PHySIM: Neural-Augmented Physical Simulation of Networked Systems

Anonymous authors
Paper under double-blind review

Abstract

Simulating network-based systems — power grids, circuits, gas pipelines — requires solving coupled subsystem equations sharing physical boundary variables. Physics-based solvers exploit this structure but are slow to converge when subsystems have severe nonlinearities or behaviors driven by exogenous factors that first-principles models cannot capture. Black-box approaches offer low-cost approximations that capture hidden behavior but cannot enforce conservation laws at inference time, require large training datasets covering the network's full state space, and struggle to generalize across topology changes. To address these limitations, we introduce *PHySIM*, a flexible framework for modeling network-based systems through subsystem partitioning, where each subsystem can be represented by either a physics-based model or a DNN coupled at inference time through shared state variables at subsystem boundaries. *PHySIM* implicitly captures interactions across coupled subsystems while combining the predictive capabilities of DNNs to capture hidden behaviors seamlessly with the explainability and accuracy of physics-based models. The coupled system is solved inside the Newton–Raphson solver loop that extracts the DNN's sensitivities via backpropagation and assembles them into the same Jacobian matrix as physics-based terms, thereby solving the coupled system simultaneously rather than sequentially. The inference-time coupling inherits the benefits of both physics-based and DNN models: exact conservation-law satisfaction; superlinear state-space scaling from DNN reuse; data requirements that scale with device complexity rather than network size; zero-shot topology generalization without retraining; and accurate system-wide simulation under exogenous factors. We demonstrate the benefits of *PHySIM* over black-box and physics-based simulation across power systems, analog circuits, and gas pipeline networks.

1 Introduction

Network-based systems — power grids, integrated circuits, gas pipelines, biological networks — are composed of discrete, nonlinear components interconnected through a network topology. Their governing equations decompose into coupled subsystems that interact through shared variables:

$$h(z, u) = h_1(z, u) + h_2(z, u) = 0, \tag{1}$$

where z denotes system state, u external inputs, and h_1, h_2 are coupled through shared boundary states. This modularity underlies network-based systems including Kirchhoff's Current Law in circuits and power-flow equations in energy systems.

Newton solvers (Ly et al., 2024; Pillage et al., 1995) are the standard tool for simulating the physics-defined system behavior. However, two challenges limit physics-based simulation as subsystem complexity grows: **(1) Nonlinear, high-dimensional subsystem models**: complex components (power-electronic inverters, nanoscale CMOS transistors) require highly nonlinear models with many internal states, degrading Newton convergence (Ly et al., 2024). **(2) Hidden behavior**: many subsystems are driven by exogenous factors — weather, time-of-day, user patterns — that are not modeled by first-principles equations.

Deep neural networks offer a low-cost approximation to capture nonlinear, data-driven behavior, but a *global* DNN surrogate for the whole network inherits three structural problems. First, it has no mechanism to satisfy conservation laws at inference time as physical constraints are soft targets

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

that degrade under distribution shift. Second, data requirements grow with network size as training datasets must cover the joint state space of the entire network. Third, topology is encoded implicitly in the weights, so any structural change (a line outage, a new device) requires retraining. Prior hybrid (gray-box) methods combine physics and DNNs, but as a preprocessing step or an additive correction independent of network state, thereby only approximating conservation laws during inference.

Our approach. We introduce a framework *PHySIM* (Physics-based HYbrid Simulation using Implicit gray-box Models) that exploits the natural subsystem decomposition of network-based systems directly inside the numerical solver. Rather than modeling the full network with a DNN, we assign DNN *macromodels* only to subsystems where physics is insufficient. This introduces a subsystem level partitioning that models components of equation 1 using either physics-based models, h_{ph}, or pre-trained DNN models, h_{nn}. The two models are coupled directly at inference time, within the solver, according to:

$$h_{ph}(z, u) + h_{nn}(z, u) = 0 \tag{2}$$

where, unlike prior gray-box methods, both DNN subsystems and physics-based models share state variables, z. *PHySIM* captures interactions between subsystems implicitly within the model definition and introduces a partitioning scheme that maximizes the reusability of DNN macromodels to improve simulation runtime and data efficiency.

The coupled system is solved simultaneously by embedding both physics-based and DNN models within a Newton-Raphson loop, where at each iteration, Jacobian elements relating to DNN macromodels are extracted by backpropagation and assembled into the same Jacobian matrix as physics-based terms. As a result, conservation laws hold at every converged solution.

PHySIM provides the following properties that inherits the advantages of both physics-based and DNN simulations: (i) exact conservation-law satisfaction and ability to capture effects of exogenous factors on system dynamics; (ii) superlinear runtime/state-space scaling from DNN reuse across device instances; (iii) data requirements that scale with device complexity rather than network size and (iv) zero-shot generalization to new network topologies without retraining. We validate these properties across a range of network-based systems including the simulation of power grids, analog circuits, and gas pipeline networks. *PHySIM* also enables the simulation of a cyber-physical system — combining ML inverter-based resource controller with grid dynamics — that is cannot be accurately simulated by standard methods.

