MF-PINN: MODEL FUSION FOR MULTI-TASK PHYSICS-INFORMED NEURAL NETWORKS SOLVING THE NAVIER–STOKES EQUATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Navier–Stokes equations at different Reynolds numbers exhibit substantial differences, so their solution models must capture both the shared physical structures across flow regimes and the specific local flow patterns of each regime. A conventional single shared multi-task model is easily affected by task interference. Model fusion can reuse information from multiple task models, but for the complex characteristics of the Navier–Stokes equations, direct parameter fusion may introduce conflicting updates and weaken physical consistency.To address this issue, we propose MF-PINN (Model-Fusion Physics-Informed Neural Network), a framework that combines task-vector fusion with physics-informed task adaptation. MF-PINN extracts task-specific updates from a commonly pretrained PINN and fuses them into a shared Base model. It then keeps the Base model fixed and uses lightweight task-specific Adapters to recover local flow characteristics under different flow regimes. The model is trained using the Navier–Stokes equations, boundary conditions, and real flow-field data, while a stream-function representation enforces incompressibility by construction.Experiments on six Reynolds-number tasks from PDEBench show that MF-PINN achieves lower and more stable physics-informed test losses across tasks than independent PINNs, shared multi-task PINNs, parameterized PINNs, and general model-fusion methods. It also requires substantially fewer task-specific parameters than storing a complete model for each task. These results demonstrate that task-vector fusion and physics-informed adaptation can effectively balance cross-task knowledge sharing and task-specific prediction accuracy.

1 INTRODUCTION

Physics-informed neural networks (PINNs) embed governing equations, initial conditions, and boundary conditions into the neural network training process, providing a unified framework for solving partial differential equations (PDEs) by combining data with physical constraints (Raissi et al., 2019). As an important branch of physics-informed machine learning, PINNs have gradually become an important tool in scientific computing (Karniadakis et al., 2021). Among the many PDE application scenarios, fluid mechanics is particularly representative: the underlying Navier–Stokes equations involve strong nonlinearities and multiscale coupling, while conventional numerical methods often face limitations in computational cost and the availability of observational data. Existing studies have applied PINNs to solving the Navier–Stokes equations, including Hidden Fluid Mechanics (Raissi et al., 2020) and NSFnets (Jin et al., 2021). However, when flow tasks involve different Reynolds numbers, PINN modeling becomes more challenging. The Reynolds number characterizes the relative strengths of inertial and viscous effects, and its variation affects velocity distributions and local flow structures. Although tasks at different Reynolds numbers follow the same governing equations, their flow-field solutions exhibit clear differences. Moreover, gradient imbalance among the data, boundary, and equation-residual losses can further increase the optimization difficulty across multiple flow regimes (Wang et al., 2021).

A direct solution is to train an independent single-task PINN model for each Reynolds number. Although this strategy can adapt to a specific flow regime, every task requires a complete model, increasing parameter redundancy while leaving shared physical information underused. As shown

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.

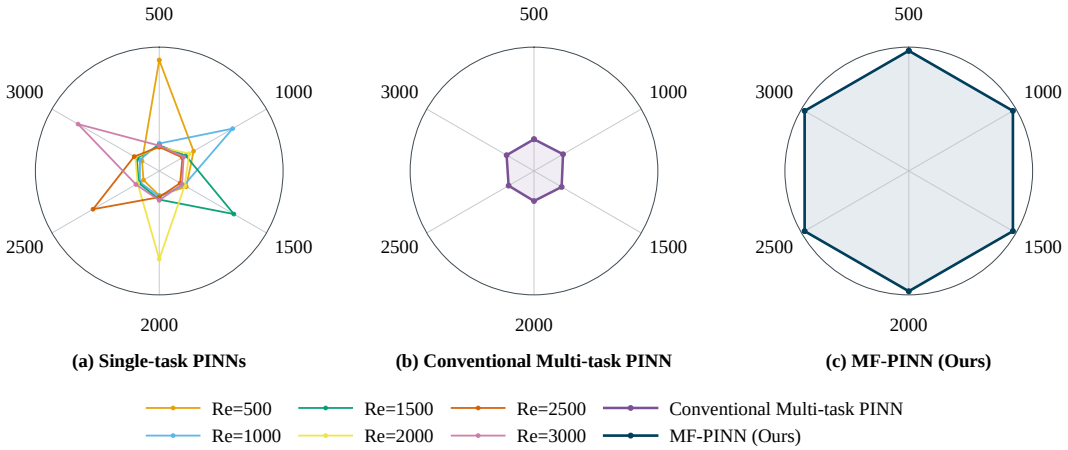


Figure 1: Relative test performance of different PINN strategies across six Reynolds-number tasks. (a) Single-task PINNs show poor cross-task performance. (b) The conventional multi-task PINN struggles to adapt to different flow regimes. (c) MF-PINN maintains lower test losses across all tasks. Points closer to the outer boundary indicate better performance.

