ADAMFLIP: ADAPTIVE MOMENTUM FEEDBACK LINEARIZATION OPTIMIZATION FOR HARD CONSTRAINED PINN TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Physics-informed neural networks (PINNs) provide a flexible framework for solving forward and inverse problems governed by partial differential equations (PDEs), but standard PINN training typically relies on soft penalty formulations that combine PDE residuals, data mismatch, and initial/boundary conditions using manually chosen weights. This often leads to ill-conditioning, sensitivity to loss weights, and poor constraint satisfaction. In this work, we reformulate PINN training as an equality-constrained optimization problem and propose a novel Adaptive Momentum Feedback Linearization Optimization for Hard Constrained PINN (AdamFLIP). The key idea is to view the constraint residuals as the output of a controlled dynamical system and to compute the Lagrange multiplier as a system input that drives these residuals toward stable linear contraction dynamics. AdamFLIP then applies Adam-style first- and second-moment adaptation to the resulting feedback-linearized Lagrangian gradient, combining principled constraint handling with the scalability and robustness of adaptive neural-network optimization. We test AdamFLIP on a range of benchmark forward and inverse PDE problems, and it consistently outperforms both the standard soft-constrained PINN and state-of-the-art constrained optimizers. Specifically, on the Navier–Stokes equations benchmark, AdamFLIP **reduces relative L_2 error by more than two thirds** for the predicted solution compared to the next best method. Our AdamFLIP framework provides an effective and computationally scalable hard constraint optimization method for PINN training.

1 INTRODUCTION

Physics informed neural networks (PINNs) have become increasingly popular in scientific machine learning which helps bridge the gap between data-driven modeling and classical PDE-based simulation. In a standard PINN (c.f. Raissi et al. (2019); Karniadakis et al. (2021); Cuomo et al. (2022)), one represents the solution of a PDE with a neural network and trains it by minimizing a composite objective loss function that includes the residual of the governing differential equation (via automatic differentiation of the network) together with initial/boundary conditions and any available data. The learned solution is therefore expected to be approximate both the available data and the governing equations. This framework can naturally extend to both forward and inverse problems; in the inverse problem setting, unknown PDE parameters are treated as additional trainable variables and inferred jointly with the solution during optimization. PINNs have consequently been applied in a wide range of engineering and scientific fields Cai et al. (2021); Guo et al. (2025).

Most existing PINN training approaches use a soft-penalty formulation, in which the PDE residual, boundary conditions (BCs), and initial conditions (ICs) are summed as separately weighted terms in a composite loss function. However, improper weight selection can lead to ill-conditioning, slow convergence, or solutions that fit the data while significantly violating the governing equations or boundary and initial conditions. Moreover, because each loss term may differ in scale, minimizing the composite loss does not guarantee that each term is individually minimized. Hence, the PINN's performance is highly sensitive to choice of penalty weights which has been shown in extensive empirical and theoretical studies Wang et al. (2021); Krishnapriyan et al. (2021); Wang et al. (2023); Maddu et al. (2022).

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.

These challenges have motivated several recent methods that incorporate constrained-optimization ideas into PINN training while retaining the soft-penalty objective. For example, Xu and Darve Xu & Darve (2021) adopt a trust-region strategy, and ADMM-PINNs Song et al. (2024) use operator splitting to decouple training. However, both methods still remain penalty weights dependent. A more fundamental shift recasts PINN training entirely as a constrained optimization problem. One prominent direction uses augmented Lagrangian (AL) methods Lu et al. (2021); Son et al. (2023); Hu et al. (2025), which treats ICs and BCs as equality constraints and performs primal updates coupled with multiplier (and penalty) updates to adaptively emphasize violated constraints. A second direction leverages sequential quadratic programming (SQP) and trust-region formulations; for example, Cheng & Na (2024) use sequential quadratic programming with trust-region to mitigate ill-conditioning and instability observed in standard PINNs (trSQP-PINN). Rather than enforcing constraints through the optimizer, a third class of methods builds them directly into the network architecture. Early work in this direction embedded constraints into the trial solution Lagaris et al. (1998); Djeumou et al. (2022). More recent approaches hard-impose boundary conditions through Fourier-feature parameterizations Straub et al. (2025); LI2 (2024) or add a projection layer derived from Karush–Kuhn–Tucker (KKT) conditions to exactly satisfy linear equality constraints Chen et al. (2024).

Beyond PINNs, constrained optimization is a long-standing theme in machine learning and reinforcement learning Kotary et al. (2021); Garcıa & Fernández (2015), where the goal is to optimize a learning objective while satisfying explicit constraints subject to explicit operational or physical constraints. Standard approaches in this broader literature include primal–dual Lagrangian methods (e.g. Nandwani et al. (2019); Stooke et al. (2020)), SQP-style methods (e.g. Berahas et al. (2021)), and hard-constraint-by-construction approaches (e.g. Min & Azizan (2024); Grontas et al. (2025)). Recently, a separate line of work Zhang et al. (2025a;b); Cerone et al. (2025) has brought a control-theoretic perspective to constrained optimization that proposes a feedback linearization (FL) solver that imports a classical nonlinear control idea into constrained learning. Rather than relying on primal-dual updates or second-order subproblems, FL designs the multiplier/feedback update to impose a stable, approximately linear contraction on constraint violations, and has been developed with rigorous convergence theories for general constrained nonconvex problems.

Our Contributions. We bring this FL framework to PINN training and make the following contributions. First, we formulate PINN training as an equality-constrained problem that explicitly separates feasibility (IC/BC/PDE satisfaction) from optimality, and design the multiplier/feedback update to shape the constraint-violation dynamics into a stable linear system. Second, we introduce AdamFLIP[1], an adaptive variant that integrates FL with moment estimation following Adam Kingma & Ba (2014). AdamFLIP preserves the fast convergence and robustness of adaptive gradient methods while incorporating principled constraint handling through FL. Unlike existing hard-constrained PINN optimizers that often rely on augmented-Lagrangian schedules or sovling sequential constrained subproblem with quasi-Newton updates, AdamFLIP remains fully first-order while incorporating Adam-style momentum and coordinate-wise adaptive scaling to achieve acceleration and efficiency. To the best of our knowledge, this makes AdamFLIP the first adaptive first-order optimizer for hard-constrained PINN training that achieves strong empirical performance without requiring second-order information or solving trust-region constrained subproblems. Empirically, AdamFLIP consistently outperforms both standard soft-penalty PINN training with Adam and representative constrained PINN optimizers, achieving significantly improved constraint satisfaction and solution accuracy for both forward and inverse problems. Third, we provide finite-time convergence analysis for a metric-compatible variant of AdamFLIP, showing that both the stationarity residual and constraint violation achieves best-iterate converge at a rate of $\mathcal{O}(\log T/\sqrt{T})$ under standard assumptions. Together, these results demonstrate that embedding feedback-linearization principles into modern adaptive optimizers offers a simple yet powerful alternative to penalty reweighting and classical constrained-training schemes.

Due to space constraints, we defer a more comprehensive review of the related literature and a detailed comparison with our work to Appendix B.

[1]Anonymous code repository: `https://anonymous.4open.science/r/AdamFLIP-5B54`

2 PRELIMINARIES: PHYSICS INFORMED NEURAL NETWORKS (PINNS)

To illustrate the framework of *physics informed neural networks* (PINNs) Cai et al. (2021); Raissi et al. (2019); Karniadakis et al. (2021), we start with the general nonlinear PDE which takes the following form:

$$u_t + \mathcal{N}[u] = 0,\ x \in \Omega,\ t \in [0, T], \tag{1}$$

where $\Omega \subset \mathbb{R}^D$ is an open, bounded D dimensional domain, $u : [0, T] \times \Omega \to \mathbb{R}$ denotes the unknown solution field, and $\mathcal{N}[\cdot]$ is a (generally nonlinear) differential operator acting on u.

Equation (1) is supplemented with the initial condition (IC):

$$u(0, x) = u_0(x), \qquad x \in \Omega, \tag{2}$$

and suitable boundary conditions (BCs) prescribed on the spatial boundary $\partial\Omega$. Without loss of generality, we denote these conditions abstractly by

$$\mathcal{B}[u](t, x) = h(t, x), \qquad (t, x) \in [0, T] \times \partial\Omega, \tag{3}$$

where $\mathcal{B}$ is a boundary operator (e.g., Dirichlet, Neumann, Robin, or a combination) and h is a given function.

PINNs offer a unified framework for combining observational data with physical laws. The key idea is to incorporate the governing PDE (1) together with the associated initial and boundary conditions (3)) directly into the loss function of a neural network where the required derivatives are obtained via automatic differentiation. In this setting, the solution u is approximated by a neural network which is trained both to fit available data and to minimize the residuals of the PDE and its boundary conditions. Consequently, the learned surrogate remains consistent with the underlying physics. More specifically, the PINN framework introduces the PDE residual loss (physics loss) denoted as ℓ_{phy} , initial condition loss denoted as ℓ_{ic} and boundary condition loss denoted as ℓ_{bc} according to (1), (2), (3) as follows:

$$\ell_{\text{phy}}(u; t, x) := u_t(t, x) + \mathcal{N}[u](t, x); \tag{4}$$

$$\ell_{\text{ic}}(u; t, x) := (u(0, x) - u_0(x))^2; \tag{5}$$

$$\ell_{\text{bc}}(u; t, x) := (\mathcal{B}[u](t, x) - h(t, x))^2 \tag{6}$$

In specific, in PINN we collect samples (generally from a mesh) $\mathcal{D} := \{(t_i, x_i)\}$ and we define

$$\begin{aligned} \mathcal{D}_{\text{phy}} &:= \{(x, t) \in \mathcal{D} : x \in \Omega,\ t \in [0, T]\}, \\ \mathcal{D}_{\text{ic}} &:= \{(x, t) \in \mathcal{D} : x \in \Omega,\ t = 0\}, \\ \mathcal{D}_{\text{bc}} &:= \{(x, t) \in \mathcal{D} : (t, x) \in [0, T] \times \partial\Omega\} \end{aligned} \tag{7}$$

Then assume that u is parameterized by the parameter of a neural network, i.e. $u = u_\theta$, then the total PDE residual loss $\mathcal{L}_{\text{phy}}$, boundary loss $\mathcal{L}_{\text{bc}}$ and initial condition $\mathcal{L}_{\text{ic}}$ loss with respect to the parameter θ are given by

$$\mathcal{L}_{\text{label}}(\theta) := \frac{1}{|\mathcal{D}_{\text{label}}|} \sum_{(t_i, x_i) \in \mathcal{D}_{\text{label}}} \ell_{\text{label}}(u_\theta; t_i, x_i);\ \ \text{label} \in \{\text{phy}, \text{ic}, \text{bc}\}$$

Forward Problem For the forward problem, we assume that the differential operator $\mathcal{N}$ is fully known and the task is to find a proper u_θ that makes $\mathcal{L}_{\text{phy}}, \mathcal{L}_{\text{ic}}, \mathcal{L}_{\text{bc}}$ as small as possible, and in general, the classical approach solves this by solving the unconstrained optimization problem

$$\min_\theta \mathcal{L}(\theta) := w_{\text{phy}}\mathcal{L}_{\text{phy}}(\theta) + w_{\text{ic}}\mathcal{L}_{\text{ic}}(\theta) + w_{\text{bc}}\mathcal{L}_{\text{bc}}(\theta), \tag{8}$$

where $w_{\text{phy}}, w_{\text{ic}}, w_{\text{bc}}$ are some hand-tuned weight on different losses. Then we solve this by first-order methods such as gradient descent or Adam Kingma & Ba (2014).

Inverse Problem The inverse problem considers a more challenging setting in which the differential operator $\mathcal{N}$ in (1) depends on unknown parameters κ (e.g. the unknown viscosity term v in 1D burgers equation). The goal is to jointly estimate the true parameter κ while learning an accurate solution u_θ from noisy observational data. To clarify the notation, we explicitly denote the parameterized differential operator by $\mathcal{N}_\kappa$. Accordingly, the associated physics loss is redefined as

$$\ell_{\text{phy}}(u, \kappa; t, x) := u_t(t, x) + \mathcal{N}_\kappa[u](t, x), \quad \mathcal{L}_{\text{phy}}(\theta, \kappa) := \frac{1}{|\mathcal{D}_{\text{phy}}|} \sum_{(t_i, x_i) \in \mathcal{D}_{\text{phy}}} \ell_{\text{phy}}(u_\theta, \kappa; t_i, x_i)$$

whereas the definitions of $\mathcal{L}_{\text{ic}}, \mathcal{L}_{\text{bc}}$ still remain unchanged. Note that the goal is to both recover the optimal parameter of the neural network θ and the parameter of the differential operator κ. To achieve this, we also need to collect observation data set $u(x_i, t_i)$ where $(x_i, t_i) \in \mathcal{D}$, and define the data loss as

$$\mathcal{L}_{\text{data}} := \frac{1}{|\mathcal{D}|} \sum_{(t_i,x_i)\in\mathcal{D}} \left(u_\theta(t_i, x_i) - u_i\right)^2, \tag{9}$$

and similar to solving the forward problem classical approach solves this by solving a minimization problem on a weighted loss that is a weighted summation of all the losses and then use first order method to find a solution.

$$\min_{\theta,\kappa} \mathcal{L}(\theta, \kappa) := w_{\text{phy}}\mathcal{L}_{\text{phy}}(\theta, \kappa) + w_{\text{ic}}\mathcal{L}_{\text{ic}}(\theta) + w_{\text{bc}}\mathcal{L}_{\text{bc}}(\theta) + w_{\text{data}}\mathcal{L}_{\text{data}}(\theta) \tag{10}$$

3 Feedback Linearization Constrained Optimization for Solving PINNs

Constrained Optimization for PINNs As mentioned in the previous section, PINNs typically solve optimization problems where the total objective loss $\mathcal{L}(\theta), \mathcal{L}(\theta, \kappa)$ is a linear combination of different losses such as data loss, physics loss, boundary condition loss and initial condition loss. While these constraints ideally should be satisfied during training, the highly nonlinear structure of the network and the discretized PDE makes it difficult to achieve objective loss going to strictly zero. Further, it often requires delicate tuning in the penalty weights $w_{\text{data}}, w_{\text{phy}}, w_{\text{ic}}, w_{\text{bc}}$.

Another way to handle this is via constrained optimization rather than optimizing the linear combination of different loss. Instead of solving (8) or (10), rather, we choose to put some of the losses into the constraint. In specific, for the *forward problem*, we consider the objective function is the physics loss $\mathcal{L}_{\text{phy}}(\theta)$, and the equality constraints aggregate the initial and boundary condition losses into a single vector $h(\theta) := [\mathcal{L}_{\text{ic}}(\theta), \mathcal{L}_{\text{bc}}(\theta)]^\top \in \mathbb{R}^2$:

$$\min_\theta \ \ \mathcal{L}_{\text{phy}}(\theta) \qquad s.t.\ \ \mathcal{L}_{\text{bc}}(\theta) = 0, \ \ \mathcal{L}_{\text{ic}}(\theta) = 0 \tag{11}$$

and for the inverse problem we consider the optimization variable expands to include both the neural network weights and the unknown PDE parameters, effectively replacing θ with the joint variable (θ, κ). The objective becomes the data loss $f(\theta, \kappa) := \mathcal{L}_{\text{data}}(\theta)$, and the constraint vector expands to enforce the PDE residual alongside the initial and boundary conditions, yielding $[\mathcal{L}_{\text{phy}}(\theta, \kappa), \mathcal{L}_{\text{ic}}(\theta), \mathcal{L}_{\text{bc}}(\theta)]^\top \in \mathbb{R}^3$.

$$\min_{\theta,\kappa} \ \ \mathcal{L}_{\text{data}}(\theta) \qquad s.t.\ \ \mathcal{L}_{\text{phy}}(\theta, \kappa), \ \mathcal{L}_{\text{bc}}(\theta), \ \mathcal{L}_{\text{ic}}(\theta) = 0 \tag{12}$$

Hence, rather than consider a unconstrained optimization problem, we put some of the losses as constraint and then we can use constrained optimization solvers. Note that both the forward problem 11 and inverse problem 12 fit into the general equality-constrained optimization framework as follows

$$\min_x f(\theta) \quad \text{s.t.} \quad h(\theta) = 0. \tag{13}$$

Remark 1 (Why Constrained Optimization for PINNs)**.** *Comparing with the penalty-based formulations* (8)*,* (10)*, constrained optimization offers a principled alternative by explicitly treating physical laws and boundary conditions as constraints, rather than soft penalties, thereby providing a clearer separation between feasibility and optimality. Moreover, constrained formulations can be interpreted as implicitly performing adaptive penalty adjustment (e.g. Li & Feng (2022)) through dual variables, alleviating the need for manual loss reweighting and improving robustness across different problem instances.*

We observe in Table 1 that constrained methods achieve substantially smaller boundary and initial condition errors than unconstrained optimizers such as Adam, even though the physics loss is occasionally larger, which is consistent with prior observations Rathore et al. (2024); Wang et al. (2024); De Ryck et al. (2024). This behavior can be attributed to two factors: first, the physics loss computed via automatic differentiation is sensitive to local approximation errors and network smoothness; second, PINN training involves an inherent tradeoff among physics residuals, boundary conditions, and initial conditions, so enforcing feasibility more strictly can redistribute errors across these components.

Feedback Linearization (FL) for equality-constrained optimization. Having reformulated PINN training as an equality-constrained problem, the next question is how to solve it efficiently and robustly. In this work, inspired by recent control-theoretic approaches to constrained optimization Zhang et al. (2025a); Cerone et al. (2025), we adopt a feedback linearization (FL) perspective, which views optimization dynamics as a controlled system and designs the multiplier to directly shape constraint evolution.

Consider the equality-constrained problem in (13), whose first-order KKT conditions are

$$\nabla f(\theta) + J_h(\theta)^\top \lambda = 0, \qquad h(\theta) = 0.$$

A common approach to constrained optimization is to perform gradient descent on the Lagrangian,

$$\theta_{t+1} = \theta_t - \eta\big(\nabla f(\theta_t) + J_h(\theta_t)^\top \lambda_t\big),$$

which reduces the problem to designing the dual variable λ_t.

