

LLM-DRIVEN LOSS BALANCING FOR PHYSICS-INFORMED NEURAL NETWORKS

Denis Degtyarev, Anastassiya Ryabkova
 {d.degtyarev, a.ryabkova}@innopolis.university
 Faculty of Computer Science and Engineering,
 Innopolis University, Innopolis, Russia

Alexei Kornaev
 a.kornaev@innopolis.ru, RC for AI,
 N.N. Blokhin National Medical Research
 Center of Oncology, Moscow, Russia
 Center for Top-Level Educational Programs
 in AI, Innopolis University, Innopolis, Russia

ABSTRACT

We test whether large language models (LLMs) can mitigate loss imbalance in Physics-Informed Neural Networks (PINNs) by either controlling loss weights during training or ranking established balancing methods from PDE and method descriptions. Experiments show that LLMs fail as direct weight controllers, but can often identify strong candidates (e.g., GradNorm, LB-PINN, RAD, DB-PINN) when acting as meta-evaluators on harder multiscale and stiff problems. Our implementation is publicly available at <https://github.com/PINN-agent/PINN-trainer>.

1 INTRODUCTION

Physics-Informed Neural Networks (PINNs) approximate solutions of differential equations by minimizing a composite loss combining PDE residuals with boundary and initial conditions (Raissi et al., 2019). Training is often unstable because these terms evolve at different scales, causing loss imbalance. Various strategies attempt to mitigate this issue, including adaptive gradient normalization (GradNorm), probabilistic weighting (LB-PINN), and residual-based adaptive sampling such as RAD or RAR-D (Chen et al., 2018; Xiang et al., 2022; Wu et al., 2022), but selecting an effective method typically requires expert knowledge of the PDE and training dynamics. Recent work suggests that LLMs can assist scientific ML workflows by interpreting model descriptions or recommending algorithmic configurations (Wu et al., 2025; He et al., 2025). This motivates exploring whether LLMs can help address PINN loss imbalance either by adjusting loss weights during training or by selecting suitable balancing strategies from textual descriptions of the PDE and numerical method. Such assistance may be useful in engineering and biomedical modeling, where PINNs are used for tasks such as vascular flow simulation or contrast transport modeling in medical imaging.

We evaluate both roles of LLMs empirically. Specifically, we (i) test LLMs as direct controllers of loss weights in PINN training, (ii) study their ability to rank existing balancing strategies from PDE and method descriptions, and (iii) benchmark four representative approaches across five differential-equation problems, comparing LLM rankings with a metric-based reference.

2 METHODOLOGY

We solve each benchmark with a physics-informed neural network u_θ that maps spatio-temporal coordinates to the target field variables. Training minimizes a weighted multi-term objective

$$\mathcal{L}(\theta, \lambda) = \sum_{i=1}^K \lambda_i \mathcal{L}_i(\theta), \tag{1}$$

where \mathcal{L}_i correspond to PDE residual, boundary/initial-condition, and (when available) data-misfit terms. We evaluate on five benchmarks: damped oscillator, 1D viscous Burgers, Kovasznay flow, Beltrami flow, and 2D cylinder wake; full PDE definitions and domains are provided in the Appendix 3 Unless stated otherwise, we use fully connected networks with tanh activations and train with

Adam (learning rate 10^{-3}) for **15000** steps/epochs using the same base sampler and initial point budget across methods; the only exception is RAD, which adaptively resamples collocation points based on residuals by design, and therefore changes the sampling distribution during training.

Across experiments, we keep the PINN architecture and training hyperparameters fixed. For methods that explicitly optimize loss weights, we vary only the mechanism used to update these weights (baseline strategies versus LLM-based controllers). In contrast, the proposed two-agent LLM system does not participate in PINN training; instead, it uses analyses of PDE/ODE descriptions together with structured summaries of method characteristics to rank existing loss-balancing method.

PINNs are implemented in PyTorch using automatic differentiation. Experiments without LLM components are trained on an NVIDIA GeForce GTX 1650 GPU. Experiments involving LLMs use HuggingFace Transformers and are executed on NVIDIA A100 GPUs. All runs use a fixed random seeds (0, 5, 42, 111), deterministic settings where applicable, and log all hyperparameters.

2.1 LOSS-WEIGHT CONTROL

Baselines. We evaluate GradNorm (per-epoch reweighting to equalize gradient norms; $\lambda_i \in [10^{-4}, 10^3]$), LB-PINN (treat λ_i as trainable parameters; update every 100 epochs for oscillator and Burgers and 500 for other PDEs), DB-PINN (dual balancing from gradient statistics; update every 100 epochs), and RAD (residual-based adaptive resampling every 400 epochs with (k, c) ; see supplement for hyperparameters).

LLM-controlled loss weights (damped oscillator). At each control intervention, the LLM receives *both current and previous* values of the losses and the weights,

$$(L_{bc}^t, L_{bc\text{-grad}}^t, L_{phys}^t, \lambda_1^t, \lambda_2^t, \lambda_3^t) \quad \text{and} \quad (L_{bc}^{t-1}, L_{bc\text{-grad}}^{t-1}, L_{phys}^{t-1}, \lambda_1^{t-1}, \lambda_2^{t-1}, \lambda_3^{t-1}).$$

The prompt also includes the problem specification (ODE and initial conditions). The LLM is instructed to return its decision in a machine-readable JSON format with the new $\lambda_1^{t+1}, \lambda_2^{t+1}, \lambda_3^{t+1}$. After each intervention, the weights are clipped to $\lambda_i \in [10^{-7}, 10^7]$ and parse back for the new epoch.

In our experiments, we instantiate this controller with several instruction-tuned Qwen-family models. Specifically, for LLM (Simple) and LLM (Prompt) we use Qwen models (Qwen3-4B, Qwen2.5-7B, and Qwen2.5-14B), while LLM (Method) uses Qwen2.5-Math-1.5B 8. **LLM (Simple).** See C The LLM outputs a discrete action $a \in \{1, 2, 3, 4\}$: double λ_1 , double λ_2 , double λ_3 , or keep the weights unchanged. Control is applied every `control_steps= 1000` training steps. **LLM (Prompt).** See C The LLM outputs updated weights $(\lambda_1, \lambda_2, \lambda_3)$. Unlike **LLM (Simple)**, the action space is not restricted, allowing arbitrary weight updates. Control is applied every `control_steps= 1000` training steps. **LLM (Method).** See C The protocol matches **LLM (Prompt)**, but the prompt additionally includes a description of the GradNorm algorithm; the LLM is expected to adapt its updates to follow a GradNorm-like balancing behavior. Control is applied every `update_epoch= 4000` steps; the control interval is selected manually based on empirical training behavior.

2.2 LLM-BASED TWO-AGENT RANKING SYSTEM

A two-agent LLM pipeline is used to recommend a loss-balancing strategy for each PDE benchmark, using the Qwen2.5-14B model (see Fig. 1).

Phase A (task understanding). The *PDE Analyst* interprets the PDE description, the structure of loss terms, and available training signals to extract task features and potential optimization risks (e.g., stiffness, multi-scale behavior, gradient imbalance). Prompt texts are provided in Appendix D.

Phase B (method ranking). The *Method Ranker* receives (i) the PDE description and extracted features produced by the PDE Analyst, and (ii) a single machine-readable JSON catalog of methods, where each candidate (GradNorm, LB-PINN, RAD, DB-PINN) is described by its core mechanism and intended use, together with a short summary of its positioning/usage in the literature. The JSON catalog is provided in Appendix D. Based on these inputs, the Method Ranker outputs an ordered

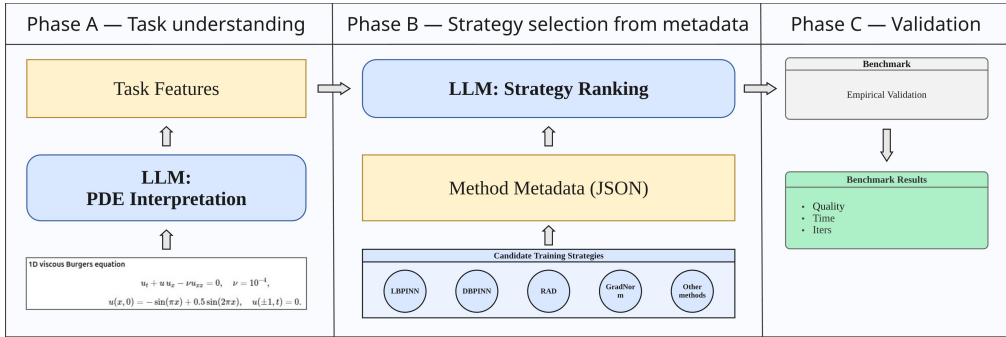


Figure 1: Agent pipeline for methods ranking.

Table 1: Selected-epoch loss components for the Damped Oscillator benchmark (epoch selected by: minimize MSE if available, else minimize unweighted loss). All values are taken directly from the training logs at the selected epoch.

Method	$(\times 10^{-1})$				
	Total loss	Phys. loss	IC _{du}	IC _u	MSE
GradNorm	0.2460	397.2649	7.391e-04	8.0752	0.9902
LB-PINN	1.3291	2329.7500	77.5878	0.0023	0.0742
RAD	10.0400	0.2110	0.0312	9.7967	0.3179
DB-PINN	1382.8850	1014.4400	0.0098	0.4014	0.1439

ranking (best→worst) with brief comments and a confidence label; the LLM outputs `D` and `D` are also included in Appendix.

Phase C (validation and benchmarking). For each PDE and each method in the candidate pool, we run training and construct a *reference ranking* from logs via a *multi-metric* ground-truth (GT) score. Metrics are computed at a *selected epoch*: we choose the epoch minimizing an MSE-type metric when available; Otherwise, we minimize the **unweighted** sum of component losses. The GT score is a weighted sum of within-task min-max normalized terms with $\log(1+x)$ stabilization for nonnegative metrics (primary weight 1.0, each secondary weight 0.2). The primary solution-quality term is the first available among **best/selected MSE** or **selected relative L_2** . Secondary terms capture convergence and constraint satisfaction: **selected max normalized loss**, **time-to- ϵ** for all normalized components, and **AUC** over normalized losses (summed across components up to the selected epoch). We compare the LLM-predicted ranking (Phase B) to the GT ranking using tie-aware, decision-oriented metrics: **NDCG@3**, **Top-1 hit**, **Hit@2**, **pairwise accuracy** (Kendall τ_b mapped to $[0, 1]$), and **normalized top-choice regret** on the primary metric. Full definitions and tie handling are provided in Appendix E.

3 RESULTS

3.1 BASELINE LOSS IMBALANCE ACROSS PDE BENCHMARKS

Tables 1–7 report the selected-epoch loss decomposition for GradNorm, LB-PINN, RAD, and DB-PINN. Overall, small total/physics losses do not necessarily imply small MSE, and the learned weights vary strongly across PDEs. **DampedOscillator**(Table 1): LB-PINN achieves the lowest MSE with large `ic_du_loss`; GradNorm has small total loss but large `ic_u_loss`. **BeltramiFlow** (Table 4): GradNorm has low total loss and tiny `phys_momentum_loss` but extremely large MSE; DB-PINN/LB-PINN achieve much smaller MSE. **CylinderWake2D** (Table 5): RAD yields orders-of-magnitude smaller total and boundary losses; GradNorm/LB-PINN retain large `bc_inlet_loss`. **Burgers1DHard** (Table 6): GradNorm has the smallest total loss, while RAD achieves the best MSE; DB-PINN shows very large total loss despite tiny `phys_loss` and `bc_loss`. **KovasznyNS** (Table 7): GradNorm has tiny `phys_u_loss/phys_v_loss` but ex-

tremely large MSE; DB-PINN/LB-PINN achieve low MSE with different u/v balances and highly non-uniform weights.