in Fig. 1(a), single-task PINNs perform well on their corresponding training tasks but degrade substantially when applied to other Reynolds numbers. In contrast, multi-task learning jointly models related tasks through shared representations (Caruana, 1997). For example, parameterized PINNs, as a type of multi-task PINN, take physical parameters as network inputs and train a unified model to learn the solution map of a parameterized PDE (Cho et al., 2024). However, tasks associated with different Reynolds numbers may induce competing optimization objectives (Sener & Koltun, 2018), and inconsistent gradient directions may cause interference and negative transfer among tasks (Yu et al., 2020). As shown in Fig. 1(b), a conventional multi-task PINN uses a shared model to process all tasks and struggles to accommodate different flow regimes. Therefore, although a unified multi-task model can reduce parameter redundancy, it may fail to capture local characteristics across different flow regimes.

Model merging provides another way to reuse parameter information from multiple task models. It converts different tasks into task vectors based on changes in model parameters and then merges these vectors, thereby retaining task-specific information to some extent. However, **parameter fusion alone may still fail to recover the local physical characteristics associated with different Reynolds numbers (Challenge 1).** Therefore, applying model merging to Navier–Stokes problems across multiple Reynolds numbers also requires task-specific physics-informed adaptation. Inspired by Model Soups (Wortsman et al., 2022) and Task Arithmetic (Ilharco et al., 2023), we combine fine-tuned models with a common initialization, represent the parameter change from the initial model to each fine-tuned model as a task vector, and combine task capabilities through vector operations. However, **task vectors may contain conflicting directions and redundant updates, which can cause interference during direct merging (Challenge 2)** (Yadav et al., 2023). For PINNs, this issue is more complex because of not only the differences among Reynolds numbers, but also the requirement that the merged model simultaneously fit observations and satisfy the governing equations and boundary conditions. Existing general model-merging methods mainly focus on integrating task parameters, and their merged models may not directly adapt to local flow characteristics at different Reynolds numbers.

Motivated by these challenges, we propose MF-PINN (Model-Fusion Physics-Informed Neural Network), which combines task-vector model merging with physics-informed adaptation for solving the two-dimensional incompressible Navier–Stokes equations at different Reynolds numbers. Specifically, **For challenge 1**, MF-PINN first jointly trains a shared pretrained model and then performs full-parameter fine-tuning for each task to extract task vectors in a common parameter space. These task vectors are subsequently fused into a shared Base model. **For challenge 2**, MF-PINN assigns a task-specific Physics-Informed Adapter to each Reynolds number. Here, the Adapter is initialized by matching the output of the corresponding Fine-tuned model, allowing it to acquire task-relevant flow information before refinement. During subsequent optimization, the Base model remains frozen

and only the Adapter is updated using the physics-informed loss to recover local flow characteristics at the current Reynolds number. This framework reuses shared parameter information while preserving task-specific adaptation capability. As shown in Fig. 1(c), MF-PINN maintains more consistent performance across all six tasks. We evaluate MF-PINN on six Reynolds-number tasks from the PDEBench dataset (Takamoto et al., 2022). The results show that neither a single shared model nor direct task-vector merging can fully adapt to all tasks. Compared with multiple baselines, MF-PINN achieves lower and more stable physics-informed test losses across the six tasks, while requiring fewer task-specific parameters than storing a complete model for every task. Overall, the main contributions of this work are summarized as follows:

• We construct the MF-PINN framework, which combines shared pretraining, task-specific fine-tuning, task-vector merging, and lightweight adaptation to integrate shared parameter information across Reynolds-number tasks.

• We introduce task-vector model merging to multi-Reynolds-number PINNs, evaluate the applicability of general merging methods to physics-constrained models, and investigate the effect of task-parameter conflict on the merged results.

• We design task-specific Physics-Informed Adapters and transfer task information from the corresponding Fine-tuned models into Adapter initialization through output matching. With the shared Base model fixed, only the lightweight Adapters are optimized to recover local flow characteristics at different Reynolds numbers.

• Under a unified data split, training configuration, and evaluation metric, we compare MF-PINN with single-task PINNs, multi-task PINNs, parameterized PINNs, and model-merging methods.