2 Related Work

Physics-based and black-box simulation. Physics-based solvers exploit network sparsity and modularity across circuits (Pillage et al., 1995), power systems (Pandey et al., 2023), and biological systems (Hines & Carnevale, 1997), but can become expensive for strongly nonlinear or poorly modeled subsystems. Classical approaches such as domain decomposition (Toselli & Widlund, 2005) and model order reduction (Benner et al., 2015) reduce this cost through interface approximations or low-dimensional projections, but are limited by their underlying physics and operating regimes. In contrast, black-box learned simulators, including MLPs (Zurada, 1992), GNNs (Zhou et al., 2020), MeshGraphNets (Pfaff et al., 2021), PINNs (Raissi et al., 2019), and neural operators (Li et al., 2020), can capture complex nonlinear behavior directly from data. However, they do not guarantee physical constraints at inference time (Ljung, 2001; Camporeale, 2019), require data that scales with the joint network state space (Reichstein et al., 2019), and generally require retraining when topology changes because network structure is encoded in the learned model (Loyola-Gonzalez, 2019). These limitations arise from modeling the network globally rather than learning localized device behavior.

Gray-box hybrid methods. Gray-box methods combine learned and physics-based models through parameter fitting (Law & Hutson, 1997; Sebastião, 2013; Cheng & Hu, 2007), serial coupling (Arahal et al., 2008; Zhang et al., 2012; Everett & Dubay, 2017; Bidarvatan et al., 2014), or parallel architectures (Menesklou et al., 2021; Yang et al., 2017; Xiong & Jutan, 2002). However, existing parallel formulations typically use an explicit combination such as $0 = h_{ph}(z, u) + h_{nn}(u)$, where the learned component depends only on external inputs. This breaks the feedback between the learned subsystem and physical state, preventing the learned model from participating in the coupled nonlinear solve.

Differentiable physics and neural-solver coupling. Differentiable physics frameworks propagate gradients between neural networks and simulators for system identification and control (Degrave et al., 2022; Heiden et al., 2021; Qiao et al., 2021; Um et al., 2020). Learned models have also been coupled to high-fidelity solvers through domain decomposition or used to initialize subsequent Newton iterations (Shukla et al., 2025). These approaches remain sequential, with learning providing an initialization or external prediction rather than a component of the nonlinear solve. Universal Differential Equations (Rackauckas et al., 2020) embed learned terms directly into ODE dynamics, but target time-integrated ODEs rather than algebraic and DAE network systems solved by Newton iteration.

Deep Equilibrium Models (Bai et al., 2019; 2020) similarly solve learned fixed-point equations, but their equilibrium represents a latent state rather than a physical state constrained by known conservation laws. Hamiltonian and Lagrangian neural networks (Greydanus et al., 2019) enforce conservation through network architecture, restricting the learned dynamics to physically structured classes. *PHySIM* instead embeds arbitrary DNN macromodels within a physics-based Newton solve, enforcing physical constraints through the solver rather than through the DNN architecture.

***PHySIM* Contribution.** *PHySIM* conditions each DNN on local boundary *state* variables and assembles its Jacobian directly into the physics-based Newton matrix at every iteration. The learned and physics-based subsystems therefore form a single coupled nonlinear system and are solved *simultaneously*. This solver-level integration distinguishes *PHySIM* from input-only gray-box models, sequential differentiable-physics coupling, and latent fixed-point methods, while enabling constraint satisfaction of physics-defined models.

3 METHOD

3.1 SUBSYSTEM PARTITIONING

A network-based system is represented by a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node, $v_i \in \mathcal{V}$, denotes a physical state variable and each edge, $(v_i, v_j) \in \mathcal{E}$, represents a physical interconnection between states. The system behavior is governed by the equations:

$$h(z, u) = 0 \tag{3}$$

where $z \in \mathbb{R}^n$ is the system state and u denotes external inputs. We partition $\mathcal{G}$ into non-overlapping subgraphs, $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}i)$, each characterized by internal states, x_i, and boundary states, y_i, such that $z \triangleq [x, y]$. The behavior of each subsystem is described by its internal equations

$$f_i(x_i, y_i, u) = 0, \tag{4}$$

while interactions between subsystem boundaries are captured by coupling equations

$$g_i(x_1, \ldots, x_m, y_i, u) = 0, \tag{5}$$

involving the m neighboring subsystems. Under the separability condition in equation 1, these boundary interactions can be decomposed as

$$\sum_{k=1}^{m} g_i^k(x_k, y_i, u) = 0. \tag{6}$$

PHySIM represents each subsystem using either physics-based equations or a trained DNN. For a DNN-modeled subsystem, the internal state, x_i, is eliminated by learning the input-output behavior at each subsystem boundary. Specifically, the DNN takes the boundary state, y_i, and external inputs, u, as inputs and predicts the corresponding boundary output seen by the rest of the network:

$$g_{nn}(y_i, u) \approx g_i(x_i, y_i, u). \tag{7}$$

This macromodel is integrated within the system-level model as:

$$\textstyle\sum_{j=1}^{m_p} g_{ph}(x_j, y_j, u) + \sum_{j=1}^{m_n} g_{nn}(y_j, u) = 0, \tag{8}$$

where m_p and m_n denote the numbers of physics-based and DNN-modeled subsystems, respectively. Both types of models are coupled through shared boundary states, y. This coupled model between physics-based and learned subsystems allows DNNs to replace computationally expensive or difficult-to-model internal dynamics. Appendix D.1 works through an example of this partitioning for a power grid with two renewable sources and a load (Figure 7).

