IFS/ACM Transactions on Machine Learning, vol. 3, no. 1, pp. 38-47, 2026

A Progressive Instructional Paradigm for Physics-informed Neural Networks

Wen Zhang[a,*], Dongshuai Liu[b], Imran Khan Keerio[c] and Haifeng Li[a,*]

[a]Department of Mathematics and Statistics, Fuyang Normal University, Fuyang, China
[b]Department of Physics and Electronic Engineering, Fuyang Normal University, Fuyang, China
[c]Department of Computer Science, Sindh Madressatul Islam University, Karachi, Pakistan

ARTICLE INFO

Keywords:
PINNs
mathematical modeling
progressive teaching design

ABSTRACT

Physics-informed neural networks as an interdisciplinary field combining deep learning and physical modeling, which needs high demands on student ability to transfer knowledge across disciplines. To address the cognitive gap students face when transitioning from traditional mathematical modeling to deep learning methods, this paper develops a three-stage progressive instructional paradigm. Taking the one-dimensional heat conduction equation as an example, teacher guides students through three successive cognitive stage, i.e., mechanistic modeling, data-driven modeling, and physics-informed modeling, corresponding respectively to the finite difference method, pure data-driven MLP fitting, and the PINNs approach. Rather than introducing PINNs directly as a new algorithm, the proposed design organizes their learning as a problem-driven cognitive progression. This can help students understand the paradigm shift from traditional numerical computation to data-driven learning and ultimately to physics-informed integration.

1. Introduction

Physics-informed neural networks (PINNs) integrate observational data and differential equations within a single training paradigm [1, 2, 3]. In PINNs, neural networks represent the solution, and loss functions combines a data term with a physical residual term [4]. PINNs are trained to fit measurements while remaining constrained by the governing physical law. In contrast to previous methods, PINNs do not require a mesh and can operate with sparse or indirect observations[5]. The properties have attracted considerable attention in various domains, such as, fluid mechanics [6, 7] and solid mechanics [8, 9]. The attractive properties, however, make PINNs a urgent demand to teach: students must coordinate partial differential equations, numerical reasoning, neural networks, and optimization; These topics are normally taught in separate courses. Therefore, there induces an instructional difficulty that arises from the integration of knowledge rather than from the mastery of any single technique.

The instructional obstacle is curricular fragmentation, and its consequences become visible when students encounter a method that requires previously separated bodies of knowledge to be used together [10, 11, 12]. For example, in a mathematical modeling course, students learn to derive an equation, discretize it, and examine a stability condition. In a data analytics course, they learn to define a loss function, train a network, and interpret a learning curve [13]. Each course is internally coherent, but the two are rarely connected [14]. Students may implement an explicit finite difference scheme yet be unable to explain why the ratio between the time step and the spatial step must satisfy a stability condition. When such students first encounter a PINN loss composed of a data term and a physical term, several questions arise: why the physical term is introduced, what the weight of the physical term controls, and how a failed training run can be attributed to the data, the physical term, the network, or the optimizer. These questions are fundamental because they distinguish procedural competence from conceptual understanding [15].

The instructional difficulty can be described in terms of three related gaps. ① **Representation Gap**: Finite difference and finite element methods transform an equation into a finite algebraic system on a mesh, whereas PINNs retain the solution as a function represented by a network and optimize the network parameters. The mathematical object, the implementation, and the modes of failure therefore differ. This difference in representation prepares the second gap. ② **Loss-design Gap**: A purely data-driven model is required only to match observations, whereas a PINN is required to match observations and to satisfy the equation at collocation points where no labels are available. Students must understand why unlabeled points can still carry information and why the data loss and the physical loss may act in different directions. Once the loss contains two terms, the third gap follows. ③ **Diagnosis gap**: In a numerical methods course, error behavior is explained through consistency, stability, and convergence, whereas in PINN training the outcome depends on initialization, sampling, network architecture, loss weighting, and non-convex optimization. A student who observes only a final error value cannot determine whether the model failed because the data were too sparse, because the physical term was too weak or too strong, or because the

DOI: https://doi.org/10.70891/TML.2026.040053
ISSN of IFS/ACM TML: 3078-5510
License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/
*Corresponding author
zw15290229577@163.com (W. Zhang); dongs_liu@yeah.net (D. Liu); IK@smiu.edu.pk (I.K. Keerio); lihaifengdlut@163.com (H. Li)

optimizer did not train long enough. These three gaps explain why a direct lecture on the PINN algorithm often leaves students able to reproduce the code but unable to explain the method [16, 17, 18].

Existing teaching materials on PINNs generally begin with the algorithm. They introduce the loss function, explain automatic differentiation, and then present a code example. This order is appropriate for readers who already possess a background in partial differential equations and neural networks, but it offers limited support to students who are moving from a traditional modeling course into physics-informed learning. What is often missing is a staged pathway in which the need for physical constraints emerges from a problem that students have already encountered. Teaching designs that employ a single equation case, a single analytical reference, and a single set of error measures to compare finite differences, a purely data-driven multilayer perceptron, and a PINN are also uncommon. Classroom activities organized around deliberate failure are rarer still: a stability failure in the numerical stage, a generalization failure in the data-driven stage, and a loss-balance problem in the PINN stage. Without such activities, students observe successful code but do not examine the reasons behind the design choices. The present paper is organized around this missing pathway.