Following Zhang et al. (2025a); Cerone et al. (2025), we interpret θ as the system state and λ as a feedback input, leading to a control-theoretic viewpoint in which the goal is to design a stabilizing feedback law. Let $y_t = h(\theta_t)$ denote the constraint violation. A first-order approximation gives

$$y_{t+1} - y_t \approx -\eta \left(J_h(\theta_t)\nabla f(\theta_t) + J_h(\theta_t)J_h(\theta_t)^\top \lambda_t\right).$$

FL selects λ_t to impose stable linear dynamics on y_t, namely

$$y_{t+1} - y_t = -\eta K y_t,$$

where $0 \prec K \preceq \frac{1}{\eta} I$. Solving for λ_t yields

$$\lambda_t = -\big(J_h(\theta_t)J_h(\theta_t)^\top\big)^{-1}\big(J_h(\theta_t)\nabla f(\theta_t) - Kh(\theta_t)\big), \tag{14}$$

which enforces a controlled contraction of constraint violations. Substituting back gives the FL update

$$\theta_{t+1} = \theta_t - \eta\left(\nabla f(\theta_t) - J_h(\theta_t)^\top \big(J_h(\theta_t)J_h(\theta_t)^\top\big)^{-1}\big(J_h(\theta_t)\nabla f(\theta_t) - Kh(\theta_t)\big)\right).$$

The analysis and empirical results in Cerone et al. (2025); Zhang et al. (2025a) show that the resulting algorithm performs strongly and achieves $O(\frac{1}{\sqrt{T}})$ even in nonconvex settings.

Remark 2 (Connection and Difference with SQP)**.** *Recent results in Zhang et al. (2025a) establish a formal connection between FL and sequential quadratic programming (SQP), showing that for the specific choice $K = \frac{1}{\eta} I$, the FL update recovers the following SQP*

$$\min_d \ \ \nabla f(\theta_t)^\top d + \tfrac{1}{2\eta}\|d\|^2 \qquad \textit{s.t.}\ \ h(\theta_t) + J_h(\theta_t) d = 0.$$

A key distinction, however, is that FL introduces an explicit gain matrix K to directly shape the constraint dynamics, enabling flexible control over the stability and rate of feasibility enforcement. Moreover, the control-theoretic formulation of FL naturally admits momentum extensions that incorporate temporal smoothing into the dynamics, and this observation directly motivates the momentum-based variants developed next.

FL with Momentum (FL-momentum) In Zhang et al. (2025a), a momentum-based variant of feedback linearization, termed FL-momentum, was proposed to accelerate convergence. The method augments the feedback-linearized update with a momentum variable as

$$\begin{aligned} m_t &= \beta m_{t-1} + \big(\nabla f(\theta_t) + J_h(\theta_t)^\top \lambda_t\big), \qquad (\lambda_t \text{ is set according to (14)}) \\ \theta_{t+1} &= \theta_t - \eta m_t. \end{aligned}$$

Compared to naive FL, which updates the primal variable θ using only the instantaneous feedback-linearized gradient $\nabla f(\theta_t) + J_h(\theta_t)^\top \lambda_t$, FL-momentum incorporates information from previous iterations through the momentum variable m_t. From an optimization perspective, the momentum term introduces an inertial effect that extrapolates the update direction based on past descent trends, allowing the iterate to move more aggressively along persistent descent directions. As a result, FL-momentum empirically achieves accelerated convergence relative to naive FL.

4 AdamFLIP: An Optimizer for Solving Constrained Optimization in PINNs

While FL-momentum introduced at the end of previous section achieves acceleration, it still uses a single global stepsize across all coordinates, which can be suboptimal for neural-network parametrizations where gradients are often highly anisotropic and unevenly scaled. In contrast, in unconstrained optimization, Adam Kingma & Ba (2014) has achieved strong empirical success in training neural networks across a wide range of applications. Motivated by this observation, we propose AdamFLIP (**Ada**ptive **m**omentum **F**eedback **Li**nearization Optimization for Hard Constrained **P**INN Training), which augments FL-momentum with coordinate-wise adaptive scaling; see Algorithm 1. where $x^{\circ 2}$

Algorithm 1 AdamFLIP Algorithm

Require: Initial parameters θ_0, step size η, moments $m_0 = 0$, $v_0 = 0$, hyperparameters $\beta_1, \beta_2, \epsilon, \delta$

1: **for** $t = 1, 2, \dots$ **do**

2: Compute constraint multiplier $\lambda_t = -\big(J_h(\theta_t)J_h(\theta_t)^\top + \epsilon I\big)^{-1}\big(J_h(\theta_t)\nabla f(\theta_t) - Kh(\theta_t)\big)$

3: $g_t \leftarrow \nabla f(\theta_t) + J_h(\theta_t)^\top \lambda_t$

4: $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$

5: $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^{\circ 2}$

6: $\hat{m}_t \leftarrow \dfrac{m_t}{1-\beta_1^t}, \quad \hat{v}_t \leftarrow \dfrac{v_t}{1-\beta_2^t}$

7: $\theta_{t+1} \leftarrow \theta_t - \eta\, \hat{m}_t \oslash \left(\hat{v}_t^{\circ \frac{1}{2}} + \delta\right)$

8: **end for**

and $x^{\circ \frac{1}{2}}$ denote the elementwise square and square root of x, and $x \oslash y$ denotes elementwise division.

In addition to the first-moment estimate m_t, AdamFLIP maintains a second-moment accumulator v_t that tracks the elementwise squared magnitude of the feedback-linearized gradient. The resulting update rescales each coordinate of the momentum direction by an estimate of its historical variability, allowing parameters with consistently large gradients to take smaller steps while amplifying progress along flatter directions.

The design of AdamFLIP is directly inspired by the Adam optimizer. Specifically, AdamFLIP mirrors Adam's use of bias-corrected first- and second-moment estimates, but replaces the standard gradient with the gradient of the Lagrangian $\nabla f(\theta_t) + J_h(\theta_t)^\top \lambda_t$. From this perspective, AdamFLIP can be interpreted as applying Adam-style adaptive normalization to feedback-linearized constrained optimization, preserving the structure and intuition of FL while inheriting the robustness and scalability that make Adam effective in large-scale neural network training.

We would also like to point out one caveat that FL-based algorithms are designed under the assumption that $J_h(\theta)J_h(\theta)^\top$ is invertible, an assumption that may not hold in PINN training, particularly near exact constraint satisfaction where some residual gradients can become near zero. In Algorithm 1, we mitigate potential ill-conditioning by introducing a damped solve using $J_h(\theta_t)J_h(\theta_t)^\top + \epsilon I$.

4.1 Theoretical Guarantees

In this section, we provide a finite-time convergence guarantee for a variant of AdamFLIP. We note that a rigorous analysis of the original AdamFLIP remains challenging. To address this, we introduce a modified version (Algorithm 2), for which we establish convergence.

The key difficulty with the original AdamFLIP (Algorithm 1) lies in a mismatch of metrics: the feedback-linearization multiplier is computed in the Euclidean metric, while the final update is preconditioned by an Adam-style diagonal metric $D_t = \mathrm{diag}\left(\frac{1}{\sqrt{\hat{v}_t}+\delta}\right)$. This inconsistency makes it difficult to control the optimization dynamics. To resolve this issue, the variant computes the multiplier directly in the same diagonal metric D_t used for the update, ensuring metric consistency throughout. In addition, following Reddi et al. (2018), we adopt a monotone second-moment estimate (AMSGrad-style) to stabilize the adaptive preconditioner. This modification prevents oscillations in the effective stepsizes and enables a tractable convergence analysis.

We emphasize that, despite strong empirical performance, providing a convergence guarantee for the original AdamFLIP remains an open problem. We leave this question to future work. In the following, we introduce standard assumptions for nonconvex constrained optimization (cf. Nocedal & Wright (2006); Peterson (1973)).

Algorithm 2 AdamFLIP variant

Require: Initial point θ_1, moments $m_0 = 0$, $v_0 = 0$, $\bar{v}_0 = 0$, stepsizes $\{\eta_t\}$, parameters $\beta_1, \beta_2 \in [0, 1)$, denominator constant $\delta > 0$, gain $K = \kappa I$

1: **for** $t = 1, 2, \ldots$ **do**
2: Define $D_t = \operatorname{diag}\left(\frac{1}{\sqrt{\bar{v}_{t-1}}+\delta}\right)$.
3: Compute $\lambda_t = -\big(J_h(\theta_t) D_t J_h(\theta_t)^\top\big)^{-1}\big(J_h(\theta_t) D_t \nabla f(\theta_t) - \kappa h(\theta_t)\big)$.
4: $g_t = \nabla f(\theta_t) + J_h(\theta_t)^\top \lambda_t$.
5: $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^{\circ 2}$.
6: $\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \tilde{v}_t = \frac{v_t}{1-\beta_2^t}$.
7: $\bar{v}_t = \max_{\text{elem}}\{\bar{v}_{t-1}, \tilde{v}_t\}$.
8: $\theta_{t+1} = \theta_t - \eta_t D_t \hat{m}_t$.
9: **end for**

Assumption 1 (Smoothness and regularity). *f, h are C^2 on a neighborhood of a compact set $\mathcal{C}$, the iterates remain in $\mathcal{C}$, and $J_h(\theta)$ has full row rank on $\mathcal{C}$. Consequently, there exists G such that $\|g_t\| \le G$ for all t.*

Assumption 2 (Stepsizes). $\eta_t > 0$, $\eta_{t+1} \le \eta_t$, $\eta_t \le 1/\kappa$, $\sum_t \eta_t = \infty$, $\sum_t \eta_t^2 < \infty$.

Define $M_t := J_h(\theta_t) D_t J_h(\theta_t)^\top, \quad \lambda_t^\dagger := -M_t^{-1} J_h(\theta_t) D_t \nabla f(\theta_t)$, and $r_t := \nabla f(\theta_t) + J_h(\theta_t)^\top \lambda_t^\dagger$, then we can prove the following theorem

Theorem 1 (Finite-time convergence, proof see Appendix F). *Under Assumptions 1–2, the iterates generated by Algorithm 2 satisfy*

$$\sum_{t=1}^T \eta_t \left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) = \mathcal{O}\left(1 + \sum_{t=1}^T \eta_t^2\right).$$

Consequently,

$$\min_{1 \le t \le T} \left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) = \mathcal{O}\left(\frac{1}{\sum_{t=1}^T \eta_t}\right).$$

Remark 3 (Discussion on the convergence implications and assumptions). *The quantity $\|r_t\|$ corresponds to a first-order stationarity residual, while $\|h(\theta_t)\|_1$ measures feasibility violation. Together, $\|r_t\|^2 + \|h(\theta_t)\|_1$ serves as a natural surrogate for the KKT residual of the constrained problem. Consequently, Theorem 1 establishes that this KKT residual diminishes along the iterates in a weighted average sense, which in turn implies a best-iterate convergence guarantee.*

In particular, choosing $\eta_t = \frac{\eta}{\sqrt{t \log t}}$ for $t \ge 2$ satisfies the stepsize conditions. In this case, Theorem 1 implies the best-iterate guarantee

$$\min_{2 \le t \le T} \left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) = \mathcal{O}\Big(\frac{\log T}{\sqrt{T}}\Big).$$

We note, however, that the obtained rate is still weaker than the classical rate typically achieved by first-order methods for nonconvex constrained optimization with $\min_{1 \le t \le T} \left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) = \mathcal{O}\big(\frac{1}{T}\big)$. This gap reflects the additional challenges introduced by the adaptive metric and the coupled multiplier update, and closing this gap remains an interesting direction for future work.

We also note that the full-row-rank condition on $J_h(\theta)$ cannot be guaranteed in general for PINN training, particularly near exact constraint satisfaction where some residual gradients may become degenerate. In practice, we do not observe this issue to be problematic, and we further stabilize the multiplier computation by using a damped solve with $J_h(\theta_t) J_h(\theta_t)^\top + \epsilon I$. Extending the convergence analysis to rank-deficient or relaxed constraint settings remains an open direction.

Table 1: Consolidated error comparison of evaluated models. Metrics include relative L^2 error, wall-clock time (WT, in seconds) measured on an NVIDIA H100 GPU (80 GB), and constraint errors (CE) as MSE across initial/boundary conditions (IC/BC), physics residuals (Phy), and data loss (DL). Parameter recovery absolute errors are denoted by Err. Best results are highlighted in blue and second-best in light blue.

Dataset	Metric	Standard PINN	AdamFLIP	FL-PINN	AL-PINN	trSQP-PINN
Burgers Fwd	Rel. L_2	2.51×10^{-1}	3.12×10^{-2}	1.06×10^{-1}	8.20×10^{-2}	4.43×10^{-1}
	CE (IC)	4.26×10^{-3}	1.00×10^{-4}	6.82×10^{-4}	2.34×10^{-3}	9.91×10^{-2}
	CE (BC)	2.44×10^{-3}	1.00×10^{-4}	4.40×10^{-4}	6.25×10^{-5}	9.80×10^{-2}
	CE (Phy)	1.49×10^{-3}	2.60×10^{-3}	1.13×10^{-2}	2.97×10^{-3}	1.31×10^{-3}
	WT (s)	113	115	133	155	625
Burgers Inv	Rel. L_2	5.46×10^{-1}	5.74×10^{-2}	4.19×10^{-1}	4.02×10^{-1}	4.54×10^{-1}
	CE (IC)	9.00×10^{-3}	2.90×10^{-3}	2.26×10^{-2}	3.06×10^{-2}	4.41×10^{-1}
	CE (BC)	1.00×10^{-4}	3.50×10^{-3}	2.04×10^{-2}	2.73×10^{-2}	2.56×10^{-1}
	CE (Phy)	1.00×10^{-4}	2.11×10^{-2}	1.80×10^{-3}	2.48×10^{-2}	5.20×10^{-3}
	CE (DL)	1.13×10^{-1}	1.20×10^{-3}	6.62×10^{-2}	5.94×10^{-2}	4.07×10^{-1}
	Err (κ_1)	3.33×10^{-2}	1.54×10^{-2}	1.28×10^{-1}	5.94×10^{-1}	9.82×10^{-1}
	Err (κ_2)	6.50×10^{-3}	3.60×10^{-3}	9.65×10^{-2}	7.97×10^{-2}	1.90×10^{-2}
	WT (s)	248	324	334	334	621
TFMDWEs Fwd	Rel. L_2	7.71×10^{-1}	2.85×10^{-1}	4.26×10^{-1}	4.93×10^{-1}	4.56×10^{-1}
	CE (IC)	1.65×10^{-3}	1.42×10^{-5}	1.36×10^{-4}	1.11×10^{-3}	1.69×10^{-3}
	CE (BC)	2.19×10^{-2}	1.47×10^{-4}	4.20×10^{-3}	5.16×10^{-2}	2.90×10^{-2}
	CE (Phy)	1.07×10^{-1}	8.87×10^{-2}	1.19×10^{-1}	7.88×10^{-1}	8.13×10^{-1}
	WT (s)	56	50	52	53	105
TFMDWEs Inv	Rel. L_2	2.22×10^{-1}	3.95×10^{-2}	5.99×10^{-1}	8.81×10^{-1}	6.67×10^{-1}
	CE (IC)	9.01×10^{-4}	1.77×10^{-5}	1.56×10^{-5}	2.60×10^{-4}	1.19×10^{-4}
	CE (BC)	2.27×10^{-2}	2.57×10^{-4}	4.08×10^{-4}	3.74×10^{-3}	3.59×10^{-3}
	CE (Phy)	7.01×10^{-2}	6.53×10^{-2}	1.82×10^{-1}	4.92×10^{-3}	3.78×10^{-3}
	CE (DL)	3.59×10^{-3}	1.13×10^{-4}	2.61×10^{-2}	5.62×10^{-2}	5.44×10^{-2}
	Err (α)	1.93×10^{-1}	3.40×10^{-2}	7.40×10^{-2}	9.54×10^{-2}	1.64×10^{-1}
	WT (s)	120	133	141	134	144
Heat Fwd	Rel. L_2	3.43×10^{-2}	7.53×10^{-3}	9.49×10^{-3}	3.09×10^{-2}	6.60×10^{-1}
	CE (IC)	1.20×10^{-4}	5.59×10^{-5}	1.13×10^{-4}	3.65×10^{-5}	1.22×10^{-1}
	CE (BC)	1.56×10^{-4}	6.35×10^{-5}	1.70×10^{-4}	7.89×10^{-5}	2.62×10^{-2}
	CE (Phy)	1.82×10^{-3}	2.00×10^{-4}	7.35×10^{-4}	7.41×10^{-4}	1.07×10^{-2}
	WT (s)	261	336	354	364	1260
Heat Inv	Rel. L_2	2.90×10^{-2}	1.59×10^{-2}	4.06×10^{-2}	3.98×10^{-2}	6.42×10^{-1}
	CE (IC)	3.06×10^{-5}	1.43×10^{-5}	1.80×10^{-5}	1.71×10^{-4}	4.81×10^{-4}
	CE (BC)	1.19×10^{-4}	1.03×10^{-5}	8.73×10^{-6}	3.07×10^{-4}	4.57×10^{-3}
	CE (Phy)	2.79×10^{-4}	9.13×10^{-5}	7.44×10^{-5}	2.29×10^{-4}	3.73×10^{-3}
	CE (DL)	5.22×10^{-5}	1.51×10^{-5}	8.81×10^{-5}	9.18×10^{-5}	2.59×10^{-2}
	Err (κ)	9.00×10^{-4}	5.00×10^{-4}	6.70×10^{-3}	2.00×10^{-3}	8.84×10^{-1}
	WT (s)	297	346	360	361	1300
Navier–Stokes	Rel. $L_2(u)$	3.62×10^{-2}	1.18×10^{-2}	7.70×10^{-2}	5.19×10^{-2}	1.96×10^{0}
	Rel. $L_2(v)$	3.48×10^{-2}	1.06×10^{-2}	8.13×10^{-2}	3.69×10^{-2}	9.68×10^{-1}
	CE (IC)	2.15×10^{-5}	1.26×10^{-5}	1.98×10^{-3}	1.10×10^{-3}	7.82×10^{-4}
	CE (BC)	1.02×10^{-4}	4.86×10^{-5}	2.72×10^{-3}	8.13×10^{-4}	3.56×10^{-4}
	CE (Phy)	6.11×10^{-4}	2.53×10^{-4}	1.08×10^{-2}	8.56×10^{-4}	1.29×10^{1}
	WT (s)	155	335	414	410	720

5 EXPERIMENT RESULTS

We evaluate AdamFLIP on four benchmark PDEs spanning 1D and 2D settings, linear and nonlinear dynamics, and integer- and fractional-order operators: (i) 1D viscous Burgers equation, (ii) 1D time-fractional mixed diffusion-wave equation (TFMDWE), (iii) 2D heat equation, and (iv) 2D incompressible Navier–Stokes equations (NSE). Each problem is tested in both *forward* (all PDE parameters known) and *inverse* (unknown parameters co-optimized with the network) settings, except NSE which is tested in the forward setting only. We compare against a standard PINN (Adam), AL-PINN Son et al. (2023), trSQP-PINN Cheng & Na (2024), and FL-PINN (feedback linearization without adaptive moments), as summarized in Table 2. The rationale for selecting these baselines is discussed in Appendix B and Remark 4. All methods share identical network architectures, training data, and stopping criteria per problem to ensure a fair comparison. Full experimental details (architecture, collocation points, hyperparameters, et al.) are deferred to Appendix D.

