. To further examine the influence of inequality constraints in DC-PINNs, Fig. 2 compares the learned second derivative profiles with the analytical curves over time. Standard PINNs deviate markedly near the boundaries, including sign reversals that violate $u_{xx} \leqslant 0$. Fixed-inequality PINNs stabilize the profiles but retain magnitude bias, indicating that static penalties cannot guarantee global compliance. AL- and hPINNs produce smoother, yet overly flattened curvature, suppressing

TABLE I. Validation (RMSE: **bold** is best) for the heat equation.

	Conditions			Accuracy				
Model	Ineq.	Hard-b	AL	E_u	E_f	E_b	$E_{h_{xx}}$	E_{h_t}
PINNs				0.013	0.032	0.027	0.677	0.068
PINNs+Ineq.	✓			0.010	0.035	0.010	0.522	0.037
hPINN (pen.)	✓	✓		0.144	0.010	**0.000**	1.433	0.142
hPINN (AL)	✓	✓	✓	0.131	0.010	**0.000**	1.310	0.129
AL-PINNs	✓		✓	0.463	0.008	0.004	4.572	0.454
DC-PINNs	✓			**0.002**	**0.006**	0.003	**0.091**	**0.009**

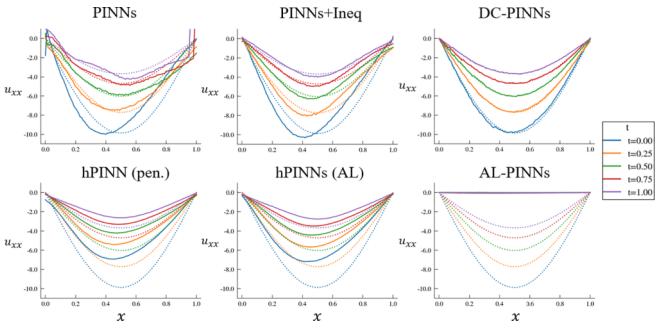


FIG. 2. Second-derivative profiles with respect to x at multiple time snapshots. Dashed: analytical solution.

temporal evolution. AL-PINNs almost collapse to zero curvature. In contrast, DC-PINNs preserve the parabolic shape and maintain nonpositivity of u_{xx} everywhere, confirming that embedding derivative inequalities in the loss is essential for both physical realism and temporal coherence.

B. Quantitative analysis and ablation

We analyze the impact of the enforcement of inequality and adaptive balancing through controlled ablations. The hyperparameter sensitivity to $\eta_m \in \{10^{-4}, 10^{-3}, 10^{-2}\}$ and $p_m, p_\lambda \in \{10, 100, 1000\}$ is summarized in Tables VII–IX in Appendix C. The results consistently favor $\eta_m \approx 10^{-3}$-10^{-2} and moderate update intervals, indicating robustness to tuning. The ablation in Table II clarifies the contributions of inequality enforcement (L_h) and adaptive balance (λ, m). Introducing L_h alone improves the satisfaction of the constraint and reduces RMSE relative to the unconstrained baseline, indicating that inequality enforcement directly improves physical fidelity. The greatest gains occur when the balance between components (λ) and sample-wise (m) balance is active, producing up to an order of magnitude reduction in RMSE and constraint errors. Removing either element degrades performance, particularly in E_{h_t}, suggesting that temporal derivatives are sensitive to unbalanced weighting. Static weights do not maintain consistent constraint satisfaction, underscoring the value of dynamic adaptation.

C. Volatility surface calibration in finance

The volatility-surface calibration problem is an inverse PDE setting in quantitative finance, where the objective is to

TABLE II. Ablation on validation (RMSE: **bold** is best) and adaptive balancing in the heat equation. λ: component-wise; m: sample-wise.

	Conditions			Accuracy				
Model	Ineq.	λ	m	E_0	E_f	E_b	$E_{h_{xx}}$	E_{h_t}
PINNs				0.013	0.032	0.027	0.678	0.068
PINNs-Ineq. (fixed)	✓			0.010	0.036	0.010	0.524	0.037
DC-PINNs (no λ)	✓		✓	0.012	0.056	0.020	0.501	0.050
DC-PINNs (no m)	✓	✓		0.004	0.008	0.006	0.160	0.016
DC-PINNs (full)	✓	✓	✓	**0.002**	**0.006**	**0.003**	**0.091**	**0.009**

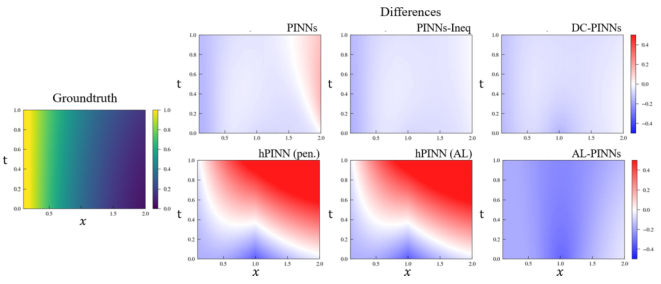


FIG. 3. Predicted solutions for the volatility-surface calibration. Left: ground-truth magnitude. Right: model–ground-truth difference with a diverging scale.

recover all the option price data from the observed data [32]. The model satisfies

$$f(x,t) = \frac{\partial u}{\partial t} - \frac{1}{2}\sigma(x,t)x^2\frac{\partial^2 u}{\partial x^2} + rx\frac{\partial u}{\partial x},$$
$$\sigma(x,t) = \sigma_A + \frac{A}{x} + Bt, \tag{19}$$

$$\text{s.t.} \quad u_{t=0} = (s_0 - x)^+, \quad u_{x=0} = s_t,$$
$$\frac{\partial u}{\partial x} \leqslant 0, \quad \frac{\partial^2 u}{\partial x^2} \geqslant 0, \quad \frac{\partial u}{\partial t} \geqslant 0, \tag{20}$$

where u is the European call option price. Synthetic data are generated from the analytical $\sigma(\cdot)$ with $\sigma_A = 0.2$, $A = 0.1$, $B = 0.2$ [33,34]. The domain is $x \in [0, 2]$, $t \in [0, 1]$ with $s_0 = 1.0$, $r = 0.05$, and Gaussian noise added to Monte Carlo prices (Appendix E). During training, $u := \psi_\theta(x, t)$. The no-arbitrage inequalities $\partial_x u \leqslant 0$, $\partial_{xx} u \geqslant 0$, and $\partial_t u \geqslant 0$ ensure economically valid surfaces. Embedding these constraints in the objective enables DC-PINNs to produce PDE-consistent, arbitrage-free surfaces. Figure 3 shows reconstructed surfaces. DC-PINNs yield smooth, monotonic profiles consistent with the ground truth, whereas unconstrained and fixed-penalty PINNs exhibit locally irregular curvature near strike extremes. Table III summarizes the calibration performance. DC-PINNs achieve the lowest overall RMSE, particularly in the initial and boundary conditions, while maintaining RMSEs of zero or near-zero constraints for $\partial_x u$, $\partial_{xx} u$, and $\partial_t u$. Fixed-penalty PINNs moderately improve over the unconstrained baseline, yet still violate constraints near maturity boundaries. AL- and hPINNs enforce hard constraints, but show optimization

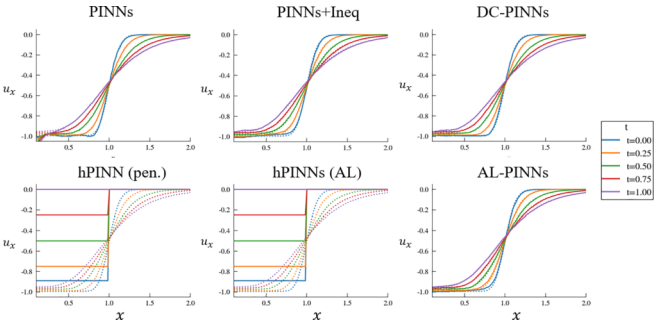


FIG. 4. Spatial derivative $\partial u/\partial x$ at multiple time snapshots. Dashed: numerical reference.

TABLE III. Validation (RMSE: **bold** is best) for the volatility surface calibration.