PHySIM is most effective for networked systems with a natural modular structure, where some subsystems are well described by physics while others exhibit nonlinear or exogenous behavior that is difficult to capture from first principles. Shared boundary variables provide a natural interface, allowing DNNs to model local subsystem behavior while physics-based equations capture network interactions and the Newton solver enforces their consistency. However, *PHySIM* admits many possible partitions, not all of which provide the same benefits. The largest runtime and data-efficiency gains arise from *DNN reuse*. When multiple subsystems share the same underlying device physics, a single DNN trained on one instance can be reused across every occurrence of that device type. A good partition should therefore identify subsystems that are both suitable for DNN modeling and reusable across the network, motivating the partitioning scheme developed next.

Principled partition selection. The governing equations define a computation graph whose primitives form connected, non-overlapping subgraphs $\mathcal{G}_1, \ldots, \mathcal{G}_K$, with boundaries defined by physical state variables. Subgraphs that are *computation-isomorphic*, sharing the same structure and primitive types, represent instances of the same device type. This allows one DNN to be trained for each device type and reused across all matching instances, regardless of the number of instances. Appendix A quantifies this reusability across all three application domains (Table 2), with savings of up to $39\times$ in power and scaling directly with instance count in circuits and gas. The partitioning problem is therefore to identify which subgraphs benefit most from this reuse.

We score each candidate group $\mathcal{G}_k$ along two dimensions. Its *physics fidelity*, ϕ_k, measures how well its behavior can be captured without the exogenous driver, while its *cost ratio*, ρ_k, compares the physics-based evaluation cost with the amortized cost of a reusable DNN. We assign $\mathcal{G}_k$ to DNN modeling when $\phi_k < \phi^*$ or $\rho_k > 1$, and retain physics otherwise.

This rule is the solution to the underlying cost-minimization problem, as shown in Appendix B.4. Because the candidate groups have non-overlapping boundaries, the assignment is separable (Proposition 2), and the two conditions determine the optimal model for each group (Theorems 3–4). In general, determining the optimal partition is challenging. However, for networked systems, the problem admits a natural solution: linear transmission elements that define network connectivity remain physics-based while complex nonlinear edge devices are replaced by DNN macromodels. This structure follows from the optimality results (Corollary 2) and is validated against alternative partition schemes in Appendix E.3.

Jacobian-constrained training. With this partitioning, each DNN *macromodels the boundary behavior* of its subsystem, replacing its internal states with a learned mapping from local boundary variables to boundary outputs. Each macromodel is trained independently on local data, so training covers only the subsystem's operating region and data requirements scale with device complexity rather than network size. A single model and training set can also be reused across all instances of the same device type, while each macromodel remains independently retrainable and replaceable.

Following (Czarnecki et al., 2017), we augment the output loss with a Jacobian-matching penalty,

$$\mathcal{L} = \|g_{nn}(y) - g_i(x, y, u)\|^2 + \lambda \left\| \frac{\partial g_{nn}}{\partial y_p} - \frac{\partial g_i}{\partial y_p} \right\|^2, \tag{9}$$

where $y_p \subseteq y$ denotes the boundary variables with available ground-truth sensitivities and $\lambda > 0$. This directly controls the Jacobian error $\|E^k\|$ governing Newton stability (Theorem 1). When boundary variables are bounded within a known operating region $\mathcal{D}$, training data can cover $\mathcal{D}$ uniformly and Newton damping keeps iterates within this region, ensuring in-distribution operation (Appendix B.2). Appendix B.1 identifies the exact Jacobian this penalty targets, Appendix B.2 proves the resulting stability bound, and Appendix B.3 extends it beyond the training points.

Algorithm 1 DNN Jacobian Extraction for the Newton Step

Input: trained DNN $g_{nn}(\cdot)$, boundary state y

1: $g_{nn} \leftarrow g_{nn}(y)$
2: g_{nn}.backward(retain_graph=True)
3: $\partial g_{nn}/\partial y \leftarrow y$.grad
4: **return** $g_{nn}, \partial g_{nn}/\partial y$