Benchmarks. We use four representative PDE systems: the 1D viscous Burgers equation with $\nu = 0.01/\pi$; a time-fractional mixed diffusion–wave equation (TFMDWE) with Caputo order α; the 2D heat equation with thermal diffusivity ν; and the 2D incompressible Navier–Stokes equations with $\nu = 0.01$ on the Taylor–Green vortex. For the inverse tasks, we infer the Burgers coefficients (κ_1, κ_2), the TFMDWE fractional order α, and the heat-equation diffusivity κ, respectively. Full governing equations, domains, exact solutions, data-generation procedures, architectures, and hyperparameters are provided in Appendix D.

Discussion. Table 1 summarizes the seven evaluated configurations. AdamFLIP obtains the lowest relative L_2 error in every configuration, with especially large gains on Burgers inverse, TFMDWE inverse, and Navier–Stokes. For forward problems, it improves Burgers accuracy by roughly $3\times$ over the next-best baseline and reduces both Navier–Stokes velocity errors to about 10^{-2}; for inverse problems, it also yields the most accurate recovery of the unknown PDE parameters. These gains come with modest overhead relative to standard PINN training, while avoiding the much larger cost of trSQP-PINN. Detailed per-problem comparisons, constraint-error trade-offs, and additional solution/error visualizations are provided in Appendix A and Appendix D.

6 CONCLUSION AND FUTURE WORK

In this paper, we introduced AdamFLIP, an adaptive momentum-based feedback linearization optimizer for hard-constrained PINN training. By viewing constraint residuals as the outputs of a controlled dynamical system and designing the Lagrange multiplier as a system input that drives these residuals toward stable linear contraction dynamics, AdamFLIP provides principled constraint enforcement while accelerating optimization. Importantly, the proposed framework avoids manual loss-weight tuning while maintaining computational efficiency. We evaluated AdamFLIP against representative constrained PINN baselines on a range of benchmark PDEs and time-fractional partial differential equations (TFMDWEs) in both forward and inverse problem settings. Empirical results show that AdamFLIP consistently achieves improved constraint satisfaction and lower relative L_2 errors, while incurring only a modest increase in training time.

These results suggest several promising directions for future work. First, extending the AdamFLIP framework to stochastic optimization settings could further broaden its applicability to large-scale and data-driven scientific machine learning problems. Second, applying AdamFLIP to enforce PDE symmetries, particularly Lie point symmetries Brandstetter et al. (2022); Akhound-Sadegh et al. (2023); Mialon et al. (2023), presents an interesting direction toward developing more physically consistent and symmetry-aware neural solvers.

AI USE STATEMENT

We used generative AI tools as writing and coding assistants during the preparation of this manuscript. Specifically, generative AI tools were used to improve language clarity and presentation, assist with coding and debugging for numerical experiments, and support proofreading and refinement of proof exposition. All technical ideas, theoretical results, experimental design, and final interpretations

were developed, verified, and approved by the authors. We reviewed all AI-assisted work and take responsibility for the final content of this manuscript.

REFERENCES

Physical informed neural networks with soft and hard boundary constraints for solving advection-diffusion equations using fourier expansions. *Computers & Mathematics with Applications*, 159: 60–75, 2024. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2024.01.021.

T. Akhound-Sadegh, L. Perreault-Levasseur, J. Brandstetter, M. Welling, and S. Ravanbakhsh. Lie point symmetry and physics-informed networks. In *Advances in Neural Information Processing Systems*, volume 36, pp. 42468–42481, 2023.

Albert S Berahas, Frank E Curtis, Daniel Robinson, and Baoyu Zhou. Sequential quadratic optimization for nonlinear equality constrained stochastic optimization. *SIAM Journal on Optimization*, 31 (2):1352–1379, 2021.

Johannes Brandstetter, Max Welling, and Daniel E Worrall. Lie point symmetry data augmentation for neural PDE solvers. In *International Conference on Machine Learning*, pp. 2241–2256. PMLR, 2022.

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

Vito Cerone, Sophie M Fosson, Simone Pirrera, and Diego Regruto. A new framework for constrained optimization via feedback control of lagrange multipliers. *IEEE Transactions on Automatic Control*, 2025.

Hao Chen, Gonzalo E Constante Flores, and Can Li. Physics-informed neural networks with hard linear equality constraints. *Computers & Chemical Engineering*, 189:108764, 2024.

Xiaoran Cheng and Sen Na. Physics-informed neural networks with trust-region sequential quadratic programming. *arXiv preprint arXiv:2409.10777*, 2024.

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

Tim De Ryck, Ameya D Jagtap, and Siddhartha Mishra. Error estimates for physics-informed neural networks approximating the navier–stokes equations. *IMA Journal of Numerical Analysis*, 44(1): 83–119, 2024.

Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. In *Learning for Dynamics and Control Conference*, pp. 263–277. PMLR, 2022.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Panagiotis D Grontas, Antonio Terpin, Efe C Balta, Raffaello D'Andrea, and John Lygeros. Pinet: Optimizing hard-constrained neural networks with orthogonal projection layers. *arXiv preprint arXiv:2508.10480*, 2025.

Jiangtao Guo, Hao Zhu, Yujie Yang, and Chenrui Guo. Advances in physics-informed neural networks for solving complex partial differential equations and their engineering applications: A systematic review. *Engineering Applications of Artificial Intelligence*, 161:112044, 2025.

Qifeng Hu, Shamsulhaq Basir, and Inanc Senocak. Conditionally adaptive augmented lagrangian method for physics-informed learning of forward and inverse problems. *arXiv preprint arXiv:2508.15695*, 2025.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

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

James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*, 2021.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

Shirong Li and Xinlong Feng. Dynamic weight strategy of physics-informed neural networks for the 2d navier–stokes equations. *Entropy*, 24(9):1254, 2022.

Lu Lu, Raphaël Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G. Johnson. Physics-informed neural networks with hard constraints for inverse design. *arXiv preprint arXiv:2102.04626*, 2021. URL https://arxiv.org/pdf/2102.04626.

Yuri Luchko. Initial-boundary-value problems for the generalized multi-term time-fractional diffusion equation. *Journal of Mathematical Analysis and Applications*, 374(2):538–548, 2011.

Suryanarayana Maddu, Dominik Sturm, Christian L Müller, and Ivo F Sbalzarini. Inverse dirichlet weighting enables reliable training of physics informed neural networks. *Machine Learning: Science and Technology*, 3(1):015026, 2022.

G. Mialon, Q. Garrido, H. Lawrence, D. Rehman, Y. LeCun, and B. Kiani. Self-supervised learning with lie symmetries for partial differential equations. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28973–29004, 2023.

Youngjae Min and Navid Azizan. Hardnet: Hard-constrained neural networks with universal approximation guarantees. *arXiv preprint arXiv:2410.10807*, 2024.

Yatin Nandwani, Abhishek Pathak, and Parag Singla. A primal dual formulation for deep learning with constraints. *Advances in neural information processing systems*, 32, 2019.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

David W Peterson. A review of constraint qualifications in finite-dimensional spaces. *Siam Review*, 15(3):639–654, 1973.

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.

Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training pinns: A loss landscape perspective. *arXiv preprint arXiv:2402.01868*, 2024.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Hwijae Son, Sung Woong Cho, and Hyung Ju Hwang. Enhanced physics-informed neural networks with augmented lagrangian relaxation method (al-pinns). *Neurocomputing*, 548:126424, 2023.

Yongcun Song, Xiaoming Yuan, and Hangrui Yue. The admm-pinns algorithmic framework for nonsmooth pde-constrained optimization: A deep learning approach. *SIAM Journal on Scientific Computing*, 46(6):C659–C687, 2024. doi: 10.1137/23M1566935.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International conference on machine learning*, pp. 9133–9143. PMLR, 2020.

Christopher Straub, Philipp Brendel, Vlad Medvedev, and Andreas Rosskopf. Hard-constraining neumann boundary conditions in physics-informed neural networks via fourier feature embeddings. *arXiv preprint arXiv:2504.01093*, 2025.

Utkarsh Utkarsh, Pengfei Cai, Alan Edelman, Rafael Gomez-Bombarelli, and Christopher Vincent Rackauckas. Physics-constrained flow matching: Sampling generative models with hard constraints. *arXiv preprint arXiv:2506.04171*, 2025.

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

Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An expert's guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*, 2023.

Tao Wang, Bo Zhao, Sicun Gao, and Rose Yu. Understanding the difficulty of solving cauchy problems with pinns. In *6th Annual Learning for Dynamics & Control Conference*, pp. 453–465. PMLR, 2024.

Kailai Xu and Eric Darve. Trust region method for coupled systems of pde solvers and deep neural networks. *arXiv preprint arXiv:2105.07552*, 2021.

Runyu Zhang, Arvind Raghunathan, Jeff S Shamma, and Na Li. Constrained optimization from a control perspective via feedback linearization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=zhgfM0dJ3F.

Runyu Zhang, Gioele Zardini, Asuman Ozdaglar, Jeff Shamma, and Na Li. Zeroth-order constrained optimization from a control perspective via feedback linearization. *arXiv preprint arXiv:2509.24056*, 2025b.

A Additional per-benchmark discussion

Discussion. Table 1 shows all results. For all seven problem configurations, AdamFLIP achieves the lowest relative L_2 error, typically improving over the next-best baseline by a factor of 2 to 10. We highlight several key observations.

Forward problems. On Burgers (forward), AdamFLIP attains a relative L_2 error of 3.12×10^{-2}, roughly $3\times$ lower than the next-best AL-PINN (8.20×10^{-2}) and an order of magnitude lower than Standard PINN (2.51×10^{-1}), while simultaneously achieving the best IC/BC constraint satisfaction. On the 2D heat equation, AdamFLIP (7.53×10^{-3}) and FL-PINN (9.49×10^{-3}) both substantially outperform Standard PINN (3.43×10^{-2}), confirming the benefit of constrained formulations; notably, trSQP-PINN diverges (6.60×10^{-1}) on this problem. For the TFMDWE, all methods have difficulty due to the non-local fractional operator, yet AdamFLIP (2.85×10^{-1}) still improves upon the next-best FL-PINN (4.26×10^{-1}) while reducing IC/BC errors by one–two orders of magnitude. On the Navier–Stokes benchmark, AdamFLIP reduces L_2 errors for both velocity fields to $\sim1\times10^{-2}$, roughly $3\times$ better than Standard PINN and an order of magnitude better than FL-PINN; trSQP-PINN again fails catastrophically on this system.

Inverse problems. The advantages of AdamFLIP are even more pronounced in parameter inference. On Burgers (inverse), AdamFLIP achieves $L_2 = 5.74\times10^{-2}$, nearly an order of magnitude below all baselines ($\sim4-5\times10^{-1}$), and recovers κ_1, κ_2 with the smallest absolute errors (1.54×10^{-2} and 3.60×10^{-3}, respectively). For the TFMDWE inverse problem, AdamFLIP attains $L_2 = 3.95\times10^{-2}$ and the most accurate recovery of the fractional order α (error 3.40×10^{-2}), while AL-PINN and FL-PINN degrade to errors larger than 5×10^{-1}. On the 2D heat inverse problem, all methods except trSQP-PINN perform reasonably, with AdamFLIP (1.59×10^{-2}) achieving the best L_2 accuracy and parameter error (5.00×10^{-4}).

Computational cost. AdamFLIP incurs only a modest overhead relative to Standard PINN (typically $\leq30\%$), while trSQP-PINN requires $2-5\times$ longer wall-clock time due to trust-region subproblem solves and pretraining requirements. AL-PINN and FL-PINN have comparable cost to AdamFLIP.

Constraint satisfaction trade-offs. Standard PINN occasionally achieves the lowest physics-loss CE (e.g., Burgers inverse), yet this does not translate to more accurate solution, consistent with the observation in Remark 1 that lower physics residual does not imply better constraint satisfaction in solution space. AdamFLIP's feedback mechanism adaptively redistributes enforcement effort across IC, BC, and physics constraints, yielding the best overall accuracy.

Additional spatiotemporal solution fields, absolute error maps, and per-problem analysis are provided in Appendix D.

B Related Works: Constrained Optimization Methods for PINN, and comparison with AdamFLIP

In this section, we review recent constrained-optimization approaches for PINN training and compare their algorithmic principles with AdamFLIP.

One line of work improves the optimization dynamics while largely preserving the penalty-based PINN objective. For example, Xu & Darve (2021) uses a trust-region strategy that restricts each update to a region where a local model of the composite loss is reliable, thereby improving stability in ill-conditioned regimes. ADMM-PINNs Song et al. (2024) instead use operator splitting: auxiliary variables are introduced so that the original problem decomposes into a PINN-based smooth subproblem and additional subproblems that may admit proximal or closed-form updates. Although these methods introduce constraints, such as the trust-region constraint in trust-region methods and the variable-splitting consistency constraint in ADMM, the main PINN training objective still relies on a weighted combination of PDE, data, and boundary/initial residuals. Therefore, they remain closer to soft-penalty-based PINN methods rather than fully hard-constrained formulations.

A more fundamental shift is to recast PINN training itself as a constrained optimization problem. In this view, some loss components are no longer treated merely as weighted penalties but are instead imposed as equality constraints. A prominent family of methods follows the augmented Lagrangian (AL) principle Lu et al. (2021); Son et al. (2023); Song et al. (2024); Hu et al. (2025). For instance, hPINNs Lu et al. (2021) use penalty and augmented-Lagrangian mechanisms to impose hard constraints in inverse-design problems, where the design variables and the PDE solution are learned through neural parameterizations. AL-PINNs Son et al. (2023) further formulate the PINN training problem by treating initial and boundary conditions as constraints and optimizing a sequence of augmented-Lagrangian objectives. In such methods, the training objective typically takes the form of the PDE residual augmented by multiplier terms and quadratic penalties for constraint violations. The multipliers and penalty parameters are updated over training so that violated constraints are emphasized more strongly. More recently, conditionally adaptive AL methods Hu et al. (2025) introduce separate penalty parameters for heterogeneous constraints and update them conditionally, so that difficult constraints receive stronger enforcement. These AL-based approaches provide an important step beyond fixed penalty weighting, since the effective weights on the constraints are adapted through dual and penalty updates. However, their practical behavior still depends on the design of penalty schedules, multiplier updates, inner-loop training accuracy, and stopping criteria for the sequential augmented problems. In contrast, AdamFLIP does not enforce constraints by repeatedly increasing penalty weights or performing dual ascent on an augmented objective. Instead, the multiplier in AdamFLIP is computed from the local Jacobian geometry and the desired feedback dynamics, rather than learned through an outer penalty/multiplier update loop.

A second constrained-optimization direction is based on SQP and trust-region methods. For example, trSQP-PINN Cheng & Na (2024) formulates PINN training as a hard-constrained problem and, at each iteration, solves a local approximation obtained by linearizing the constraints and using a quadratic model of the objective within a trust region. To reduce the cost of classical SQP, it uses quasi-Newton approximations and a pretraining stage to improve initialization. For instance, complex constraint implementations like trSQP-PINN require a pre-trained model and use quasi-Newton update to enforce the constraints, which doubling or tripling the total training time (as shown in our experiments in WT comparison in Table 1). AdamFLIP is related in spirit to SQP because both methods use local constraint linearization; see Remark 2. However, their algorithmic structures are different. trSQP-PINN solves a constrained quadratic/trust-region subproblem at each iteration and requires quasi-Newton updates, trust-region management, and pretraining. AdamFLIP instead computes a feedback-linearization multiplier in closed form and then applies an Adam-style adaptive

update to the resulting Lagrangian gradient. Thus, AdamFLIP retains local constraint awareness while remaining closer to a first-order neural-network optimizer.

Rather than enforcing constraints through the optimizer, a third class of methods builds them directly into the network architecture. This idea has early roots in trial-solution constructions Lagaris et al. (1998), where the neural ansatz is designed to satisfy prescribed initial or boundary conditions by construction. More recent work develops this principle in more systematic and problem-adapted ways, including physics-informed architectures for dynamical systems with built-in structural constraints Djeumou et al. (2022), Fourier-based parameterizations that hard-impose certain boundary conditions Straub et al. (2025); LI2 (2024), and KKT-hPINN Chen et al. (2024), which uses a KKT-derived projection layer to enforce linear equality constraints exactly. These hard-constraint-by-construction methods can guarantee exact satisfaction of eligible constraints at every forward pass. However, they are often problem-specific: the ansatz, embedding, or projection layer must be tailored to the boundary condition, domain geometry, or constraint structure. Moreover, they do not directly address the optimization difficulty associated with the remaining PDE residual, data loss, or inverse-parameter identification objective. AdamFLIP is complementary to this line of work: it does not require a specialized architecture, but instead operates at the optimizer level once the relevant PINN losses are formulated as equality constraints.