3.2 DIRECT LLM CONTROL OF LOSS WEIGHTS IS UNSTABLE

Table 8 shows that all three LLM variants (**LLM (Simple)**, **LLM (Prompt)**, and **LLM (Method)**) produce extreme, skewed weights (e.g., 10^6 – 10^7 on a single term), resulting in severely imbalanced loss components. Critically, as training history lengthens, the LLM increasingly hallucinates invalid weight updates - suggesting fundamental limitations in numerical reasoning under temporal context. Since these failures manifest even on simple ODE problems, we concluded that unconstrained LLM-based weight control is unreliable for real-time loss balancing and did not proceed to PDE benchmarks.

3.3 LLM-BASED RANKING PARTIALLY MATCHES METRIC-BASED RANKS

Table 2: Rank disagreement between LLM and metric-based evaluation. $\Delta = \text{LLM} - \text{Overall}$; \uparrow means LLM ranks the method worse, \downarrow means better.

PDE	GradNorm	LB-PINN	RAD	DB-PINN
Damped Oscillator	$\downarrow 3$	0	$\uparrow 1$	$\uparrow 2$
Burgers 1D (Hard)	$\uparrow 1$	$\downarrow 2$	0	$\uparrow 1$
Kovaszny NS	$\downarrow 1$	$\downarrow 1$	0	$\uparrow 2$
Cylinder Wake 2D	$\downarrow 1$	0	0	$\uparrow 1$
Beltrami Flow	$\downarrow 3$	$\uparrow 1$	$\uparrow 1$	$\uparrow 1$

Finally, we evaluate the two-agent LLM system as a meta-evaluator that ranks loss-balancing methods from PDE and method descriptions. Table 2 compares the metric-based ground-truth ranks (“Overall”) to the LLM-predicted ranks (“LLM”) across benchmarks and methods. The LLM exhibits mixed behavior: it often preserves the ranking for RAD (zero shifts on Burgers 1D (Hard) and Kovaszny NS), while producing systematic mis-rankings on other tasks - most notably on the Damped Oscillator where it markedly promotes GradNorm and demotes RAD/DB-PINN relative to the metric-based evaluation (large Δ magnitudes). These results suggest that the LLM can predict the best-performing methods reasonably well, making it a useful and interpretable advisor for narrowing the candidate set, rather than a full ranker or a standalone method selector.

4 CONCLUSION AND FUTURE WORK

We study whether large language models (LLMs) can reduce the loss-imbalance problem in physics-informed neural networks (PINNs) in two ways: (i) directly setting loss weights during training and (ii) acting as meta-evaluators that choose among established balancing methods from a PDE and methods description. For simple ODE problem, LLM-driven weight control (discrete or continuous) was generally unreliable: it frequently produced extreme/unstable weights that destabilized optimization and led to poor convergence, reflecting current LLM limits in numerical control and prompt sensitivity. In contrast, in the meta-evaluation setting, LLMs were generally effective at identifying the strongest loss-balancing candidates, but their fine-grained numerical ordering was less stable: occasional rank inconsistencies appeared, with the largest discrepancies on the Damped Oscillator (and, for GradNorm, also on Beltrami Flow), indicating limited precision in exact ranking regardless of PDE complexity. Overall, LLMs are better positioned today as interpretable “front-end” guides for selecting a balancing strategy-helping reduce manual trial-and-error-rather than as replacements for adaptive weighting algorithms; future work should add more structured diagnostics and hybridize LLM reasoning with automated tuning.

ACKNOWLEDGMENTS

This work was supported by the Grant Agreement from the Federal Budget in the form of a Subsidy to the Federal State Budgetary Institution "N.N. Blokhin National Medical Research Center of Oncology" of the Ministry of Health of the Russian Federation dated April 16, 2025, No. 139-15-2025-008, State Contract ID 000000C313925P3R0002.

REFERENCES

- Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018.
- Xin He, Liangliang You, Hongduan Tian, Bo Han, Ivor Tsang, and Yew-Soon Ong. Lang-PINN: From language to physics-informed neural networks via a multi-agent framework. *arXiv preprint arXiv:2510.05158*, 2025.
- 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.
- Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, 2022. URL <https://arxiv.org/abs/2207.10289>.
- Qingpo Wu, Chonghan Gao, Tianyu Chen, Yihang Huang, Yuekai Zhang, et al. PINNsAgent: Automated PDE surrogation with large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.
- Zixue Xiang, Wei Peng, Xu Liu, and Wen Yao. Self-adaptive loss balanced physics-informed neural networks. *Neurocomputing*, 496:11–34, 2022.

A CODE AVAILABILITY

Our implementation is available at: <https://github.com/PINN-agent/PINN-trainer>.

B PDE BENCHMARKING

We evaluate weight-control and training strategies across standard PINN benchmarks spanning ODE, 1D, 2D, and 3D PDE settings.

Table 3: Benchmark PDE Problems and Their Key Challenges

PDE	Variables	Key Challenges
Damped Oscillator	t	Baseline ODE problem (described above)
Cylinder Wake	x, y, t	Complex geometry, mixed BC, flow separation
1D Viscous Burgers	x, t	Convection-dominated regime ($\nu = 1 \times 10^{-4}$), shock form.
Kovaszny Flow	x, y	Stationary Navier-Stokes, incompressibility constraint
Beltrami Flow	x, y, z, t	3D unsteady Navier-Stokes, multiple velocity components

For each benchmark below, we report the decomposition of the training objective at the selected epoch (minimize MSE if available, else minimize unweighted loss), averaged over runs.

Table 4: Selected-epoch loss components for the Beltrami flow benchmark (epoch selected by: minimize MSE if available, else minimize unweighted loss). All values are taken from the training logs at the selected epoch and averaged over runs.

Method	$(\times 10^{-3})$					
	Total loss	Phys. mom.	Phys. div.	IC u	BC u	MSE
GradNorm	1.251	0.023	0.163	3580	419	340
LB-PINN	7.97	0.618	0.056	1.0e+04	9647	0.045
RAD	1.0856	0.804	0.054	1000	1000	0.1154
DB-PINN	4.107	2.097	0.226	1000	1000	0.0117
Reference benchmarks (relative L_2 errors) from prior work Xiang et al. (2022)						
lbPINNs $\varepsilon_f, \varepsilon_i, \varepsilon_b = 2, 2, 2$		3.21	0.990			1.87
lbPINNs $\varepsilon_f, \varepsilon_b, \varepsilon_i = 2, 2, 10$		1.83	0.865			1.29

Table 5: Selected-epoch loss components for the Cylinder Wake 2D benchmark (epoch selected by: minimize MSE if available, else minimize unweighted loss). All values are taken from the training logs at the selected epoch and averaged over runs.

Method	$(\times 10^{-4})$					
	Total loss	Phys. loss	BC inlet	BC cylinder	BC periodic	BC outlet
GradNorm	5597	0.866	5505	0.0457	0.000	0.000
LB-PINN	5597	0.866	5505	0.0457	0.0	0.0
RAD	0.115	0.0768	0.0011	0.0017	0.0	0.0
DB-PINN	0.29	0.51	0.0	0.0106	0.0	0.0
Reference benchmarks (relative L_2 errors) from prior work Xiang et al. (2022)						
lbPINNs ($\varepsilon_f, \varepsilon_d = 2$)		0.0456	0.0482			1.07
lbPINNs ($\varepsilon_f = 2, \varepsilon_d = 0.04$)		0.0851	0.0671			5.37

Table 6: Selected-epoch loss components for the Burgers 1D (Hard) benchmark (epoch selected by: minimize MSE if available, else minimize unweighted loss). All values are taken from the training logs at the selected epoch and averaged over runs.

Method	$(\times 10^{-1})$			
	Total loss	IC loss	BC loss	MSE
GradNorm	0.854	0.0	1.70e-04	4.62
LB-PINN	6.53	0.0	0.0	4.62
RAD	6.15	0.0	2.16e-04	4.46
DB-PINN	591	0.0	0.0026	4.61

Table 7: Selected-epoch loss components for the Kovaszny flow benchmark (epoch selected by: minimize MSE if available, else minimize unweighted loss). All values are taken from the training logs at the selected epoch and averaged over runs.

Method	$(\times 10^{-4})$					
	Total loss	Phys. u	Phys. v	Div. loss	BC loss	MSE
GradNorm	2.25	0.006	0.0063	9250	18200	2286
LB-PINN	3.039	1.95	32.83	3775	1.0	0.127
RAD	4.55	2.56	0.668	10000	10000	1.11
DB-PINN	75.6	22.0	1.97	10000	10000	0.128
Reference benchmarks (relative L_2 errors) from prior work Xiang et al. (2022)						
lbPINNs ($\varepsilon_f = \varepsilon_b = 2$)		6.41	2.82			0.432
lbPINNs ($\varepsilon_f = \varepsilon_b = 0.2$)		2.16	3.27			0.328

C LLM AS AN OPTIMIZER

We treat the LLM as a black-box controller that maps short training diagnostics to loss-weight updates. All prompts are reproduced verbatim as used in the codebase; placeholders (e.g., `{history_text}`) are substituted at runtime. Below, the exact prompt templates are provided for the three LLM controllers.

LLM (Simple) controller

```

user_prompt = """
Loss history (old → new):

{history_text}

Rule:
- Focus on the LAST few values.
- If a loss stays above 1e-2 and does NOT decrease, it converges
poorly t increase its weight.
- If multiple losses satisfy this, choose the one with the highest
value.
- If all losses are below 1e-2 and decreasing, choose 4 (no change).

Respond with ONE digit:
1 = increase weight for loss1
2 = increase weight for loss2
3 = increase weight for loss3
4 = no change
"""

system_prompt = (
    "You are a PINN training controller. "
    "Your task is to identify which loss term remains large and is
not decreasing. "
    "A loss is considered poorly converging if it stays above 1e-2
and does not show improvement. "
    "Choose the term that converges the worst. "
    "If all losses are small (<1e-2) and decreasing, choose 4. "
    "Output ONLY ONE digit: 1, 2, 3, or 4."
)

```

The *Simple* controller outputs a discrete action, whereas the following controllers output explicit continuous weight values.