Model	Conditions			Accuracy					
	Ineq.	Hard-b	AL	E_0	E_b	E_f	E_{h_x}	$E_{h_{xx}}$	E_{h_t}
PINNs				0.002	**0.002**	**0.002**	0.154	0.065	**0.000**
PINNs+Ineq. (fixed)	✓			0.002	0.003	0.004	0.098	0.019	0.029
hPINN (pen.)	✓	✓		0.477	0.007	0.878	0.533	0.423	**0.000**
hPINN (AL)	✓	✓	✓	0.477	0.007	0.878	0.533	0.419	**0.000**
AL-PINNs	✓		✓	0.002	0.003	0.002	0.057	**0.005**	0.032
DC-PINNs	✓			**0.001**	**0.002**	**0.002**	**0.053**	0.095	**0.000**

instability and poorer fits. Adaptive balancing in DC-PINNs ensures stable convergence and joint enforcement of derivative constraints, improving both data fidelity and physical admissibility. Figure 4 shows $\partial u/\partial x$ over time. Standard PINNs and fixed-inequality variants capture the overall trend but deviate near the strike boundaries. DC-PINNs remain closer to the reference in the outer regions, consistent with the lower E_{h_x} in Table III. AL- and hPINNs introduce discontinuities around $x \approx 1$, producing piecewise or over-smoothed gradients that break the expected monotonicity. AL-PINNs recover the general trend but underestimate slopes in the interior, indicating excessive penalization of higher-order terms.

As in the other experiments, derivative profile analyses (Fig. 5) show that DC-PINNs capture the solution while satisfying nonlinear constraints, outperforming traditional PINNs. The framework is versatile and applies to data-driven calibration of PDE parameters.

D. Navier–stokes equations in fluid dynamics

We now consider the incompressible Navier–Stokes equations, a challenging benchmark in fluid dynamics [1]. The flow past a 2D circular cylinder (von Kármán vortex street) is modeled. The governing equations in convective form are

$$\begin{aligned} f^u &= \frac{\partial u}{\partial t} + \mu_1\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) + \frac{\partial p}{\partial x} - \mu_2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0 \\ f^v &= \frac{\partial v}{\partial t} + \mu_1\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) + \frac{\partial p}{\partial y} - \mu_2\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0 \end{aligned}, \tag{21}$$

$$\text{s.t.} \quad (u, v)|_{x=-15} = (u, v)|_{y=-8,8} = (1, 0), \quad \left.\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}\right|_{x=25} = p|_{x=-15} = 0, \quad u = v = 0 \text{ on the cylinder surface}, \tag{22}$$

$$\nabla \cdot \mathbf{V} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \quad |\nabla p| \leqslant \frac{1}{2}\rho U^2 \quad , \tag{23}$$

where $\mathbf{V} = (u, v)$, p is pressure, and $\mu_1 = 1$, $\mu_2 = 1/\text{Re}$ with Re the Reynolds number. We treat this as PDE discovery, estimating $\mu = (\mu_1, \mu_2)$ from the data. The flow past a cylinder of diameter $D = 1$ at $\text{Re} = 100$ is simulated. Training samples of $\mathbf{V}$ are drawn at 5000 random points in $x \in [1, 8]$, $y \in [-2, 2]$ (cylinder centered at the origin), and $t \in [0, 20]$ in periodic steady flow. Incompressibility ($\nabla \cdot \mathbf{V} = 0$) enforces mass conservation [35]; the bound of the pressure gradient follows the Bernoulli scale with $\rho = U = 1$ [36]. Reference solutions are computed with the NEKTAR++ SPECTRAL / HP ELEMENT code [37] (Appendix F).

For the two-dimensional cylinder wake, Table IV shows that the relative ordering depends on which objective is prioritized. AL-PINNs attain the lowest pressure-constraint error $E_{h_{\nabla p}}$ and the lowest E_{0_u}, while the hPINN (pen.) baseline yields the lowest E_{0_v} and the smallest PDE residuals (E_{f_u}, E_{f_v}). DC-PINNs reduce $E_{h_{\nabla p}}$ relative to unconstrained PINNs and recover μ_2 close to its reference value, but they do not improve E_0 or E_f over the best-performing baselines in this benchmark. We therefore interpret the Navier–Stokes results primarily as evidence of improved constraint handling,

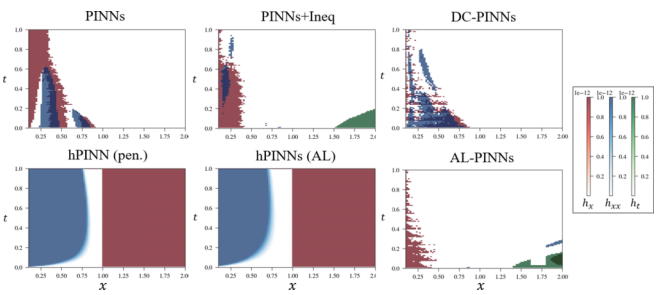


FIG. 5. Inequality-violation maps for learned local volatilities; dashed lines indicate exact references.

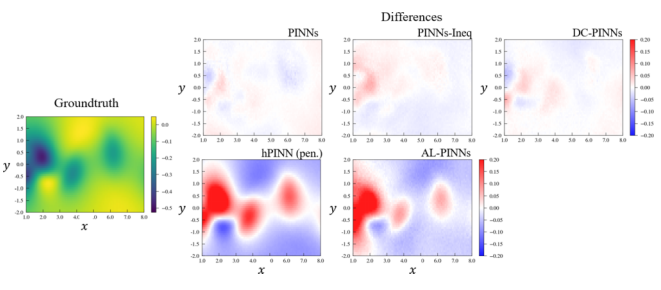


FIG. 6. Predicted pressure $p(x, y, t)$ (mean-adjusted) at $t = 10$ for the Navier–Stokes case. Left: ground-truth magnitude. Right: model–ground-truth difference with a diverging scale.

TABLE IV. Validation (RMSE: **bold** is best) for Navier–Stokes equations.

Model	Conditions		Accuracy					μ	
	Ineq.	AL	E_{0_u}	E_{0_v}	E_{f_u}	E_{f_v}	$E_{h_{\nabla p}}$	μ_1	μ_2
PINNs			0.395	0.395	0.018	0.019	0.159	**0.977**	**0.010**
PINNs+Ineq. (fixed)	✓		0.395	0.391	0.020	0.019	0.154	**0.977**	0.011
hPINN (pen.)	✓		0.350	**0.304**	**0.004**	**0.004**	0.165	0.758	0.020
AL-PINNs	✓	✓	**0.343**	0.350	**0.004**	0.005	**0.142**	0.875	0.012
DC-PINNs	✓		0.395	0.391	0.018	0.016	0.153	**0.977**	**0.010**

with an explicit accuracy–feasibility trade-off under a fixed training budget. A systematic study of the dependence on network capacity and training time is left for future work. Across hyperparameter sweeps (Table IX), the recovered viscosity consistently converges to $\mu_2 \approx 0.01$ for $\nu = 10^{-2}$ (Re $= 100$), confirming accurate inference of the underlying physics (Appendix C). Figure 6 compares reconstructed pressure fields in a representative phase of the shedding cycle. Most methods reproduce the gross wake symmetry, and the snapshot-level differences are subtle; consequently, we use the figure as qualitative context and rely on Table IV for quantitative assessment. As discussed above, these results reinforce the central trade-off in this benchmark: improved feasibility of the derivative constraint does not necessarily coincide with the lowest data error or residual.

V. DISCUSSIONS

A. Computational complexity

We report wall-clock training times for all methods on the three benchmark problems using identical multilayer perceptron (MLP) architectures and data pipelines as in the accuracy experiments. For the heat equation, the network consists of two hidden layers with 100 neurones each, trained on grids of 31×31 interior points, 1001 initial points, and 2×1001 boundary points. The volatility surface problem employs a fully connected network with four hidden layers of 50 neurones each, using an interior grid of 50×50, 1000 observed prices, and 2×1001 boundary points. For the two-dimensional Navier-Stokes case, we used eight hidden layers with 20 neurones per layer and $N_\chi = 5000$ collocation samples. Table V summarizes the training times per run, measured in seconds until the reported evaluation, under identical JIT compilation and data loading settings.

Across all experiments, DC-PINNs incur roughly a 1.5–2× training-time overhead relative to unconstrained PINNs, but this cost yields systematic gains in physical consistency and, in most cases, accuracy. For the Heat problem, second derivative violations ($u_{xx} \geqslant 0$, $u_t \geqslant 0$) are markedly reduced; in the case of the volatility surface, boundary and derivative errors decline, producing price surfaces consistent with Dupire's constraints; and in Navier–Stokes, velocity and pressure errors remain comparable while divergence and momentum residuals tighten, improving stability. Overall, the results show that DC-PINNs are most beneficial when derivative constraints dominate the ill-posedness (as in heat and volatility), whereas for coupled multifield systems (Navier–Stokes), they primarily enhance consistency rather than headline accuracy, with AL- or hPINN variants offering faster but occasionally less stable alternatives.

TABLE V. Wall-clock training time (s) across benchmark problems.