Remark 4 (Choice of constrained PINN baselines)**.** *In Section 5, we compare AdamFLIP with AL-PINN and trSQP-PINN as two representative optimizer-level constrained PINN baselines. AL-PINN represents augmented-Lagrangian methods, which enforce constraints through multiplier and penalty updates, while trSQP-PINN represents SQP/trust-region methods, which enforce constraints by solving local constrained subproblems. These two methods are therefore the closest classical constrained-optimization counterparts to AdamFLIP.*

We do not use architecture-based hard-constraint methods as primary baselines because they change the hypothesis class through problem-specific trial functions, embeddings, or projection layers. Such comparisons would mix the effect of the optimizer with the effect of a specialized constrained parameterization. We also include standard PINN with Adam as the soft-penalty baseline and FL-PINN as an ablation of AdamFLIP without Adam-style adaptive moments.

C Discussion on the Limitations

While AdamFLIP shows strong empirical performance, several limitations remain. First, our convergence analysis applies to a variant of AdamFLIP rather than the original AdamFLIP used in the experiments. The original algorithm computes the feedback-linearization multiplier in the Euclidean metric but applies an Adam-style preconditioned update, creating a metric mismatch that complicates the analysis. Second, the analysis assumes that $J_h(\theta)$ has full row rank, which may fail in PINN training, especially near exact constraint satisfaction. We mitigate possible ill-conditioning in practice by using the damped solve $J_h(\theta_t)J_h(\theta_t)^\top + \epsilon I$, but extending the theory to rank-deficient or relaxed constraint settings remains open. Finally, AdamFLIP requires choosing which losses are treated as objectives and which are imposed as constraints. Our choices work well for the tested benchmarks, but other PDEs, noisy or sparse data, and higher-dimensional problems may require different formulations and further empirical validation.

D Detailed Experimental Setup and Additional Results

This appendix provides the full experimental details for the numerical results reported in Section 5. We summarize the shared experimental settings below; problem-specific settings (architecture, collocation points, inverse problem parameterization) are given in each subsequent subsection.

D.1 Physics-Informed Neural Networks (PINNs)

PINN architecture. Figure 1 illustrates the PINN framework used throughout our experiments. The neural network takes spatiotemporal coordinates as input, processes them through multiple fully connected hidden layers with $\tanh$ activations, and outputs the predicted solution field $\hat{u}$. For inverse problems, the unknown PDE parameters (e.g., κ_1, κ_2, α) are included as learnable scalars in the

computational graph and co-optimized with the network weights θ. All required spatial and temporal derivatives are computed via automatic differentiation.

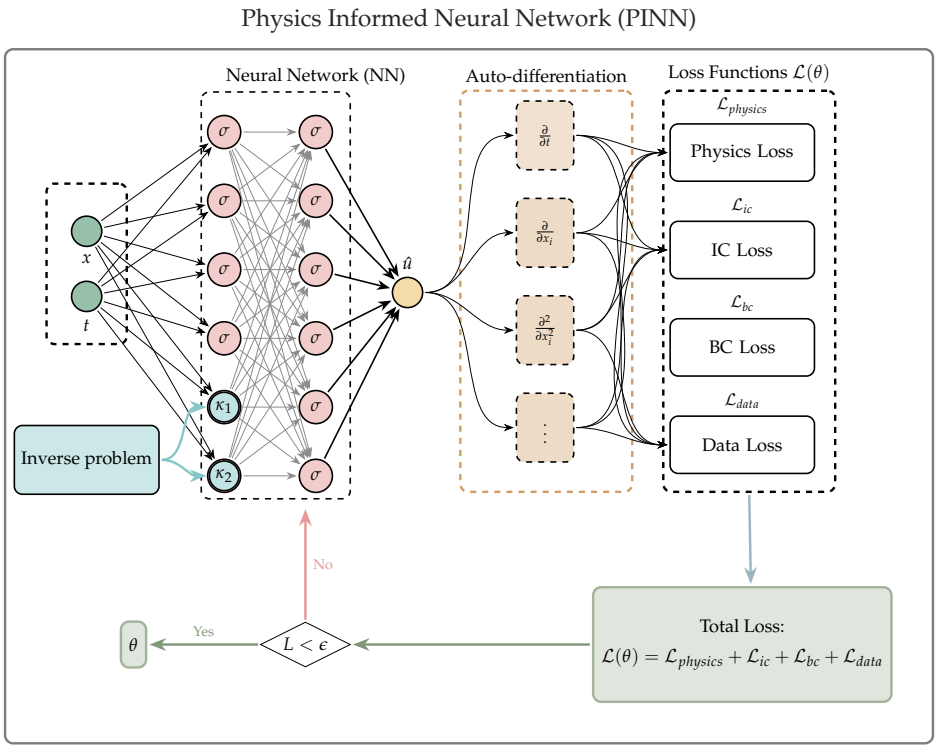


Figure 1: Illustration of the PINN framework. The network maps spatiotemporal inputs to the solution field $\hat{u}$; automatic differentiation provides exact derivatives for constructing the physics, initial condition, and boundary condition losses. In inverse problems, unknown PDE parameters are jointly optimized.

Training procedure. For each test problem, all five methods (Standard PINN, AL-PINN, trSQP-PINN, FL-PINN, and AdamFLIP) use an identical network architecture, the same training data, and the same stopping criterion: training terminates when the total loss falls below a predefined threshold ϵ, or a maximum iteration count is reached. The Adam optimizer Kingma & Ba (2014) is used for the unconstrained baseline (Standard PINN) and as the base optimizer within AdamFLIP, with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$.

Evaluation metrics. We report: (i) the relative L^2 error between the predicted and analytical solutions on a held-out test grid; (ii) the constraint errors (CE), computed as the mean squared error (MSE) at the initial condition (IC), boundary condition (BC), physics residual (Phy), and—for inverse problems—data loss (DL) collocation points; (iii) the absolute parameter recovery error for inverse problems; and (iv) the training wall-clock time (WT), measured on an HPC node equipped with an NVIDIA H100 GPU (80 GB).

Choice of Objectives and Constraints in the Algorithm For the forward problem, we choose the boundary loss and initial loss as constraints because the boundary and initial conditions define the admissible solution space of the PDE. If these conditions are not well satisfied, the learned solution may be physically inconsistent even when the residual loss is small inside the domain. Therefore, we optimize the physics residual as the main objective while enforcing the boundary and initial losses as constraints. This design encourages the model to reduce the PDE residual without violating the prescribed boundary and initial behaviors.

For the inverse problem, we use the data loss as the objective and treat the physics, boundary, and initial losses as constraints. This is because the main goal of the inverse problem is to fit the observed

data and recover the unknown quantities, while the PDE residual and prescribed conditions should still be satisfied to ensure physical consistency. In this setting, the data loss drives parameter identification, and the constraints prevent the learned solution from deviating from the governing equation and known problem conditions.

We performed experiments where all loss terms were treated as constraints with the target loss value set to zero. The final experimental results show that the above objective–constraint choices provide the best overall performance, giving a better balance between accuracy, constraint satisfaction, and training stability.

Constraint Error (CE) To evaluate the physical consistency of our learned solutions, we adopt the constraint error metric introduced by Utkarsh et al. (2025). Specifically, we compute the ℓ_2 norm of the residuals evaluated by the constraint functions $\mathcal{R}_*$ on each predicted sample $\hat{u}^{(n)}$, and average these norms across all N samples:

$$\text{CE}(*) = \frac{1}{N}\sum_{n=1}^{N}\left\|\mathcal{R}_*\left(\hat{u}^{(n)}\right)\right\|_2, \quad * \in \{\text{IC}, \text{BC}, \text{Phy}\}.$$

Here, the residual functions $\mathcal{R}_*$ quantify the violations of the prescribed Dirichlet initial and boundary conditions, as well as conservation laws.

Collocation sampling. Training collocation points are sampled uniformly at random from the respective domains: $\mathcal{D}_{\text{ic}}$ at $t = 0$, $\mathcal{D}_{\text{bc}}$ along the spatial boundaries over time, and $\mathcal{D}_{\text{phy}}$ in the spatiotemporal interior. The specific numbers of collocation points (N_{ic}, N_{bc}, N_f) vary by problem and are detailed in each subsection below.

D.2 One-dimensional Burgers' Equation

The one-dimensional viscous Burgers' equation is a nonlinear partial differential equation frequently used as a benchmark Karniadakis et al. (2021). The Burgers' equation with Dirichlet boundary conditions on the spatial domain $\Omega = [-1, 1]$ and time domain $[0, T]$ yields the following:

$$\frac{\partial u}{\partial t} - \nu\frac{\partial^2 u}{\partial x^2} + u\frac{\partial u}{\partial x} = 0, \quad x \in \Omega,\ t \in [0, T], \tag{15}$$

$$u(x, t) = 0, \quad \forall x \in \partial\Omega, \tag{16}$$

$$u(x, 0) = -\sin(\pi x). \tag{17}$$

Here, $u(x, t)$ represents the solution over space and time, and the viscosity ν is chosen to be $0.01/\pi$. The benchmark solution clearly displays the formation and evolution of steep gradients over time.

D.2.1 Numerical Settings

Neural network architecture To ensure a fair and consistent comparison across all different models, we employ an identical deep neural network architecture. Specifically, each model utilizes a fully connected network consisting of 8 hidden layers with 20 neurons per layer. During the training phase, each model was trained until its loss dropped below the predefined threshold, ϵ.

Training and testing data In the training stage, the IC and BC losses are evaluated using data sampled in space at the initial time $t = 0$ and in time along the spatial boundaries ($x = -1$ and $x = 1$). In contrast, the physics loss is computed on 100 unlabeled collocation points randomly selected from the interior of the spatiotemporal domain. During the testing stage, the evaluation data are sampled over the spatiotemporal domain $\Omega \times [0, T]$, where $\Omega = [0, 1]$. The data is generated on a uniform grid with a spatial mesh size of $\Delta x = 0.01$ and a temporal step size of $\Delta t = 0.01$.

Quantitative comparison metric To quantitatively assess and compare the performance across all tested models, we calculate the mean squared error (MSE) between the benchmark solution and the model prediction.

Acronyms of all different models

Table 2: Acronyms of all different models in numerical results.

Model Acronym	Full name	Reference
PINN	Physics-Informed Neural Network	Raissi et al. (2019)
AL-PINN	PINN With Augmented Lagrangian Relaxation Method	Son et al. (2023)
trSQP-PINN	PINN With Trust-Region Sequential Quadratic Programming	Cheng & Na (2024)
FL-PINN	Feedback linearization PINN	n/a
AdamFLIP	Adam feedback linearization PINN	n/a

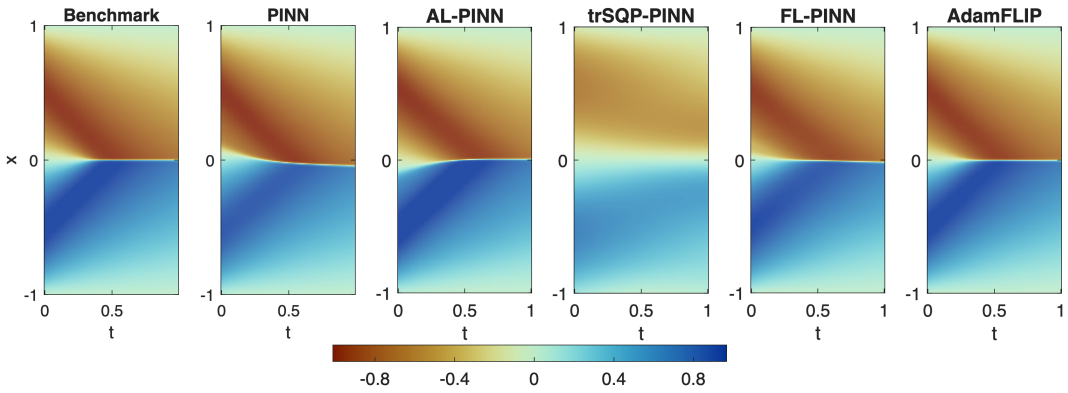
D.2.2 FORWARD PROBLEM

In this section, we consider the forward problem setting where the physics loss is fully specified and does not involve any unknown parameters to be inferred. The objective is therefore to approximate the solution accurately while ensuring consistency with the prescribed initial condition, boundary condition, and governing equation.

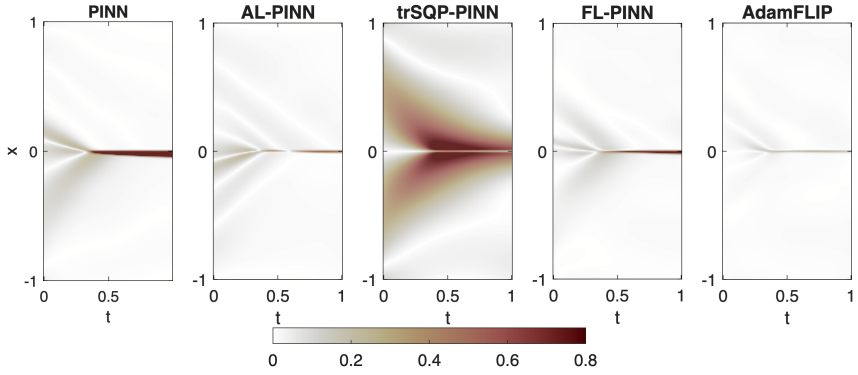
Table 3 presents a quantitative comparison of standard PINN (Adam), AL-PINN, trSQP-PINN, FL-PINN, and AdamFLIP for the Burgers' equation, measured in terms of the relative L^2 error and the constraint errors (CE) for the initial condition (IC), boundary condition (BC), and physics loss (PL). Among all models considered, AdamFLIP performs best as achieving a relative L^2 error that is approximately one order of magnitude smaller than those of the other models while also yielding the lowest constraint errors (CE) in most cases. Although trSQP-PINN attains the smallest physics-loss CE, which is several orders of magnitude lower than those of the other models, this advantage does not translate into improved relative L^2 error accuracy or better performance in the other constraint errors. It is also worth noting that all models yield comparable wall-clock times, indicating that the improved performance of AdamFLIP is achieved without introducing additional computational cost.

Table 3: Quantitative error comparison for the 1D Burgers equation (forward problem). We report relative L^2 errors, constraint errors (CE) at initial (IC), boundary (BC), and physics (Phy) collocation points, and training wall-clock time (WT).

Method	**Constraint**	**Rel.** L^2	**CE (IC)**	**CE (BC)**	**CE (Phy)**	**WT (s)**
Standard PINN	✗	2.51×10^{-1}	4.26×10^{-3}	2.44×10^{-3}	1.49×10^{-3}	**113**
AL-PINN	✓	8.20×10^{-2}	2.34×10^{-3}	6.25×10^{-5}	2.97×10^{-3}	155
trSQP-PINN	✓	4.43×10^{-1}	9.91×10^{-2}	9.80×10^{-2}	$\mathbf{1.31 \times 10^{-3}}$	625
FL-PINN	✓	1.06×10^{-1}	6.82×10^{-4}	4.40×10^{-4}	1.13×10^{-2}	133
AdamFLIP	✓	$\mathbf{3.12 \times 10^{-2}}$	$\mathbf{1.00 \times 10^{-4}}$	$\mathbf{1.00 \times 10^{-4}}$	2.60×10^{-3}	115



(a) Spatiotemporal fields



(b) Absolute errors

Figure 2: Comparison of spatiotemporal solutions and absolute errors for the one-dimensional Burgers' equation. (a) Predicted fields $u(x,t)$ from the benchmark solution and from Standard PINN, FL-PINN, trSQP-PINN, and AdamFLIP. (b) Corresponding absolute error distributions for each method.

This is also confirmed in Figure 2. In particular, Figure 2a compares the spatiotemporal fields predicted by different models with the benchmark solution (first panel). Compared with the benchmark solution, the standard PINN (second panel) captures the overall spatiotemporal structure but exhibits noticeable smoothing in regions that should contain sharp gradients. The FL-PINN (third panel) improves the representation of these features by more accurately capturing the solution structure along the characteristic directions of the dynamics, while AdamFLIP (forth panel) performs best overall, effectively preserving both the magnitude and the sharpness of the solution profile.

Figure 2b further supports these observations by presenting the spatiotemporal absolute error distributions for the different models.It is clearly observed that AdamFLIP attains the smallest error throughout the entire spatiotemporal domain, with substantially smaller errors in the shock regions than the other models. FL-PINN reduces these errors by incorporating feedback linearization, yet residual discrepancies remain visible. In contrast, AdamFLIP achieves the lowest absolute error throughout the domain, with errors significantly suppressed in both magnitude and spatial extent. These observations are consistent with the quantitative results in Table 3, confirming that AdamFLIP provides superior accuracy while enforcing physical constraints more effectively.

D.2.3 Inverse Problem

In the inverse problem setting, we assume that there are two unknown PDE parameters κ_1 and κ_2 in Eq. (15) , which yields the following setting:

$$\frac{\partial u}{\partial t} + \kappa_1 u \frac{\partial u}{\partial x} = \kappa_2 \frac{\partial^2 u}{\partial x^2}, \quad x \in \Omega,\ t \in [0, T], \tag{18}$$

$$u(x,t) = 0, \quad \forall x \in \partial\Omega, \tag{19}$$

$$u(x, 0) = -\sin(\pi x). \tag{20}$$

Within the PINN frameworks, κ_1 and κ_2 are parameterized as learnable scalars, which are co-optimized with the neural network weights by minimizing the loss function same as in the forward problem. In this setup, for all models we initialize the parameter estimates at $\kappa_1 = 2, \kappa_2 = 0$, whereas their ground truth values are $\kappa_1 = 1$ and$\kappa_2 = \nu = 0.01/\pi \approx 0.00318$.