LLM (Prompt) controller

```

pde_osci = r"""
Damped oscillator ODE:

u_tt(t) + 4 u_t(t) + 400 u(t) = 0,
t [0, 1],

initial conditions:
u(0) = 1,
u_t(0) = 0.
"""

system_prompt = (
    "You are an expert adaptive controller for PINNs. "
    "Your goal is to improve convergence by adjusting loss weights dynamically. "
    "You will receive: (1) recent loss values (boundary, gradient, physics), "
    "(2) current weights (lambda1, lambda2, lambda3). "
    "You may INCREASE, DECREASE or KEEP each weight, "
    "depending on the behavior of its loss.\n"
    "Rules:\n"
    "- If a loss is **high (>1e-2) AND not decreasing over recent steps**, "
    "it is poorly converging | you should **increase its weight**.\n"
    "- The **higher the loss or the longer it stagnates** "
    "*", the **more aggressively** you may increase its weight (e.g., "
    "x2, x5).\n"
    "- If a loss is **very dominating** (its weight is much larger "
    "than the others or its loss already decreases much faster than "
    "the rest), "
    "you may **slightly decrease its weight** to rebalance training, "
    "even if the loss is not yet very small.\n"
    "- If a loss is already low (<1e-3) and decreasing | you may "
    "**slightly reduce its weight** (e.g., x0.5) to focus on harder "
    "terms.\n"
    "- If a loss is moderate and behaves similarly to the others, "
    "prefer to **keep its weight roughly unchanged**.\n"
    "- All weights must stay in [1e-7, 1e7].\n"
    "- Prefer moderate changes (e.g., x0.5{x2}) unless convergence "
    "is severely imbalanced.\n"
    f"- pde: {pde_osci}\n"
    "- Output format:\n"
    "###THINKING: <step-by-step analysis>\n"
    "###ANSWER: lambda1=<value>, lambda2=<value>, lambda3=<value>"
)

user_prompt = """
Current loss weights:
- lambda1 (BC loss)      = {lambda1:.3e}
- lambda2 (BC grad loss) = {lambda2:.3e}
- lambda3 (PDE loss)     = {lambda3:.3e}

Loss history (last 50 steps, newest at bottom):
{history_text}

→ Analyze: which losses are large/stagnant? Which are small/improving?
→ Adjust weights accordingly.
→ Output reasoning and new weights **strictly** in the required format.
"""

---

## GRADIENT STATISTICS INTERPRETATION:

### Primary Rules:
- High Gradient Norm → The loss is EASY to optimize → DECREASE its weight
- Low Gradient Norm → The loss is HARD to optimize → INCREASE its weight
- Use gradient statistics as the primary guide for weight updates

```

```

### Strategy:
- Follow the inverse gradient relationship: 1/gradient_norm
- If gradients suggest conflicting changes with loss values,
prioritize gradients
- Use quality metrics to validate decisions
- Consider exploratory changes if stagnation persists

---

Please:
1. Briefly (3-5 bullet points) explain the logic behind
the weight changes (text only).
2. Then output ONLY JSON with the new weights in a separate fenced block:

{{ "lambda1": <number>, "lambda2": <number>, "lambda3": <number> }}
"""

```

LLM (Method) controller

```

prompt_template = """
I am training PINN (Physics-Informed Neural Network) for oscillator
processes using Gradient Statistics method.

## Problem Equation:
The damped harmonic equation is solved:


$$d^2u/dt^2 + du/dt + k u(t) = 0$$


Initial Conditions:
-  $u(0) = 1$ 
-  $u'(0) = 0$ 

## GRADIENT STATISTICS METHOD:

### How Gradient Statistics Works:
- Core Principle: Monitor gradient norms to balance loss influence
automatically
- Inverse Relationship: Losses with LARGE gradients dominate
training → get LOWER weights
- EMA Smoothing: Use Exponential Moving Average of gradient norms
for stability
- Dynamic Balancing: Automatically finds optimal weight distribution
during training

## Current State:

### Loss Values:
1. loss_bc | violation of  $u(0)=1$  = {loss_bc}
2. loss_bc_grad | violation of  $u(0)=0$  = {loss_bc_grad}
3. loss_phys | physical residual violation = {loss_phys}

### Current Weights:
1 = {lambda1}, 2 = {lambda2}, 3 = {lambda3}

## Quality Metrics:
- mse_test = {mse_test}
- max_residual = {max_residual}
- residual_L2 = {residual_L2}
- u_error_L2 = {u_error_L2}
- grad_norm = {grad_norm}
- u_mid = {u_mid}

## Training History:
{history_log}

```

```

---
## GRADIENT STATISTICS INTERPRETATION:

### Primary Rules:
- High Gradient Norm → The loss is EASY to optimize → DECREASE its weight
- Low Gradient Norm → The loss is HARD to optimize → INCREASE its weight
- Use gradient statistics as the primary guide for weight updates

### Strategy:
- Follow the inverse gradient relationship: 1/gradient_norm
- If gradients suggest conflicting changes with loss values, prioritize gradients
- Use quality metrics to validate decisions
- Consider exploratory changes if stagnation persists

---

```

Please:

1. Briefly (3-5 bullet points) explain the logic behind the weight changes (text only).
2. Then output ONLY JSON with the new weights in a separate fenced block:

```

{{ "lambda1": <number>, "lambda2": <number>, "lambda3": <number> }}
"""

```

Table 8 summarizes the final weights and corresponding loss components obtained under each LLM controller and backbone model.

Table 8: Final loss components and weight values at the last training iteration ($\sim 30k$). Rows correspond to LLM control strategies, columns correspond to backbone models. All values are taken verbatim from logs.

Strategy	Model	λ_1	λ_2	λ_3	$(\times 10^{-5})$		
					BC loss	BC-grad loss	PDE residual
LLM (Simple)	Qwen3-4B	1.0e7	1.0	1.0	0.000101	19200	336000
	Qwen2.5-7B	1.0e7	1.0	1.0	1.33	8.58	464000
	Qwen2.5-14B	1.0e7	1.0	1.0	1.33	8.58	464000
LLM (Prompt)	Qwen3-4B	1.0	1.0	1.0	100000	0.0309	8.01
	Qwen2.5-7B	1.0e7	0.625	1.0e7	98400	21.3	7040
	Qwen2.5-14B	3.2e6	0.25	3.0e5	99900	1.79	683
LLM (Method)	Qwen2.5-Math-1.5B	1.0	1.0	3.0	3.6380e-12	1.8527e-02	88.8631e+02

D LLM AS AN RANKER

This appendix documents our two-stage ranking pipeline: an LLM first produces a PINN-relevant structured description of the PDE, which—together with a JSON summary of candidate methods—is then fed to a ranking LLM to select and order the loss-balancing method. D.

PDE texts

```

pde_osci = r"""
Damped oscillator ODE:

u_tt(t) + 4 u_t(t) + 400 u(t) = 0,
t [0, 1],

initial conditions:
u(0) = 1,
u_t(0) = 0.
"""

```

```

pde_burgers = r"""
1D viscous Burgers equation (hard, convection-dominated):

 $u_t(x,t) + u(x,t) * u_x(x,t) - 1e-4 * u_{xx}(x,t) = 0,$ 
 $x \in [-1, 1], t \in [0, 1].$ 

Initial condition (multi-frequency, hard for PINNs):
 $u(x, 0) = -\sin(x) + 0.5 \sin(2x).$ 

Boundary conditions (Dirichlet):
 $u(-1, t) = 0,$ 
 $u(1, t) = 0.$ 

Here  $\nu = 1e-4$  is very small (convection-dominated regime with thin
shock / boundary layer).
"""

pde_kovasznyay = r"""
2D stationary incompressible Navier{Stokes (Kovasznyay flow, Re = 40):

Unknown fields:
 $u(x, y), v(x, y)$  | velocity components,
 $p(x, y)$  | pressure.

Domain:
 $(x, y) \in [-0.5, 1.0] \times [-0.5, 1.5].$ 

Parameters:
 $\text{Re} = 40,$ 
 $\nu = 1 / \text{Re} = 0.025.$ 

Governing equations (stationary Navier{Stokes in 2D):

 $u * u_x + v * u_y = -p_x + \nu * (u_{xx} + u_{yy}),$ 
 $u * v_x + v * v_y = -p_y + \nu * (v_{xx} + v_{yy}),$ 

incompressibility constraint:
 $u_x + v_y = 0,$ 

where subscripts denote partial derivatives, e.g.
 $u_x = u/x, u_{yy} = \partial^2 u / \partial y^2,$  etc.

Boundary conditions (Dirichlet, on the whole boundary ):
 $u(x, y) = u_{BC}(x, y),$ 
 $v(x, y) = v_{BC}(x, y),$ 

for all  $(x, y) \in \partial\Omega$ , where  $u_{BC}, v_{BC}$  are prescribed boundary
velocity fields
corresponding to the Kovasznyay flow at  $\text{Re} = 40.$ 
"""

pde_beltrami = r"""
3D incompressible Navier{Stokes (Beltrami flow benchmark):

Unknown fields:
 $u(x, y, z, t), v(x, y, z, t), w(x, y, z, t)$  | velocity components,
 $p(x, y, z, t)$  | pressure.

Governing equations ( $\text{Re} = 1$ ):

 $u_t + u u_x + v u_y + w u_z = -p_x + \nu (u_{xx} + u_{yy} + u_{zz}),$ 
 $v_t + u v_x + v v_y + w v_z = -p_y + \nu (v_{xx} + v_{yy} + v_{zz}),$ 
 $w_t + u w_x + v w_y + w w_z = -p_z + \nu (w_{xx} + w_{yy} + w_{zz}),$ 

```

incompressibility constraint:
 $u_x + v_y + w_z = 0,$

where subscripts denote partial derivatives, e.g. $u_t = u/t$, $u_{xx} = \partial^2 u / \partial x^2$, etc.

Domain:

$(x, y, z, t) \in [-1, 1] \times [-1, 1] \times [-1, 1] \times [0, 1].$

Parameters:

$Re = 1.$

Initial condition ($t = 0$):

$u(x, y, z, 0) = u_{IC}(x, y, z),$

$v(x, y, z, 0) = v_{IC}(x, y, z),$

$w(x, y, z, 0) = w_{IC}(x, y, z),$

Boundary conditions (Dirichlet, all faces of the cube $x = \pm 1$, $y = \pm 1$, $z = \pm 1$):

$u(x, y, z, t) = u_{BC}(x, y, z, t),$

$v(x, y, z, t) = v_{BC}(x, y, z, t),$

$w(x, y, z, t) = w_{BC}(x, y, z, t),$

for all $t \in [0, 1]$ on the spatial boundary.

This setup corresponds to the Beltrami flow benchmark, where IC/BC are induced by an exact smooth solution of the incompressible Navier-Stokes equations.

"""

pde_cylinder_wake = r"""

2D incompressible Navier-Stokes (cylinder wake behind a circular cylinder, $Re = 100$):

Unknowns:

$u(x, y, t), v(x, y, t)$ | velocity components,
 $p(x, y, t)$ | pressure.

Governing equations ($\nu = 0.01$, $Re = 1/\nu = 100$):

$u_t + u * u_x + v * u_y = -p_x + \nu * (u_{xx} + u_{yy}),$

$v_t + u * v_x + v * v_y = -p_y + \nu * (v_{xx} + v_{yy}),$

incompressibility:

$u_x + v_y = 0,$

on the spatial{temporal domain:

$(x, y) \in ([-15, 25] \times [-8, 8]) \setminus \{(x, y) : x^2 + y^2 \leq (D/2)^2\},$

$D = 1,$

$t \in [0, T].$

Boundary conditions:

Inlet (left boundary $x = -15$, $-8 \leq y \leq 8$, outside the cylinder):

$u(-15, y, t) = U_{in},$

$v(-15, y, t) = 0,$

(uniform inflow, typically $U_{in} = 1$).

Cylinder surface ($x^2 + y^2 = (D/2)^2$):

$u(x, y, t) = 0,$

$v(x, y, t) = 0,$

(no-slip boundary condition).

Top and bottom boundaries ($y = \pm 8$):

```

u(x, -8, t) = u(x, 8, t),
v(x, -8, t) = v(x, 8, t),
p(x, -8, t) = p(x, 8, t),
(periodic boundary conditions in y).