The proposed design consists of three stages, all based on the same one-dimensional heat conduction equation. The first stage addresses the representation gap. Students derive the heat equation from Fourier's law and energy conservation, solve it with the finite difference method, and compare the result with the analytical solution. They also conduct a deliberate failure experiment: they first select a stable ratio between the time step and the spatial step, then increase the time step until the stability condition is violated, and observe oscillations and divergence. The stability condition thus becomes an observed phenomenon rather than a memorized formula. This stage provides a numerical baseline and clarifies how discretization and step sizes affect the solution. The second stage addresses the loss-design gap. The equation is removed from the model, and students receive only a small number of spatiotemporal samples for training a plain multilayer perceptron. The objective is not to obtain a low training error but to compare training error with test error, examine the spatial and temporal distribution of the error, and recognize that a network can fit the samples while behaving poorly in regions where no samples were provided. This stage demonstrates why data alone may be insufficient and prepares the question that the third stage answers. The third stage addresses the diagnosis gap. The same network architecture is retained, but a physical residual is added at a set of collocation points. The physical term is not introduced as an additional technique; it is introduced as a response to the question left by the second stage: if several functions fit the samples equally well, how can those that violate heat conduction be excluded? Students then compare data loss, physical loss, and total loss during training and examine how the weight of the physical term changes the result. Each stage thus creates the need for the next, and the three stages form a single line of reasoning rather than three separate lessons.

To make the design usable in classroom practice, the paper also specifies the target students, the learning objectives, and the experimental settings. The target students have completed an introductory programming course and are familiar with the basic concepts of partial differential equations and feedforward neural networks. The learning objectives are divided into three levels. At the knowledge level, students should be able to explain the basic principles of finite difference discretization, data fitting, and physical constraints. At the methodological level, they should be able to implement the three models and compare their results under a common evaluation reference. At the cognitive level, they should understand the role of physical information when data are sparse and recognize the complementary roles of data and mechanism knowledge. The experimental settings are kept as consistent as possible across stages, so that differences in results can be attributed to the information used by each model rather than to changes in the example or the evaluation metric.

This paper is guided by three research questions. ① How can a course be structured to support progressive learning from mechanism-based modeling to data-driven learning and subsequently to physics-informed integration, while helping students appreciate both the distinctions and interconnections among these three modeling paradigms? ② How can a benchmark case built upon a single governing equation, one analytical reference solution, and a unified suite of quantitative and visual indicators assist students in diagnosing issues including data scarcity, loss imbalance, and anomalous training dynamics? ③ What classroom activities and experimental configurations are necessary to ensure that this instructional design is practicable, reproducible, and enables fair comparative analysis? To address these questions, this paper makes three following contributions. First, it proposes a three-stage instructional framework that frames finite difference methods, multilayer perceptrons, and physics-informed neural networks as a coherent learning progression instead of three isolated techniques. Second, it establishes a benchmark case and a standardized evaluation workflow, encompassing stability-failure tests, generalization diagnosis under sparse data, loss evolution analysis, and sensitivity analysis for the weighting coefficient of the physics loss term. Third, it presents a sequence of classroom activities and comparative reasoning scaffolds to guide students from basic code execution toward deeper model interpretation and diagnostic thinking. Its central focus lies in instructional design and a readily implementable experimental workflow.

The remainder of the paper is organized as follows. Section 2 presents the teaching framework, the target students, the learning objectives, and the common heat equation case. Section 3 develops the first stage: derivation, finite difference discretization, the stability condition, and the deliberate failure experiment. Section 4 develops the second stage: sparse

sampling, multilayer perceptron training, and the diagnosis of overfitting and poor generalization. Section 5 develops the third stage: the physical residual, collocation points, loss weighting, automatic differentiation, and training diagnosis. Section 6 compares the three methods and discusses assessment and limitations. Section 7 reports the numerical experiments. Section 8 concludes the paper.

2. Progressive Teaching Framework

This section frames the core instructional challenge in terms of three interrelated gaps. Accordingly, an instructional framework is built on the premise that each stage targets one distinct gap, that each stage deliberately leaves an open problem to motivate progression to the subsequent stage, and that all three stages share a unified benchmark case and evaluation reference to ensure valid comparisons. The section defines the intended student cohort and learning objectives, outlines the learning progression across the three stages, and introduces the canonical heat conduction problem that acts as the unifying thread throughout the design.

2.1. Target Students and Learning Objectives

The framework is intended for senior undergraduate students that have completed an introductory Python programming course. Students are expected to understand the basic concepts of partial differential equations, to perform elementary numerical computation in Python, and to be familiar with feedforward neural networks. This choice follows directly from the instructional diagnosis in the introduction. If students were already fluent in both numerical methods and machine learning, the three gaps would not arise as obstacles; the framework is therefore aimed precisely at students who possess each body of knowledge in isolation but have not yet been asked to integrate them.

The learning objectives are organized at three cumulative levels, and each level is aligned with one of the gaps identified in the introduction. At the knowledge level, students should be able to explain the basic principles of finite difference discretization, data fitting, and physical constraints, and to articulate why these principles lead to different computational strategies. This level addresses the representation gap: students must first recognize that the finite difference method and the neural network approach construct and manipulate fundamentally different mathematical objects before they can appreciate why the latter requires a different form of supervision. At the methodological level, students should be able to implement a finite difference solver, a purely data-driven multilayer perceptron, and a PINN within a common computational environment, and to compare their outputs under identical evaluation conditions. This level addresses the loss-design gap: by constructing the data loss and the physical loss themselves, students observe how the two terms are defined, how they interact, and how the weight assigned to the physical term changes the behavior of the trained model. At the cognitive level, students should understand the role of physical information when observational data are sparse, and they should recognize that data and mechanism knowledge play complementary rather than substitutable roles. This level addresses the diagnosis gap: students should be able to look at a training outcome and reason about whether the observed behavior originates in insufficient data, in an inappropriate loss balance, in an inadequate network, or in an insufficient optimization budget.