Table 4 shows the learned PDE parameters and their L^1 error compared to the ground truth for the Burgers' equation in the inverse problem setting. It is shown that among all models, AdamFLIP demonstrates the highest accuracy by achieving the lowest L^1 errors in estimating both κ_1 and κ_2. Specifically, it reduces the estimation error of κ_1 by nearly an order of magnitude compared to the next best model, FL-PINN, while simultaneously improving the estimation accuracy for κ_2 by at least one-third relative to the other methods. We use the same neural network structure same as used in forward problem but treat the unknown parameters as the trainable parameters in the neural network. Notably, trSQP-PINN exhibits the largest error in κ_1 (0.9821), indicating unstable or inaccurate parameter identification. By enforcing parameter bounds and adaptively balancing constraints, AdamFLIP substantially improves robustness and accuracy in parameter discovery.

Table 4: Parameter estimation results for the one-dimensional Burgers' equation in the inverse problem setting. The exact PDE parameters are $\kappa_1 = 1$ and $\kappa_2 = 0.003183$.

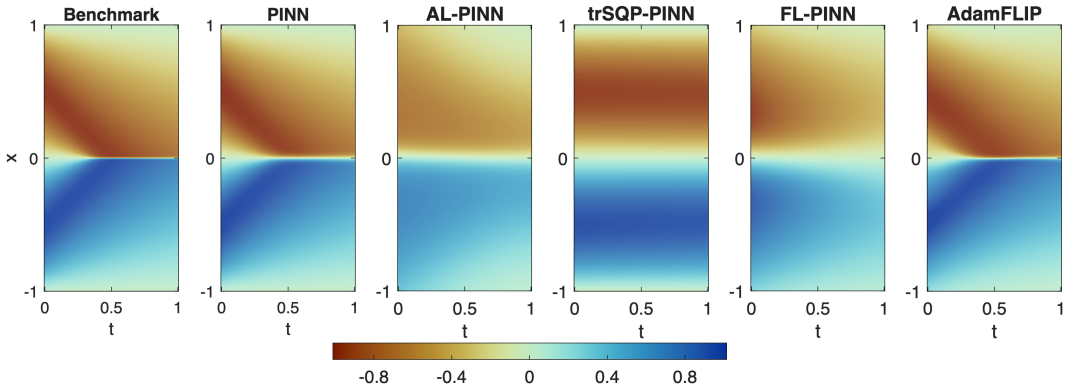
Method	$\hat{\kappa}_1$	$\lvert\hat{\kappa}_1 - \kappa_1\rvert$	$\hat{\kappa}_2$	$\lvert\hat{\kappa}_2 - \kappa_2\rvert$
Standard PINN	0.9667	0.0333	0.0097	0.0065
AL-PINN	1.5940	0.5940	0.0829	0.0797
trSQP-PINN	1.9821	0.9821	0.0222	0.0190
FL-PINN	0.8723	0.1277	0.0996	0.0965
AdamFLIP	**1.0154**	**0.0154**	**0.0067**	**0.0036**

Similar to Table 3, Table 5 presents a quantitative comparison of standard PINN (Adam), AL-PINN, trSQP-PINN, FL-PINN, and AdamFLIP for the Burgers' equation in the inverse problem setting, where the relative L^2 error and the constraint errors (CE) for the initial condition (IC), boundary condition (BC), and physics loss (PL) are compared. Consistent with the forward problem setting, AdamFLIP performs best, achieving a relative L^2 error that is approximately one order of magnitude smaller than those of the other models. Furthermore, it yields the lowest constraint errors in most categories. For example, while AL-PINN and FL-PINN manage to reduce the data loss constraint error, CE(DL), to roughly half of that produced by the standard PINN, AdamFLIP minimizes this error by nearly two orders of magnitude relative to the baseline. Similarly, AdamFLIP reduces the initial and boundary constraint errors to approximately one-tenth of those observed in the AL-PINN and FL-PINN models. While constrained models incur overhead in computational cost, i.e., AL-PINN, FL-PINN and trSQP-PINN take about one-third longer to train than standard PINN, AdamFLIP remains relative efficient.

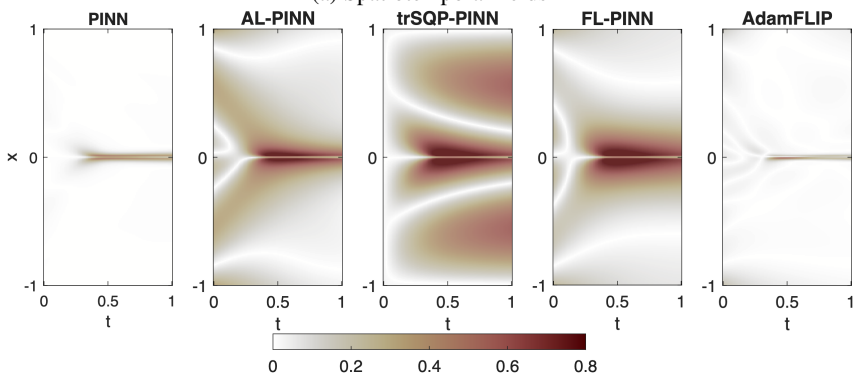
Consistent with the quantitative results in Tables 4–5, Figure 3 presents a qualitative comparison of the reconstructed spatiotemporal solutions and their corresponding absolute errors. The benchmark solution is characterized by a sharp transition near $x = 0$, a feature that AdamFLIP accurately captures across the entire temporal domain. In contrast, the standard PINN fails to resolve the underlying physical structure, instead converging to a trivial, highly diffuse profile that neglects the sharp interface. While FL-PINN improves upon the baseline, it still exhibits noticeable discrepancies near the transition region. These differences are further emphasized in the absolute error plots: the standard PINN shows significant error concentrations around the discontinuity, and FL-PINN achieves only moderate error reduction, whereas AdamFLIP maintains uniformly low errors throughout the entire domain. Overall, AdamFLIP proves to be the most effective framework, significantly outperforming other models with or without constraints by accurately identifying the governing physics while reconstructing the underlying spatiotemporal dynamics with moderate computational overhead.

Table 5: Quantitative error comparison for the 1D Burgers equation (inverse problem). We report relative L^2 errors, constraint errors (CE) at initial (IC), boundary (BC), physics (Phy), and data loss (DL) collocation points, and training wall-clock time (WT).

Method	Constraint	Rel. L_2	CE (IC)	CE (BC)	CE (Phy)	CE (DL)	WT (s)
Standard PINN	✗	5.46×10^{-1}	9.00×10^{-3}	$\mathbf{1.00 \times 10^{-4}}$	$\mathbf{1.00 \times 10^{-4}}$	1.13×10^{-1}	248
AL-PINN	✓	4.02×10^{-1}	3.06×10^{-2}	2.73×10^{-2}	2.48×10^{-2}	5.94×10^{-2}	334
trSQP-PINN	✓	4.54×10^{-1}	4.41×10^{-1}	2.56×10^{-1}	5.20×10^{-3}	4.07×10^{-1}	621
FL-PINN	✓	4.19×10^{-1}	2.26×10^{-2}	2.04×10^{-2}	1.80×10^{-3}	6.62×10^{-2}	334
AdamFLIP	✓	$\mathbf{5.74 \times 10^{-2}}$	$\mathbf{2.90 \times 10^{-3}}$	3.50×10^{-3}	2.11×10^{-2}	$\mathbf{1.20 \times 10^{-3}}$	324



(a) Spatiotemporal fields



(b) Absolute errors

Figure 3: Comparison of spatiotemporal solutions and absolute errors for the inverse one-dimensional Burgers' equation. (a) Predicted fields $u(x,t)$ from the benchmark solution and from Standard PINN, FL-PINN, trSQP-PINN ,and AdamFLIP. (b) Corresponding absolute error distributions for each method.

D.3 Time-fractional Mixed Diffusion-Wave Equations (TFMDWEs)

The third example we consider is the time-fractional mixed diffusion-wave equation (TFMDWE). This class of equations is widely used to model anomalous transport processes and memory effects in complex media, such as viscoelastic materials, where the dynamics transition between pure diffusion and wave propagation Luchko (2011). Specifically, we consider the tTFMDWEs on the spatial domain $\Omega = (0, \pi)$ with homogeneous Dirichlet boundary conditions:

$$D_t^{\alpha} u(x,t) = \frac{\partial^2 u}{\partial x^2} + f(x,t), \quad t \in [0,1],\ x \in \Omega \equiv [0,\pi], \tag{21}$$

$$u(x,t) = 0, \quad \forall x \in \partial\Omega, \tag{22}$$

$$u(x,0) = 0, \quad x \in \Omega, \tag{23}$$

where D_t^α denotes the Caputo fractional derivative of order $\alpha \in (0,1]$. In addition, we prescribe the forcing term as:

$$f(x,t) = \left[\frac{\Gamma(4)}{\Gamma(4-\alpha)} t^{3-\alpha} + t^3\right] \sin(x), \tag{24}$$

which corresponds to the analytical solution $u(x,t) = t^3 \sin(x)$. Here, $\Gamma(\cdot)$ represents the Euler gamma function.

D.3.1 PINN'S SETTINGS

Neural network architecture To ensure a fair and consistent comparison across all different models, we employ an identical deep neural network architecture. Specifically, each model utilizes a fully connected network consisting of 3 hidden layers with 100 neurons per layer. During the training phase, each model was trained until its loss dropped below the predefined threshold, ϵ.

Training and testing data In the training stage, the initial condition (IC) and boundary condition (BC) losses are evaluated using data uniformly sampled in space at $t = 0$ and in time along the spatial boundaries ($x = 0$ and $x = \pi$), with $N_{\text{ic}} = 100$ and $N_{\text{bc}} = 100$ points, respectively. In contrast, the physics loss for the TFMDWE is computed on $N_f = 5000$ interior collocation points randomly selected from the spatiotemporal domain $(0,\pi)\times(0,1]$. Additionally, $N_{\text{data}} = 100$ synthetic observation points are sampled from the exact solution $u(x,t) = t^3 \sin(x)$ to inform the training process. During the testing stage, the evaluation data are sampled over the spatiotemporal domain $\Omega \times [0,T]$, where $\Omega = [0,\pi]$ and $T = 1$. The data is generated on a uniform grid with a spatial mesh size of $\Delta x = \pi/100$ and a temporal step size of $\Delta t = 0.01$, resulting in a 100×100 grid for quantitative performance assessment.

D.3.2 FORWARD

Table 6 reports the numerical performance of Standard PINN, AL-PINN, trSQP-PINN, FL-PINN, and AdamFLIP on the forward fractional PDE problem. Overall, AdamFLIP provides the best accuracy and constraint satisfaction, achieving the lowest relative L_2 error of 2.85×10^{-1} while substantially reducing the constraint errors to $\mathrm{CE(IC)} = 1.42 \times 10^{-5}$ and $\mathrm{CE(BC)} = 1.47 \times 10^{-4}$. In comparison, FL-PINN improves over the baselines with a relative error of 4.26×10^{-1}, but its constraint errors remain larger ($\mathrm{CE(IC)} = 1.36 \times 10^{-4}$, $\mathrm{CE(BC)} = 4.20 \times 10^{-3}$) and its physics-loss constraint error is higher than AdamFLIP. Both AL-PINN and trSQP-PINN reduce the relative error compared to Standard PINN, yet they exhibit markedly poorer physics-loss enforcement, with $\mathrm{CE(PL)}$ on the order of 10^{-1}–10^{0}, indicating unstable or imbalanced training. Standard PINN performs worst overall, yielding a large relative error of 7.71×10^{-1} with comparatively weak constraint satisfaction. Moreover, AdamFLIP achieves these improvements with the lowest wall-clock time (50 s), demonstrating a favorable trade-off between accuracy, constraint enforcement, and computational efficiency for forward fractional PDE learning.

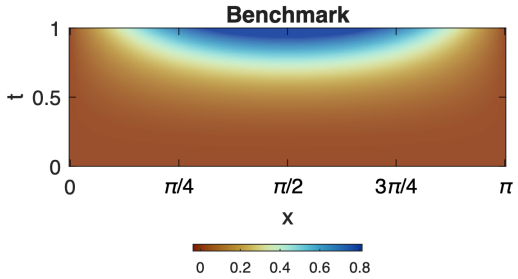
Table 6: Quantitative error comparison for the forward fractional PDE. We report relative L^2 errors, constraint errors (CE) at initial (IC), boundary (BC), and physics (Phy) collocation points, and training wall-clock time (WT).

Method	Constraint	Rel. L_2	CE (IC)	CE (BC)	CE (Phy)	WT (s)
Standard PINN	✗	7.71×10^{-1}	1.65×10^{-3}	2.19×10^{-2}	1.07×10^{-1}	56
AL-PINN	✓	4.93×10^{-1}	1.11×10^{-3}	5.16×10^{-2}	7.88×10^{-1}	53
trSQP-PINN	✓	4.56×10^{-1}	1.69×10^{-3}	2.90×10^{-2}	8.13×10^{-1}	105
FL-PINN	✓	4.26×10^{-1}	1.36×10^{-4}	4.20×10^{-3}	1.19×10^{-1}	52
AdamFLIP	✓	$\mathbf{2.85 \times 10^{-1}}$	$\mathbf{1.42 \times 10^{-5}}$	$\mathbf{1.47 \times 10^{-4}}$	$\mathbf{8.87 \times 10^{-2}}$	**50**

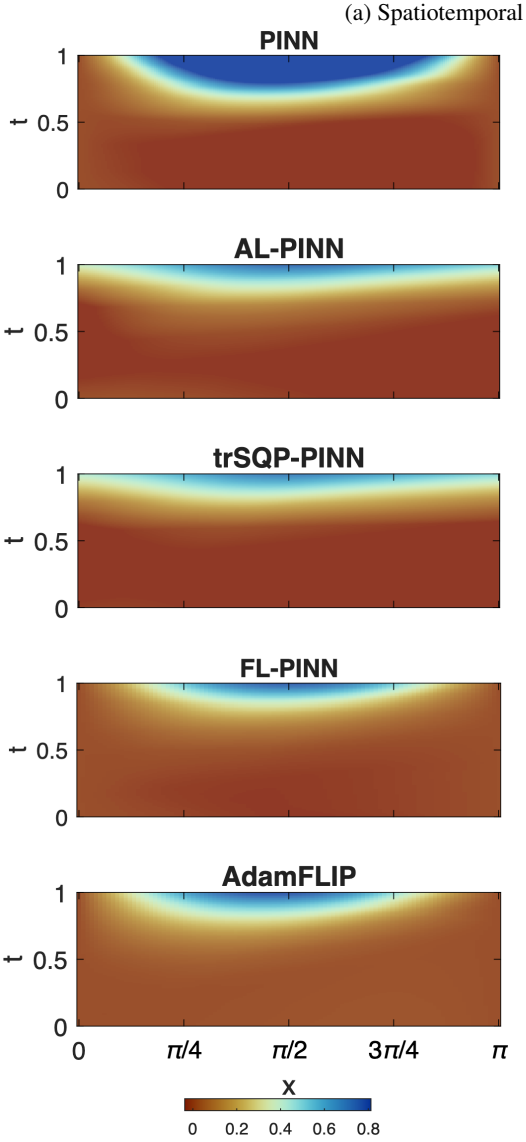
Figure 4 visualizes the spatiotemporal solution fields and corresponding absolute error maps for the forward one-dimensional time-fractional mixed diffusion–wave equation. As shown in Fig. 7(a), the benchmark solution exhibits a smooth high-amplitude region concentrated near the upper time boundary, with the strongest response occurring around the center of the spatial domain. The Standard

PINN prediction in Fig. 7(b) deviates noticeably from the benchmark, particularly in the high-gradient region near the top boundary, where the solution amplitude and spread are misrepresented. This discrepancy is reflected in the error map in Fig. 7(c), where the Standard PINN displays pronounced localized errors concentrated near the upper portion of the domain, indicating difficulty in resolving the sharp spatiotemporal transition governed by the fractional dynamics.

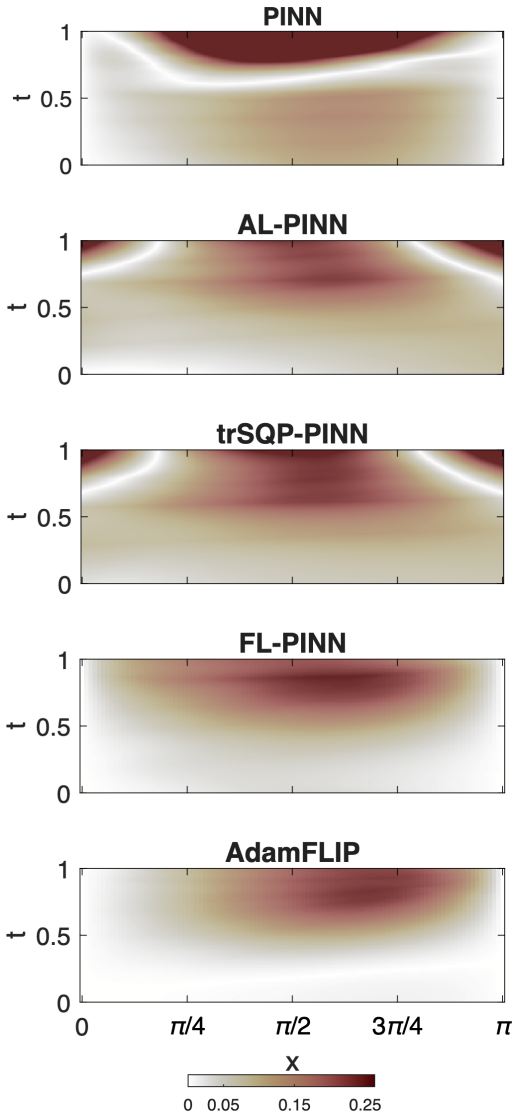
In contrast, FL-PINN produces a solution field that more closely matches the benchmark structure, reducing the mismatch in the dominant high-amplitude region. The corresponding absolute error map shows a substantial reduction in error magnitude and spatial extent compared with the Standard PINN, suggesting that feedback linearization improves optimization stability and accelerates convergence toward a physically consistent solution. AdamFLIP yields the closest agreement with the benchmark among all methods. Its predicted spatiotemporal field preserves both the location and smooth shape of the dominant response region, and its absolute error map exhibits the smallest and most spatially confined residuals. In particular, the remaining errors are primarily limited to a narrow band near the upper time boundary, consistent with a challenging region where the solution gradients are largest and where PINN training typically struggles. Overall, Fig. 4 demonstrates that adaptive feedback linearization not only improves solution accuracy relative to the Standard PINN but also yields more uniform and reduced errors across the full spatiotemporal domain, aligning with the quantitative improvements reported in Table 6.