```

```

Outlet (right boundary x = 25, -8 < y < 8, outside the cylinder):
p(25, y, t) = 0, (zero-pressure outflow),
u_x(25, y, t) = 0,
v_x(25, y, t) = 0, (zero normal gradient of velocity, opt.).

```

```

Initial condition:
u(x, y, 0) = u_0(x, y),
v(x, y, 0) = v_0(x, y),
p(x, y, 0) = p_0(x, y),

```

where (u_0, v_0, p_0) is a divergence-free initial field (e.g., taken from a CFD snapshot of the cylinder wake).

These PDE descriptions are passed verbatim to the analyst agent as the sole source of problem context.

Method JSON

```

{
  "methods": [
    {
      "Method": "LB-PINN (self-adaptive loss weighting via uncertainty / probabilistic formulation)",
      "What_it_adapts_(mechanism)": "Learns **loss-term weights** as trainable \uncertainty/noise" parameters; automatically reweights PDE/BC/IC/data contributions",
      "When_it_is_useful_(PINN_problem_signals)": "You have **multi-term loss with different scales** (PDE vs BC/IC vs auxiliary constraints) and want an **automatic** alternative to manual - tuning; imbalance is mainly about **loss magnitudes** rather than where errors are localized",
      "When_it_is_not_useful_low_ROI_risky": "If the main bottleneck is **where to sample** (thin layers, sharp fronts, localized residual), reweighting alone may not fix accuracy; depends on a meaningful, stable **loss decomposition** (poorly separated terms reduce benefit)",
      "What_to_monitor_in_logs_(decision_signals)": "Track learned weights and per-term losses: one term being consistently down-weighted to near-zero; some constraint losses stagnate while others decrease (scale-driven imbalance)"
    },
    {
      "Method": "GradNorm (gradient-based multi-objective balancing)",
      "What_it_adapts_(mechanism)": "Adapts weights using **per-term gradient norms** (balances training rates across objectives)",
      "When_it_is_useful_(PINN_problem_signals)": "You observe **gradient dominance** (e.g., PDE residual gradients overpower BC/IC gradients, or vice versa), causing underfitting of some constraints; multi-loss setup where aligning gradient flow is critical",
      "When_it_is_not_useful_low_ROI_risky": "Requires extra gradient computations and can be **noisy**; may oscillate if gradients are highly stochastic; does not directly address **sampling bottlenecks** or localized structures",
      "What_to_monitor_in_logs_(decision_signals)": "Log per-term **gradient norms** and relative decrease rates: a persistent gap in gradient norms is a strong indicator; look for stabilization of \ignored" terms once reweighting is enabled"
    }
  ],
}

```

```

"Method": "DB-PINN (Dual-Balancing: inter + intra)",

"What_it_adapts_(mechanism)": "(1) **Inter-balancing** between
PDE residual and constraint fitting; (2) **Intra-balancing**
across **multiple condition groups** using difficulty/history
signals",
"When_it_is_useful_(PINN_problem_signals)": "Best when you have
**many heterogeneous constraint groups** (multiple BC segments,
IC + BC + auxiliary constraints, possibly data) and see both:
(a) PDE-vs-constraints imbalance and (b) **some constraint
groups lagging** far behind others",
"When_it_is_not_useful_low_ROI_risky": "Overkill for simple
problems with few condition terms; added complexity/compute; if
the dominant issue is **point placement** (localized residual),
balancing alone may not recover accuracy",
"What_to_monitor_in_logs_(decision_signals)": "Track weights at
two levels: PDE-vs-constraints and **per-condition-group
weights**; signal: persistent \stuck" groups (e.g., certain
boundary segments) while others converge"
},
{
"Method": "RAD (Residual-based Adaptive Distribution / residual-
adaptive sampling)",
"What_it_adapts_(mechanism)": "Adapts **collocation point
distribution** using residual magnitude (focuses training where
residual is high)",
"When_it_is_useful_(PINN_problem_signals)": "You expect
**localized difficulty** (thin boundary layers, steep
gradients, fronts) or observe residual concentrated in small
regions under uniform sampling; you need to spend compute
**where the solution is hard**",
"When_it_is_not_useful_low_ROI_risky": "Requires residual
evaluation over candidates and resampling (can be costly); may
be challenging in very high dimensions; for globally
smooth/easy solutions, gains can be limited",
"What_to_monitor_in_logs_(decision_signals)": "Monitor residual
statistics and spatial concentration (quantiles / top-k
residual regions): signal is a persistent high-residual
\hotspot" that does not resolve with uniform points"
}
]
}

```

The method table is provided to the ranker as structured text, restricting it to the listed mechanisms and stated applicability conditions.

PDE Analyst

```

pde_analyst_agent = Agent(
    role="PDE behavior analyst",
    goal=(
        "Given a PDE description, produce a concise analysis of its
        behavior for numerical solvers "
        "and PINN training. Focus on PDE traits that affect
        difficulty, loss balancing, stiffness, "
        "sampling, and temporal strategy."
    ),
    backstory=(
        "You are an expert in qualitative PDE behavior for Physics-
        Informed Neural Networks (PINNs). "
        "You map PDE properties (type, stiffness, regularity, shocks,
        chaos, IC/BC sensitivity, "
        "domain geometry) to numerical difficulty. "
        "You do NOT prove theorems or add physics not present
        in the input. "
    )
)

```

```

        "If something is unclear from the PDE text or standard
        theory, you say 'unknown' or 'uncertain' "
        "instead of guessing."
    ),
    llm=hf_llm,
    verbose=True,
    memory=False, # PDE
)
pde_analysis_task = Task(
    description=(
        "Given the PDE:\n{{pde}}\n\n"
        "Act as a PDE behavior analyst for PINNs. Extract only the
        properties that matter for:\n"
        "- training stability and convergence,\n"
        "- loss balancing (PDE vs IC/BC vs data),\n"
        "- stiffness and gradient pathologies,\n"
        "- sampling strategy (uniform vs residual-based adaptive
        sampling),\n"
        "- temporal strategy (single-shot vs causal/time-
        marching).\n\n"
        "Use standard PDE intuition only. Do NOT invent extra terms,
        parameters, or BCs. "
        "If a trait cannot be clearly inferred, mark it as 'unknown'
        or 'uncertain' and keep confidence low."
    ),
    expected_output=""
)
You are a PINN-oriented PDE analyst. Your job is NOT
to explain the PDE in general.
Your job is to extract a structured \PINN difficulty profile"
that helps predict:
(1) convergence speed / stability, (2) final accuracy risk, and (3)
which intervention class is most relevant:
- loss-weight balancing (LB-PINN),
- gradient-based loss balancing (GradNorm),
- dual inter+intra balancing across condition groups (DB-PINN),
- residual-based adaptive sampling (RAD).

Rules:
- Use ONLY information explicitly present in the PDE description plus
standard, widely known PDE/PINN facts.
- Do NOT invent missing IC/BC, parameters, turbulence/chaos,
etc. If not stated, mark as "unknown".
- When uncertain, prefer "unknown" + low confidence.
- Focus on properties that impact composite-loss optimization,
gradient imbalance, stiffness, and sampling.

INPUT PDE TEXT:
{{pde}}

Return THREE parts:

PART A | PINN-relevant structural analysis (concise bullets)
A1) PDE & solution setting (minimal but precise):
- time-dependent vs stationary; PDE/ODE; spatial dimension;
order of derivatives; linear/semilinear/quasilinear/fully nonlinear.
- number of unknown fields; presence of auxiliary constraints
(e.g., incompressibility/divergence-free) if stated.
A2) Loss decomposition that a standard PINN would implement
(from the text):
- List loss term groups you would include: PDE residual(s),
IC, BC, data (if any), auxiliary constraints.
- For BC/IC: identify TYPES (Dirichlet/Neumann/periodic/mixed) and
whether there are MULTIPLE boundary segments or heterogeneous
condition groups.
- Note any interior boundaries/obstacles or complex geometry that

```

creates additional condition groups.

A3) Mechanisms that drive optimization difficulty:

- Dominant mechanisms mentioned (advection/convection, diffusion/viscosity, reaction/source, incompressibility coupling, etc.).
- Multi-scale indicators explicitly present: small parameters, high frequencies, thin layers implied by small diffusion, large domains, high derivative order.

A4) Expected PINN failure-mode risks (state as risks, not certainties):

- Gradient imbalance risk between term groups (PDE vs conditions vs auxiliary constraints): low/medium/high with 1{2 reasons grounded in A2{A3.
- Stiffness risk (none/mild/moderate/severe) and likely source category: temporal / spatial / small-parameter / nonlinear / multi-field coupling / unknown.
- Sampling risk (uniform points likely OK vs adaptive sampling likely needed): low/medium/high with reasons grounded in A3 (localized structures, boundary layers, shocks/fronts).
- Constraint heterogeneity risk: do different BC/IC segments likely have different means/variance/difficulty? (low/medium/high/unknown)

A5) \Signals to look for in logs" (actionable diagnostics):

- If gradient imbalance suspected: what observable patterns would appear (e.g., one term's gradient norms dominate; some constraints never decrease).
- If sampling is the bottleneck: what patterns would appear (e.g., residual concentrated in narrow regions; persistent local error).

A6) Method relevance summary

(ONLY these four methods; do not propose others):

For each method {LB-PINN, GradNorm, DB-PINN, RAD} output:

- Expected usefulness: high/medium/low/unknown
- Main justification: 1{2 sentences tied to A2{A4
- Main caveat: 1 sentence (when it might be low ROI or risky)
- Confidence: high/medium/low

PART B | Machine-readable traits JSON (valid JSON)

```
{
  "pde_setting": {
    "time_dependence": "time_dependent | stationary | unknown",
    "equation_class": "ODE | elliptic | parabolic |
hyperbolic | mixed | unknown",
    "linearity": "linear | semilinear | quasilinear |
fully_nonlinear | unknown",
    "num_spatial_dims": <int_or_null>,
    "num_fields": <int_or_null>,
    "max_derivative_order": <int_or_null>,
    "aux_constraints": ["incompressibility", "divergence_free",
"unknown", ...]
  },
  "pinn_loss_structure": {
    "term_groups": ["pde_residual", "bc", "ic", "data",
"aux_constraint"],
    "bc_types": ["dirichlet", "neumann", "periodic",
"mixed", "unknown"],
    "num_condition_groups": <int_or_null>,
    "has_complex_geometry_or_obstacles": true/false/unknown
  },
  "difficulty_drivers": {
    "dominant_mechanisms": ["convection", "diffusion", "reaction",
"coupling", "unknown"],
    "multi_scale_indicators": ["small_parameter", "thin_layers",
"high_frequency_ic", "large_domain", "unknown"],
    "stiffness_level": "none | mild | moderate | severe | unknown",
    "stiffness_sources": ["temporal", "spatial", "small_parameter",
"nonlinear", "multi_field_coupling", "unknown"]
  }
}
```

```

    },
    "pinn_risk_profile": {
        "gradient_imbalance_risk": "low | medium | high | unknown",
        "sampling_bottleneck_risk": "low | medium | high | unknown",
        "constraint_heterogeneity_risk": "low | medium | high | unknown",
        "overall_training_difficulty": "low | medium | high | unknown",
        "confidence_notes": "short"
    },
}

```

PART C | One-paragraph decision rationale (very short)

A tight paragraph explaining which risk dominates (gradient imbalance vs heterogeneous conditions vs sampling), and therefore which method class is most directly targeted, with explicit uncertainty if needed.

```

"""
    agent=pde_analyst_agent,
    inputs={"pde": "{{pde}}"},
    output_key="pde_analysis",
)

```

The analyst produces a structured traits JSON, which is then used as the primary input for method ranking.