2.2. Progression Across the Three Stages

The three stages follow a single organizing principle: the preceding stage generates a problem to which the subsequent stage responds. This principle distinguishes the present design from a conventional sequence in which finite differences, neural networks, and PINNs are introduced as three separate techniques. In a conventional sequence, each method is presented as a self-contained procedure, and students are left to infer the relationships among them. The result is often a collection of disconnected skills: a student can implement a finite difference scheme, can train a network, and can run a PINN code, but cannot explain why the third method is needed or how it differs from the first two. The present design makes the relationships explicit by structuring the course around the problems that each stage leaves unresolved, so that the transition from one method to the next is experienced as a response to a limitation rather than as an additional topic to be memorized.

Stage I establishes an interpretable mechanism-based model and a conventional numerical baseline. Students derive the governing equation from physical principles, discretize it, implement an explicit scheme, and examine the stability condition through a deliberate failure experiment. The deliberate failure is central to this stage. Rather than presenting the stability condition as a theoretical result to be accepted, the experiment allows students to observe oscillations and divergence when the condition is violated, so that stability becomes a phenomenon they have encountered rather than a formula they have memorized. The stage produces a numerical solution that serves as a reference for the subsequent stages, and it provides students with a concrete understanding of how discretization choices affect accuracy and stability. The problem left unresolved at the end of Stage I is that the numerical solution depends on a mesh and on step sizes that must be chosen with care. This limitation does not invalidate the numerical approach; it establishes the baseline against which alternative approaches will be judged, and it prepares students to appreciate why a different computational paradigm might be useful.

Stage II changes the source of information. The governing equation is withheld from the learning model, and only a limited number of spatiotemporal observations are supplied to a plain multilayer perceptron. Students train the network, compare training error with test error, and visualize the spatial and temporal distribution of the error. The stage produces a data-driven model and a diagnosis of its generalization behavior. The problem left unresolved at the end of Stage II is that a network trained on sparse samples can fit those samples while producing a function

Table 1
Three-stage progressive teaching framework.

Stage	Core question	Main method	Key learning activities	Stage outcome
Mechanism based modeling	Given physical laws, how can a computable temperature field be obtained?	Finite difference method	Equation derivation, discretization, and stability experiments	Numerical baseline and understanding of error
Data-driven learning	Can a network recover the temperature field using only limited samples?	Data-driven MLP	Sparse sampling, training, and error visualization	Data-driven model and generalization diagnosis
Physics-informed integration	How can a network use both data and physical laws?	PINNs	Automatic differentiation, residual construction, and loss balancing	Physics-informed model and understanding of integration

that violates the governing equation in unsampled regions. This problem is not merely acknowledged; it is demonstrated through the error visualization, so that students encounter the limitation directly rather than being told about it. The demonstration is essential because it creates the need for the next stage. If students were simply informed that data-driven models generalize poorly under data scarcity, they would have no reason to seek a remedy; having observed the failure themselves, they are in a position to understand why additional information might help.

Stage III retains the same network architecture and the same training samples, and introduces the governing equation as an additional constraint at a set of collocation points. The physical residual is presented not as a new algorithmic component but as a response to the question left by Stage II: among the functions that fit the available samples, how can those that violate the governing equation be excluded? Students compare data loss, physical loss, and total loss during training, and they examine how the weight assigned to the physical term changes the trade-off between data fitting and physical consistency. The stage produces a physics-informed model and an understanding of how data and physical constraints interact. The diagnosis gap is addressed here as well: because students have already seen the numerical baseline in Stage I and the data-driven failure in Stage II, they are in a position to interpret the training curves and error maps of Stage III in relation to both. They can ask whether a low physical loss accompanied by a high data error indicates an imbalance in the loss weights, and they can compare the resulting model against the numerical baseline and the data-driven prediction using the same criteria. Table 1 summarizes the three-stage framework, including the core question, the main method, the key learning activities, and the expected outcome of each stage.

2.3. A Common Case and a Unified Evaluation Reference

To reduce the additional cognitive cost caused by changing examples, all three stages employ the one-dimensional heat equation in Equation 1. Consider the following initial-boundary value problem [19]:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \qquad x \in [0, L], \quad t \in [0, T], \tag{1}$$

where $u(x, t)$ denotes the temperature at position x and time t, α is the thermal diffusivity. Homogeneous Dirichlet boundary conditions are imposed,

$$u(0, t) = 0, \qquad u(L, t) = 0, \tag{2}$$

and the initial condition is

$$u(x, 0) = \sin\left(\frac{\pi x}{L}\right). \tag{3}$$

The problem has the analytical solution

$$u(x, t) = \exp\left[-\alpha \left(\frac{\pi}{L}\right)^2 t\right] \sin\left(\frac{\pi x}{L}\right). \tag{4}$$

The problem is selected for two reasons, and both reasons bear directly on the instructional logic developed above. First, its physical meaning is clear and its mathematical form is simple, so that students can focus on methodological differences rather than on the complexity of the model itself. This is consistent with the aim of isolating the three gaps: if the case were physically or mathematically intricate, students might attribute difficulties to the complexity of the problem rather than to the differences among the methods. Second, the analytical solution provides a unified reference against which the finite difference solution, the data-driven prediction, and the physics-informed prediction can be compared using identical error measures. This is essential for the comparative activities that run through all three stages, because it ensures that differences in performance reflect differences in the information used by each method rather than differences in the example, the domain, or the metric.

The finite difference solution is treated as a traditional numerical baseline rather than as the standard answer, and the neural models are evaluated over the same computational domain with the same criteria. This treatment is deliberate. If the finite difference solution were presented as the correct answer, students might interpret the neural models

as approximations to a known target rather than as alternative ways of incorporating information. By treating it as a baseline, the design keeps the three methods on comparable footing: each method uses a different source of information, and each is evaluated against the same analytical reference. The shared case and the shared reference thus serve the same purpose as the shared organizing principle across stages: they allow students to compare the methods within a single frame of reference, and they make it possible to attribute differences in outcome to differences in information rather than to incidental changes in the setup. This arrangement supports the central instructional claim of the paper, namely that the transition from mechanism-based modeling to data-driven learning and then to physics-informed integration can be presented as a coherent progression rather than as a sequence of unrelated techniques.