Method	Heat Problem	Volatility Surface	Navier–Stokes
PINNs	2.16	3.93	58.52
PINNs+Ineq	2.19	4.05	82.14
hPINNs (pen.)	2.82	3.85	42.55
hPINNs (AL)	2.88	4.36	52.68
AL-PINNs	2.02	3.51	
DC-PINNs	**4.41**	**6.17**	**97.10**

B. Evaluations

We evaluated four main aspects of performance, namely accuracy, stability, efficiency, and violation. Accuracy refers to the final validation error obtained at the end of training. Stability is measured through the normalized total variation of the validation error trajectory, which captures oscillations during optimization. Efficiency is assessed by the time required for the error to fall to half of its initial value, together with the area under the improvement curve, both measured with respect to wall-clock time. The violation metric quantifies the degree to which physical or structural constraints remain unsatisfied at the end of the training.

Let $\mathbf{e} = [e_1, \ldots, e_n]$ denote the validation errors recorded at times $\mathbf{t} = [t_1, \ldots, t_n]$ with $e_{\min} = \min_i e_i$, $e_{\max} = \max_i e_i$, and a small positive constant ε. The final accuracy is given by $e_{\text{final}} = e_n$. The stability measure is expressed as

$$\text{TVN} = \frac{\sum_{i=1}^{n-1} |e_{i+1} - e_i|}{e_{\max} - e_{\min} + \varepsilon}, \tag{24}$$

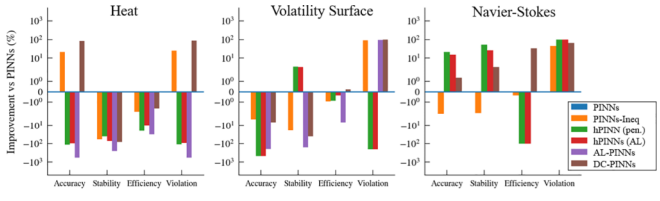


FIG. 7. Percentage improvement with respect to PINNs on a symmetric logarithmic scale. Positive values indicate improvement relative to the baseline.

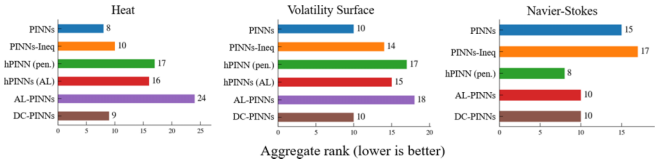


FIG. 8. Aggregate ranks across all metrics. Left: heat PDE. Center: volatility surface PDE. Right: Navier–Stokes PDE. Lower ranks indicate stronger overall performance. DC-PINNs rank highly on violation- and stability-related metrics, with overall rank depending on how accuracy and feasibility are weighted.

while efficiency is expressed as

$$\mathrm{nAUC} = \frac{1}{t_n - t_1 + \varepsilon} \int_{t_1}^{t_n} \frac{e_{\max} - e(t)}{e_{\max} - e_{\min} + \varepsilon}\, dt. \tag{25}$$

This definition rewards models that sustain low error values over time. The violation score is calculated as the average magnitude of the constraint residuals on the validation grid. Improvements are reported as percentage changes relative to PINNs. Figure 7 reports these percentage improvements on a symmetric logarithmic scale. For metrics to be minimized: $100\,(m_{\mathrm{PINNs}} - m)/\max(m_{\mathrm{PINNs}}, \varepsilon)$; for metrics to be maximized: $100\,(m - m_{\mathrm{PINNs}})/\max(m_{\mathrm{PINNs}}, \varepsilon)$. The methods are ranked per metric, and the overall performance is the Borda sum of ranks. Figure 8 reports the aggregate ranks across all metrics. The aggregated results highlight a consistent pattern across tasks: DC-PINNs reduce constraint violations and damp oscillatory training behavior, but they do so with a measurable cost in computational efficiency. In particular, DC-PINNs typically require additional derivative evaluations and longer wall-clock time, which is reflected in a lower efficiency score even when the final accuracy is comparable. Moreover, accuracy improvements are task-dependent: in Navier–Stokes, for example, AL- and hPINN baselines achieve lower data error and residual metrics. We therefore present DC-PINNs as a method that prioritizes feasibility and stability, rather than a uniformly dominating replacement for existing approaches. Aggregates confirm the trend: DC-PINNs most consistently cut violations and oscillations while preserving accuracy. Training time rises by roughly 1.5–2× versus PINNs, reflecting constraint enforcement, but the added cost is offset by improved stability and reliability when derivatives govern the physics.

C. Scalability to higher-dimensional DC-PDEs

DC-PINNs scale to higher spatial dimensions without architectural changes: as d grows, the loss still combines PDE residuals with derivative equality/inequality constraints, and automatic differentiation provides all required mixed derivatives.

a. Multidimensional Heat Equation. To illustrate this property, consider the d-dimensional heat equation defined on the unit hypercube $\Omega = (0, 1)^d$ with zero Dirichlet boundary conditions. The governing equation in variables $\boldsymbol{x} = (x_1, \ldots, x_d, t)$ is

$$f(\boldsymbol{x}, t) = \frac{\partial u}{\partial t} - \lambda \sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2}, \tag{26}$$

TABLE VI. Wall-clock training time (s) for the d-dimensional heat equation with collocation size N_χ: Fixed ($N_\chi = 100$), root ($N_\chi = \lfloor 100\sqrt{d} \rfloor$), linear ($N_\chi = 100d$), power-1.5 ($N_\chi = 100d^{1.5}$).

d	Fixed	Root	Linear	Power-1.5
1	1.86	1.78	1.73	1.88
2	1.92	2.06	2.17	2.44
8	2.52	3.76	5.28	5.63
32	5.76	6.78	9.58	30.72
128	8.64	6.81	45.01	[a]

[a]Out-of-GPU-memory.

$$\text{s.t.} \quad u(\boldsymbol{x}, 0) = \prod_{i=1}^{d} \sin(\pi x_i), \quad u(\boldsymbol{x}, t) = 0 \text{ for } \boldsymbol{x} \in \partial\Omega,$$
$$\frac{\partial u}{\partial t} \leqslant 0, \quad \frac{\partial^2 u}{\partial x_i^2} \leqslant 0 \text{ for all } i = 1, \ldots, d. \tag{27}$$

The analytical solution $u(\boldsymbol{x}, t) = e^{-\lambda \pi^2 d\, t} \prod_{i=1}^{d} \sin(\pi x_i)$ decays monotonically and satisfies the derivative signs. We set $\lambda = 0.1$ and test $d \in \{1, 2, 8, 32, 128\}$; other settings follow Appendix D.

Table VI shows that the training cost increases with dimension because denser sampling is required to approximate higher-dimensional solution manifolds. In this experiment, the practical limitation at large d is GPU memory rather than numerical instability. However, we emphasise that this is a linear heat-equation benchmark used as a controlled stress test; these timings and stability observations do not, on their own, establish scalability for complex nonlinear PDE systems.

Moving to three dimensions and beyond mainly requires two adjustments. First, the neural network must have enough capacity to represent more complex fields. Second, the collocation points must be chosen carefully so that they adequately cover the larger domain. Methods such as Latin hypercube sampling [38] or adaptive refinement [39,40] are often effective for this purpose. Soft enforcement of constraints with adaptive weighting also avoids the numerical stiffness that often appears in hard-constraint approaches. On the computational side, DC-PINNs are well suited for data parallel training on modern accelerators like [41] and can incorporate techniques developed for large-scale PINNs, such as domain decomposition in XPINNs [42], which splits high-dimensional domains into parallel subproblems [42,43], and adaptive sampling.

VI. CONCLUSION AND FUTURE WORK

We cast the problem of solving PDEs with derivative-based equality and inequality constraints into a unified derivative-constrained optimization framework, and proposed derivative-constrained physics-informed neural networks (DC-PINNs) to realize this framework. DC-PINNs combine constraint-aware losses with self-adaptive balancing to prioritize feasibility while maintaining competitive accuracy. Across the test problems considered, DC-PINNs consistently reduce violation-related metrics and stabilize training trajectories, at the cost

TABLE VII. Sensitivity of DC-PINNs on the heat problem. Reported values are RMSE in log-scale, given as ($\mathcal{L}_0$, $\mathcal{L}_f$, $\mathcal{L}_b$, $\mathcal{L}_{(h1+h2)}$) in dense evaluation grids. Bold entries indicate the overall top-four choices (by the sum of the four metrics; more negative is better).