3.2 Hybrid Newton–Raphson Solver

The partitioned system, $f_{ph}(x, y, u) = 0$ and $g_{ph}(x, y, u) + g_{nn}(y, u) = 0$, forms a coupled set of physics-based and DNN equations that must be solved simultaneously. *PHySIM* solves the complete system using Newton–Raphson, evaluating and coupling both model types within each iteration k as:

$$\begin{bmatrix} \frac{\partial}{\partial x} f_{ph} & \frac{\partial}{\partial y} f_{ph} \\ \frac{\partial}{\partial x} g_{ph} & \frac{\partial}{\partial y} g_{ph} + \frac{\partial}{\partial y} g_{nn} \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \end{bmatrix} = -\alpha \begin{bmatrix} f_{ph}(x^k, y^k) \\ g_{ph}(x^k, y^k) + g_{nn}(y^k) \end{bmatrix}, \tag{10}$$

evaluated at (x^k, y^k), with damping scalar α. The DNN contribution, $g_{nn}(y^k)$, and Jacobian, $\partial g_{nn}/\partial y$, are obtained from one forward pass and one backpropagation call through the pre-trained network (Algorithm 1), then assembled into equation 10 alongside the physics-based Jacobian. The stability of the Newton solver under this hybrid coupling is established in Theorem 1 (Appendix B) as the Sobolev loss equation 9 bounds the DNN Jacobian entries to ensure stable convergence.

3.3 Extension to Time-Domain (Transient) Simulation

Many network-based systems are governed by differential-algebraic equations (DAEs):

$$\dot{x}(t) = f_{ph}(x(t), y(t)), \quad \dot{y}(t) = g_{ph}(x(t), y(t)) + g_{nn}(y(t)) \tag{11}$$

PHySIM extends to this setting via trapezoidal integration, a semi-implicit numerical method commonly used in physics-based simulations. At each numerical integration step, the trapezoidal integration defines a set of nonlinear equations that define the state at a time-point t:

$$x(t{+}\Delta t) - x(t) - \tfrac{\Delta t}{2}\left[f_{ph}(x(t{+}\Delta t), y(t{+}\Delta t)) + f_{ph}(x(t), y(t))\right] = 0, \tag{12}$$

$$y(t{+}\Delta t) - y(t) - \tfrac{\Delta t}{2}\Big[g_{ph}(x(t{+}\Delta t), y(t{+}\Delta t)) + g_{nn}(y(t{+}\Delta t)) + g_{ph}(x(t), y(t)) + g_{nn}(y(t))\Big] = 0, \tag{13}$$

The resulting nonlinear equations from the coupled DNN and physics dynamical system is solved by Newton iterations. Each Newton step of the trapezoidal integration is then computed as:

$$\begin{bmatrix} I - \frac{\Delta t}{2}\frac{\partial f_{ph}}{\partial x} & -\frac{\Delta t}{2}\frac{\partial f_{ph}}{\partial y} \\ -\frac{\Delta t}{2}\frac{\partial g_{ph}}{\partial x} & I - \frac{\Delta t}{2}\left(\frac{\partial g_{ph}}{\partial y} + \frac{\partial g_{nn}}{\partial y}\right) \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \end{bmatrix} = -\alpha\, r^k(t + \Delta t), \tag{14}$$

where $r^k(t + \Delta t)$ is the trapezoidal residual of x and y at $t + \Delta t$ relative to their values at t (full expressions in Appendix C), and the DNN's forward pass $g_{nn}(y^k(t{+}\Delta t))$ and Jacobian are again extracted via Algorithm 1. Because the DNN macromodel replaces a subsystem's full internal DAE state with a single reusable input-output map, each additional device instance removes its internal states from the transient solve at zero additional training cost.

4 Experiments

We evaluate *PHySIM* across five properties, shown in Table 1, that capture its advantages over physics-based and black-box simulation methods. We assess each property across power systems, analog circuits, and gas pipeline networks to examine its consistency across application domains, as well as a cyber-physical system of a ML-defined controller for an inverter-based resource connected to the grid. Baselines include physics-based simulation, fully connected DNNs (FC-DNNs), PINNs (Raissi et al., 2019), GC-NNs (Zhou et al., 2020), and a Fourier Neural Operator (FNO) (Li et al., 2020) treating the network's node ordering as a 1D domain for spectral convolution. Application details, model architectures, and training procedures are provided in Appendix D.

	Physics-based	*PHySIM*	FC-DNN	PINN	GC-NN	FNO
Guaranteed constraint satisfaction	✓	✓	✗	✗	✗	✗
Accurate under exogenous changes	✗	✓	✗	✗	✗	✗
Zero-shot topology generalization	✓	✓	✗	✗	✗	✗
Data scales with device, not network	✓	✓	✗	✗	✗	✗
Improved simulation runtime	✗	✓	✓	✓	✓	✓

Table 1: *PHySIM* is the only approach satisfying all five properties. Physics-based methods cannot capture hidden exogenous behaviors; black-box methods cannot enforce constraints, remain accurate outside their training distribution, generalize across topologies, or decouple data requirements from network size.