3. Stage I: Mechanism Based Modeling

The framework assigns Stage I a specific function: to establish an interpretable mechanism-based model and a conventional numerical baseline, and in doing so to address the representation gap.

3.1. From Physical Laws to a Mathematical Equation

The objective of the first stage is to help students establish a complete chain from physical laws to mathematical equation and then to numerical computation. Instruction begins with Fourier's law of heat conduction, i.e., $q = -k\frac{\partial u}{\partial x}$. And it is combined with energy conservation. Under the assumptions of a homogeneous material and no internal heat source, the one-dimensional heat equation is obtained. This process emphasizes that a PDEs is not merely an abstract formula but a mathematical representation of a physical mechanism. The initial and boundary conditions then specify the particular problem to be solved.

3.2. Finite Difference Discretization and Recurrence

Divide the spatial interval $[0, L]$ into N_x subintervals with spatial step $\Delta x = L/N_x$, and divide $[0, T]$ into N_t time steps with $\Delta t = T/N_t$. Let $u_i^n \approx u(i\Delta x, n\Delta t)$. Using a forward difference for the time derivative and a centered difference for the second spatial derivative gives

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2}. \tag{5}$$

Rearranging yields the explicit recurrence relation

$$u_i^{n+1} = u_i^n + r\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right), \qquad r = \frac{\alpha \Delta t}{(\Delta x)^2}. \tag{6}$$

This recurrence relation directly shows how the temperature at the next time level is determined by neighboring grid values at the current time level. Students can implement the time-marching procedure in Python while continuously imposing the prescribed Dirichlet boundary conditions.

3.3. A Failure Experiment on Stability

Instead of presenting the stability condition only as a theoretical conclusion, this stage includes a deliberate-failure experiment. For the explicit scheme in Eq. (6), stability requires $r \leq 1/2$.

Students first choose Δx and Δt that satisfy this condition and obtain a stable numerical solution. They then deliberately increase the time step so that $r > 1/2$ and observe numerical oscillations and divergence. The observed behavior is subsequently explained using von Neumann stability analysis. Through the sequence of observing failure first and explaining the cause afterward,stability changes from a formula to be memorized into an observable and diagnosable numerical phenomenon.

3.4. Stage Assessment and Outputs

Students compare the finite difference result with the analytical solution and evaluate discretization error using the mean squared error or maximum absolute error. The outputs of this stage include a finite difference program, the complete spatiotemporal temperature field, and error visualizations. More importantly, students should be able to explain how the grid, step sizes, stability condition, and boundary conditions jointly influence the numerical solution. This provides a reference for the later comparison between grid-based numerical computation and learning methods based on function approximation.

4. Stage II: Pure Data-Driven MLP Learning

4.1. From a Complete Numerical Field to Sparse Observations

The second stage deliberately changes the information available to the model. The PDE is no longer provided to the learning model; only a limited number of spatiotemporal samples are supplied. Training samples may be drawn from the numerical field generated in Stage I or from the analytical solution. Denote the training set by $\mathcal{D}_{\text{train}} = \left\{(x_i, t_i, u_i)\right\}_{i=1}^{N_{\text{train}}}$. Reference values over the complete computational domain are used only for testing and are not included in model training. This setup helps students distinguish between training information and evaluation information and understand how data coverage affects learning.

Sampling strategies can be treated as an extension activity. The basic experiment uses random or uniform sampling so that variables are controlled in the comparison between the MLP and PINNs. More advanced experiments may then consider boundary-prioritized sampling or gradient-aware sampling. This arrangement avoids introducing too many variables into the main teaching sequence while retaining room for further investigation.

4.2. MLP Function Approximation and Data Loss

A feedforward neural network takes (x, t) as input and predicts the temperature $\hat{u}$: $\hat{u} = f_\theta(x, t)$, where θ denotes the trainable parameters. The hidden layers use the tanh activation function. Its smooth differentiability is suitable

Table 2
Unified comparison of the three methods

Method	Main information source	Learning mechanism	Typical focus	Teaching role
FDM	PDEs, initial and boundary conditions	Grid discretization	Step size, stability	Mechanism-based numerical baseline
MLP	Limited observations	Neural network function fitting	Overfitting, generalization	Reveal limitations of relying only on data
PINNs	PDEs and data	Optimization of data and physical loss	Loss balancing	Integration of data and physical constraint

for function approximation in this stage and, importantly, allows the same network architecture to be retained in Stage III when second derivatives are computed by automatic differentiation.

The data loss is defined by the mean squared error:

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \left| f_\theta(x_i, t_i) - u_i \right|^2 . \tag{7}$$

A gradient-based optimization algorithm is used to update θ. The teaching emphasis is not on constructing increasingly complicated network architectures, but on making students recognize that the only supervision available to the network at this point is the mapping $(x, t) \mapsto u$ at a limited number of samples. The network itself has no knowledge that these temperature values should satisfy the heat equation.

4.3. From Successful Fitting to Generalization Diagnosis

The key objective of this stage is not simply to obtain a low training loss, but to compare the training error with the error over the complete test domain and to inspect the spatial and temporal distribution of the error. When the training error continues to decrease while the test error increases, the behavior is characteristic of overfitting [20]. When test errors are concentrated primarily in regions insufficiently covered by the training samples, it is more accurate to describe the problem as insufficient generalization in unsampled regions rather than simply as "extrapolation failure."