η_m	$p_m \backslash p_\lambda$	10	100	1000
10^{-4}	10	(−1.47,−2.78,−0.77,0.38)	(−1.52,−2.92,−0.83,0.30)	(−1.51,−2.89,−0.82,0.31)
	100	(−1.40,−2.51,−0.69,0.52)	(−1.41,−2.52,−0.70,0.50)	(−1.40,−2.53,−0.69,0.51)
	1000	(−1.33,−3.00,−0.63,0.63)	(−1.32,−2.99,−0.63,0.63)	(−1.32,−2.99,−0.62,0.63)
10^{-3}	10	(−4.98,−4.11,−4.50,−4.54)	(−5.20,−4.25,−4.61,−4.97)	(−5.28,−4.20,−4.69,−5.10)
	100	(−5.12,−4.31,−4.55,−4.93)	(−5.00,−4.40,−4.47,−4.77)	(−5.23,−4.43,−4.71,−5.15)
	1000	(−4.99,−4.15,−4.44,−4.57)	(−4.84,−4.15,−4.29,−4.33)	(−5.20,−4.36,−4.62,−5.06)
10^{-2}	10	(−5.61,−4.86,−5.19,−5.77)	**(−6.21,−5.25,−6.16,−7.75)**	**(−6.01,−5.10,−5.80,−6.77)**
	100	(−5.08,−4.62,−4.45,−4.93)	**(−6.03,−5.10,−5.86,−7.11)**	**(−6.19,−5.28,−6.20,−7.97)**
	1000	(−5.10,−4.56,−4.46,−4.89)	(−5.24,−4.54,−4.56,−5.18)	(−5.90,−5.06,−5.45,−6.60)

of additional derivative evaluations and increased wall-clock time. Accuracy outcomes are problem-dependent: for Navier–Stokes, AL- and hPINNs baselines achieve lower data and residual errors under the same training budget, whereas DC-PINNs primarily improve constraint-related measures. Finally, our higher-dimensional study is based on a linear heat-equation benchmark and should be interpreted as a controlled scalability stress test rather than a proof of tractability for complex nonlinear systems.

Future extensions include exploring adaptive learning rates for constraint parameters, alternative penalty formulations near inequality boundaries, and coupling with multifidelity or operator-learning architectures to further improve tractability. These directions aim to extend the applicability of DC-PINNs to broader classes of PDE-constrained systems while preserving the balance between accuracy, stability, and physical consistency.

ACKNOWLEDGMENTS

We gratefully acknowledge anonymous reviewers for their insightful comments and constructive suggestions that have substantially improved the quality of this manuscript. We also thank the participants of the ICML 2024 AI for Science Workshop for their stimulating discussions and valuable feedback. The authors also acknowledge the Department of Computer Science at University College London for providing the high-performance computing resources that enabled this study. This work also benefited from complimentary access to cloud TPUs through Google's TPU Research Cloud (TRC) program, whose support is sincerely acknowledged herein.

DATA AVAILABILITY

All scripts to reproduce all tables and figures are available in the repository in Ref. [44].

APPENDIX A: THE MULTILAYER PERCEPTRON (MLP)

This Appendix summarizes the multilayer perceptron (MLP) used as the parametric map $u_\theta \equiv \phi_\theta$ throughout the paper and highlights how its structure supports the computation of partial derivatives with respect to inputs via automatic differentiation (AD). Let $L \geqslant 2$ denote the number of affine layers. The network has input dimension $d_0 = n$ and output dimension $d_L = d$ (with $d \geqslant 1$), so that $\phi_\theta : \mathbb{R}^n \to \mathbb{R}^d$. The model is written as a composition,

$$\phi_\theta(\mathbf{x}) \;=\; A_L \circ f_{L-1} \circ A_{L-1} \circ \cdots \circ f_1 \circ A_1(\mathbf{x}), \tag{A1}$$

where, for $l = 1, \ldots, L$, each $A_l : \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_l}$ is an affine map of the form

$$A_l(\mathbf{x}_{l-1}) \;=\; W_l\,\mathbf{x}_{l-1} + \mathbf{b}_l, \quad W_l \in \mathbb{R}^{d_l \times d_{l-1}}, \quad \mathbf{b}_l \in \mathbb{R}^{d_l}, \tag{A2}$$

and, for $l = 1, \ldots, L-1$, the activation $f_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_l}$ acts component-wise. With the conventions $\mathbf{x}_0 = \mathbf{x}$ and $\mathbf{x}_l = f_l(\mathbf{z}_l)$

TABLE VIII. Sensitivity of DC-PINNs on the volatility surface calibration. The reported values are RMSE on the logarithmic scale, given as ($\mathcal{L}0$, $\mathcal{L}b$, $\mathcal{L}f$, $\mathcal{L}h(x + xx + t)$) on dense evaluation grids. The bold entries indicate the overall top-four configurations, and the lower values correspond to better performance.

ν	$p_m \backslash p\lambda$	10	100	1000
10^{-4}	10	(−4.88,−5.30,−4.16,−3.82)	(−4.86,−5.34,−4.26,−3.97)	(−4.98,−5.34,−4.33,−4.02)
	100	(−4.97,−5.03,−4.66,−4.40)	(−5.02,−4.99,−4.69,−4.41)	(−5.00,−5.05,−4.73,−4.50)
	1000	(−4.89,−4.90,−4.79,−4.50)	(−4.96,−4.94,−4.80,−4.52)	(−4.99,−4.95,−4.87,−4.51)
10^{-3}	10	(−6.76,−6.24,−6.07,−5.65)	**(−6.82,−6.40,−5.69,−5.93)**	**(−6.87,−6.35,−5.68,−5.64)**
	100	(−6.13,−5.47,−6.13,−5.05)	**(−6.57,−6.18,−6.16,−5.86)**	(−6.34,−6.00,−6.17,−5.93)
	1000	(−6.51,−6.11,−6.11,−5.71)	**(−6.59,−6.20,−6.24,−5.67)**	(−6.55,−6.10,−6.25,−5.74)
10^{-2}	10	(−1.26,−0.48,−∞,−∞)	(−5.27,−5.06,−4.78,−3.93)	(−6.44,−6.20,−6.26,−5.49)
	100	(−6.54,−6.09,−5.05,−4.54)	(−5.78,−5.60,−5.35,−4.95)	(−4.26,−4.29,−4.59,−3.13)
	1000	(−3.82,−3.97,−4.27,−3.20)	(−5.53,−5.45,−5.17,−4.88)	(−3.87,−4.36,−4.09,−5.90)

TABLE IX. Sensitivity of DC-PINNs on the Navier–Stokes equation. Reported values are RMSE on the logarithmic scale, evaluated on dense grids and expressed as $(\mathcal{L}_0, \mathcal{L}_f, \mathcal{L}_h, \mu_1/\mu_2)$, where $\mathcal{L}_0 = \max\{\mathcal{L}_{0_u}, \mathcal{L}_{0_v}, \mathcal{L}_{0_p}\}$, $\mathcal{L}_f = \max\{\mathcal{L}_{f_u}, \mathcal{L}_{f_v}\}$, and $\mathcal{L}_h = \mathcal{L}_{h_{\nabla p}}$.

ν	$p_m \backslash p_\lambda$	10	100	1000
10^{-4}	10	(−1.00,−3.67,−1.31; 0.0483/−0.0020)	(−1.00,−3.42,−1.31; 0.0442/−0.0046)	(−1.00,−2.65,−1.31; 0.0461/−0.0084)
	100	(−1.01,−3.97,−1.31; 0.0432/0.0002)	(−1.01,−3.96,−1.31; 0.0405/0.0004)	(−1.01,−3.83,−1.31; 0.0430/−0.0008)
	1000	(−1.01,−4.28,−1.32; 0.0430/0.0012)	(−1.01,−4.27,−1.32; 0.0405/0.0013)	(−1.01,−4.12,−1.32; 0.0435/0.0010)
10^{-3}	10	(−0.94,−2.73,−1.12; 0.7695/0.0134)	(−0.94,−2.61,−1.09; 0.7109/0.0073)	(−0.94,−2.63,−1.09; 0.6992/0.0070)
	100	(−0.94,−2.79,−1.14; 0.7852/0.0160)	(−0.95,−2.78,−1.11; 0.7383/0.0095)	(−0.95,−2.98,−1.11; 0.7812/0.0117)
	1000	(−0.96,−2.99,−1.15; 0.8008/0.0165)	(−0.95,−2.90,−1.12; 0.7578/0.0118)	(−0.96,−2.92,−1.12; 0.7500/0.0099)
10^{-2}	10	(−0.93,−4.10,−1.07; 0.9805/0.0098)	(−0.93,−4.17,−1.07; 0.9805/0.0104)	(−0.92,−4.12,−1.06; 0.9844/0.0102)
	100	(−0.93,−3.99,−1.08; 0.9766/0.0115)	(−0.94,−4.00,−1.08; 0.9766/0.0099)	(−0.94,−4.12,−1.07; 0.9805/0.0104)
	1000	(−0.94,−4.14,−1.08; 0.9805/0.0112)	(−0.94,−4.16,−1.07; 0.9805/0.0106)	(−0.93,−4.06,−1.07; 0.9844/0.0103)

for $l = 1, \ldots, L-1$, the forward relations are

$$\mathbf{z}_l = W_l\,\mathbf{x}_{l-1} + \mathbf{b}_l, \quad \mathbf{x}_l = f_l(\mathbf{z}_l), \quad l = 1, \ldots, L-1,$$
$$\phi_\theta(\mathbf{x}) = A_L(\mathbf{x}_{L-1}) = W_L\,\mathbf{x}_{L-1} + \mathbf{b}_L. \tag{A3}$$

The trainable parameters are $\theta := \{(W_l, \mathbf{b}_l)\}_{l=1}^{L}$. Given a dataset $D_0 = \{(x_i^{(0)}, u_i^{(0)})\}_{i=1}^{N_0}$ and a prescribed empirical risk or objective informed by physics (as defined in the main text), the parameters are obtained by minimizing the loss chosen with respect to θ.