4.1 Physical Consistency and Predictive Accuracy, Simultaneously

Reliable simulation requires both physical consistency and accurate prediction under changing and previously unseen conditions. We first evaluate these properties under changes that remain physically representable by the governing equations. Figure 1 compares *PHySIM* with physics-based and black-box simulators across power systems, analog circuits, and gas pipeline networks by varying system loading, circuit operating conditions, and pipeline parameters (loading factor, V_1 differential input, and source pressure p_0, respectively) beyond the DNN training range. *PHySIM* maintains low constraint residuals and prediction errors because its localized DNNs accurately capture individual device behavior, while physics-based equations define the overall network topology and interactions. This separation allows well-trained device macromodels to remain accurate as operating conditions change without requiring a global model to generalize over the full network state space. In contrast, global DNN simulators must learn both device behavior and network interactions, causing accuracy and physical consistency to degrade together outside their training distribution. Full sweep ranges, baseline architectures, and training procedures are given in Appendix D.

We then consider behavior driven by exogenous factors that physics alone cannot represent including weather-dependent electric loads, temperature-dependent circuit behavior (via BSIM-CMG device physics Lin (2018)), and weather-driven gas demand. *PHySIM*'s localized DNNs learn how these factors affect individual devices, while the physics-based network equations propagate their local effects throughout the system. Figure 17 shows that this combination captures the resulting system-wide behavior accurately across all three domains. Physics-based simulators cannot represent the exogenous dependence, while black-box simulators can learn it but cannot enforce physical consistency. By coupling localized learned behavior with physics-based network equations, *PHySIM* captures both the effects of exogenous factors and their system-wide consequences while maintaining physical constraint satisfaction across the full range tested in every domain.

The physical accuracy of the system-wide simulation from *PHySIM* is achieved from both the proposed partitioning scheme and Sobolev training of the DNN macromodels. The partitioning scheme preserves network-level physics, with partitioning ablations showing that all tested alternatives incur higher residuals, longer runtimes, or both (Appendix E.3). Additionally, Sobolev training enforces consistency within each macromodel to reduce passivity violations under extrapolation by roughly 3× (Appendix E.1–E.2).

4.2 Zero-Shot Topology Generalization

Simulation engines must remain accurate and efficient as network topology changes, such as through line outages, added devices, or circuit redesigns. Physics-based models represent network topology explicitly through their governing equations, allowing topology changes to be incorporated directly without retraining. In contrast, black-box models implicitly encode topology in their learned parameters, so changes in network structure generally require new training data and retraining, increasing both computational cost and data requirements. *PHySIM* retains the physics-based representation of network interactions while reusing localized DNNs for device-level behavior, enabling topology changes without retraining while preserving simulation accuracy.

PHySIM combines physics-based network equations with DNNs that learn only device-level boundary behavior $(y_i, u_{ex}) \rightarrow g_{nn}$. Topology changes are therefore handled by updating the physics-based