Students can visualize the reference temperature field, the MLP prediction, and the absolute-error field, and then vary the number of training samples to obtain learning curves. For the present example, the temperature changes rapidly in time near the initial instant, whereas regions with large spatial gradients are mainly located near the two boundaries. At $x = L/2$, the initial temperature reaches its maximum and the first spatial derivative is zero. Relating the error distribution to the characteristics of the physical field helps students recognize that both the amount and spatial distribution of data directly affect the reliability of a purely data-driven model.

4.4. Cognitive Transition at the End of Stage II

After the experiments, the instructor can pose a central question: If several functions fit a limited set of training samples equally well, how can functions that violate the heat-conduction law be excluded? This question reconnects the data insufficiency observed in Stage II with the physical equation already learned in Stage Stage I. Stage III therefore does not merely add another algorithm; instead, it attempts to transform known physical laws into training information that can be used by the neural network.

Table 3
FDM convergence at $t = 1$ with $r = 0.4$.

N_x	Δx	Δt	Relative L^2	Max. abs. error
10	0.1000	0.040000	1.1521×10^{-2}	4.2941×10^{-3}
20	0.0500	0.010000	2.8508×10^{-3}	1.0625×10^{-3}
40	0.0250	0.002500	7.1088×10^{-4}	2.6495×10^{-4}
80	0.0125	0.000625	1.7761×10^{-4}	6.6195×10^{-5}

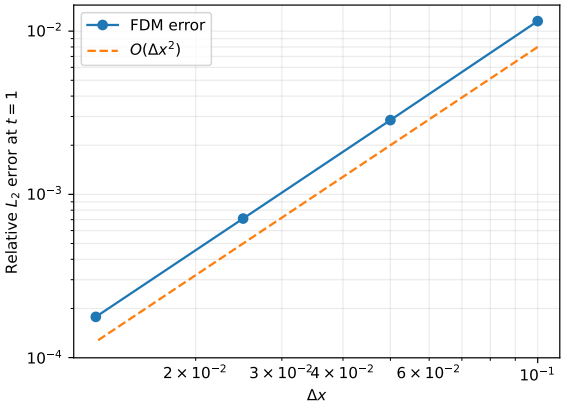


Figure 1: FDM convergence under the stable setting $r = 0.4$.

5. Stage III: The PINNs Method

5.1. From Data Fitting to Physical Constraints

Stage III retains the inputs, outputs, and basic network architecture used in Stage II and changes only the training objective. Let the neural network approximation be $\hat{u}(x, t) = f_\theta(x, t)$. Substituting it into the heat equation defines the

physical residual

$$\mathcal{R}(x,t;\theta) = \frac{\partial \hat{u}}{\partial t} - \alpha \frac{\partial^2 \hat{u}}{\partial x^2}. \tag{8}$$

If the network output satisfies the heat equation, $\mathcal{R}$ should be close to zero. A set of collocation points $\left\{(x_j^c, t_j^c)\right\}_{j=1}^{N_{\text{coll}}}$ is therefore selected in the computational domain, and the physical loss is defined as

$$\mathcal{L}_{\text{PDE}}(\theta) = \frac{1}{N_{\text{coll}}} \sum_{j=1}^{N_{\text{coll}}} \left| \mathcal{R}(x_j^c, t_j^c; \theta) \right|^2. \tag{9}$$

The collocation points do not require temperature labels. Their purpose is to constrain the functional form of the network throughout the computational domain using known physical laws. From the perspective of function spaces, the data loss requires the model to pass through the observations, while the physical loss further narrows the set of candidate functions consistent with those observations. This interpretation helps students understand the practical role of physical information more clearly than simply describing a PINNs as a neural network with an additional regularization term.

5.2. Joint Loss and Weight Balancing

The total PINNs loss is

$$\mathcal{L}_{\text{total}}(\theta) = \mathcal{L}_{\text{data}}(\theta) + \lambda \mathcal{L}_{\text{PDE}}(\theta), \tag{10}$$

where $\lambda > 0$ controls the relative strength of the data and physical constraints. When $\lambda = 0$, the model reduces to the purely data-driven MLP in Stage II. When λ is excessively large, optimization may overemphasize the physical residual and weaken the fit to the observational data. Thus, λ should not be interpreted as implying that more physics is always better, but rather as a balancing parameter in a multi-objective optimization problem. In teaching experiments, several values $\lambda = 0, 0.1, 1$, and 10 can be compared in terms of data error, PDEs residual, and test error. This can help students understand the effect of loss weighting empirically.

5.3. Automatic Differentiation

Automatic differentiation is a key techniche of PINNs. A computational graph records the dependence of the network output on x and t, enabling the calculation of $\partial\hat{u}/\partial t$, $\partial\hat{u}/\partial x$, and $\partial^2\hat{u}/\partial x^2$, which are then used to construct $\mathcal{R}(x,t;\theta)$. Instruction should explicitly distinguish automatic differentiation from finite differences. FDM approximates derivative using discrete points and introduce truncation errors. In contrast, automatic differentiation applies the chain rule to compute derivative and avoids truncation errors. This comparison also creates a methodological connection between Stage I and Stage III.

5.4. Design of Comparative Experiments

To highlight the role of physical constraints, the MLP and PINNs should use the same network architecture, training samples, and optimization settings. The only difference is whether $\mathcal{L}_{\text{PDE}}$ is included. At least two training conditions are recommended: a sparse-data condition and a relatively data-rich condition. The analytical reference, MLP prediction, PINNs prediction, absolute error, and PDE residual can then be compared. Any reported error reduction should be supported by actual experimental results. Repeated independent runs and consistent evaluation metrics should be used.

Table 4
Deterministic MLP/PINNs comparison on the common test grid.