(a) Spatiotemporal field of Benchmark



(b) Spatiotemporal fields of different models



(c) Absolute errors

Figure 4: Forward time-fractional mixed diffusion–wave equation (1D) results: benchmark spatiotemporal field and model predictions. (a) Reference solution $u(x, t)$. (b) Predicted spatiotemporal fields obtained by Standard PINN, FL-PINN, trSQP-PINN, and AdamFLIP. (c) Corresponding absolute error maps , showing that AdamFLIP yields the smallest and most localized errors, especially near the high-gradient region at large t.

D.3.3 INVERSE

Table 7 reports a quantitative comparison for the inverse fractional PDE problem across Standard PINN, AL-PINN, trSQP-PINN, FL-PINN, and AdamFLIP in terms of the relative L_2 error, constraint errors for the initial condition (IC), boundary condition (BC), physics loss (PL), and data loss (DL), as well as wall-clock time. Overall, AdamFLIP achieves the best performance, attaining the lowest relative L_2 error of 3.95×10^{-2} and the smallest data loss $\mathrm{CE(DL)} = 1.13 \times 10^{-4}$, indicating substantially improved data fitting and more reliable parameter identification compared with all baselines.

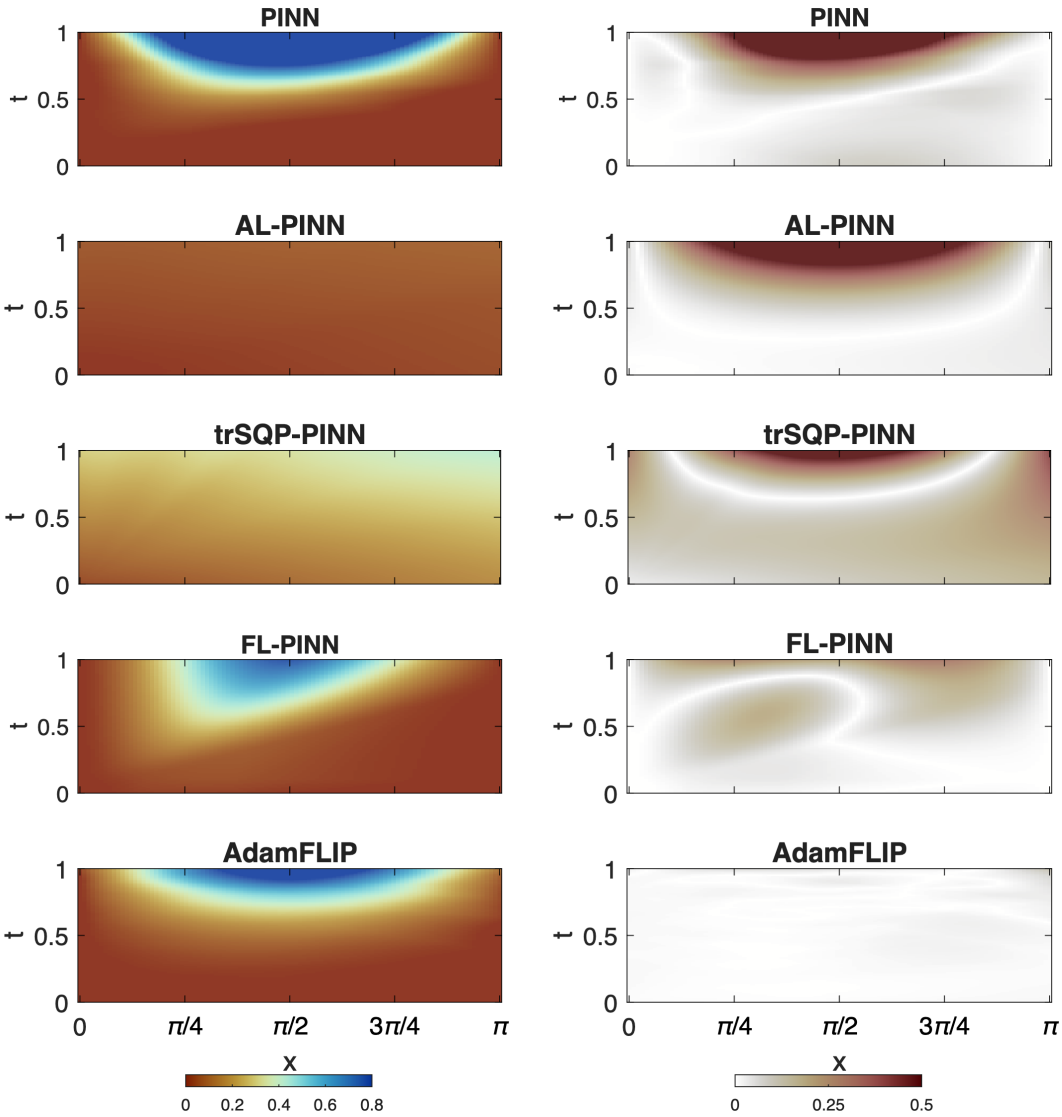
The Standard PINN exhibits moderate solution accuracy (2.22×10^{-1}) but suffers from noticeably larger violations of the boundary condition and physics constraints ($\mathrm{CE(BC)} = 2.27 \times 10^{-2}$, $\mathrm{CE(PL)} = 7.01 \times 10^{-2}$). AL-PINN and trSQP-PINN perform poorly in this inverse setting, producing large relative errors on the order of 10^{-1}–10^{0} and comparatively large data losses, suggesting that they either overemphasize certain constraints or converge to suboptimal parameter values. FL-PINN partially improves constraint enforcement, achieving a small $\mathrm{CE(IC)} = 1.56 \times 10^{-5}$ and $\mathrm{CE(BC)} = 4.08 \times 10^{-4}$, but its overall accuracy remains limited (5.99×10^{-1}). This behavior is consistent with a failure mode in inverse optimization where the method collapses to a trivial solution, yielding low constraint residuals while failing to match observational data. In contrast, AdamFLIP adaptively balances constraints and data fitting during training, preserving strong IC/BC enforcement ($\mathrm{CE(IC)} = 1.77 \times 10^{-5}$, $\mathrm{CE(BC)} = 2.57 \times 10^{-4}$) while dramatically reducing the data loss and improving overall solution accuracy, with a comparable computational cost.

Table 7: Quantitative error comparison for the inverse fractional PDE problem. We report relative L^2 errors, constraint errors (CE) at initial (IC), boundary (BC), physics (Phy), and data loss (DL) collocation points, and training wall-clock time (WT).

Method	**Constraint**	**Rel.** L_2	**CE (IC)**	**CE (BC)**	**CE (Phy)**	**CE (DL)**	**WT (s)**
Standard PINN	✗	2.22×10^{-1}	9.01×10^{-4}	2.27×10^{-2}	7.01×10^{-2}	3.59×10^{-3}	**120**
AL-PINN	✓	8.81×10^{-1}	2.60×10^{-4}	3.74×10^{-3}	4.92×10^{-3}	5.62×10^{-2}	134
trSQP-PINN	✓	6.67×10^{-1}	1.19×10^{-4}	3.59×10^{-3}	$\mathbf{3.78 \times 10^{-3}}$	5.44×10^{-2}	144
FL-PINN	✓	5.99×10^{-1}	$\mathbf{1.56 \times 10^{-5}}$	4.08×10^{-4}	1.82×10^{-1}	2.61×10^{-2}	141
AdamFLIP	✓	$\mathbf{3.95 \times 10^{-2}}$	1.77×10^{-5}	$\mathbf{2.57 \times 10^{-4}}$	6.53×10^{-2}	$\mathbf{1.13 \times 10^{-4}}$	133

Figure 5 illustrates the spatiotemporal solution fields and corresponding absolute error maps for the inverse one-dimensional time-fractional mixed diffusion–wave equation. Compared with the forward case, the inverse problem is more challenging due to the additional requirement of parameter identification while simultaneously fitting observational data. The Standard PINN solution exhibits noticeable distortion in the upper time region, where the dominant response is expected. The predicted field appears overly smoothed and shifted, leading to large localized errors near the top boundary, as highlighted in the absolute error map. This behavior reflects the tendency of unconstrained optimization to converge to suboptimal parameter values that partially satisfy the PDE residual but fail to accurately reconstruct the underlying dynamics.

FL-PINN improves constraint enforcement and produces a solution with a clearer dominant structure compared to Standard PINN. However, the reconstructed field still deviates from the expected spatiotemporal pattern, particularly in the central region, where the amplitude and curvature are misrepresented. The corresponding error map reveals persistent residual bands, indicating that strict constraint minimization alone does not guarantee accurate parameter recovery in the inverse setting. AdamFLIP yields a solution field that most closely matches the expected physical structure, accurately capturing both the spatial localization and temporal evolution of the dominant response. The absolute error map is nearly uniform and close to zero across the entire domain, with only minor residuals near regions of steep gradient. This demonstrates that adaptive constraint balancing effectively mitigates the degeneracy observed in pure feedback linearization and prevents convergence to trivial or physically inconsistent parameter values. Overall, Fig. 5 visually confirms the quantitative results reported in Table 7: AdamFLIP provides substantially improved accuracy, more reliable parameter identification, and significantly reduced global error compared to Standard PINN and FL-PINN in the inverse fractional PDE setting.



(a) Spatiotemporal fields of different models

(b) Absolute errors

Figure 5: Inverse time-fractional mixed diffusion–wave equation (1D) results: spatiotemporal reconstructions and absolute errors. (a) Predicted solution fields $u(x,t)$ obtained by Standard PINN, FL-PINN, trSQP-PINN, and AdamFLIP. (b) Corresponding absolute error maps, showing that AdamFLIP produces the smallest and most uniform errors across the spatiotemporal domain, whereas Standard PINN and FL-PINN exhibit larger localized error bands, especially near the upper-time region.

D.4 2D Heat Equation

We next consider the classical two-dimensional heat equation on the unit square $\Omega = (0,1)^2$ with homogeneous Dirichlet boundary conditions:

$$\frac{\partial u(x,y,t)}{\partial t} - \nu\,\Delta u(x,y,t) \;=\; f(x,y,t), \qquad (x,y) \in \Omega,\ t \in [0,1], \tag{25}$$

$$u(x,y,t) = 0, \quad \forall x \in \partial\Omega, \tag{26}$$

$$u(x,y,0) = u_0(x,y), \tag{27}$$

where Δ is the Laplacian operator and $\partial\Omega$ denotes the boundary of the domain. To enable quantitative comparison, we consider the following analytical solution

$$u(x, y, t) = \sin(\pi x)\sin(\pi y)\,\mathrm{e}^{-2\pi^2\kappa t}, \tag{28}$$

with no forcing, i.e., $f \equiv 0$.

D.4.1 PINN'S SETTINGS

Neural network architecture To ensure a fair and consistent comparison across all different models, we employ an identical deep neural network architecture. Specifically, each model utilizes a fully connected network consisting of 6 hidden layers with 64 neurons per layer. During the training phase, each model was trained until its loss dropped below the predefined threshold, ϵ.

Training and testing data In the training stage, the initial and boundary condition losses are evaluated using data randomly sampled in space at the initial time $t = 0$ and in time along the spatial boundaries ($(x, y) \in \partial\Omega$). Specifically, we sample $N_{\text{ic}} = 500$ points for the initial condition and $N_{\text{bc}} = 500$ points for the boundary conditions. Additionally, we sample $N_f = 2000$ collocation points within the interior of the spatio-temporal domain to compute the residual loss. During the testing stage, the evaluation data are sampled over the spatiotemporal domain $\Omega \times [0, 1]$. The data is generated on a uniform grid with a spatial mesh size of $\Delta x = \Delta y = 0.01$ and a temporal step size of $\Delta t = 0.01$. All samples are drawn uniformly at random from their respective domains. The homogeneous Dirichlet condition $u = 0$ is imposed at all boundary points.

D.4.2 FORWARD PROBLEM

In this section, we consider the forward problem setting where the physics loss is fully specified and does not involve any unknown parameters to be inferred which is same as in Section D.2.2. Table 8 presents a quantitative comparison of standard PINN (Adam), AL-PINN, trSQP-PINN, FL-PINN, and AdamFLIP for the two-dimensional heat equation. The performance is measured in terms of the relative L^2 error and the constraint errors (CE) for the initial condition (IC), boundary condition (BC), and physics loss (PL). Among all models considered, ***AdamFLIP*** performs best, achieving a relative L^2 error that is approximately one order of magnitude smaller than that of the standard PINN. Furthermore, AdamFLIP yields the lowest constraint errors for both the BC and PL components, with its PL constraint error being roughly one-third of that achieved by the standard PINN.

In contrast to the Burgers' equation results, ***trSQP-PINN*** struggles significantly in this setting, exhibiting a relative L^2 error that is more than one order of magnitude higher than the other models. This indicates that the optimization advantages of trSQP-PINN do not necessarily translate to improved accuracy for the heat equation. It is also worth noting that all models yield comparable wall-clock times that implies the improved performance of ***AdamFLIP*** is achieved without introducing prohibitive computational cost, as its training time remains within approximately one-third of the baseline PINN's overhead.

Figure 6 shows the absolute error distributions at the two different time steps. The standard PINN exhibits a noticeable error distribution that intensifies near the boundaries and the center of the domain as the system evolves toward $t = 1.0$. While maintaining the overall physical structure, the FL-PINN shows a concentrated region of higher absolute error near the domain center ($x \approx 0.5, y \approx 0.5$). Notably, the peak error for this method is approximately one order of magnitude larger than that of the best-performing model. In contrast, the ***AdamFLIP*** demonstrates the highest precision among all methods. The absolute error maps for AdamFLIP are significantly clearer, with error magnitudes consistently remaining near the lower bound of the scale across both time steps, which is consistent with the results in Table 8.

Table 8: Quantitative error comparison for the 2D heat equation (forward problem). We report relative L^2 errors, constraint errors (CE) at initial (IC), boundary (BC), and physics (Phy) collocation points, and training wall-clock time (WT).

Method	**Constraint**	**Rel.** L_2	**CE (IC)**	**CE (BC)**	**CE (Phy)**	**WT (s)**
Standard PINN	✗	3.43×10^{-2}	1.20×10^{-4}	1.56×10^{-4}	1.82×10^{-3}	**261**
AL-PINN	✓	3.09×10^{-2}	$\mathbf{3.65 \times 10^{-5}}$	7.89×10^{-5}	7.41×10^{-4}	364
trSQP-PINN	✓	6.60×10^{-1}	1.22×10^{-1}	2.62×10^{-2}	1.07×10^{-2}	1260
FL-PINN	✓	9.49×10^{-3}	1.13×10^{-4}	1.70×10^{-4}	7.35×10^{-4}	354
AdamFLIP	✓	$\mathbf{7.53 \times 10^{-3}}$	5.59×10^{-5}	$\mathbf{6.35 \times 10^{-5}}$	$\mathbf{2.00 \times 10^{-4}}$	336

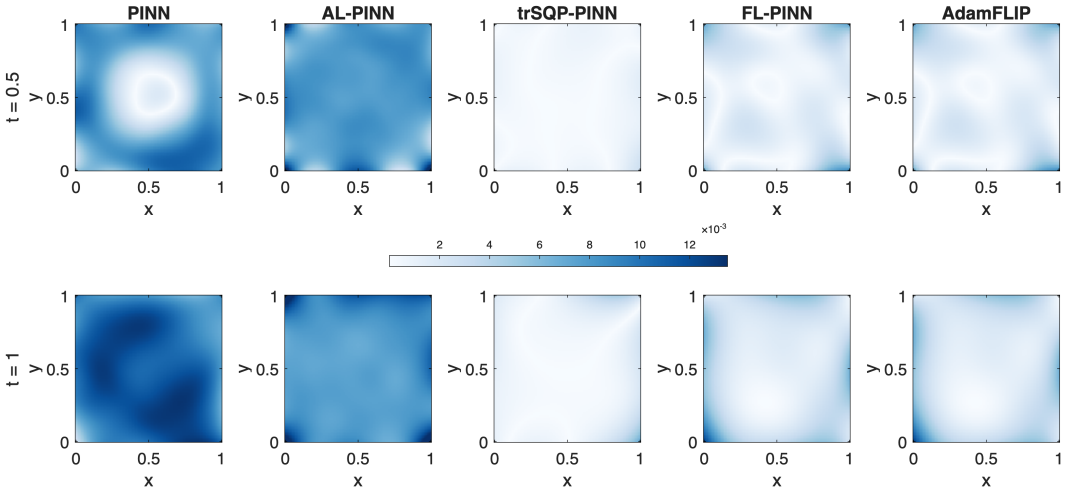


Figure 6: Comparison of spatiotemporal solutions and absolute errors in forward problem for the 2d Heat equation: Standard PINN, AL-PINN, trSQP-PINN, FL-PINN, and AdamFLIP.

D.4.3 Inverse Problem

In the inverse problem setting, we consider the thermal diffusivity ν in Eq. (25) to be an unknown parameter, hereafter denoted as κ, which leads to the following formulation:

$$\frac{\partial u(x,y,t)}{\partial t} - \kappa\, \Delta u(x,y,t) \;=\; f(x,y,t), \qquad (x,y) \in \Omega,\ t \in [0,1], \tag{29}$$

$$u(x,y,t) = 0, \quad \forall x \in \partial\Omega, \tag{30}$$

$$u(x,y,0) = u_0(x,y), \tag{31}$$

Consistent with the treatement in Section D.2.3, κ is parameterized as learnable scalars, which are co-optimized with the neural network weights by minimizing the loss function same as in the forward problem. In this setup, for all models we initialize the parameter estimates at $\kappa = 1$, whereas their ground truth values are $\kappa = 0.1$.