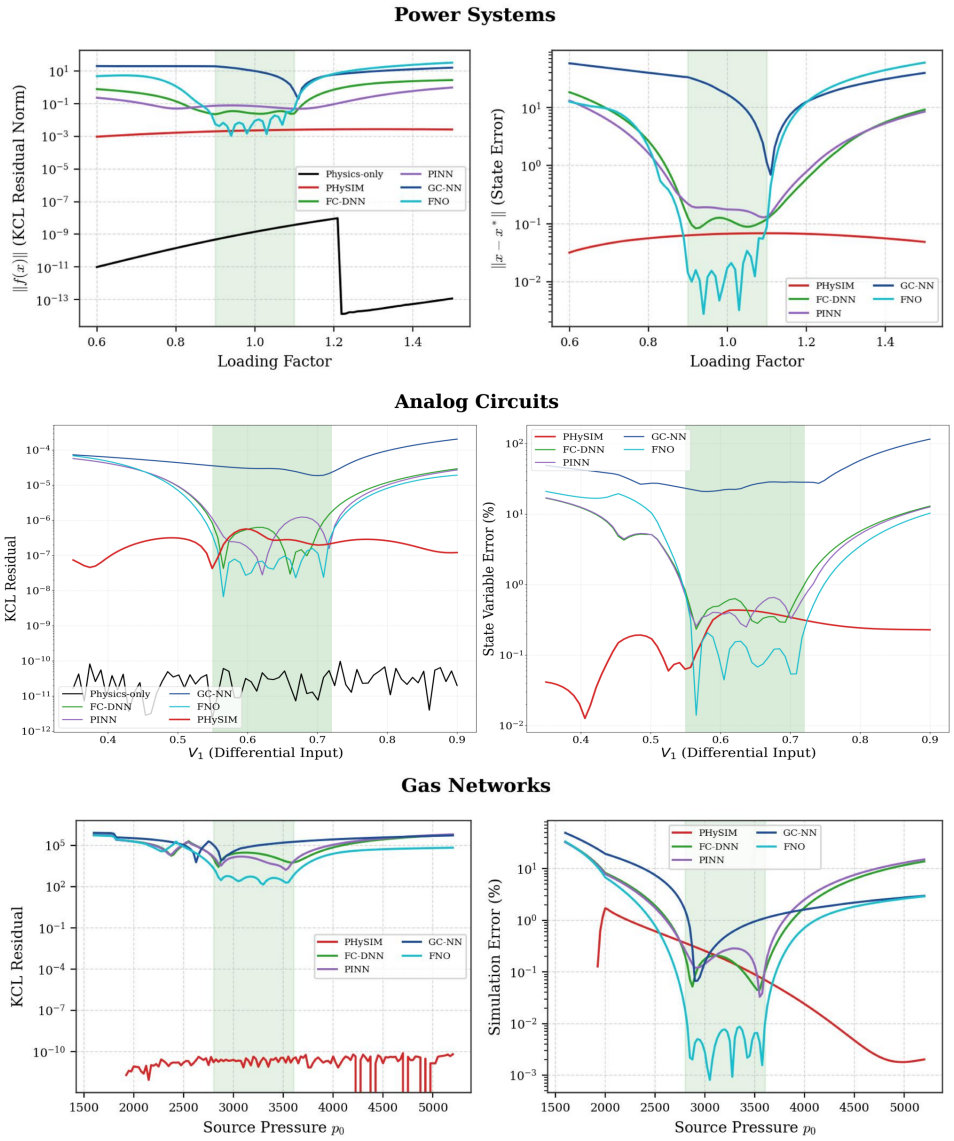


Figure 1: Constraint residual and state-prediction error together, across all three domains, vs. out-of-distribution operating conditions, against the full baseline set (FC-DNN, PINN, GC-NN, and FNO). *PHySIM* (red) holds both near machine precision through each domain.

equations while localized DNNs are reused without retraining. Figure 2 demonstrates this zero-shot generalization against black-bos methods across three domains: (a) N-1 contingencies on the IEEE 118-bus system (b) analog circuit topologies ranging between 4–32 transistors and (c) a source-pressure sweep in gas network. Across all domains, one DNN per device type can be reused under new network configurations, allowing *PHySIM* to maintain accuracy while black-box errors increase as the topology departs from the training configuration. On the other hand, the performance of black-box methods trained on a single network instance degrades as the network topology and conditions change outside the training sets.

4.3 Efficiency: Data and Runtime

Simulation efficiency has two key dimensions: runtime and data requirements.

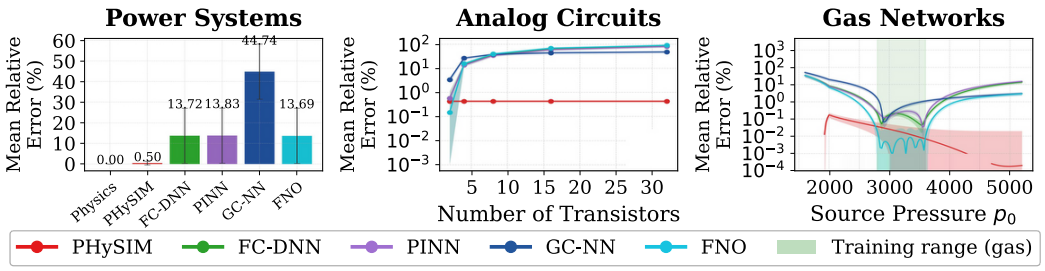


Figure 2: Zero-shot deployment under topology changes, full baseline set (FC-DNN, PINN, GC-NN, FNO). *PHySIM* reuses a single trained DNN across contingencies (a, mean ± std over 5 outages, error measured against the post-contingency ground truth), circuit sizes (b), and source-pressure conditions (c) without retraining.

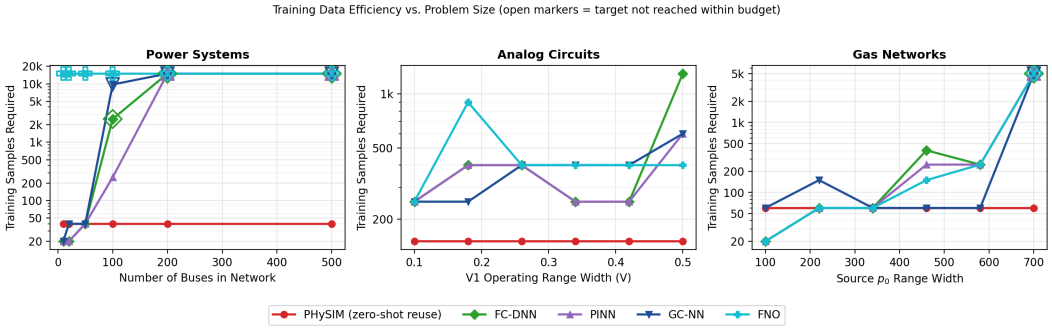


Figure 3: Training samples required to reach a fixed accuracy target (1% KCL residual error), as problem size scales. *Power:* number of buses; *Circuits:* voltage source operating range width; *Gas:* source range width. *PHySIM*'s device-level DNN (red) is trained once and reused zero-shot, so its sample requirement stays flat; black-box baselines must retrain on the full swept range and training data requirements scale poorly for larger networks.

Runtime efficiency. *PHySIM* reduces the dimension of the coupled solve by macromodeling subsystem boundaries with DNNs, eliminating their internal states from the global Jacobian.