For derivative-constrained formulations, derivatives of ϕ_θ with respect to inputs are obtained efficiently by AD through the computational graph. Denote by $D_l(\mathbf{z}_l) := \mathrm{diag}\,(f_l'(\mathbf{z}_l)) \in \mathbb{R}^{d_l \times d_l}$ the diagonal matrix of the first derivatives of f_l, evaluated at $\mathbf{z}_l$. The Jacobian of ϕ_θ with respect to $\mathbf{x}$ admits the compact representation,

$$\begin{aligned} J_{\phi_\theta}(\mathbf{x}) &= \frac{\partial \phi_\theta}{\partial \mathbf{x}}(\mathbf{x}) \\ &= W_L\,D_{L-1}(\mathbf{z}_{L-1})\,W_{L-1}\,\cdots\,D_1(\mathbf{z}_1)\,W_1 \;\in\; \mathbb{R}^{d\times n}, \end{aligned} \tag{A4}$$

which reduces to a row vector in the scalar output case $d = 1$. Higher-order derivatives, for example, Hessians, can be computed via AD using compositions of Jacobian vector and vector-Jacobian products without explicitly forming dense higher-order tensors. This is standard in modern AD frameworks and suffices for the operators $\mathcal{D}$ and $\mathcal{D}_h$ used in the main text. To ensure that all derivatives appearing in the training objective are well defined, the activation functions should possess the appropriate smoothness: if derivatives of the order k of u_θ enter the loss, it is sufficient to choose $f_l \in C^k$ for $l = 1, \ldots, L-1$, for example, tanh satisfies C^∞ smoothness. A formal treatment of training with input derivatives in neural networks can be found in [16].

APPENDIX B: ALGORITHMS

This section introduces the DC-PINNs algorithm for multi-objective problems, which controls the inequality loss of the partial derivatives of a neural network function with respect to its input features and applies the combination of loss balancing techniques for both categorized and individual losses.

Algorithm 1 exhibits DC-PINNs characteristics that separate it from conventional learning methods. First, the calculation points $x_{\{f,h\}}$ for the MLP derivatives do not correspond to the points of the training dataset x_0. The algorithm adjusts the derivatives to fit wide mesh grids in the defined space, thus capturing derivative data across a wide array of input features. Secondly, the objective function $\mathcal{L}$ does not depend only on the MLP's direct output but also on its derivatives as specified in Eq. (8), all of which depend on identical network parameters. DC-PINNs facilitate the balance among the categorized losses in addition to the enhanced individual losses, which consist of PDE residuals and various scaled losses resulting from the violation of inequality constraints.

APPENDIX C: SENSITIVITY ANALYSIS OF ADAPTIVE PARAMETERS

We investigate the sensitivity of DC-PINNs to the adaptive parameters $\eta_m \in \{10^{-4}, 10^{-3}, 10^{-2}\}$ and $p_m, p_\lambda \in \{10, 100, 1000\}$ across heat, volatility calibration, and Navier–Stokes on dense grids with log–RMSE.

In the heat case, η_m is decisive: 10^{-4} underenforces constraints and destabilizes the inequality term, 10^{-3} stabilizes all losses, and 10^{-2} produces the lowest overall errors, especially for $\mathcal{L}_0$ and $\mathcal{L}b$. Moderate penalties ($p_m \in 10, 100$, $p\lambda \in 100, 1000$) give the most consistent gains; extremes occasionally induce stiffness and degrade derivative terms. Runtimes remain flat, so improvements reflect optimization rather than extra computation.

In volatility calibration, the addition of $\mathcal{L}_b$ clarifies the fit–constraint trade-off, with the best joint minima concentrated at $\nu = 10^{-3}$ and p_m, $p\lambda \in 100, 1000$. At $\nu = 10^{-4}$ performance is stable but shallower; at $\nu = 10^{-2}$ results become bimodal, with strong penalties sometimes overstiffening derivatives. For Navier–Stokes, $\mathcal{L}_0$ varies little, $\mathcal{L}_f$ improves with larger penalties (but not extreme), and $\mathcal{L}_h$ is stable; the learned viscosity aligns with scaling, recovering $\mu_2 \approx 10^{-2}$.

APPENDIX D: ONE-DIMENSIONAL HEAT EQUATION: IMPLEMENTATION AND EXPERIMENTAL DETAILS

Network, training, and sampling

PDE and domain. $u_t - \lambda\,u_{xx} = 0$ on $(x, t) \in [0, 1] \times [0, 1]$ with $u(x, 0) = \sin(\pi x)$ and $u(0, t) = u(1, t) = 0$; constraints of inequality $u_{xx} \leqslant 0$ and $u_t \leqslant 0$ in the domain. *Network.*

ALGORITHM 1. DC-PINNs with balancing processes.

Input: Dataset D_0, x_b, x_f, x_h, η, η_m, p_m, p_λ, k_{max}
Output: Neural network parameters θ
Consider a deep NN $\varphi_\theta(x)$ with parameters θ, and the loss

$$\mathcal{L} := \sum_\chi \lambda_\chi \hat{\mathcal{L}}_\chi(m_\chi, x_\chi),$$

where $\hat{\mathcal{L}}_\chi$ denotes the categorised loss with $\chi \in \{0, b, f, h\}$, $m_\chi = \mathbf{1}$ are soft-weighting vectors for individual losses, and $\lambda_\chi = 1$ are dynamic multipliers.
for $k \leftarrow 1$ **to** k_{max} **do**
  Compute $\nabla_\theta \hat{\mathcal{L}}_\chi(k)$ by automatic differentiation.
  if $k \equiv 0 \pmod{p_m}$ **then**
    Update m_χ by

$$m_\chi^{(j)}(k{+}1) = m_\chi^{(j)}(k) + \eta_m \nabla_{m_\chi^{(j)}} \hat{\mathcal{L}}_\chi(k),$$

    where $m_\chi(k)$ and $\hat{\mathcal{L}}_\chi(k)$ denote values at the k-th iteration.
  if $k \equiv 0 \pmod{p_\lambda}$ **then**
    Update λ_χ by

$$\lambda_\chi(k{+}1) = \begin{cases} 1, & \text{if } \alpha = 0, \\ \lambda_\chi(k) + \dfrac{\sum_\chi \alpha_\chi}{\alpha_\chi}, & \text{otherwise,} \end{cases}$$

    where $\alpha_\chi = \overline{|\nabla_\theta \hat{\mathcal{L}}_\chi(k)|}$.
  Update the parameters θ via gradient descent, e.g.

$$\theta(k+1) = \theta(k) - \eta \nabla_\theta \mathcal{L}(k).$$

return θ

MLP $(x, t) \mapsto u$ with two hidden layers of width 100 and an absolute-value output activation. *Optimizer and schedule.* The Adam optimizer is used with an exponentially decaying learning rate initialized at 10^{-3}, where the transition step is 2000 and the decay rate is 0.9. The training runs for 10 000 epochs in the main comparisons. *Sampling.* Interior collocation: 31×31 uniform grid; validation: 101×101; boundaries: dense grid in t (1001 points) at $x \in \{0, 1\}$ and the initial line at $t = 0$. *Evaluation metrics and grids.* We compute log–RMSE on the initial line (RMSE_0), the interior (RMSE_f), and the boundary (RMSE_b); log-violation rates for $u_{xx} \leqslant 0$ and $u_t \leqslant 0$ ("$-\infty$" indicates no violations); and log–RMSE of u_{xx} and u_t against the analytic solution on a 101×101 validation mesh. *Loss function.* For the heat equation, the total loss is defined as $\mathcal{L} = \lambda_0 \hat{\mathcal{L}}_0 + \lambda_f \hat{\mathcal{L}}_f + \lambda_b \hat{\mathcal{L}}_b + \lambda_{h_{xx}} \hat{\mathcal{L}}_{h_{xx}} + \lambda_{h_t} \hat{\mathcal{L}}_{h_t}$, where $\hat{\mathcal{L}}_0$ represents the data mismatch, $\hat{\mathcal{L}}_f$ the residual of the heat equation $f = \partial_t u - \lambda \partial_x^2 u$, and $\hat{\mathcal{L}}_b$ the boundary loss. The additional components $\hat{\mathcal{L}}_{h_x}$, $\hat{\mathcal{L}}_{h_{xx}}$, and $\hat{\mathcal{L}}_{h_t}$ measure violations of derivative-based smoothness and monotonicity conditions. The coefficients λ are adaptively balanced during training to ensure stable convergence.