N_{data}	Method	Rel. L^2	RMSE	MAE	PDEs RMS
20	MLP	0.04066	0.01892	0.01449	0.23226
20	PINNs	0.02559	0.01191	0.00923	0.13213
50	MLP	0.03555	0.01655	0.01243	0.25463
50	PINNs	0.02609	0.01214	0.00902	0.12519
100	MLP	0.01815	0.00845	0.00619	0.18022
100	PINNs	0.01451	0.00675	0.00507	0.07713
200	MLP	0.02057	0.00957	0.00744	0.17081
200	PINNs	0.01493	0.00695	0.00484	0.07769

5.5. Loss Evolution and Training Diagnosis

During training, $\mathcal{L}_{\text{data}}$, $\mathcal{L}_{\text{PDE}}$, and $\mathcal{L}_{\text{total}}$ are recorded simultaneously. This allows the training process to be analyzed rather than focusing only on the final output. If the data loss decreases substantially while the physical loss remains nearly unchanged, the current training process is biased toward data fitting. Conversely, if the physical loss is low while the data error remains large, the physical and observational constraints may be imbalanced.

In addition to comparing loss magnitudes, students may further examine

$$g_{\text{data}} = \left\| \nabla_\theta \mathcal{L}_{\text{data}} \right\|_2, \qquad g_{\text{PDE}} = \left\| \nabla_\theta \mathcal{L}_{\text{PDE}} \right\|_2, \tag{11}$$

to discuss the actual contributions of different loss terms to parameter updates. Gradient normalization or adaptive weighting may be introduced as extension topics rather than required components of the basic lesson.

6. Unified Comparison and Teaching Discussion

6.1. A Unified Logic for Comparing the Three Methods

After completing all three stages, the methods should be compared within a common evaluation framework. The analytical solution provides a unified reference. The finite difference method represents the conventional mechanism-based approach, in which the PDEs is solved through numerical discretization. The MLP represents a purely data-driven approach that learns the solution from sample data. The PINNs integrates observational data with physical constraints through an optimization framework. The comparison is not to identify the best method. Instead, it examines the information used by each method and the assumptions involved. It also analyzes the sources of error.

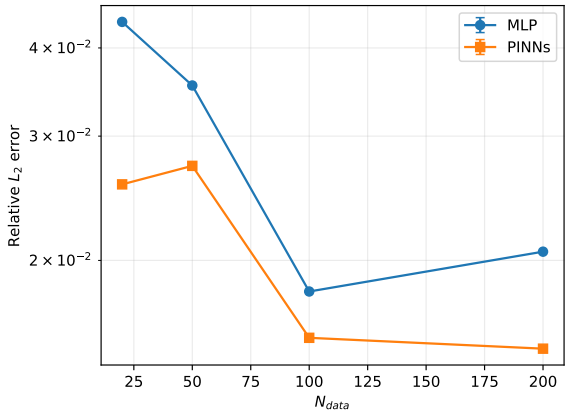


Figure 2: Relative L^2 error versus the number of labeled samples.

6.2. From Outcome Evaluation to Cognitive Evaluation

Teaching assessment should not simply rely on whether the code runs or whether the test error decreases. Learning objectives can be divided into three levels. First, students are expected to correctly implement the FDM, MLP, and PINNs. They also need to understand the concepts of inputs, outputs, and loss function. Second, students can diagnose models using error maps, learning curves, and physical residuals. Third, they should be able to answer conceptual questions, such as: Why can physical constraints improve generalization when data are sparse? Why must λ be balanced? How do PINNs and finite differences differ in their use of the PDE? Such assessment can reflect whether students have progressed from algorithmic operation to methodological understanding.

6.3. Applicability and Limitations of the Teaching Design

The one-dimensional heat equation has analytical solution and simple mathematical form, it is suitable as an introductory PINNs example. These advantages also define the limits of the present teaching design. The linear one-dimensional problem does not represent the training difficulty of complex nonlinear, high-dimensional, or multiscale PDEs. In addition, the observations in the current case are mainly generated from analytical or numerical solutions. Thus, they do not reflect measurement noise or model error in real experimental data.Further evidence is also required to demonstrate the effectiveness of the teaching design itself. Such evidence includes pre-tests and post-tests, learning-task scores, and student feedback. Accordingly, this paper focuses on an implementable and comparable teaching framework. Algorithmic results are not considered as direct evidence of teaching effectiveness.

7. Numerical Experiments

7.1. Experimental Configuration

The experiments use $L = T = 1$ and $\alpha = 0.1$. The reference solution is Eq. (4). All neural models use the architecture 2–32–32–32–1 with tanh activation and Xavier initialization. Adam is used with learning rate 10^{-3}. The main MLP and PINNs comparison uses 1000 Adam iterations. PINNs use $N_f = 192$ collocation points and $\lambda = 0.01$. The random seed is 1234. The reported neural network values are deterministic single seed runs. They are used to demonstrate an instructional workflow, not to estimate population-level performance or uncertainty.

The principal metric is the relative L^2 error, i.e., $E_{L^2} = \frac{\|\hat{u}-u\|_2}{\|u\|_2}$, computed on a 101×101 evaluation grid, where $\hat{u}$ is numerical solution, u is analytical soultion. Root mean square error (RMSE), mean absolute error (MAE), maximum absolute error, and PDEs residual are also recorded.

7.2. Experiment 1: FDM Accuracy and Stability

For the convergence experiment, the stable ratio is fixed at $r = 0.4$ while N_x is doubled from 10 to 80. Table 3 shows the relative error. The error decreases by approximately a factor of four whenever Δx is halved. This confirms the expected second-order spatial convergence. The time step is chosen as $\Delta t \propto \Delta x^2$.