Method Ranker

```

optimizer_selector_agent = Agent(
    role="PINN optimization method ranker",
    goal=(
        "Given PDE traits and a table of optimization methods
        for PINNs, "
        "rank the methods from most to least suitable for this PDE
        and briefly explain why."
    ),
    backstory=(
        "You are an expert in
        training Physics-Informed Neural Networks (PINNs). "
        "You know how PDE traits
        (stiffness, shocks, smoothness, scaling, IC/BC sensitivity, "
        "temporal structure, chaos)
        affect optimization and loss balancing.\n"
        "You ALWAYS base your ranking only on:\n"
        "- the structured PDE traits JSON from the previous agent,\n"
        "- and the method descriptions in the CSV table "
        " ('When it performs best',
        'When it performs poorly', 'Convergence behavior').\n"
        "You do not invent new methods and
        do not add properties that are not in the table. "
        "Your job is to compare, rank, and explain as simply
        and clearly as possible."
    ),
    llm=hf_llm,
    verbose=True,
    memory=False,
)

```

```

optimizer_selection_task = Task(
    description=(
        "You are the second-stage agent in a two-step pipeline for
        PINN optimization.\n\n"
        "You receive three inputs:\n"
        "1) Raw PDE text:\n"
        "{pde}\n\n"
        "2) Structured PDE analysis from the previous agent.
        It contains:\n"
        " - PART A: human-readable analysis (you may skim it),\n"

```

```

" - PART B: a machine-readable traits JSON
(this is your MAIN source),\n"
"   with fields such as:\n"
"     'pde_type', 'dominant_mechanisms', 'stiffness_level',
'regularity_issues',\n"
"     'loss_imbalance_risk', 'gradient_pathology_risk',
'sampling_difficulty',\n"
"     'temporal_causality_strength',
'long_time_horizon_difficulty', etc.\n\n"
"3) A CSV table of candidate optimization methods for PINN
training with columns:\n"
"  - 'Method'\n"
"  - 'When it performs best (PDE traits)'\n"
"  - 'When it performs poorly or gives little benefit'\n"
"  - 'Convergence behavior'\n\n"
"CSV table:\n"
"{methods_table}\n\n"
"Your task:\n"
"1) Read the PDE traits JSON (PART B) and identify the key
traits of this PDE:\n"
"   e.g. stiffness_level, regularity_issues (shocks, layers,
chaos),\n"
"   loss_imbalance_risk, gradient_pathology_risk,
sampling_difficulty,\n"
"   temporal_causality_strength,
long_time_horizon_difficulty.\n"
"2) For EACH method in the CSV:\n"
"   - Compare the PDE traits to 'When it performs best (PDE
traits)'. \n"
"   - Compare the PDE traits to 'When it performs poorly or
gives little benefit'. \n"
"   - Decide if this method is a GOOD, MEDIUM, or POOR match
for this PDE.\n"
"3) Use the 'Convergence behavior' column only to break ties
and refine the order among\n"
"   methods that are at least MEDIUM or GOOD matches.\n"
"4) Produce a single ranking of all methods from most to
least suitable for this PDE.\n\n"
"Important rules:\n"
"- Use ONLY the PDE traits JSON and the CSV descriptions.
Do NOT invent new methods.\n"
"- If the 'poorly' column clearly matches the PDE traits
(e.g. method is bad for shocks, "
"   and the PDE has strong shocks),
treat this method as a POOR match and rank it near the end.\n"
"- Always choose exactly ONE method as the best overall
(rank = 1).\n"
),
expected_output=(
"Return a **valid JSON** object with this structure
(no trailing commas):\n"
"{\n"
'  "primary_method": "<name of the best method>"\n'
'  "ranking": [\n'
'    {\n'
'      "method": "<name>"\n'
'      "rank": 1,\n'
'      "match": "good | medium | poor"\n'
'      "reason": "1-3 short sentences linking specific PDE
traits to the CSV descriptions."\n'
'    },\n'
'    {\n'
'      "method": "<name>"\n'
'      "rank": 2,\n'
'      "match": "good | medium | poor"\n'

```

```

        \ "reason\": \ "... \ "\n'
        } \n'
        // one object for each method, sorted from best (rank=1)
        to worst \n'
        ], \n'
        \ "notes\": \ "<optional brief comment: e.g. top-2 are very
        close, or a suggestion to combine methods> \ "\n'
        } \n \n"
        "Guidelines: \n"
        "- Keep reasons short but concrete: always mention at least
        one PDE trait name \n"
        " (e.g. 'stiffness_level=severe', 'sampling_difficulty=high')
        and one phrase from the CSV \n"
        " ('performs best for stiff multi-loss problems', 'performs
        poorly when shocks are present', etc.) \n"
        "- Do NOT output anything outside of the JSON object."
    ),
    agent=optimizer_selector_agent,
    context=[pde_analysis_task],
    markdown=True,
    output_file="report.json",
)

```

Below we report representative raw outputs from both stages; occasional logging/casting errors are kept verbatim to reflect the exact runtime traces.

Ranking Results Damped Oscillator *Note:* The following lines starting with Logger failed casting... indicate a parsing issue in the logging layer, not the ranking logic itself.

1. ****PDE Type****:
 - Class: ODE (Ordinary Differential Equation)
 - Time-dependent: Yes
 - Linear: Yes
 - Dimension: 1 (time only)
 - Number of Fields: 1
 - Meaning for PINN difficulty: The ODE nature simplifies the problem compared to PDEs, but the damping term introduces a non-trivial temporal dynamics that requires careful handling for stability and convergence.
2. ****Dominant Mechanisms****: Diffusion (from the second-order derivative in time) and reaction (from the first-order and zeroth-order terms).
3. ****Stability****: Decays to steady state. The equation models a damped harmonic oscillator where energy dissipates over time, leading to a stable decay without oscillations or blow-up.
4. ****Regularity****: Mostly smooth. There are no shocks, fronts, or other irregularities; the solution will smoothly approach zero over time.
5. ****Stiffness****: Mild. The presence of both first-order and second-order time derivatives suggests some stiffness due to rapid changes in the early stages, but the overall dynamics are well-behaved.
6. ****Characteristic Scales****: Temporal scale is on the order of the inverse of the dominant frequency (around 1/20), but the damping ensures that the system settles quickly. No significant spatial scales since it's an ODE.
7. ****IC/BC Sensitivity****: Low. The initial condition directly sets the starting point of the solution trajectory, and the boundary condition is implicitly defined by the ODE itself.

8. **Symmetries & Conserved Quantities**: None. The equation does not exhibit any symmetries or conservation laws beyond the trivial ones.

9. **Numerical Challenges for PINNs**:

- Loss imbalance risk: Medium (the balance between PDE loss and initial condition loss may require tuning).
- Gradient pathology risk: Low (the linear nature reduces the likelihood of severe gradient issues).
- Sampling difficulty: Low (uniform sampling should suffice given the smooth nature of the solution).
- Temporal issues: Weak (time-marching might be beneficial but is not strictly necessary).
- Long-time error accumulation: Low (the system stabilizes quickly, minimizing long-term error propagation).

10. **Confidence**: High confidence in the classification of the PDE type, linearity, and dominant mechanisms. Moderate confidence in the assessment of stability, regularity, and stiffness. Low confidence in detailed characteristics like specific scales and sensitivities due to limited information.

PART B | Machine-readable traits JSON (valid JSON, no comments):

```
```json
{
 "pde_type": "ode",
 "linearity": "linear",
 "num_spatial_dims": 0,
 "num_fields": 1,
 "dominant_mechanisms": ["reaction", "diffusion"],
 "stability_behavior": "decays_to_steady",
 "regularity_issues": ["smooth"],
 "stiffness_level": "mild",
 "stiffness_sources": ["temporal"],
 "characteristic_scales": {
 "spatial": "unknown",
 "temporal": "around 1/20",
 "key_parameters": ["damping_coefficient", "frequency"]
 },
 "ic_bc_sensitivity": "low",
 "symmetries": "none",
 "conserved_quantities": "none",
 "loss_imbalance_risk": "medium",
 "gradient_pathology_risk": "low",
 "sampling_difficulty": "low",
 "temporal_causality_strength": "weak",
 "long_time_horizon_difficulty": "low",
 "overall_pinn_difficulty": "medium",
 "confidence_notes": "High confidence in basic PDE classification and properties. Moderate confidence in stability and stiffness. Low confidence in detailed characteristic scales."
}
``` to integer
```

```
PDE analysis { osci { characteristics
Logger failed casting log level ```markdown
{
  "primary_method": "LBPINN",
  "ranking": [
    {
      "method": "LBPINN",
      "rank": 1,
      "match": "good",
      "reason": "The PDE has a medium risk of loss imbalance
(\`loss_imbalance_risk: \`medium\`), which LBPINN handles well
```

```

    by adapting loss weights automatically (`When it performs best
    (PDE traits)` includes stiff PDEs and IC/BC-sensitive models)."
```

- {
 "method": "DB-PINN",
 "rank": 2,
 "match": "good",
 "reason": "Although the PDE has low sensitivity to IC/BC
 (`ic_bc_sensitivity: \"low\"`), DB-PINN is still a good fit
 because it addresses complex PDEs with diverse constraints
 (`When it performs best (PDE traits)` mentions problems with
 many BC/IC/data groups)."
- {
 "method": "GradNorm",
 "rank": 3,
 "match": "medium",
 "reason": "While the PDE has a low risk of gradient pathology
 (`gradient_pathology_risk: \"low\"`), GradNorm could still help
 manage the mild stiffness (`stiffness_level: \"mild\"`) by
 adjusting weights based on gradient norms."
- {
 "method": "RAR / RAD / RAR-D",
 "rank": 4,
 "match": "poor",
 "reason": "The PDE is smooth with no shocks or fronts
 (`regularity_issues: [\"smooth\"]`), making adaptive
 sampling methods less effective (`When it performs
 poorly or gives little benefit` includes problems
 without shocks or layers)."
- {
 "method": "Causal Training / Time-Marching PINN",
 "rank": 5,
 "match": "poor",
 "reason": "Temporal causality strength is weak
 (`temporal_causality_strength: \"weak\"`) and there's
 no indication of long-time horizon difficulty
 (`long_time_horizon_difficulty: \"low\"`), so causal
 training methods are not ideal for this PDE."

```

  }
  ],
  "notes": "LBPINN and DB-PINN are both strong candidates
  due to their ability to handle loss imbalances, though
  LBPINN is slightly more tailored to the specific needs of
  this PDE."
}

```

Cylinder Wake

PART A | Short structured analysis (bulleted or numbered):

1. PDE Type: Mixed (elliptic-parabolic-hyperbolic) due to the incompressibility constraint and Navier-Stokes equations, time-dependent, quasilinear. The presence of nonlinearity and multiple coupled fields makes this PDE type challenging for PINNs, especially in capturing the complex flow dynamics accurately.
2. Dominant Mechanisms: Convection (nonlinear advection terms), diffusion (viscous terms), and pressure gradient terms. Nonlinear effects dominate due to the quadratic convective terms.
3. Stability: Oscillatory with possible transient growth of

perturbations, but overall bounded due to dissipative nature of viscous terms. No clear indication of blow-up within the specified time horizon.