Table 4 shows the learned PDE parameters and their L^1 error compared to the ground truth for the Burgers' equation in the inverse problem setting. It is shown that among all models, AdamFLIP demonstrates the highest accuracy by achieving the lowest L^1 errors in estimating κ. Specifically, it reduces the estimation error of κ by nearly half of magnitude compared to the next best model, standard PINN.

Table 10 presents a quantitative error comparison of different models for the two-dimensional heat equation in the inverse problem setting. Beyond the standard constraint errors (CE) for the initial condition (IC), boundary condition (BC), and physics loss (PL), we also report the data loss error (CE(DL)) used to infer the unknown diffusivity κ. Consistent with the forward problem results, *AdamFLIP* exhibits the most robust performance, achieving a relative L^2 error that is approximately half that of the standard PINN. Furthermore, the data loss error (CE(DL)) for AdamFLIP is roughly

Table 9: Parameter estimation results for the two-dimensional heat equation in the inverse problem setting. The exact PDE parameters are $\kappa = 0.1$.

Method	$\hat{\kappa}$	$\lvert\hat{\kappa} - \kappa\rvert$
Standard PINN	0.0991	0.0009
AL-PINN	0.1020	0.0020
trSQP-PINN	0.9840	0.8840
FL-PINN	0.0933	0.0067
AdamFLIP	**0.0995**	**0.0005**

Table 10: Quantitative error comparison for the 2D heat equation (inverse problem). We report relative L^2 errors, constraint errors (CE) at initial (IC), boundary (BC), physics (Phy), and data loss (DL) collocation points, and training wall-clock time (WT).

Method	**Constraint**	**Rel.** L_2	**CE (IC)**	**CE (BC)**	**CE (Phy)**	**CE (DL)**	**WT (s)**
Standard PINN	✗	2.90×10^{-2}	3.06×10^{-5}	1.19×10^{-4}	2.79×10^{-4}	5.22×10^{-5}	297
AL-PINN	✓	3.98×10^{-2}	1.71×10^{-4}	3.07×10^{-4}	2.29×10^{-4}	9.18×10^{-5}	361
trSQP-PINN	✓	6.42×10^{-1}	4.81×10^{-4}	4.57×10^{-3}	3.73×10^{-3}	2.59×10^{-2}	1300
FL-PINN	✓	4.06×10^{-2}	1.80×10^{-5}	$\mathbf{8.73 \times 10^{-6}}$	7.44×10^{-5}	8.81×10^{-5}	360
AdamFLIP	✓	$\mathbf{1.59 \times 10^{-2}}$	$\mathbf{1.43 \times 10^{-5}}$	1.03×10^{-5}	9.13×10^{-5}	$\mathbf{1.51 \times 10^{-5}}$	346

one-third of the error observed in the standard baseline, indicating a significantly more accurate reconstruction of the underlying field from observed data.

It is also noteworthy that *AdamFLIP* achieves the lowest initial condition error (CE(IC)), which is approximately half the magnitude of the error produced by the standard PINN. In contrast, *trSQP-PINN* continues to struggle in this setting, yielding a relative L^2 error and a data loss error that are more than one order of magnitude higher than those of the other models. Regarding computational efficiency, all frameworks yield comparable wall-clock times (WT) though the training time for AdamFLIP is approximately one-fifth higher than the standard PINN.

These results are also confirmed in Figure 7. While all methods capture the general diffusion dynamics (panel a.), the spatial distributions of absolute error at different time instances (panel. b) reveal that *AdamFLIP* yields a much smaller error distribution in space that effectively mitigating the concentrated interior errors observed in the standard PINN and FL-PINN models.

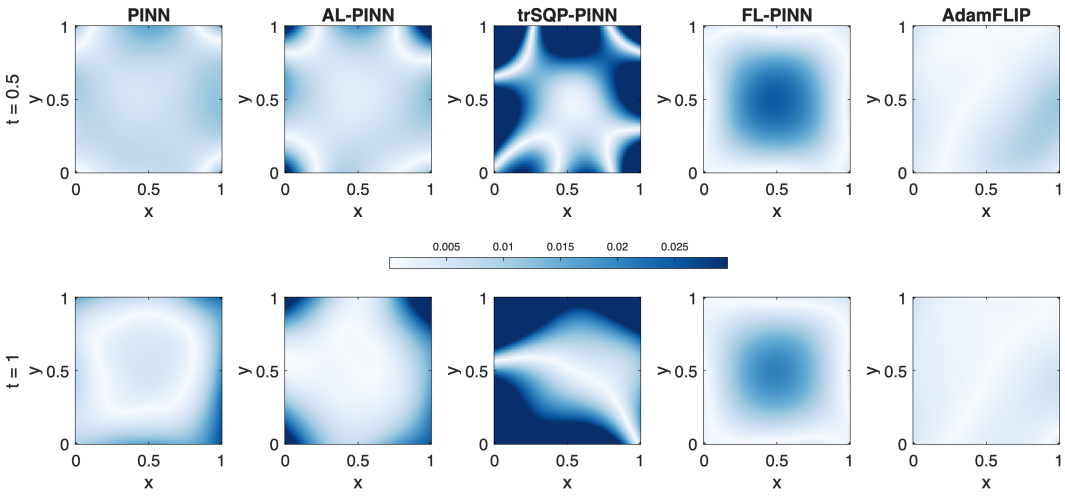


Figure 7: Inverse 2D heat equation results: comparison of absolute errors fields for each method at the same time instants,for PINN, FL-PINN, trSQP-PINN, and AdamFLIP. It shows that AdamFLIP yields the smallest and most uniformly distributed errors across the spatial domain.

D.5 2D INCOMPRESSIBLE NAVIER–STOKES

We evaluate all methods on the 2D incompressible Navier–Stokes equations, a coupled nonlinear system that is substantially more challenging than the scalar PDEs considered above. The governing equations on $\Omega = [0, 2\pi]^2$, $t \in [0, T]$ are

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{32}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \tag{33}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{34}$$

where $\mathbf{u} = (u, v)$ is the velocity field, p is the pressure, and $\nu = 0.01$. We use the Taylor–Green vortex as the benchmark, which admits the following closed-form analytical solution:

$$u(x, y, t) = -\cos(x)\sin(y)\, e^{-2\nu t}, \tag{35}$$

$$v(x, y, t) = \sin(x)\cos(y)\, e^{-2\nu t}, \tag{36}$$

$$p(x, y, t) = -\tfrac{1}{4}\left(\cos(2x) + \cos(2y)\right) e^{-4\nu t}. \tag{37}$$

D.5.1 NUMERICAL SETTINGS

Neural network architecture To ensure a fair and consistent comparison across all models, we employ an identical fully connected network consisting of 6 hidden layers with 64 neurons per layer. The network takes (x, y, t) as input and predicts the velocity-pressure field (u, v, p). During training, each model was trained until its loss dropped below the predefined threshold, ϵ.

Training and testing data In the training stage, the initial condition loss is evaluated at $N_{\text{ic}} = 500$ randomly sampled spatial points at $t = 0$. The boundary condition loss is evaluated at $N_{\text{bc}} = 500$ points sampled along the spatial boundaries over time. The physics loss, comprising the two momentum equations (32)–(33) and the incompressibility constraint (34), is computed on $N_f = 2000$ collocation points randomly sampled from the interior of the spatiotemporal domain. During the testing stage, the evaluation data are generated on a uniform grid over $\Omega \times [0, T]$ with spatial mesh size $\Delta x = \Delta y = 0.05$ and temporal step size $\Delta t = 0.1$.

We consider the forward problem setting where the viscosity ν is known and the objective is to accurately approximate the velocity and pressure fields. Table 11 presents a quantitative comparison of Standard PINN (Adam), AL-PINN, trSQP-PINN, FL-PINN, and AdamFLIP for the Navier–Stokes equations. The performance is measured in terms of the mean relative L^2 errors of the velocity components u and v, the constraint errors (CE) for the initial condition (IC), boundary condition (BC), and physics residual (Phy), and the training wall-clock time (WT).

Among all models, AdamFLIP achieves the best performance across all accuracy metrics. It attains mean relative L_2 errors of 1.18×10^{-2} and 1.06×10^{-2} for the u and v velocity components, respectively—roughly $3\times$ lower than Standard PINN (3.62×10^{-2} and 3.48×10^{-2}) and substantially better than all constrained baselines. AdamFLIP also yields the lowest constraint errors across IC, BC, and physics residual, demonstrating that the feedback linearization mechanism effectively balances multi-component constraint satisfaction in this coupled system.

Notably, trSQP-PINN fails catastrophically on this problem, producing L_2 errors on the order of 10^0 and a physics CE of 1.29×10^1. This is consistent with the sensitivity of trust-region SQP methods to the increased dimensionality and nonlinearity of the Navier–Stokes system. FL-PINN also struggles relative to the scalar PDE benchmarks, with L_2 errors (7.70×10^{-2}, 8.13×10^{-2}) that are worse than Standard PINN, suggesting that naive feedback linearization without adaptive moment estimation is insufficient for this more complex problem. AL-PINN performs comparably to Standard PINN on the v-component but shows degraded IC/BC enforcement.

In terms of computational cost, Standard PINN is the fastest at 155 s, while AdamFLIP requires 335 s—roughly $2\times$ the baseline, which is a larger overhead than observed on the scalar PDEs. This additional cost is attributable to the Jacobian computations required for the multi-output, multi-constraint structure of the Navier–Stokes system. Nevertheless, the accuracy improvement is substantial.

Table 11: Quantitative error comparison for the 2D Navier–Stokes equations (forward problem). We report mean relative L^2 errors for velocity components u and v, constraint errors (CE) at initial (IC), boundary (BC), and physics (Phy) collocation points.

Method	**Constraint**	**Mean** $L_2(u)$	**Mean** $L_2(v)$	**CE (IC)**	**CE (BC)**	**CE (Phy)**
Standard PINN	✗	3.62×10^{-2}	3.48×10^{-2}	2.15×10^{-5}	1.02×10^{-4}	6.11×10^{-4}
AL-PINN	✓	5.19×10^{-2}	3.69×10^{-2}	1.10×10^{-3}	8.13×10^{-4}	8.56×10^{-4}
trSQP-PINN	✓	1.96×10^{0}	9.68×10^{-1}	7.82×10^{-4}	3.56×10^{-4}	1.29×10^{1}
FL-PINN	✓	7.70×10^{-2}	8.13×10^{-2}	1.98×10^{-3}	2.72×10^{-3}	1.08×10^{-2}
AdamFLIP	✓	$\mathbf{1.18 \times 10^{-2}}$	$\mathbf{1.06 \times 10^{-2}}$	$\mathbf{1.26 \times 10^{-5}}$	$\mathbf{4.86 \times 10^{-5}}$	$\mathbf{2.53 \times 10^{-4}}$

Figure 8 presents the pointwise absolute error maps $|u - \hat{u}|$ and $|v - \hat{v}|$ at $t = 1$ for all five methods. Standard PINN exhibits moderate, spatially distributed errors that are roughly uniform across the domain. AL-PINN shows a similar error pattern with slightly larger magnitudes. trSQP-PINN produces large-scale errors spanning the entire domain, consistent with its failure to converge to a physically meaningful solution. FL-PINN displays concentrated error regions, particularly near the vortex cores where the velocity gradients are steepest. In contrast, AdamFLIP achieves the smallest and most spatially uniform errors for both velocity components, with residual errors primarily confined to narrow bands near regions of maximum shear. These qualitative observations are fully consistent with the quantitative results reported in Table 11.

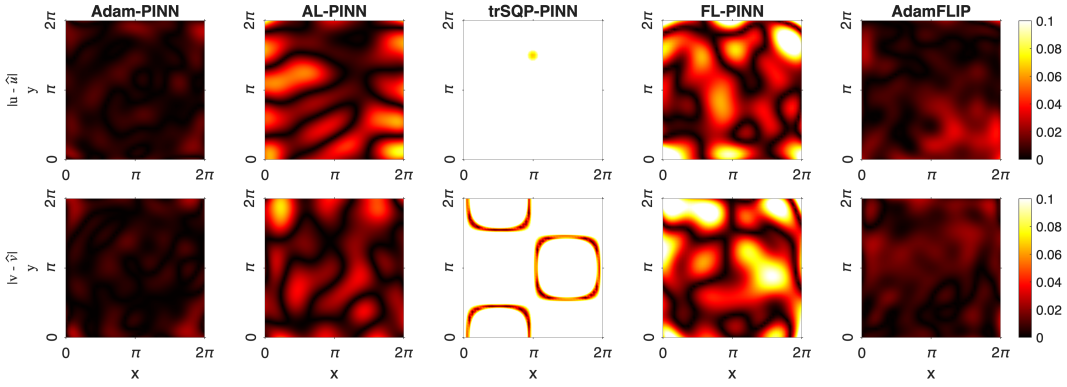


Figure 8: Pointwise absolute error $|u - \hat{u}|$ (top row) and $|v - \hat{v}|$ (bottom row) at $t = 1$ for the 2D Navier–Stokes equations. From left to right: Standard PINN, AL-PINN, trSQP-PINN, FL-PINN, and AdamFLIP. AdamFLIP produces the smallest and most spatially uniform errors across both velocity components.

E Further Ablations

Table 12 studies the effect of the momentum parameters (β_1, β_2) under fixed $K_p = 1000$ and $K_i = 0.01$. Overall, introducing stronger first- and second-moment smoothing significantly reduces the relative L_2 error compared with the weak-momentum baseline. The setting $(\beta_1, \beta_2) = (0.99, 0.9999)$ achieves the lowest mean relative L_2 error, indicating that aggressive momentum smoothing can improve the final solution accuracy. However, this configuration may be more sensitive to random initialization and training dynamics. In comparison, $(\beta_1, \beta_2) = (0.95, 0.999)$ also yields a very low relative L_2 error and provides a competitive balance between accuracy and stability. Therefore, we use $(\beta_1, \beta_2) = (0.95, 0.999)$ as the default configuration in our experiments.

Table 13: Ablation study on the proportional penalty gain K_p. We report the physics loss as the objective and the initial condition (IC) and boundary condition (BC) losses as constraints. Results are averaged over three random seeds and reported as mean $\pm$ standard deviation. Lower values are better. The selected default configuration is highlighted in bold.

K_p	Objective	Constraints	
	Physics Loss	IC Loss	BC Loss
10	$3.83\times10^{-4} \pm 5.66\times10^{-5}$	$9.83\times10^{-1} \pm 1.17\times10^{-1}$	$8.48\times10^{-2} \pm 4.87\times10^{-2}$
50	$4.71\times10^{-3} \pm 5.96\times10^{-4}$	$3.88\times10^{-2} \pm 5.31\times10^{-3}$	$3.55\times10^{-2} \pm 7.72\times10^{-3}$
100	$7.34\times10^{-3} \pm 4.68\times10^{-4}$	$2.54\times10^{-2} \pm 1.33\times10^{-2}$	$2.95\times10^{-2} \pm 1.48\times10^{-2}$
200	$1.01\times10^{-2} \pm 1.07\times10^{-3}$	$3.55\times10^{-2} \pm 3.08\times10^{-2}$	$2.68\times10^{-2} \pm 1.49\times10^{-2}$
500	$1.12\times10^{-2} \pm 6.52\times10^{-4}$	$2.27\times10^{-2} \pm 1.76\times10^{-2}$	$1.89\times10^{-2} \pm 1.29\times10^{-2}$
1000	$\mathbf{1.30\times10^{-2} \pm 1.02\times10^{-3}}$	$\mathbf{1.26\times10^{-2} \pm 4.99\times10^{-3}}$	$\mathbf{1.31\times10^{-2} \pm 4.80\times10^{-3}}$
2000	$1.43\times10^{-2} \pm 1.56\times10^{-3}$	$8.16\times10^{-3} \pm 6.20\times10^{-4}$	$9.25\times10^{-3} \pm 1.50\times10^{-3}$
5000	$2.25\times10^{-2} \pm 3.38\times10^{-3}$	$1.10\times10^{-2} \pm 4.69\times10^{-3}$	$9.28\times10^{-3} \pm 4.49\times10^{-3}$

Table 12: Ablation study on the momentum parameters (β_1, β_2) with fixed $K_p = 1000$ and $K_i = 0.01$. We report the relative L_2 error averaged over three random seeds. Lower values are better. The best result is highlighted in bold.

β_1	β_2	Rel. L_2 Error
0.00	0.9000	$2.15\times10^{-1} \pm 3.34\times10^{-2}$
0.50	0.9900	$1.00\times10^{-1} \pm 1.28\times10^{-2}$
0.90	0.9900	$1.33\times10^{-1} \pm 1.56\times10^{-2}$
0.90	0.9990	$1.10\times10^{-1} \pm 1.89\times10^{-2}$
0.95	0.9990	$8.48\times10^{-3} \pm 5.01\times10^{-2}$
0.99	0.9999	$\mathbf{7.53\times10^{-3} \pm 6.99\times10^{-3}}$

Table 13 studies the effect of the proportional penalty gain K_p on the objective loss and constraint satisfaction. Here, the physics loss is treated as the optimization objective, while the initial condition and boundary condition losses are treated as constraints. As K_p increases, both constraint losses decrease substantially, indicating that a larger proportional gain improves enforcement of the initial and boundary conditions. However, this comes at the cost of a larger physics loss, revealing a clear trade-off between minimizing the PDE residual objective and satisfying the constraints. Small values such as $K_p = 10$ achieve the lowest physics loss but fail to enforce the constraints, especially the initial condition loss. In contrast, very large values such as $K_p = 5000$ provide strong boundary constraint satisfaction but significantly increase the physics loss. The setting $K_p = 1000$ provides a balanced trade-off, achieving much lower IC and BC losses than small-K_p settings while avoiding the larger objective degradation observed at $K_p = 5000$. Therefore, we select $K_p = 1000$ as the default configuration.