7.3. Experiment 2: MLP versus PINNs under Different Data Sets

Table 4 compares matched MLP and PINNs models. In these deterministic runs, the PINNs gives a lower relative L^2 error at all four data sets. The reductions relative to the corresponding MLP are 37.1%, 26.6%, 20.0% and 27.4% for $N_{\text{data}} = 20, 50, 100, 200$, respectively. More importantly for the pedagogical objective, the PDEs residual is consistently smaller for the PINNs. This makes visible the difference between fitting observations and learning a function that is also closer to the governing equation.

The error is not strictly monotone in N_{data} for a single finite training run. This is a useful teaching point: neural network optimization depends on initialization, optimization length, sample placement and competing objectives. More data do not guarantee a monotone error curve in every individual run. Repeated-seed experiments would be required before making statistical claims about the magnitude of improvement.

For $N_{\text{data}} = 50$, Fig. 3 compares the analytical field, MLP prediction, PINNs prediction and absolute error. The visualization complements the scalar metrics by showing where each model deviates from the reference solution.

7.4. Experiment 3: Loss Evolution

Fig. 4 records the total, data and PDEs loss for the representative PINNs run with $N_{\text{data}} = 50$. Plotting the loss components separately reveals the multi-objective nature of PINNs training. It also helps students identify imbalances among different loss terms. Therefore, training quality can be evaluated beyond the final test error.

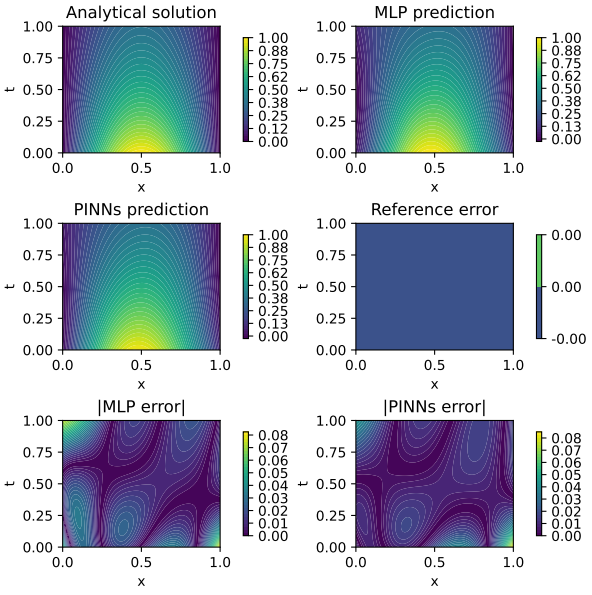


Figure 3: Analytical solution, predictions, and absolute errors for the representative $N_{\text{data}} = 50$.

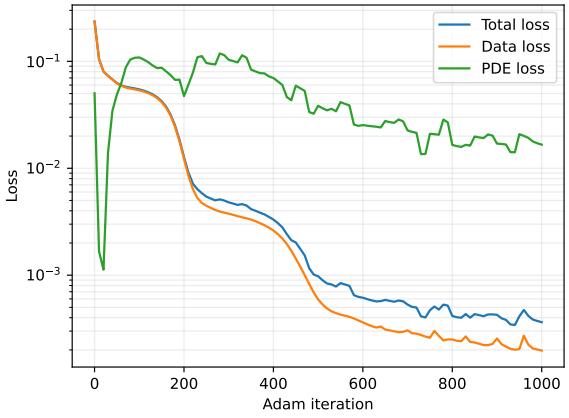


Figure 4: PINNs loss evolution for the representative $N_{\text{data}} = 50$.

7.5. Experiment 4: Sensitivity to the Physics-Loss Weight

To isolate the role of λ, a separate controlled experiment uses $N_{\text{data}} = 50$, 500 Adam iterations, $N_f = 128$, and the same seed for all weights. Table 5 shows a clear trade-off. Increasing λ strongly reduces the PDEs residual, but an excessively large value worsens the prediction error. In this particular finite run experiment, $\lambda = 0.1$ gives the smallest relative L^2 error among the tested values, whereas $\lambda = 100$ produces the smallest PDEs residual but the largest prediction error. Thus, satisfying the PDEs residual more strongly is not equivalent to obtaining the best data consistent approximation.

Table 5
Sensitivity to the physics-loss weight ($N_{\text{data}} = 50$).

λ	Rel. L^2	RMSE	MAE	PDEs RMS
0	0.04400	0.02048	0.01499	0.30019
0.01	0.03821	0.01779	0.01232	0.17822
0.1	0.03097	0.01441	0.01010	0.06401
1	0.05259	0.02448	0.01670	0.01556
10	0.13225	0.06155	0.04530	0.01060
100	0.45978	0.21399	0.17353	0.00430

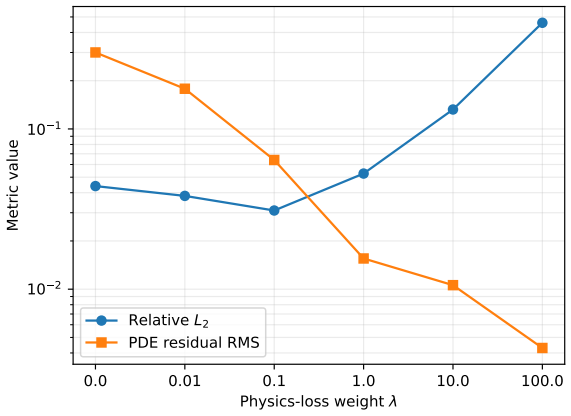


Figure 5: Prediction error and PDEs residual as functions of the physics-loss weight.

8. Conclusions and Future Work

This paper addresses the instructional challenge of connecting mechanism-based knowledge, data-driven methods, and physics-informed integration in PINNs education. It develops a three-stage progressive teaching design that moves from mechanism-based modeling to data-driven learning and then to physics-informed integration. Rather than presenting the finite difference method, MLP, and PINNs as separate approaches, the proposed design emphasizes the logical progression and connections relationship among the three stages. The finite difference stage establishes a reference linking the physical model to numerical computation. The purely data-driven stage uses sparse samples to reveal generalization limitations. The PINNs stage further incorporates the governing equation into the neural network as a physical constraint. This progression provides a clear rationale for introducing physical information into neural network training.