APPENDIX E: LOCAL VOLATILITY MODEL: METHODOLOGY AND IMPLEMENTATION DETAILS

1. Methodology

a. Black–scholes pricing and equivalence to the heat equation

We work on a complete filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{Q})$ under the risk-neutral measure $\mathbb{Q}$. The (non-dividend-paying) asset price S_t follows the geometric Brownian motion

$$dS_t = r\, S_t\, dt + \sigma\, S_t\, dW_t, \quad \sigma > 0, \tag{E1}$$

where r is the risk-free rate and W_t is a standard $\mathbb{Q}$-Brownian motion. The time-t price of a European call with strike K and maturity T is

$$C(S_t, K, \tau) = e^{-r\tau}\, \mathbb{E}^{\mathbb{Q}}[(S_T - K)^+ \mid \mathcal{F}_t], \quad \tau := T - t. \tag{E2}$$

By the Feynman–Kac theorem, C solves the Black–Scholes (BS) PDE

$$\frac{\partial C}{\partial \tau} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC, \quad C(S, 0) = (S - K)^+,$$
$$\lim_{S\to 0} C = 0, \quad \lim_{S\to\infty} \frac{C}{S} = 1. \tag{E3}$$

When market quotes are expressed through *implied volatility* $\sigma_{\text{imp}}(K, \tau)$, the corresponding BS price is the closed form

$$C_{\text{BS}}(\sigma_{\text{imp}}) = S_t\, N(d_+) - Ke^{-r\tau} N(d_-),$$
$$d_\pm = \frac{\ln(S_t/K) + \left(r \pm \frac{1}{2}\sigma_{\text{imp}}^2\right)\tau}{\sigma_{\text{imp}}\sqrt{\tau}}, \tag{E4}$$

where $N(\cdot)$ denotes the standard normal CDF and $d_- = d_+ - \sigma_{\text{imp}}\sqrt{\tau}$.

Equivalence to the heat equation (main text). The BS pricing problem is *mathematically equivalent* to the one-dimensional heat equation used in the main text, cf. (17). In fact, set

$$x = \ln(S/K), \quad \theta = \frac{\sigma^2}{2}\tau, \quad \tilde{C}(x, \tau) = \frac{e^{r\tau}}{K} C(S, t), \tag{E5}$$

and define $u(x, \theta) = e^{\alpha x + \beta\theta}\tilde{C}(x, \tau)$ with

$$\alpha = \frac{1}{2} - \frac{r}{\sigma^2}, \quad \beta = -\frac{(2r - \sigma^2)^2}{8\sigma^2}. \tag{E6}$$

A direct computation yields the *constant-coefficient* heat equation,

$$\frac{\partial u}{\partial \theta} = \frac{\partial^2 u}{\partial x^2}, \tag{E7}$$

with the initial condition induced by the transformed payoff. Thus, the BS model and the thermodynamic heat problem are equivalent after an affine change of variables and time rescaling, resulting in the same linear parabolic PDE. In particular, the PINN/DC-PINN machinery, residual structure, and inequality handling developed for (17) apply verbatim to the BS setting.

b. Local volatility as an extension of black–scholes

The *implied volatility surface* (IVS) is the map $(K,\tau)\mapsto \sigma_{\text{imp}}(K,\tau)$ obtained by inverting the Black–Scholes formula (E4) on the observed prices of European options. The local volatility model of [32] extends BS by replacing the constant volatility with a *deterministic function* of state and time. In the spot-time representation, one writes

$$dS_t = r\,S_t\,dt + \sigma_{\text{LV}}(S_t,t)\,S_t\,dW_t, \tag{E8}$$

which reduces to BS when $\sigma_{LV}(S,t)\equiv\sigma$ is constant. In the strike-maturity coordinates (K,τ), the European call prices $C(K,\tau)$ satisfy the Dupire PDE

$$r\,K\,\frac{\partial C}{\partial K} - \frac{1}{2}\,\sigma_{\text{LV}}^2(K,\tau)\,K^2\,\frac{\partial^2 C}{\partial K^2} + \frac{\partial C}{\partial \tau} = 0, \tag{E9}$$

with boundary/initial conditions

$$C(K,0) = (S_t - K)^+, \quad \lim_{K\to\infty} C = 0, \quad \lim_{K\to 0} C = S_t. \tag{E10}$$

Hence, relative to BS, LV is (i) a state- and time-inhomogeneous diffusion, which is spatially varying diffusivity in the heat-equation analogy, and (ii) rich enough to fit the entire IVS while remaining Markov and arbitrage-free at the level of European options.

Practically, one often calibrates σ_{LV} from a sufficiently smooth $\sigma_{\text{imp}}(K,\tau)$ by substituting BS prices in (E4) into Dupire's equation (E9), which yields the conversion

$$\sigma_{\text{LV}}^2(K,\tau) = \frac{\sigma_{\text{imp}}^2 + 2\sigma_{\text{imp}}\tau\left(\frac{\partial\sigma_{\text{imp}}}{\partial\tau} + rK\frac{\partial\sigma_{\text{imp}}}{\partial K}\right)}{1 + 2d_+K\sqrt{\tau}\,\frac{\partial\sigma_{\text{imp}}}{\partial K} + K^2\tau\left(d_+d_-\left(\frac{\partial\sigma_{\text{imp}}}{\partial K}\right)^2 + \sigma_{\text{imp}}\frac{\partial^2\sigma_{\text{imp}}}{\partial K^2}\right)}, \tag{E11}$$

where $d_\pm$ are the BS quantities defined below (E4). Equation (E11) makes explicit that LV is an *extension* of BS: if σ_{imp} is constant in (K,τ), then $\sigma_{LV}\equiv\sigma_{\text{imp}}$ and (E9) collapses in the BS case.

c. No-arbitrage constraints for european options

The option prices should obey the restrictions imposed by no-arbitrage conditions, which are essential financial principles that posit that market prices prevent guaranteed returns above the risk-free rate. This study considers the necessary and sufficient conditions for no-arbitrage presented in [45]. This allows us to express the call option price as a two-dimensional surface appropriately. The necessary and sufficient conditions for no-arbitrage are represented as nonstrict inequalities for several first and second derivatives,

$$-e^{-r\tau} \leqslant \frac{\partial C}{\partial K} \leqslant 0, \quad \frac{\partial^2 C}{\partial K^2} \geqslant 0, \quad \frac{\partial C}{\partial \tau} \geqslant 0. \tag{E12}$$

From the above, the no-arbitrage conditions require these derivatives to have a specific sign. The standard architecture does not automatically satisfy these conditions when calibrating with a loss function simply based on the mean squared error (MSE) of the prices.

2. Network, training, and sampling

Problem and domain. We learn a price surface $C(K,T)$ under a synthetic local volatility model with spot $s_0 = 1$, rate $r = 0.05$, and maturity $T \in (0,1]$, enforcing monotonicity/convexity in K and temporal smoothness through h_x, h_{xx}, and h_t. *Interior and validation grids.* Interior collocation uses a 50×50 uniform (K,T) grid with $K\in[0.1,2.0]$ and $T\in[0.01,1.0]$; validation uses a 100×100 grid over the same ranges. *Boundary/initial observations.* The dataset provides price observations on (i) the T=0 line, and (ii) selected boundary slices, which are used as supervised anchors alongside the PDE residual. *Network.* An MLP with four hidden layers of width 50 and a Softplus output activation maps $(K,T)\mapsto C$. *Optimizer and schedule.* Adam optimizer is used with an exponentially decaying learning rate initialized at 10^{-3}, where the transition step is 2000 and the decay rate is 0.9. The training runs for 10 000 epochs in the main comparisons. *Sampling.* The spot follows the diffusion of local volatility under the risk-neutral measure $\mathrm{d}S_t = rS_t\,\mathrm{d}t + \sigma_{\text{LV}}(S_t,t)\,S_t\,\mathrm{d}W_t$,

$$\sigma_{\text{LV}}(S,t) = \sigma_A + \frac{A}{S} + Bt, \tag{E13}$$

where the parameters are fixed as $\sigma_A = 0.2$, $A = 0.1$, and $B = 0.2$, following the setup in [33,34]. The simulation domain is $S\in[0.1S_0, 2S_0]$ and $t\in[0,1]$. European call prices are computed by Monte Carlo using a log Euler discretization with uniform time steps N_{steps} $\Delta t = T/N_{\text{steps}}$. The update rule

$$S_{t+\Delta t} = S_t \exp\left(\left(r - q - \tfrac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t}\,Z\right),$$

$$Z\sim\mathcal{N}(0,1),$$

preserves positivity and evaluates $\sigma_{\text{LV}}(S_t,t)$ pathwise at each step. Antithetic sampling is applied by pairing Z with $-Z$ to reduce variance. The discounted Monte Carlo estimator for a call with strike K is

$$C^{\text{MC}}(S_0,K,T) = e^{-rT}\,\frac{1}{N_{\text{paths}}}\sum_{i=1}^{N_{\text{paths}}}\max\left(S_T^{(i)} - K, 0\right).$$