4. Regularity: Steep gradients and possibly chaotic patterns due to the transition from laminar to turbulent flow regimes, especially near the cylinder. High sensitivity to initial conditions and parameters.

5. Stiffness: Moderate, mainly due to the interplay between convection and diffusion terms, and the nonlinearity in the convective terms. Temporal stiffness could arise from the rapid changes in flow structures.

6. Characteristic Scales: Spatial scales include the cylinder diameter ($D=1$), inlet/outlet width (26 units), and flow length (40 units). Temporal scales are related to the vortex shedding frequency and viscous diffusion timescales. Key parameters are viscosity (ν) and Reynolds number (Re).

7. IC/BC Sensitivity: Medium to high, particularly sensitive to initial flow disturbances and boundary conditions at the inlet and cylinder surface.

8. Symmetries & Conserved Quantities: Periodic in y direction due to boundary conditions. Energy is likely conserved in the absence of external forcing or dissipation.

9. Numerical Challenges for PINNs:

- Loss imbalance risk: High, due to the combination of different physical mechanisms and scales.
- Gradient pathology risk: High, due to steep gradients and nonlinearities.
- Sampling difficulty: High, due to the need for capturing fine-scale features and coherent structures.
- Temporal causality strength: Strong, due to the time-dependent nature and the requirement for accurate tracking of evolving flow patterns.
- Long-time horizon difficulty: Medium to high, due to potential accumulation of errors over time.

10. Confidence: High confidence in PDE type, dominant mechanisms, characteristic scales, and symmetry. Medium confidence in stability, regularity, stiffness, and IC/BC sensitivity. Low confidence in exact long-term behavior and specific numerical challenges without further simulation results.

PART B | Machine-readable traits JSON (valid JSON, no comments):

```
{
  "pde_type": "mixed",
  "linearity": "quasilinear",
  "num_spatial_dims": 2,
  "num_fields": 3,
  "dominant_mechanisms": ["convection", "diffusion", "pressure_gradient"],
  "stability_behavior": "bounded_oscillatory",
  "regularity_issues": ["steep_gradients", "chaotic_patterns", "unknown"],
  "stiffness_level": "moderate",
  "stiffness_sources": ["spatial", "nonlinear", "unknown"],
  "characteristic_scales": {
    "spatial": "cylinder_diameter=1, inlet_outlet_width=26, flow_length=40",
    "temporal": "vortex_shedding_frequency, viscous_diffusion_timescale",
    "key_parameters": ["viscosity", "reynolds_number"]
  },
  "ic_bc_sensitivity": "medium",
  "symmetries": ["periodic_in_y"],
  "conserved_quantities": ["energy", "unknown"],
  "loss_imbalance_risk": "high",
```

```

"gradient_pathology_risk": "high",
"sampling_difficulty": "high",
"temporal_causality_strength": "strong",
"long_time_horizon_difficulty": "medium",
"overall_pinn_difficulty": "high",
"confidence_notes": "High confidence in PDE type, dominant
mechanisms, characteristic scales, and symmetry. Medium
confidence in stability, regularity, stiffness, and IC/BC
sensitivity. Low confidence in exact long-term behavior
and specific numerical challenges."
}" to integer

```

```

PDE analysis { cylinder { characteristics
Logger failed casting log level "```json
{
  "primary_method": "DB-PINN",
  "ranking": [
    {
      "method": "DB-PINN",
      "rank": 1,
      "match": "good",
      "reason": "The DB-PINN method is well-suited for
complex PDEs with diverse constraints, which aligns
with the high loss imbalance risk and gradient
pathology risk identified in the cylinder wake
problem."
    },
    {
      "method": "GradNorm",
      "rank": 2,
      "match": "good",
      "reason": "GradNorm excels in multi-loss setups with
strong gradient imbalance, which is relevant given the
moderate stiffness level and high gradient pathology
risk in the Navier-Stokes equations."
    },
    {
      "method": "LBPINN",
      "rank": 3,
      "match": "medium",
      "reason": "LBPINN adapts loss weights through learned
uncertainty, beneficial for IC/BC-sensitive models and
stiff PDEs, but its performance is limited when
gradients differ significantly, which is common in
this problem."
    },
    {
      "method": "RAR-D",
      "rank": 4,
      "match": "medium",
      "reason": "RAR-D adapts collocation point distribution
based on PDE residuals, useful for problems with steep
gradients, but it may undersample smooth regions,
limiting its effectiveness here."
    },
    {
      "method": "Causal Training",
      "rank": 5,
      "match": "poor",
      "reason": "Causal Training is designed for time-
dependent PDEs with long horizons and strong IC
sensitivity, but the problem's temporal causality
strength is strong rather than extremely long, making
this method less suitable."
    }
  ]
}

```

```

    }
  ],
  "notes": "DB-PINN and GradNorm are both strong candidates
due to their ability to handle the complexity and gradient
issues of the Navier-Stokes equations. LBPINN and RAR-D
offer additional benefits but have limitations that reduce
their overall suitability compared to the top two methods."
}

```

1D Viscous Burgers

PART A | Short structured analysis (bulleted or numbered):

1. **PDE Type:** Quasilinear, parabolic, time-dependent, 1D, single field. This PDE is challenging for PINNs due to its nonlinear nature and the presence of both convection and diffusion terms, especially when the viscosity is very small
2. **Dominant Mechanisms:** Convection and diffusion. The small viscosity ($= 1e-4$) indicates a convection-dominated regime.
3. **Stability:** Unknown. The PDE could exhibit complex behavior including shocks, but the exact stability behavior is uncertain without further analysis.
4. **Regularity:** Likely to have shocks or thin boundary layers due to the convection-dominated regime and the small viscosity.
5. **Stiffness:** Moderate to severe, primarily due to the small viscosity parameter leading to thin boundary layers/shocks and the nonlinear convection term.
6. **Characteristic Scales:** Spatial scale around 1 unit (domain size), temporal scale likely on the order of 1 unit (time interval). Key parameter: $= 1e-4$.
7. **IC/BC Sensitivity:** High. The multi-frequency initial condition and Dirichlet boundary conditions may lead to sensitive solutions.
8. **Symmetries & Conserved Quantities:** None/unknown. No clear symmetries or conserved quantities are evident from the given information.
9. **Numerical Challenges for PINNs:**
 - Loss imbalance risk: High due to the combination of convection and diffusion.
 - Gradient pathology risk: High due to the presence of shocks and thin layers.
 - Sampling difficulty: High, especially near shocks or thin layers where more samples are needed.
 - Temporal issues: Strong need for causal/time-marching due to the potential for rapid changes in solution.
 - Long-time error accumulation: Medium to high due to the possibility of chaotic behavior or instability.
10. **Confidence:** High on PDE type, dominant mechanisms, characteristic scales, and stiffness; medium on stability, regularity, IC/BC sensitivity, and numerical challenges; low on symmetries and conserved quantities.

PART B | Machine-readable traits JSON (valid JSON, no comments):

```

{
  "pde_type": "quasilinear_parabolic",
  "linearity": "quasilinear",
  "num_spatial_dims": 1,
  "num_fields": 1,
  "dominant_mechanisms": ["convection", "diffusion"],
  "stability_behavior": "unknown",
  "regularity_issues": ["shocks", "thin_boundary_layers", "unknown"],
  "stiffness_level": "severe",
  "stiffness_sources": ["small_viscosity", "nonlinear"],
  "characteristic_scales": {
    "spatial": "1 unit",

```

```

    "temporal": "1 unit",
    "key_parameters": ["viscosity = 1e-4"]
  },
  "ic_bc_sensitivity": "high",
  "symmetries": "unknown",
  "conserved_quantities": "unknown",
  "loss_imbalance_risk": "high",
  "gradient_pathology_risk": "high",
  "sampling_difficulty": "high",
  "temporal_causality_strength": "strong",
  "long_time_horizon_difficulty": "medium",
  "overall_pinn_difficulty": "high",
  "confidence_notes": "High confidence in PDE type, dominant
mechanisms, characteristic scales, and stiffness. Medium
confidence in stability, regularity, IC/BC sensitivity,
and numerical challenges. Low confidence in symmetries and
conserved quantities."
}" to integer
{
  "primary_method": "LBPINN",
  "ranking": [
    {
      "method": "LBPINN",
      "rank": 1,
      "match": "good",
      "reason": "The PDE has a high loss imbalance risk and
stiffness level, which LBPINN handles well by adapting
loss weights and improving late-stage stability."
    },
    {
      "method": "GradNorm",
      "rank": 2,
      "match": "good",
      "reason": "The PDE exhibits high gradient pathology
risk and loss imbalance, making GradNorm's focus on
per-loss gradient norms beneficial despite its
sensitivity to noise."
    },
    {
      "method": "DB-PINN",
      "rank": 3,
      "match": "medium",
      "reason": "While DB-PINN excels in complex PDEs with
diverse constraints, the current PDE's complexity does
not fully justify its high compute cost."
    },
    {
      "method": "RAR / RAD / RAR-D",
      "rank": 4,
      "match": "medium",
      "reason": "The PDE has high sampling difficulty due to
shocks and thin layers, but the method's strength lies
in shock regions, not necessarily balancing losses."
    },
    {
      "method": "Causal Training / Time-Marching PINN",
      "rank": 5,
      "match": "poor",
      "reason": "The PDE's temporal causality is strong, but
the method's effectiveness is limited to problems with
long horizons, which is not the primary issue here."
    }
  ],
  "notes": "Top two methods (LBPINN and GradNorm) are very

```

```

    close. Consider combining them for enhanced performance."
}

```

Kovaszny Flow

PART A | Short structured analysis (bulleted or numbered):

1. **PDE Type**: Elliptic, stationary, quasilinear (due to the nonlinearity in the convective terms and the coupling with pressure). The stationary nature simplifies temporal aspects but introduces challenges due to the coupled system of equations.
2. **Dominant Mechanisms**: Convection (dominated by the velocity field) and diffusion (controlled by viscosity).
3. **Stability**: The steady-state solution is stable; no blow-up or chaotic behavior expected within the given domain and parameters.
4. **Regularity**: Mostly smooth, though potential for complex flow structures like vortices and boundary layers near walls or corners.
5. **Stiffness**: Moderate, primarily due to the nonlinearity in the convective terms and the coupling between the velocity and pressure fields.
6. **Characteristic Scales**: Spatial scales determined by the domain size and Reynolds number; temporal scale is irrelevant due to stationarity.
7. **IC/BC Sensitivity**: Medium to high, particularly sensitive to boundary conditions due to the incompressibility constraint and the nature of fluid dynamics.
8. **Symmetries & Conserved Quantities**: None explicitly stated, but conservation of mass and possibly energy should hold.
9. **Numerical Challenges for PINNs**:
 - Loss imbalance risk: High due to the coupling of velocity and pressure.
 - Gradient pathology risk: Moderate due to nonlinear convective terms.
 - Sampling difficulty: Medium to high, especially near boundaries where gradients may be steep.
 - Temporal issues: Not applicable due to stationarity.
 - Long-time error accumulation: Not relevant due to stationarity.
10. **Confidence**: High on PDE type, dominant mechanisms, stability, and characteristic scales; medium on regularity and stiffness; low on symmetries and conserved quantities due to lack of explicit information.