We observe this reduction in all three domains. Figure 4 shows state-space reduction for induction-motor loads growing from 3.9% to 38.4% as the power network scales, while maintaining below 6% error; circuit and gas simulations likewise achieve faster per-iteration runtimes. Appendix C demonstrates this improvement in simulating transient dynamics, with 3.4% maximum error while eliminating 15.9% of the DAE states. As the DNN macromodel absorbs a larger subsystem rather than a single device, we observe greater runtime improvements. Appendix E.5 shows that replacing an aggregate subtree of n coupled devices with a macromodel collapses $O(n)$ internal states to achieve a speedup of $50.9\times$ at n=200.

Data efficiency. Physics-based methods require no training data, whereas black-box methods must cover the joint state space of the entire network, causing data requirements to grow with network size and operating range. *PHySIM* instead trains each DNN on local device behavior and reuses it across all instances of that device type, making training requirements depend primarily on device complexity. Figure 3 shows flat or near-flat requirements as power-system size and circuit and gas operating ranges increase, while FC-DNN, PINN, GC-NN, and FNO require progressively more data or lose accuracy under a fixed training budget.

4.4 Simulating Cyber-Physical Systems

Having established *PHySIM*'s advantages from coupling physics and DNN subsystems, we now apply it to a cyber-physical system where control decisions and physical states continuously interact. We simulate a grid-connected inverter-based resource (IBR) in which a physics-based electrical plant

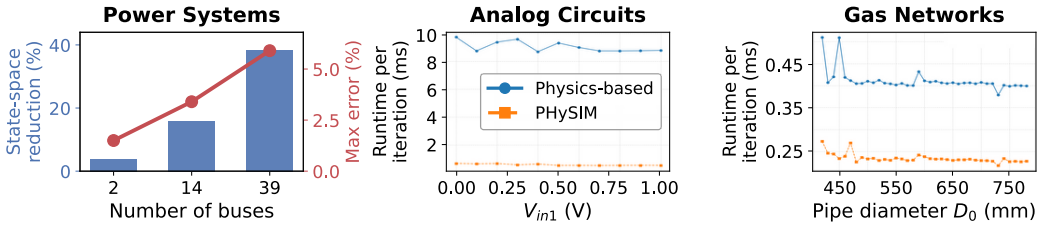


Figure 4: Simulation efficiency across domains: (a) EMT state-space reduction and error vs. network size (power systems); (b)–(c) runtime per Newton/solver iteration for circuits and gas networks, *PHySIM* (orange) vs. physics-based simulation (blue).

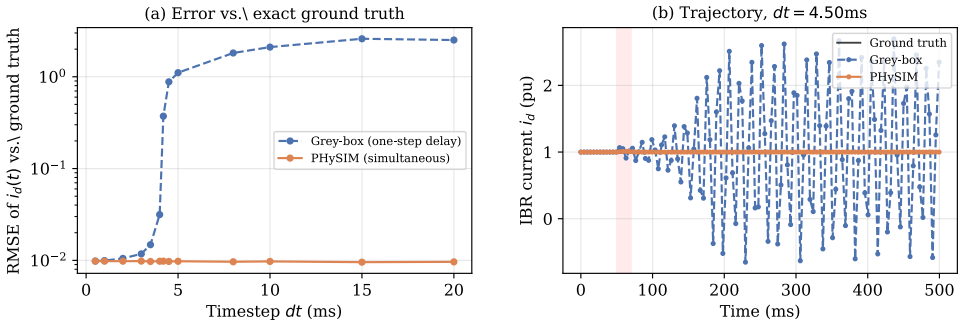


Figure 5: Inverter-based resource current control under a voltage-sag fault, evaluated against an exact ground-truth trajectory. (a) RMSE vs. ground truth across a timestep sweep: *PHySIM*'s error is bounded by DNN approximation error alone; grey-box's grows $250\times$ from delay. (b) Example trajectory of a fault: *PHySIM* tracks ground truth while grey-box's one-step delay creates numerical oscillation.

is coupled to a DNN current controller through shared input and output states. To bypass the coupling interactions, conventional grey-box methods evaluate the controller once per timestep and hold its output fixed during the physical solve, introducing a delay that can destabilize fast inverter control loops. *PHySIM* instead incorporates both subsystems into the same coupled solve, allowing their states to converge simultaneously at each timestep.

Figure 5 compares both approaches against an exact ground-truth trajectory during a voltage-sag fault while increasing the timestep. *PHySIM* maintains an RMSE of approximately 0.01, whereas grey-box coupling accumulates increasing error at larger timesteps. Simultaneous coupling avoids the numerical instability introduced by sequential DNN–physics interaction, enabling *PHySIM* to simulate tightly coupled cyber-physical systems that physics-only, black-box, and sequential grey-box approaches cannot represent at all. Full derivation, controller architecture, and a weather-dependent extension are given in Appendix E.6–E.7.

5 Conclusion

We introduced *PHySIM*, which couples DNN subsystem macromodels with physics-based models within a unified Newton–Raphson solve. Each DNN is trained on local boundary states, with its Jacobian assembled alongside physics-based terms so the coupled system is solved simultaneously. This coupling enables five properties: exact conservation-law satisfaction, runtime benefits from DNN reuse, data requirements determined by device complexity rather than network size, zero-shot topology generalization, and the ability to capture the effect of exogeneous features. Across power systems, analog circuits, and gas pipeline networks, *PHySIM* maintains constraint residuals near physics-based simulators even under distribution shift while capturing system responses due to exogeneous factors and efficient simulation runtimes and dataset sizes across new topologies.