Prices are generated on a grid of maturities and logarithmic moneyness values, with $T\in[0,1]$ and $y = \log(K/S_0)\in[-2,2]$. We set $S_0 = 1$ and $r = 0.05$. The scheme is first-order accurate in Δt, with Monte Carlo standard error $\mathcal{O}(N_{\text{paths}}^{-1/2})$. The lower bound $S\geqslant 0.1S_0$ keeps the A/S term finite, while antithetic sampling improves stability for short maturities. The computed call prices $u_{i,j}$ were then polluted by a relative noise $u^{\text{noisy}} = u(1+0.05\eta)$ with η drawn from the standard normal distribution $\mathcal{N}(0,1)$. Observed prices are sampled as $x\sim\text{TruncNorm}([-1,1])$ and $t\sim\text{TruncNorm}([0.1,2.0])$, thus covering the relevant ranges of moneyness and maturity. We use $N_0 = 1000$ observed option prices, a 51×51 collocation grid for the interior residual, and $N_h = 5000$ inequality-enforcement points. *Loss function.* The

total loss is defined as $\mathcal{L} = \lambda_0\hat{\mathcal{L}}_0 + \lambda_f\hat{\mathcal{L}}_f + \lambda_b\hat{\mathcal{L}}_b + \lambda_{h_x}\hat{\mathcal{L}}_{h_x} + \lambda_{h_{xx}}\hat{\mathcal{L}}_{h_{xx}} + \lambda_{h_t}\hat{\mathcal{L}}_{h_t}$, where $\hat{\mathcal{L}}_0$ denotes the data-mismatch term, $\hat{\mathcal{L}}_f$ the PDE residual, and $\hat{\mathcal{L}}_b$ the boundary loss. The additional terms $\hat{\mathcal{L}}_{h_x}$, $\hat{\mathcal{L}}_{h_{xx}}$, and $\hat{\mathcal{L}}_{h_t}$ quantify violations of the derivative-based no-arbitrage conditions. The coefficients λ are adaptively balanced during training to stabilize the optimization dynamics.

APPENDIX F: INCOMPRESSIBLE FLOW DYNAMICS: METHODOLOGY AND IMPLEMENTATION DETAILS

1. Reference simulation and data extraction

High-fidelity reference data were obtained using the spectral/hp element method implemented in NEKTAR++ [37]. An unstructured triangular mesh refined around the cylinder resolves the near-wall shear and the downstream vortex street. The governing equations are the two-dimensional incompressible Navier–Stokes equations in a velocity–pressure formulation, advanced by the velocity-correction (i.e., projection) scheme with a Galerkin projection. Advection was discretized in convective form and diffusion was treated implicitly. Time progression was used with a second order IMEX scheme with a fixed step $\Delta t = 0.025$.

Unless otherwise stated, the nominal Reynolds number was $\mathrm{Re} = 100$ (kinematic viscosity $\nu = 1/\mathrm{Re}$), with spectral/hp dealiasing enabled to control aliasing errors introduced by nonlinear advection. To further attenuate spurious energy in the highest-resolved modes, spectral vanishing viscosity with a DG kernel and cutoff ratio 0.5 was applied. Global systems arising from the pressure Poisson and elliptic subproblems were solved using direct static condensation. Boundary conditions follow the standard flow cylinder configuration: no-slip velocity on the surface of the cylinder (wall), a uniform inflow $(u, v) = (1, 0)$ at the inlet, traction-free conditions at the outlet and weak Dirichlet far field conditions elsewhere. Polynomial expansions used for (u, v, p).

The calculation was integrated until a statistically steady periodic shedding regime was established at $t \approx 800$. After discarding the initial transient, flow field snapshots (u, v, p) were sampled in every 0.1 time unit to form the reference dataset. In addition to volumetric fields, we recorded the aerodynamic force coefficients on the cylinder wall, a probe time series at a fixed monitoring location in the wake, and per-mode kinetic energies. These diagnostics were used to verify convergence to the limit cycle (through the lift coefficient and probe spectra) and to compute the Strouhal number.

All volumetric outputs were written in VTU format and post-processed in ParaView [46]. A reproducible Python script accompanying the data colours the scene by the streamwise velocity, applies an arrow-glyph overlay for velocity vectors, and stores camera state and screenshots used for figures in this paper.

2. Network, training, and sampling

Problem. Two-dimensional incompressible flow past a circular cylinder; we use the precomputed wake dataset provided with the codebase. The computational domain and boundary conditions are as stated in the main text. *Network.* An eight-layer MLP of width 20 maps $(x, y, t) \mapsto (\phi, p)$; inputs are linearly rescaled to $[-1, 1]$ before the final layer. Parameters are initialized by Glorot initialization. *Optimizer and schedule.* Adam optimizer is used with an exponentially decaying learning rate initialized at 10^{-3}, where the transition step is 2000 and the decay rate is 0.9. The training runs for 10 000 epochs in the main comparisons. *Physics and constraints.* To preserve physical admissibility, we enforce incompressibility, $\nabla \cdot \mathbf{V} = 0$, and bound the pressure gradient by $|\nabla p| \leqslant \frac{1}{2}\rho U^2$, which limits excessive shear and ensures numerically stable flow fields, where $\rho = 1$ and $U = 1$ in the dimensionless framework based on the diameter of the cylinder and the flow speed. The viscosity coefficients (μ_1, μ_2) are learned together with the network; throughout the sweeps, μ_2 converged to $\approx$0.01. *Regularization.* Let U denote the reference velocity scales, as nondimensionalisation, with diameter D and inflow $U_\infty, \tilde{x} = x/D, \tilde{t} = t\,U_\infty/D, (u, v) = U_\infty(\tilde{u}, \tilde{v})$, and viscosity ν, so that $\mu_1 = 1$ and $\mu_2 = \nu/(U_\infty D) = 1/\mathrm{Re}$. All losses and constraints are computed in nondimensional variables. *Sampling.* From these simulations, we extract velocity samples 5,000 in the near-wake grids ($x \in [1, 8], y \in [-2, 2]$) and the location grids for PDE residuals and constraints, as follows [1]. *Loss function.* The total loss is $\mathcal{L} := \lambda_{0_u}\hat{\mathcal{L}}_{0_u} + \lambda_{0_v}\hat{\mathcal{L}}_{0_v} + \lambda_{f^u}\hat{\mathcal{L}}_{f^u} + \lambda_{f^v}\hat{\mathcal{L}}_{f^v} + \lambda_{h_{\nabla_\mathrm{p}}}\hat{\mathcal{L}}_{h_{\nabla_\mathrm{p}}}$, where $\hat{\mathcal{L}}_{0_u/0_v}$ are the data mismatch, $\hat{\mathcal{L}}_{f_u/f_v}$ are PDE residuals, $\hat{\mathcal{L}_h}^{\nabla_\mathrm{p}}$ enforces the upper bound following Bernoulli's principle. The coefficients λ are adaptively balanced during training.

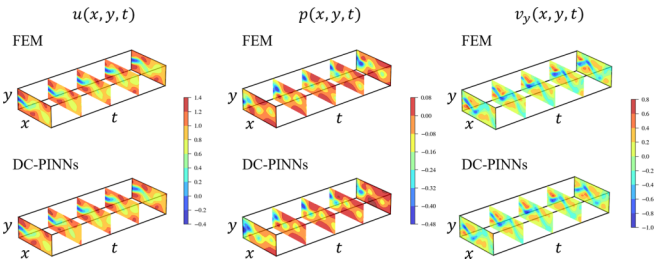


FIG. 9. Flow past a circular cylinder at $Re = 100$. Shown are the spatiotemporal fields $u(x, y, t)$, $p(x, y, t)$ (average adjusted), and $v_y(x, y, t)$. The top row provides the FEM reference and the bottom row shows DC-PINNs. The slabs indicate several time instants along the t-axis.