PART B | Machine-readable traits JSON (valid JSON, no comments):

```

{
  "pde_type": "elliptic",
  "linearity": "quasilinear",
  "num_spatial_dims": 2,
  "num_fields": 3,
  "dominant_mechanisms": ["convection", "diffusion"],
  "stability_behavior": "decays_to_steady",
  "regularity_issues": ["steep_gradients", "smooth"],
  "stiffness_level": "moderate",
  "stiffness_sources": ["nonlinear", "coupling"],
  "characteristic_scales": {
    "spatial": "domain_size",
    "temporal": "irrelevant",
    "key_parameters": ["Reynolds_number"]
  },
  "ic_bc_sensitivity": "high",
  "symmetries": "unknown",

```

```

"conserved_quantities": "unknown",
"loss_imbalance_risk": "high",
"gradient_pathology_risk": "medium",
"sampling_difficulty": "medium",
"temporal_causality_strength": "none",
"long_time_horizon_difficulty": "low",
"overall_pinn_difficulty": "medium",
"confidence_notes": "High confidence in PDE type, dominant
mechanisms, stability, and characteristic scales. Medium
confidence in regularity and stiffness. Low confidence in
symmetries and conserved quantities."
}" to integer

PDE analysis { kovasznai { characteristics
Logger failed casting log level "I now can give a great answer
```markdown
{
 "primary_method": "LBPINN",
 "ranking": [
 {
 "method": "LBPINN",
 "rank": 1,
 "match": "good",
 "reason": "The PDE has a high risk of loss imbalance
(`loss_imbalance_risk`: high) and moderate stiffness
(`stiffness_level`: moderate), making LBPINN well-
suited due to its strength in handling multi-physics
systems and stiff PDEs."
 },
 {
 "method": "DB-PINN",
 "rank": 2,
 "match": "good",
 "reason": "The PDE involves multiple fields and
constraints (`num_fields`: 3, `loss_imbalance_risk`:
high), aligning with DB-PINN's ability to handle
complex PDEs with diverse constraints."
 },
 {
 "method": "GradNorm",
 "rank": 3,
 "match": "medium",
 "reason": "While the PDE has a medium gradient
pathology risk (`gradient_pathology_risk`: medium), it
does not exhibit strong gradient imbalance that would
fully leverage GradNorm's strengths."
 },
 {
 "method": "RAR / RAD / RAR-D",
 "rank": 4,
 "match": "poor",
 "reason": "The PDE does not have prominent shocks,
layers, or steep gradients that would significantly
benefit from adaptive sampling (`regularity_issues`:
`steep_gradients`, `smooth`)."
 },
 {
 "method": "Causal Training / Time-Marching PINN",
 "rank": 5,
 "match": "poor",
 "reason": "The PDE is stationary (`pde_type`:
elliptic, `temporal_causality_strength`: none), making
the causal training approach irrelevant."
 }
]
}

```

```

 }
],
 "notes": "Top two methods (LBPINN and DB-PINN) are closely
 matched, both being highly suitable for the given PDE.
 Consider evaluating them further based on specific
 implementation details and resource availability."
}

```

### Beltrami Flow

**\*\*PART A | Short structured analysis\*\***

1. **\*\*PDE Type\*\***: The PDE is a system of three coupled nonlinear parabolic PDEs (Navier-Stokes equations) with an additional incompressibility constraint. It is time-dependent and 3-dimensional with four fields (three velocity components and one pressure field). This system is known to be highly nonlinear and challenging for PINNs due to the coupling between velocity and pressure, as well as the nonlinearity in convective terms.
2. **\*\*Dominant Mechanisms\*\***: Convection and diffusion dominate the dynamics. The nonlinear convective terms (like  $uu_x$ ) are significant, leading to complex flow behaviors.
3. **\*\*Stability\*\***: Unknown. While the Beltrami flow is known to have smooth solutions, the overall stability of the system over the entire time horizon  $\in([0, 1])$  is uncertain without specific information on the initial and boundary conditions.
4. **\*\*Regularity\*\***: Mostly smooth. The Beltrami flow typically produces smooth solutions, but the Navier-Stokes equations can exhibit complex behaviors under certain conditions, potentially introducing irregularities.
5. **\*\*Stiffness\*\***: Moderate. The combination of convection and diffusion introduces stiffness, particularly in regions where the convective term dominates. Temporal stiffness can also arise due to the need for resolving small spatial scales accurately.
6. **\*\*Characteristic Scales\*\***: Spatial scales are determined by the domain size  $\in([-1, 1]^3)$ , while temporal scales depend on the Reynolds number ( $Re=1$ ) and the characteristic velocity. Key parameters include the Reynolds number and the viscosity.
7. **\*\*IC/BC Sensitivity\*\***: Medium. The Navier-Stokes equations are generally sensitive to initial and boundary conditions, especially in turbulent regimes, although the Beltrami flow is known to be more stable.
8. **\*\*Symmetries & Conserved Quantities\*\***: None or unknown. The specific form of the Beltrami flow may introduce symmetries, but without explicit details, it's uncertain. Energy conservation is a common property of the Navier-Stokes equations.
9. **\*\*Numerical Challenges for PINNs\*\***:
  - Loss imbalance risk: High.
  - Balancing the PDE loss with the incompressibility constraint and boundary conditions is challenging.

- Gradient pathology risk: High.  
Nonlinear convective terms can lead to steep gradients and loss of gradient information.
- Sampling difficulty: Medium.  
Uniform sampling might suffice for smooth solutions, but adaptive sampling could improve accuracy near boundaries or in regions of high gradients.
- Temporal causality strength: Strong.  
Time-marching strategies may be necessary due to the temporal nature of the problem.
- Long-time error accumulation: Medium. Error accumulation over time can be significant, especially if the solution becomes complex.

10. **Confidence**: High confidence in the PDE type, dominant mechanisms, and numerical challenges. Medium confidence in stability, regularity, and IC/BC sensitivity. Low confidence in specific symmetries and conserved quantities.

**PART B | Machine-readable traits JSON**

```
```json
{
  "pde_type": "parabolic",
  "linearity": "quasilinear",
  "num_spatial_dims": 3,
  "num_fields": 4,
  "dominant_mechanisms": ["convection", "diffusion"],
  "stability_behavior": "unknown",
  "regularity_issues": ["smooth", "unknown"],
  "stiffness_level": "moderate",
  "stiffness_sources": ["nonlinear", "unknown"],
  "characteristic_scales": {
    "spatial": "[-1, 1]^3",
    "temporal": "[0, 1]",
    "key_parameters": ["Reynolds_number"]
  },
  "ic_bc_sensitivity": "medium",
  "symmetries": "unknown",
  "conserved_quantities": "unknown",
  "loss_imbalance_risk": "high",
  "gradient_pathology_risk": "high",
  "sampling_difficulty": "medium",
  "temporal_causality_strength": "strong",
  "long_time_horizon_difficulty": "medium",
  "overall_pinn_difficulty": "high",
  "confidence_notes": "High confidence in PDE type, dominant mechanisms, and numerical challenges. Medium confidence in stability, regularity, and IC/BC sensitivity. Low confidence in specific symmetries and conserved quantities."
}
``` to integer
```

```
PDE analysis { beltrami { characteristics
Logger failed casting log level ```json
{
 "primary_method": "DB-PINN",
 "ranking": [
 {
 "method": "DB-PINN",
 "rank": 1,
 "match": "good",
 "reason": "The DB-PINN method excels in handling
```