REFERENCES

Manuel R Arahal, Cristina M Cirre, and Manuel Berenguel. Serial grey-box model of a stratified thermal tank for hierarchical control of a solar plant. *Solar Energy*, 82(5):441–451, 2008.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

M Bidarvatan, V Thakkar, M Shahbakhti, B Bahri, and A Abdul Aziz. Grey-box modeling of hcci engines. *Applied Thermal Engineering*, 70(1):397–409, 2014.

Enrico Camporeale. The challenge of machine learning in space weather: Nowcasting and forecasting. *Space weather*, 17(8):1166–1207, 2019.

Yuhua Cheng and Chenming Hu. *MOSFET modeling & BSIM3 user's guide*. Springer Science & Business Media, 2007.

Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in neural information processing systems*, 30, 2017.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Charles A Desoer and Ernest S Kuh. Basic circuit theory. *(No Title)*, 1969.

Florian Dörfler and Francesco Bullo. Kron reduction of graphs with applications to electrical networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(1):150–163, 2013.

Scott E Everett and Rickey Dubay. A sub-space artificial neural network for mold cooling in injection molding. *Expert Systems with Applications*, 79:358–371, 2017.

Gene H Golub and Charles F Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2013.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. Augmenting differentiable simulators with neural networks to close the sim2real gap. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2416–2422, 2021.

Michael L Hines and Nicholas T Carnevale. The neuron simulation environment. *Neural computation*, 9(6):1179–1209, 1997.

Gabriel Kron. Tensor analysis of networks. *New York: John Wiley & Sons*, 1939.

Mark M Law and Jeremy M Hutson. I-nolls: A program for interactive nonlinear least-squares fitting of the parameters of physical models. *Computer physics communications*, 102(1-3):252–268, 1997.

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

Huan Lin. pycmg: Bsim-cmg model (dc only) in python. https://github.com/huanlinberkeley/pycmg, 2018. Updated: March 28, 2018. Accessed: 2025-07-13.

Lennart Ljung. Black-box models from input-output measurements. In *IMTC 2001. Proceedings of the 18th IEEE instrumentation and measurement technology conference. Rediscovering measurement in the age of informatics (Cat. No. 01CH 37188)*, volume 1, pp. 138–146. IEEE, 2001.

Octavio Loyola-Gonzalez. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE access*, 7:154096–154113, 2019.

Sal Ly, Kim-Hung Pho, Shin-Hung Pan, and Wing-Keung Wong. Newton-raphson method: Overview and applications. *Advances in Decision Sciences*, 28(3):1–26, 2024.

Philipp Menesklou, Tabea Sinn, Hermann Nirschl, and Marco Gleiss. Grey box modelling of decanter centrifuges by coupling a numerical process model with a neural network. *Minerals*, 11(7):755, 2021.

Amritanshu Pandey et al. Circuit-based electromagnetic transient simulation. *Foundations and Trends® in Electric Energy Systems*, 6(2):83–118, 2023.

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

Lawrence Pillage, Ronald Rohrer, and Chandramouli Visweswariah. *Electronic Circuit & System Simulation Methods (SRE)*. McGraw-Hill, Inc., 1995.

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning (ICML)*, pp. 8661–8671, 2021.

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

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.

Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, and fnm Prabhat. Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195–204, 2019.

Pedro J Sebastião. The art of model fitting to experimental results. *European Journal of Physics*, 35 (1):015017, 2013.

Khemraj Shukla, Zongren Zou, Chi Hin Chan, Additi Pandey, Zhicheng Wang, and George Em Karniadakis. Neurosem: A hybrid framework for simulating multiphysics problems by coupling pinns and spectral elements. *Computer Methods in Applied Mechanics and Engineering*, 433: 117498, 2025.

Andrea Toselli and Olof Widlund. *Domain decomposition methods-algorithms and theory*, volume 34. Springer Science & Business Media, 2005.

Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6111–6122, 2020.

Qiang Xiong and Arthur Jutan. Grey-box modelling and control of chemical processes. *Chemical Engineering Science*, 57(6):1027–1039, 2002.

Zhuo Yang, Douglas Eddy, Sundar Krishnamurty, Ian Grosse, Peter Denno, Yan Lu, and Paul Witherell. Investigating grey-box modeling for predictive analytics in smart manufacturing. In *International design engineering technical conferences and computers and information in engineering conference*, volume 58134, pp. V02BT03A024. American Society of Mechanical Engineers, 2017.

Yi Zhang, Jianguo Yang, and Hui Jiang. Machine tool thermal error modeling and prediction by grey neural network. *The International Journal of Advanced Manufacturing Technology*, 59:1065–1072, 2012.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

Jacek Zurada. *Introduction to artificial neural systems*. West Publishing Co., 1992.