3. Comparative analysis

We compare DC-PINNs with standard PINNs and the Nektar++ reference FEM solver. The metrics include RMSE of (u, v, p), divergence error, violation rates of inequality constraints, and running time efficiency. The results show that DC-PINNs maintain incompressibility, reduce shear violations, and achieve pressure/velocity errors comparable to those of FEM while outperforming baseline PINNs in stability and physical consistency.

The results are compared with those obtained using a spectral/hp element method implemented in the open source software NEKTAR++ [37], as illustrated in Fig. 9. In training, velocity and pressure fields are defined $(\phi, p) := \psi_\theta(x, y, t)$, $(u, v) = (\partial\phi/\partial y, -\partial\phi/\partial x)$, following the architecture in [1]. In the experiment, we consider the divergence-

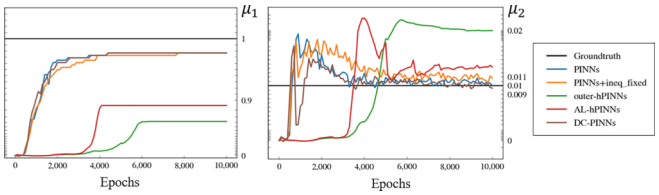


FIG. 10. Evolution of trained coefficients ($\mu_{1/2}$) by epochs.

free field and vortices shed in the wake having a characteristic size comparable to the diameter of the cylinder as derivative constraints, inspired by [47], in (23).In Fig. 9, DC-PINNs closely reproduces the FEM reference across all three fields. The large-scale vortex-shedding patterns align in phase and wavelength in u, the pressure field p exhibits comparable low-amplitude structures without spurious oscillations, and the cross-stream velocity v_y matches the alternating streaks in the wake. Relative to FEM, DC-PINNs shows mild smoothing of sharp extrema near the shear layers, yet the coherent structures and their temporal evolution remain consistent, indicating a faithful reconstruction of the wake dynamics on the evaluation grids.

To evaluate the performance of DC-PINNs, Fig. 10 presents the learning of system identification, or data-driven discovery of PDEs, showing how the parameters $\mu = (\mu_1, \mu_2)$ describe the observed data. DC-PINNs effectively converge to the reference values for the parameters μ during the training process. The results show that DC-PINNs can accurately capture complex flow features, including the von Kármán vortex street in the cylinder wake. The additional physical constraints enforced by DC-PINNs help maintain solution stability and prevent nonphysical artifact that may arise in unconstrained neural network approaches.

[1] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. **378**, 686 (2019).

[2] J. Sirignano and K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. **375**, 1339 (2018).

[3] J. Berg and K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing **317**, 28 (2018).

[4] W. E and B. Yu, The Deep Ritz method A deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. **6**, 1 (2018).

[5] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, *hp*-VPINNs: Variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Eng. **374**, 113547 (2021).

[6] L. Onsager, Reciprocal relations in irreversible processes. I., Phys. Rev. **37**, 405 (1931).

[7] Y. Patel, V. Mons, O. Marquet, and G. Rigas, Turbulence model augmented physics-informed neural networks for mean-flow reconstruction, Phys. Rev. Fluids **9**, 034605 (2024).

[8] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Physics-informed machine learning, Nat. Rev. Phys. **3**, 422 (2021).

[9] Z. Chen, Y. Liu, and H. Sun, Physics-informed learning of governing equations from scarce data, Nat. Commun. **12**, 6136 (2021).

[10] I. E. Lagaris, A. Likas, and D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Networks **9**, 987 (1998).

[11] A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, and M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural Inf. Process. Syst. **34**, 26548 (2021).

[12] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, Scientific machine learning through physics–informed neural networks: Where we are and what's next, J. Sci. Comput. **92**, 88 (2022).

[13] S. Wang, S. Sankaran, and P. Perdikaris, Respecting causality for training physics-informed neural networks, Comput. Methods Appl. Mech. Eng. **421**, 116813 (2024).

[14] S. Wang, X. Yu, and P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, J. Comput. Phys. **449**, 110768 (2022).

[15] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Comput. Methods Appl. Mech. Eng. **365**, 113028 (2020).

[16] K. Lo and D. Huang, On training derivative-constrained neural networks, arXiv:2310.01649.

[17] N. Sukumar and A. Srivastava, Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks, Comput. Methods Appl. Mech. Eng. **389**, 114333 (2022).

[18] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM J. Sci. Comput. **43**, B1105 (2021).

[19] Y. Chen, D. Huang, D. Zhang, J. Zeng, N. Wang, H. Zhang, and J. Yan, Theory-guided hard constraint projection (HCP): A knowledge-based data-driven scientific machine learning method, J. Comput. Phys. **445**, 110624 (2021).

[20] T. Beucler, M. Pritchard, S. Rasp, J. Ott, P. Baldi, and P. Gentine, Enforcing analytic constraints in neural networks emulating physical systems, Phys. Rev. Lett. **126**, 098302 (2021).

[21] L. D. McClenny and U. M. Braga-Neto, Self-adaptive physics-informed neural networks, J. Comput. Phys. **474**, 111722 (2023).

[22] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, An expert's guide to training physics-informed neural networks, arXiv:2308.08468.

[23] H. Son, S. W. Cho, and H. J. Hwang, Enhanced physics-informed neural networks with Augmented Lagrangian relaxation method (AL-PINNs), Neurocomputing **548**, 126424 (2023).

[24] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. (Springer, New York, 2006).

[25] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods* (Academic Press, New York, 1982).

[26] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, edited by Y. W. Teh and M. Titterington, Proceedings of Machine Learning Research Vol. 9 (PMLR, Chia Laguna Resort, Sardinia, Italy, 2010), pp. 249–256.

[27] D. P. Kingma and J. Ba, Adam A method for stochastic optimization, in *Proceedings of the 3rd International Conference on Learning Representations* (ICLR, San Diego, 2015)

[28] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, Y. Katariya, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX Composable transformations of Python+NumPy programs, version 0.3.13, http://github.com/jax-ml/jax (2018).

[29] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, Flax A neural network library and ecosystem for JAX, version 0.12.7, http://github.com/google/flax (2024).

[30] Google, https://research.google.com/colaboratory/faq.html (accessed on 25th Sept, 2025).

[31] J. R. Cannon, *The One-Dimensional Heat Equation, Encyclopedia of Mathematics and its Applications* (Cambridge University Press, Cambridge, 1984).

[32] B. Dupire, Pricing with a smile, Risk **7**, 18 (1994).

[33] S. Kim, H. Han, H. Jang, D. Jeong, C. Lee, W. Lee, and J. Kim, Reconstruction of the local volatility function using the Black–Scholes model, J. Comput. Sci. **51**, 101341 (2021).

[34] P. P. Boyle and D. Thangaraj, Volatility estimation from observed option prices, Decisions Economics Finance **23**, 31 (2000).

[35] R. Peyret and T. D. Taylor, Computational methods for fluid flow, *Springer Series in Computational Physics* (Springer-Verlag, New York, 1983).

[36] G. K. Batchelor, An Introduction to Fluid Dynamics, *Cambridge Mathematical Library* (Cambridge University Press, Cambridge, 2000)

[37] D. Moxey, C. D. Cantwell, Y. Bao, A. Cassinelli, G. Castiglioni, S. Chun, E. Juda, E. Kazemi, K. Lackhove, J. Marcon, *et al.*, *Nektar++*: Enhancing the capability and application of high-fidelity spectral/ *hp* element methods, Comput. Phys. Commun. **249**, 107110 (2020).

[38] M. D. McKay, R. J. Beckman, and W. J. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics **42**, 55 (2000).

[39] Z. Gao, L. Yan, and T. Zhou, Failure-informed adaptive sampling for PINNs, SIAM J. Sci. Comput. **45**, A1971 (2023).

[40] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Rev. **63**, 208 (2021).

[41] R. Frostig, M. J. Johnson, and C. Leary, Compiling machine learning programs via high-level tracing, in *Proceedings of SysML 2018* (Stanford, CA, USA, 2018).

[42] A. D. Jagtap and G. E. Karniadakis, Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. **28**, 2002 (2020).

[43] B. Moseley, A. Markham, and T. Nissen-Meyer, Finite basis physics-informed neural networks (FBPINNs): A scalable domain decomposition approach for solving differential equations, Adv. Comput. Math. **49**, 62 (2023).

[44] https://anonymous.4open.science/r/dcpinns-ef12704/.

[45] P. Carr and D. B. Madan, A note on sufficient conditions for no arbitrage, Finance Res. Lett. **2**, 125 (2005).

[46] U. Ayachit, *The ParaView Guide: A Parallel Visualization Application* (Kitware, Inc., 2015).

[47] S. Singh and S. Mittal, Vortex-induced oscillations at low Reynolds numbers: Hysteresis and vortex-shedding modes, J. Fluids Struct. **20**, 1085 (2005).