```

 complex PDEs with diverse constraints, which aligns
 well with the moderate stiffness level and high loss
 imbalance risk of the Navier-Stokes equations."
 },
 {
 "method": "GradNorm",
 "rank": 2,
 "match": "good",
 "reason": "GradNorm performs best for multi-loss
 setups with strong gradient imbalance, addressing the
 high gradient pathology risk and moderate stiffness
 level of the Navier-Stokes equations."
 },
 {
 "method": "LBPINN",
 "rank": 3,
 "match": "medium",
 "reason": "LBPINN adapts loss weights via learned
 uncertainty, suitable for stiff PDEs and multi-physics
 systems, but it may not fully address the high
 gradient pathology risk."
 },
 {
 "method": "Causal Training / Time-Marching PINN",
 "rank": 4,
 "match": "medium",
 "reason": "Causal Training is beneficial for
 time-dependent PDEs with strong IC sensitivity, but it
 may not directly address the high loss imbalance and
 gradient pathology risks."
 },
 {
 "method": "RAR / RAD / RAR-D",
 "rank": 5,
 "match": "poor",
 "reason": "RAR methods focus on adaptive sampling,
 which is less relevant for the Navier-Stokes equations
 given the high loss imbalance and gradient pathology
 risks."
 }
],
"notes": "Top two methods (DB-PINN and GradNorm) are very
close in suitability, with DB-PINN having a slight edge
due to its dual-balancing mechanism."
}

```

## E BUILDING THE FINAL RANKING

We construct a benchmark (ground-truth, GT) ranking of training strategies from training logs by combining (i) a primary solution-quality signal and (ii) secondary indicators of convergence and loss-term balance. We then evaluate an LLM-produced ranking against this GT benchmark using tie-aware, top-heavy ranking metrics and decision-oriented measures.

### E.1 STEP 1: GROUND-TRUTH (GT) RANKING OF METHODS

Fix a PDE task and a set of  $N$  candidate methods indexed by  $i \in \{1, \dots, N\}$ . Let training progress be indexed by  $t \in \{0, \dots, T\}$  (epochs). Let  $\ell_{i,k}(t)$  denote the  $k$ -th component of the composite training loss for method  $i$ , for  $k \in \mathcal{K}$  (e.g., PDE residual, boundary conditions, initial conditions), and let  $L_i(t)$  denote an (optional) logged aggregate total loss.

**Loss-term normalization.** To make heterogeneous loss components comparable, we normalize each term by its initial value:

$$\tilde{\ell}_{i,k}(t) = \frac{\ell_{i,k}(t)}{\ell_{i,k}(0) + \varepsilon}, \quad \varepsilon > 0. \quad (2)$$

**Epoch selection (endpoint definition).** All “final” metrics are evaluated at a *selected epoch*  $t_i^*$  (rather than necessarily at  $T$ ). We choose  $t_i^*$  by the following rule:

$$t_i^* = \begin{cases} \arg \min_t \text{MSE}_i(t), & \text{if an MSE-type metric is available,} \\ \arg \min_t \sum_{k \in \mathcal{K}} \ell_{i,k}(t), & \text{otherwise (unweighted component sum).} \end{cases} \quad (3)$$

This ensures a consistent endpoint even when explicit solution-error metrics are missing (e.g., for some PDEs). When multiple runs are available for the same method, we first align runs by keeping only those with the modal number of logged epochs. We then select a run-specific epoch  $t_i^*$  according to Eq. equation 3 and compute all scalar method-level statistics at  $t_i^*$ ; finally, we average these scalars across runs.

**Secondary (process) metrics (computed up to  $t_i^*$ ).** Using the normalized loss terms, we define:

(i) **Time-to- $\epsilon$  for all terms.** Given  $\epsilon$  (we use  $\epsilon = 10^{-2}$ ),

$$\tau_i(\epsilon) = \min \left\{ t \leq t_i^* : \forall k \in \mathcal{K}, \tilde{\ell}_{i,k}(t) \leq \epsilon \right\}, \quad (4)$$

and  $\tau_i(\epsilon)$  is undefined if the set is empty.

(ii) **AUC of normalized loss terms.** We quantify integrated optimization cost by summing the per-term areas under the normalized curves:

$$\text{AUC}_i = \sum_{k \in \mathcal{K}} \int_0^{t_i^*} \tilde{\ell}_{i,k}(t) dt, \quad (5)$$

approximated numerically (e.g., trapezoidal rule) on the epoch grid.

**Selected-epoch balance metric.** (iii) **Maximum normalized loss term at the selected epoch.** This penalizes methods where at least one constraint remains large at  $t_i^*$ :

$$L_i^{\max} = \max_{k \in \mathcal{K}} \tilde{\ell}_{i,k}(t_i^*). \quad (6)$$

**Primary solution-quality metric.** The primary objective is solution accuracy. We define  $Q_i$  as the *first available* metric in the following priority list (all are minimized):

$$Q_i \in \left\{ \underbrace{\text{MSE}_i^{\text{best}}}_{\min_t \text{MSE}_i(t)}, \underbrace{\text{MSE}_i(t_i^*)}_{\text{selected MSE}}, \underbrace{\text{ReL2}_i(t_i^*)}_{\text{selected relative } L^2}, \underbrace{L_i(t_i^*)}_{\text{selected total loss (fallback)}} \right\}. \quad (7)$$

When solution-quality metrics are not logged, we fall back to the selected (unweighted) total loss at  $t_i^*$ .

**Scale stabilization and within-task normalization.** For nonnegative metrics we apply a log-stabilization,

$$\phi(x) = \log(1 + x), \quad (8)$$

to  $\{Q_i, L_i^{\max}, \tau_i(\epsilon), \text{AUC}_i\}$  when defined. Then each metric is min–max normalized within the same PDE task:

$$\text{mm}(x_i) = \frac{x_i - \min_j x_j}{\max_j x_j - \min_j x_j}, \quad \text{mm}(x_i) \in [0, 1], \quad (9)$$

defined when  $\max_j x_j > \min_j x_j$ .

**Weighted GT score and GT ranking.** For each PDE we construct a single scalar score per method by combining the primary signal with secondary process indicators. Let the set of candidate metrics be

$$\mathcal{M} = \{Q, L^{\max}, \tau(\epsilon), \text{AUC}\}. \quad (10)$$

We include in the score only those metrics whose values are finite for at least one method in the task. Importantly, missing values for a method should *not* make it appear better. Therefore, after min–max normalization we assign missing entries the worst normalized value (1.0) before aggregation.

We assign a larger weight to the primary metric and a smaller, uniform weight to each secondary term:

$$w_Q = 1.0, \quad w_m = \lambda \text{ for all } m \in \mathcal{M} \setminus \{Q\}, \quad (11)$$

where  $\lambda$  is a fixed secondary weight (we use  $\lambda = 0.2$  in the main experiments). We normalize weights to sum to one over the metrics actually used:

$$\hat{w}_m = \frac{w_m}{\sum_{r \in \mathcal{M}_{\text{used}}} w_r}. \quad (12)$$

The GT score is the weighted sum of normalized (log-stabilized) costs:

$$\text{GTScore}_i = \sum_{m \in \mathcal{M}_{\text{used}}} \hat{w}_m \cdot \text{mm}(\phi(m_i)), \quad (13)$$

and the GT ranking is obtained by sorting in ascending order:

$$\text{GTRank}_i = \text{rank}(\text{GTScore}_i; \text{ascending}). \quad (14)$$

## E.2 STEP 2: EVALUATING AN LLM RANKING AGAINST GT

Let  $\text{LLMRank}_i$  denote the rank assigned by the LLM for the same set of methods. Both GT and LLM rankings may contain ties; we therefore use tie-aware evaluation metrics.

**Tie groups.** For any ranking column Rank define tie groups as

$$G_r = \{i : \text{Rank}_i = r\}. \quad (15)$$

We treat each tie group as exchangeable and compute either (i) expected metrics under uniform random tie-breaking within each tie group or (ii) tie-aware set-based decisions.

**(i) NDCG@3 (tie-aware, top-heavy).** We evaluate top-heavy ranking quality using NDCG@3. We derive graded gains from the dense GT rank. Let  $r_{\text{GT}}^{\text{dense}}(i) \in \{1, 2, \dots\}$  denote the dense rank (best is 1), and define

$$\text{rel}(i) = r_{\text{max}}^{\text{dense}} - r_{\text{GT}}^{\text{dense}}(i) + 1, \quad g(i) = 2^{\text{rel}(i)} - 1. \quad (16)$$

For a ranked list  $\pi$  (best to worst), the DCG at cutoff  $k$  is

$$\text{DCG}@k(\pi) = \sum_{j=1}^k \frac{g(\pi_j)}{\log_2(j+1)}. \quad (17)$$

With ties in the LLM ranking, we compute the *expected* DCG by treating each tie group as exchangeable: for a tie group spanning positions  $\{j, \dots, j+m-1\}$  we use the *mean gain* of its elements and multiply by the sum of position discounts  $\sum_{p=j}^{j+m-1} 1/\log_2(p+1)$ . We then report

$$\text{NDCG}@3 = \frac{\text{DCG}@3(\pi^{\text{LLM}})}{\text{DCG}@3(\pi^{\text{ideal}})}, \quad (18)$$

where  $\pi^{\text{ideal}}$  is the ideal ordering induced by the GT ranks.

**(ii) Pairwise accuracy via Kendall’s  $\tau_b$  (tie-aware global agreement).** To quantify overall agreement between two rankings (beyond top- $k$ ), we compute Kendall’s  $\tau_b$  between GTRank and LLMRank, which accounts for ties. We report a bounded “pairwise accuracy” score

$$\text{PairwiseAcc} = \frac{\tau_b + 1}{2} \in [0, 1], \quad (19)$$

where  $\text{PairwiseAcc} = 1$  indicates perfect concordance and 0 indicates perfect discordance.

**(iii) Decision-oriented metrics: Top-1 hit and Hit@2.** These metrics directly reflect method selection decisions.

Let the set of GT-best methods be

$$B = \{i : \text{GTRank}_i = \min_j \text{GTRank}_j\}. \quad (20)$$

Let the LLM top tie group be

$$T_1 = \{i : \text{LLMRank}_i = \min_j \text{LLMRank}_j\}. \quad (21)$$

We report

$$\text{Top1Hit} = \mathbb{I}[B \cap T_1 \neq \emptyset]. \quad (22)$$

For Hit@2, we construct a tie-aware top-2 set  $T_2$  by taking the union of LLM tie groups from the top until the set cardinality is at least 2:

$$T_2 = \bigcup_{r \in \{r_{\text{min}}, r_{\text{min}}+1, \dots\}} G_r^{\text{LLM}} \quad \text{s.t.} \quad |T_2| \geq 2. \quad (23)$$

We then report

$$\text{Hit@2} = \mathbb{I}[B \cap T_2 \neq \emptyset]. \quad (24)$$

**(iv) Regret of the LLM top choice (normalized).** To quantify the practical cost of selecting the LLM top method(s), we compute regret on the same *primary solution-quality* metric used in GT construction (Eq. 7). Let  $Q(i)$  denote this primary metric for method  $i$  (lower is better), and define the best and worst values among methods with defined  $Q$ :

$$Q^* = \min_i Q(i), \quad Q^{\max} = \max_i Q(i). \quad (25)$$

Under uniform selection within the LLM top tie group  $T_1$ ,

$$\mathbb{E}[Q | T_1] = \frac{1}{|T_1|} \sum_{i \in T_1} Q(i). \quad (26)$$

We report the *normalized regret* (bounded in  $[0, 1]$ ),

$$\text{Regret}_{\text{norm}} = \begin{cases} \frac{\mathbb{E}[Q | T_1] - Q^*}{Q^{\max} - Q^*}, & \text{if } Q^{\max} > Q^*, \\ 0, & \text{if } Q^{\max} = Q^*, \end{cases} \quad (27)$$

clipped to  $[0, 1]$  for numerical robustness. For reporting convenience (so that higher is better), we additionally define

$$\text{BestPickQuality} = 1 - \text{Regret}_{\text{norm}} \in [0, 1]. \quad (28)$$

**Macro-averaging across PDE tasks.** To summarize performance across a benchmark suite of PDEs, we macro-average each metric over tasks (each PDE contributes equally) and compute 95% confidence intervals via nonparametric bootstrap over PDEs.

## F LLM RANKING QUALITY METRICS

This section reports the ranking quality of the LLM-based method selector against the ground-truth (GT) ordering, both as a macro-average across PDE tasks and broken down per PDE.

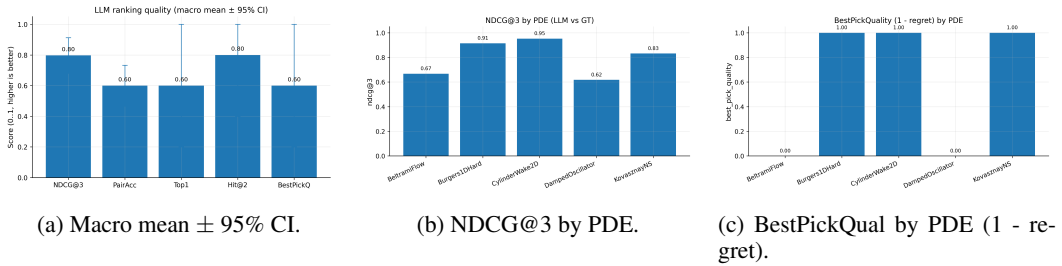


Figure 2: LLM ranking quality across benchmarks. Left: macro-averaged metrics (mean  $\pm$  95% CI). Middle/right: per-PDE breakdown for NDCG@3 and BestPickQual.

Table 9 reports the macro-average scores underlying Fig. 2 (left). Table 10 provides the corresponding per-PDE values, matching the per-task trends in Fig. 2 (middle/right).

Table 9: LLM ranking quality across PDE tasks (macro-average over tasks; 95% bootstrap CI).

Metric	Mean	95% CI
NDCG@3 ( $\uparrow$ )	0.7967	[0.6804, 0.9130]
PairwiseAcc ( $\uparrow$ )	0.6000	[0.4667, 0.7333]
Top1Hit ( $\uparrow$ )	0.6000	[0.2000, 1.0000]
Hit@2 ( $\uparrow$ )	0.8000	[0.4000, 1.0000]
BestPickQual ( $\uparrow$ )	0.5998	[0.1999, 0.9997]

Table 10: Per-task ranking quality of the LLM relative to the GT benchmark.

PDE	NDCG@3	PairAcc	Top1	Hit@2	BestPickQual
BeltramiFlow	0.6676	0.5000	0.00	1.00	0.0000
Burgers1DHard	0.9146	0.6667	1.00	1.00	1.0000
CylinderWake2D	0.9522	0.8333	1.00	1.00	0.9995
DampedOscillator	0.6175	0.3333	0.00	0.00	0.0000
KovasznyNS	0.8315	0.6667	1.00	1.00	0.9996

Across tasks, NDCG@3 and Hit@2 are generally high, while PairwiseAcc and BestPickQual reveal sharper failures on BeltramiFlow and DampedOscillator (low agreement and high top-choice regret), despite strong performance on Burgers1DHard, CylinderWake2D, and KovasznyNS.