F Proof of Theorem 1

We prove Theorem 1 using the exact penalty merit function

$$\Psi_\rho(\theta) := f(\theta) + \rho\|h(\theta)\|_1.$$

Throughout the proof, all constants are uniform over $\theta \in \mathcal{C}$ and over the admissible diagonal metrics generated by Algorithm 2.

Lemma 1 (Uniform metric bounds and finite metric variation)**.** *Under Assumption 1, the diagonal metrics D_t satisfy*

$$\frac{1}{G+\delta} I \preceq D_t \preceq \frac{1}{\delta} I \qquad \forall t.$$

Moreover, if $\Delta_t := \|D_{t+1} - D_t\|$, *then*

$$\sum_{t=1}^{\infty} \Delta_t \le \frac{n}{\delta},$$

and therefore

$$\sum_{t=1}^{\infty} \eta_t \Delta_t \le \frac{\eta_1 n}{\delta}.$$

Proof. Since $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^{\circ 2}$ and $v_0 = 0$, the bias-corrected raw second moment $\tilde{v}_t = v_t/(1-\beta_2^t)$ is a coordinatewise convex combination of $g_1^{\circ 2}, \dots, g_t^{\circ 2}$. Hence, by Assumption 1, for every coordinate j,

$$0 \le \tilde{v}_{t,j} \le G^2.$$

The monotone second-moment estimate is defined by

$$\bar{v}_t = \max_{\text{elem}}\{\bar{v}_{t-1}, \tilde{v}_t\}.$$

Thus $0 \le \bar{v}_{t,j} \le G^2$ for all t, j. Since

$$D_t = \operatorname{diag}\left(\frac{1}{\sqrt{\bar{v}_{t-1}} + \delta}\right),$$

we have, for every coordinate j,

$$\frac{1}{G+\delta} \le \frac{1}{\sqrt{\bar{v}_{t-1,j}} + \delta} \le \frac{1}{\delta}.$$

This proves the uniform metric bounds.

It remains to prove finite variation. For each coordinate j, define

$$d_{t,j} := \frac{1}{\sqrt{\bar{v}_{t-1,j}} + \delta}.$$

Since $\bar{v}_{t,j}$ is coordinatewise nondecreasing in t, the sequence $d_{t,j}$ is nonincreasing in t. Moreover, $0 < d_{t,j} \le 1/\delta$. Therefore,

$$\sum_{t=1}^{\infty} |d_{t+1,j} - d_{t,j}| = \sum_{t=1}^{\infty} (d_{t,j} - d_{t+1,j}) \le d_{1,j} \le \frac{1}{\delta}.$$

Since D_t is diagonal,

$$\|D_{t+1} - D_t\| \le \|D_{t+1} - D_t\|_F \le \sum_{j=1}^{n} |d_{t+1,j} - d_{t,j}|.$$

Summing over t gives

$$\begin{aligned}\sum_{t=1}^{\infty} \Delta_t &\le \sum_{j=1}^{n} \sum_{t=1}^{\infty} |d_{t+1,j} - d_{t,j}| \\ &\le \frac{n}{\delta}.\end{aligned}$$

Finally, since $\eta_t \le \eta_1$,

$$\sum_{t=1}^{\infty} \eta_t \Delta_t \le \eta_1 \sum_{t=1}^{\infty} \Delta_t \le \frac{\eta_1 n}{\delta}.$$

This proves the lemma. □

Lemma 2 (Momentum tracking error)**.** *Let $e_t := \hat{m}_t - g_t$. Then there exists $C_e > 0$ such that, for every $T \geq 1$,*

$$\sum_{t=1}^{T} \eta_t \|e_t\| \leq C_e \left(\sum_{t=1}^{T} \eta_t^2 + \sum_{t=1}^{T} \eta_t \Delta_t \right).$$

Consequently, by Lemma 1,

$$\sum_{t=1}^{T} \eta_t \|e_t\| \leq C_e \left(\sum_{t=1}^{T} \eta_t^2 + \frac{\eta_1 n}{\delta} \right).$$

Proof. The bias-corrected first moment can be written as

$$\hat{m}_t = \sum_{i=1}^{t} a_{t,i} g_i, \qquad a_{t,i} := \frac{(1-\beta_1)\beta_1^{t-i}}{1-\beta_1^t}.$$

The weights satisfy $a_{t,i} \geq 0$ and $\sum_{i=1}^{t} a_{t,i} = 1$. Hence

$$e_t = \hat{m}_t - g_t = \sum_{i=1}^{t} a_{t,i}(g_i - g_t).$$

By Assumption 1 and Lemma 1, the map $(\theta, D) \mapsto g_D(\theta)$ is Lipschitz on the compact admissible set

$$\mathcal{C} \times \left\{ D : \frac{1}{G+\delta} I \preceq D \preceq \frac{1}{\delta} I \right\}.$$

Let $L_g > 0$ be a corresponding Lipschitz constant. Then, for $i \leq t$,

$$\|g_i - g_t\| \leq L_g \big(\|\theta_i - \theta_t\| + \|D_i - D_t\| \big).$$

Since $\hat{m}_t$ is a convex combination of previous g_i's and $\|g_i\| \leq G$, we have $\|\hat{m}_t\| \leq G$. Together with $\|D_t\| \leq 1/\delta$, this gives

$$\|\theta_{t+1} - \theta_t\| = \eta_t \|D_t \hat{m}_t\| \leq \frac{G}{\delta} \eta_t.$$

Therefore, for $i \leq t$,

$$\|\theta_i - \theta_t\| \leq \frac{G}{\delta} \sum_{k=i}^{t-1} \eta_k, \qquad \|D_i - D_t\| \leq \sum_{k=i}^{t-1} \Delta_k.$$

Then

$$\|g_i - g_t\| \leq L_g \frac{G}{\delta} \sum_{k=i}^{t-1} (\eta_k + \Delta_k).$$

Thus

$$\begin{aligned} \|e_t\| &\leq L_g \frac{G}{\delta} \sum_{i=1}^{t} a_{t,i} \sum_{k=i}^{t-1} (\eta_k + \Delta_k) \\ &= L_g \frac{G}{\delta} \sum_{k=1}^{t-1} (\eta_k + \Delta_k) \sum_{i=1}^{k} a_{t,i}. \end{aligned}$$

Using the explicit form of $a_{t,i}$,

$$\sum_{i=1}^{k} a_{t,i} = \frac{\beta_1^{t-k} - \beta_1^t}{1-\beta_1^t} \leq \frac{\beta_1^{t-k}}{1-\beta_1}.$$

Therefore,

$$\|e_t\| \le \frac{L_g G}{(1-\beta_1)\delta} \sum_{k=1}^{t-1} \beta_1^{t-k}(\eta_k + \Delta_k).$$

Multiplying by η_t and summing from $t = 1$ to T yields

$$\begin{aligned}\sum_{t=1}^{T} \eta_t \|e_t\| &\le \frac{L_g G}{(1-\beta_1)\delta} \sum_{t=1}^{T} \eta_t \sum_{k=1}^{t-1} \beta_1^{t-k}(\eta_k + \Delta_k) \\ &= \frac{L_g G}{(1-\beta_1)\delta} \sum_{k=1}^{T-1} (\eta_k + \Delta_k) \sum_{t=k+1}^{T} \eta_t \beta_1^{t-k}.\end{aligned}$$

Since η_t is nonincreasing, for $t \ge k+1$ we have $\eta_t \le \eta_k$. Hence

$$\sum_{t=k+1}^{T} \eta_t \beta_1^{t-k} \le \eta_k \sum_{\ell=1}^{\infty} \beta_1^{\ell} = \eta_k \frac{\beta_1}{1-\beta_1}.$$

Therefore,

$$\begin{aligned}\sum_{t=1}^{T} \eta_t \|e_t\| &\le \frac{L_g G \beta_1}{(1-\beta_1)^2 \delta} \sum_{k=1}^{T-1} \eta_k (\eta_k + \Delta_k) \\ &\le C_e \left(\sum_{t=1}^{T} \eta_t^2 + \sum_{t=1}^{T} \eta_t \Delta_t \right),\end{aligned}$$

where one may take

$$C_e := \frac{L_g G \beta_1}{(1-\beta_1)^2 \delta}.$$

The second claim follows immediately from Lemma 1. This proves the lemma. □

Proof of Theorem 1. By Lemma 1, the matrices D_t are uniformly positive definite and uniformly bounded. Since $J_h(\theta)$ has full row rank on $\mathcal{C}$, the matrices

$$M_t = J_h(\theta_t) D_t J_h(\theta_t)^\top$$

are uniformly nonsingular. Therefore, $\lambda_t^\dagger$, r_t, and g_t are all uniformly bounded on the compact admissible set. In particular, define

$$G_f := \sup_{\theta \in \mathcal{C}} \|\nabla f(\theta)\|, \qquad J_1 := \sup_{\theta \in \mathcal{C}} \|J_h(\theta)\|_{2\to 1}, \qquad \Lambda := \sup_{t \ge 1} \|\lambda_t^\dagger\|_\infty.$$

All three constants are finite.

Let $L_f > 0$ be a Lipschitz constant of ∇f on the neighborhood specified in Assumption 1. Let $L_{h,1} > 0$ satisfy

$$\|h(\theta + s) - h(\theta) - J_h(\theta) s\|_1 \le \frac{L_{h,1}}{2} \|s\|^2$$

whenever the line segment between θ and $\theta + s$ lies in that neighborhood.

Let $e_t := \hat{m}_t - g_t$. The update can be written as

$$\theta_{t+1} = \theta_t - \eta_t D_t g_t - \eta_t D_t e_t.$$

We first estimate the objective decrease. By smoothness of f,

$$f(\theta_{t+1}) \le f(\theta_t) + \langle \nabla f(\theta_t), \theta_{t+1} - \theta_t \rangle + \frac{L_f}{2} \|\theta_{t+1} - \theta_t\|^2$$

$$= f(\theta_t) - \eta_t \langle \nabla f(\theta_t), D_t g_t \rangle - \eta_t \langle \nabla f(\theta_t), D_t e_t \rangle + \frac{L_f}{2} \eta_t^2 \|D_t \hat{m}_t\|^2.$$

Using $\|\nabla f(\theta_t)\| \le G_f$, $\|D_t\| \le 1/\delta$, and $\|\hat{m}_t\| \le G$, we obtain

$$f(\theta_{t+1}) \le f(\theta_t) - \eta_t \langle \nabla f(\theta_t), D_t g_t \rangle + \frac{\eta_t G_f}{\delta} \|e_t\| + \frac{L_f G^2}{2\delta^2} \eta_t^2. \tag{38}$$

We now simplify the leading inner product. By definition,

$$r_t = \nabla f(\theta_t) + J_h(\theta_t)^\top \lambda_t^\dagger,$$

so

$$\nabla f(\theta_t) = r_t - J_h(\theta_t)^\top \lambda_t^\dagger.$$

Therefore,

$$\begin{aligned} \langle \nabla f(\theta_t), D_t g_t \rangle &= \langle r_t, D_t g_t \rangle - \left\langle J_h(\theta_t)^\top \lambda_t^\dagger, D_t g_t \right\rangle \\ &= \langle r_t, D_t g_t \rangle - \left\langle \lambda_t^\dagger, J_h(\theta_t) D_t g_t \right\rangle. \end{aligned}$$

We have $J_h(\theta_t) D_t r_t = 0$ by construction. Moreover,

$$g_t = r_t + \kappa J_h(\theta_t)^\top M_t^{-1} h(\theta_t),$$

and hence

$$\begin{aligned} \langle r_t, D_t g_t \rangle &= \langle r_t, D_t r_t \rangle + \kappa \left\langle r_t, D_t J_h(\theta_t)^\top M_t^{-1} h(\theta_t) \right\rangle \\ &= \langle r_t, D_t r_t \rangle + \kappa \left\langle J_h(\theta_t) D_t r_t, M_t^{-1} h(\theta_t) \right\rangle \\ &= \langle r_t, D_t r_t \rangle. \end{aligned}$$

Also, the metric feedback-linearization identity gives

$$J_h(\theta_t) D_t g_t = \kappa h(\theta_t).$$

Thus

$$\langle \nabla f(\theta_t), D_t g_t \rangle = \langle r_t, D_t r_t \rangle - \kappa {\lambda_t^\dagger}^\top h(\theta_t).$$

Using $D_t \succeq \frac{1}{G+\delta} I$ and $\|\lambda_t^\dagger\|_\infty \le \Lambda$, we obtain

$$- \langle \nabla f(\theta_t), D_t g_t \rangle \le -\frac{1}{G+\delta} \|r_t\|^2 + \kappa \Lambda \|h(\theta_t)\|_1. \tag{39}$$

Combining (38) and (39) yields

$$f(\theta_{t+1}) \le f(\theta_t) - \frac{\eta_t}{G+\delta} \|r_t\|^2 + \eta_t \kappa \Lambda \|h(\theta_t)\|_1 + \frac{\eta_t G_f}{\delta} \|e_t\| + \frac{L_f G^2}{2\delta^2} \eta_t^2. \tag{40}$$

Next, we estimate the constraint violation. By Taylor's theorem,

$$\begin{aligned} h(\theta_{t+1}) &= h(\theta_t) + J_h(\theta_t)(\theta_{t+1} - \theta_t) + \mathcal{E}_t \\ &= h(\theta_t) - \eta_t J_h(\theta_t) D_t g_t - \eta_t J_h(\theta_t) D_t e_t + \mathcal{E}_t \\ &= (1 - \kappa \eta_t) h(\theta_t) - \eta_t J_h(\theta_t) D_t e_t + \mathcal{E}_t, \end{aligned}$$

where

$$\|\mathcal{E}_t\|_1 \le \frac{L_{h,1}}{2} \|\theta_{t+1} - \theta_t\|^2.$$

Since $\|\theta_{t+1} - \theta_t\| \le \eta_t \|D_t \hat{m}_t\| \le \eta_t G/\delta$, we have

$$\|\mathcal{E}_t\|_1 \le \frac{L_{h,1} G^2}{2\delta^2} \eta_t^2.$$

Using $\eta_t \le 1/\kappa$, we have $1 - \kappa\eta_t \ge 0$. Therefore,

$$\begin{aligned}\|h(\theta_{t+1})\|_1 &\le (1-\kappa\eta_t)\|h(\theta_t)\|_1 + \eta_t\|J_h(\theta_t)D_t e_t\|_1 + \frac{L_{h,1}G^2}{2\delta^2}\eta_t^2 \\ &\le (1-\kappa\eta_t)\|h(\theta_t)\|_1 + \frac{\eta_t J_1}{\delta}\|e_t\| + \frac{L_{h,1}G^2}{2\delta^2}\eta_t^2. \end{aligned} \tag{41}$$

Multiplying (41) by ρ and adding it to (40), we obtain

$$\begin{aligned}\Psi_\rho(\theta_{t+1}) \le \Psi_\rho(\theta_t) &- \frac{\eta_t}{G+\delta}\|r_t\|^2 - \eta_t\kappa(\rho-\Lambda)\|h(\theta_t)\|_1 \\ &+ \frac{\eta_t(G_f + \rho J_1)}{\delta}\|e_t\| + \frac{(L_f + \rho L_{h,1})G^2}{2\delta^2}\eta_t^2. \end{aligned} \tag{42}$$

Choose $\rho > \Lambda$ and define

$$c_* := \min\left\{\frac{1}{G+\delta}, \kappa(\rho-\Lambda)\right\}, \qquad B_e := \frac{G_f + \rho J_1}{\delta}, \qquad B_\eta := \frac{(L_f + \rho L_{h,1})G^2}{2\delta^2}.$$

Then (42) implies

$$\Psi_\rho(\theta_{t+1}) \le \Psi_\rho(\theta_t) - c_*\eta_t\left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) + B_e\eta_t\|e_t\| + B_\eta\eta_t^2. \tag{43}$$

Summing (43) from $t = 1$ to T, we get

$$c_*\sum_{t=1}^{T}\eta_t\left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) \le \Psi_\rho(\theta_1) - \Psi_\rho(\theta_{T+1}) + B_e\sum_{t=1}^{T}\eta_t\|e_t\| + B_\eta\sum_{t=1}^{T}\eta_t^2.$$

Since Ψ_ρ is continuous on $\mathcal{C}$, it is bounded below. Let

$$\underline{\Psi} := \inf_{\theta\in\mathcal{C}}\Psi_\rho(\theta) > -\infty.$$

Using Lemma 2, we obtain

$$\sum_{t=1}^{T}\eta_t\|e_t\| \le C_e\left(\sum_{t=1}^{T}\eta_t^2 + \frac{\eta_1 n}{\delta}\right).$$

Therefore,

$$c_*\sum_{t=1}^{T}\eta_t\left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) \le \Psi_\rho(\theta_1) - \underline{\Psi} + B_eC_e\left(\sum_{t=1}^{T}\eta_t^2 + \frac{\eta_1 n}{\delta}\right) + B_\eta\sum_{t=1}^{T}\eta_t^2.$$

Thus

$$\sum_{t=1}^{T}\eta_t\left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) \le B_0 + B_1\sum_{t=1}^{T}\eta_t^2, \tag{44}$$

where

$$B_0 := \frac{\Psi_\rho(\theta_1) - \underline{\Psi} + B_eC_e\eta_1 n/\delta}{c_*}, \qquad B_1 := \frac{B_eC_e + B_\eta}{c_*}.$$

This proves the first claim.

Finally, let $S_T := \sum_{t=1}^{T}\eta_t$. Since

$$S_T\min_{1\le t\le T}\left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) \le \sum_{t=1}^{T}\eta_t\left(\|r_t\|^2 + \|h(\theta_t)\|_1\right),$$

dividing (44) by S_T gives

$$\min_{1\le t\le T}\left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) \le \frac{B_0 + B_1\sum_{t=1}^{T}\eta_t^2}{S_T}.$$

By Assumption 2, $\sum_t \eta_t^2 < \infty$. Hence

$$\min_{1\le t\le T}\left(\|r_t\|^2 + \|h(\theta_t)\|_1\right) = \mathcal{O}\left(\frac{1}{S_T}\right).$$

This completes the proof. □