All three stages use the same one-dimensional heat conduction problem. The analytical solution provides a common reference. The evaluation progresses from numerical and prediction errors to physical residuals and loss value. This organization helps students understand PINNs as an integration of traditional mechanism-based knowledge and data-driven methods, rather than as a new algorithm disconnected from previous courses. In addition, the teaching activities include deliberate stability failure and sparse-data

generalization analysis. They also examine the sensitivity to the physical loss weight and the balance between different loss terms. These activities encourage students to move from reproducing results to interpreting and diagnosing models.

Formal classroom studies can use pre-tests and post-tests, assignment performance, and student feedback. These studies can evaluate changes in students' conceptual understanding, methodological transfer, and problem-diagnosis abilities. The results can provide further evidence for the effectiveness of the three-stage progressive teaching design.

Acknowledgement

The authors declare that there is no conflict of interest. The work was supported partially by the Doctoral Foundation of Fuyang Normal University under Grant No. 2025KYQD0080, 2025KYQD0031, and 2024KYQD0094, partially by the Natural Science Foundation of Fuyang Normal University under Grant No. 2025FSKJ2 and 2025FSKJ09, partially by the Teacher Education Collaborative Quality Improvement Plan of Fuyang Normal University under Grant No. 2025XTKYTD01, and partially by the Undergraduate Teaching Quality Enhancement Project of Fuyang Normal University under Grant No. 2024JYXM0042 and 2025JYXM0011.

References

[1] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics, 2021, vol. 3, pp. 422–440.

[2] J. Danek, J. Pospisil, Challenges in automatic differentiation and numerical integration in physics-informed neural networks modelling, International Journal for Numerical Methods in Engineering, 2025, vol. 126, no. 20, pp. e70148:1–e70148:28.

[3] C. Díaz-Cuadro, S. A. Correa Lazo, Physics-informed neural networks – A new field in mechanical engineering education, Mecánica Computacional, 2025, vol. 42, no. 16, pp. 1159–1168.

[4] M. Raissi, P. Perdikaris, G. 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, 2019, vol. 378, pp. 686–707.

[5] S. Cuomo, V. Schiano Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics-informed neural networks: Where we are and what's next, Journal of Scientific Computing, 2022, vol. 92, no. 3, pp. 88:1–88:62.

[6] C. Zhao, F. Zhang, W. Lou, X. Wang, J. Yang, A comprehensive review of advances in physics-informed neural networks and their applications in complex fluid dynamics, Physics of Fluids, 2024, vol. 36, no. 10, pp. 101301:1–101301:27.

[7] W. Zhang, J. Li, CPINNs: A coupled physics-informed neural networks for the closed-loop geothermal system, Computers and Mathetics with Applications, 2023, vol. 132, pp. 161–179.

[8] H. Hu, L. Qi, X. Chao, Physics-informed neural networks (PINN) for computational solid mechanics: Numerical frameworks and applications, Thin-Walled Structures, 2024, vol. 205, no. B, pp. 112495:1–112495:25.

[9] T. Le-Duc, S. Lee, H. Nguyen-Xuan, J. Lee, A hierarchically normalized physics-informed neural network for solving differential equations: Application for solid mechanics problems, Engineering Applications of Artificial Intelligence, 2024, vol. 133, no. E, pp. 108400:1–108400:30.

[10] R. Moore, Curriculum restructuring in South African higher education: Academic identities and policy implementation, Studies in Higher Education, 2003, vol. 28, no. 3, pp. 303–319.

[11] M. H. A. Lam, From defining to enacting interdisciplinary education: Curriculum integration strategies among Hong Kong undergraduate interdisciplinary programmes, Studies in Higher Education, 2025, vol. 50, no. 12, pp. 2762–2781.

[12] S. K. Howard, A. Chan, P. Caputi, More than beliefs: Subject areas and teachers' integration of laptops in secondary teaching, British Journal of Educational Technology, 2015, vol. 46, no. 2, pp. 360–369.

[13] X. Zhang, Y. Liu, J. Kou, H. Liu, A governed workflow for LLM-based formative assessment with score stabilization, Journal of Artificial Intelligence Research, 2026, vol. 3, no. 1, pp. 9–18.

[14] X. Wang, Personalized forgetting-aware multi-view graph knowledge tracing for vocational bachelor's practical training platform evaluating, IFS/ACM Transactions on Machine Learning, 2026, vol. 3, no. 1, pp. 20–29.

[15] T. Wang, S. Yin, Enhancing sustainability of online education via a validated cloud management framework, IFS/ACM Transactions on Machine Learning, 2026, vol. 3, no. 1, pp. 9–19.

[16] C. Cabo, Persistence of early misconceptions hinders progress in learning computer programming, IEEE Transactions on Education, 2026, vol. 69, no. 1, pp. 31–39.

[17] L. Yeomans, S. Zschaler, K. Coate, Transformative and troublesome? Students' and professional programmers' perspectives on difficult concepts in programming, ACM Transactions on Computing Education, 2019, vol. 19, no. 3, pp. 23:1–23:27.

[18] S. Fitzgerald, R. McCauley, B. Hanks, L. Murphy, B. Simon, C. Zander, Debugging from the student perspective, IEEE Transactions on Education, 2010, vol. 53, no. 3, pp. 390–396.

[19] R. J. LeVeque, Finite difference methods for ordinary and partial differential equations: Steady-state and time-dependent problems, Society for Industrial and Applied Mathematics, 2007.

[20] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT Press, 2016.