2 Related Work

Physics-informed neural networks (PINNs) incorporate governing equations, boundary conditions, and observations into neural network training. They have been applied to fluid-flow surrogate modeling (Sun et al., 2020), cardiovascular-flow inference (Kissas et al., 2020), high-speed flows (Mao et al., 2020), and Reynolds-averaged Navier–Stokes equations (Eivazi et al., 2022). DeepXDE provides a reusable PINN framework (Lu et al., 2021). Existing reviews have discussed PINN applications in flow prediction, parameter identification, and fluid inverse problems (Cai et al., 2021), as well as modeling procedures, limitations, and future directions in scientific machine learning (Cuomo et al., 2022).

However, PINN training can suffer from gradient imbalance, optimization stagnation, and failure modes (Wang et al., 2021). These issues have been studied through failure-mode analysis (Krishnapriyan et al., 2021) and neural-tangent-kernel analysis (Wang et al., 2022), and can become more pronounced when physical parameters vary across tasks. Consequently, multi-task learning has been used to coordinate shared representations and optimization objectives across related tasks.

Multi-task learning improves joint modeling by exploiting shared structures among related tasks (Caruana, 1997). Uncertainty Weighting dynamically adjusts task losses according to task uncertainty (Kendall et al., 2018). Adversarial MTL reduces distributional differences across tasks or domains through domain adapters and adversarial training (Hidayaturrahman et al., 2024); PCGrad alleviates task interference by correcting conflicting task gradients (Yu et al., 2020), while FiLM modulates intermediate features using task conditions (Perez et al., 2018). These methods mainly improve joint training through loss weighting, gradient optimization, adversarial adaptation, or conditional feature modulation.

For physical tasks, multi-task PINN methods further introduce task-oriented optimization objectives or network structures. MTO-PINN formulates PINN training as a multi-task optimization problem (Wang et al., 2023). MultiPINN uses task-specific prediction heads (Li, 2024), Multi-head UR-PINN combines multi-head structures with regularization (Tarancón-Álvarez et al., 2025), MHPINN uses multiple heads to learn task-related functional priors (Zou & Karniadakis, 2025), and AW-EL-PINNs coordinate physical constraints across tasks (Li & Zeng, 2026). Parameterized PINN takes physical parameters as network inputs to learn a parameterized PDE solution map (Cho et al., 2024). These methods improve the representation of different physical tasks, but generally still rely on a shared network or backbone to coordinate task differences.

Beyond joint training, model merging reuses information from multiple trained models through operations in parameter space. Model Soups fuses fine-tuned models through parameter averaging (Wortsman et al., 2022), while Task Arithmetic represents model changes as task vectors (Ilharco et al., 2023). Model Ratatouille studies model reuse for out-of-distribution generalization (Rame et al., 2023), whereas Git Re-Basin and ZipIt! explore alternative cross-task model-merging strategies (Ainsworth et al., 2023; Stoica et al., 2023). Fisher-weighted Fusion determines merging weights according to parameter importance (Matena & Raffel, 2022), and AdaMerging learns task-dependent fusion coefficients (Yang et al., 2024). TIES-Merging addresses sign conflicts and redundant updates in task vectors (Yadav et al., 2023), while DARE-TIES further reduces merging interference through parameter sparsification (Yu et al., 2024). However, direct merging may lose task-specific information; for PINNs, the merged model must also satisfy equation, boundary, and flow-field constraints.

Parameter-efficient adaptation corrects task-specific errors with a small number of trainable parameters. Adapter, AdapterFusion, and LoRA use lightweight modules, adapter composition, and low-rank updates, respectively (Houlsby et al., 2019; Pfeiffer et al., 2021; Hu et al., 2022). Wang et al. further investigated full-parameter fine-tuning, lightweight fine-tuning, and low-rank adaptation for PINNs (Wang et al., 2025). Building on these studies, this work combines task-vector merging with physics-informed Adapter adaptation to construct a shared Base model and recover task-specific flow characteristics across different Reynolds numbers.

3 METHOD

3.1 PROBLEM FORMULATION

We study a family of two-dimensional incompressible Navier–Stokes flow tasks indexed by Reynolds numbers, with task set $\mathcal{R} = \{\mathrm{Re}_1, \mathrm{Re}_2, \ldots, \mathrm{Re}_N\}$. Here, N is the number of tasks and Re_i is the Reynolds number of task i.

For task Re_i, the network input is $\boldsymbol{z} = (x, y, t, \mathrm{Re}_i)$, containing spatial coordinates, time, and the Reynolds-number condition. The network predicts the stream function and pressure, while the velocity components are recovered as $u = \partial\psi/\partial y$ and $v = -\partial\psi/\partial x$. This stream-function representation satisfies the incompressibility constraint by construction;the output definition and its derivation are provided in Appendix A.2.

For the task with Reynolds number Re_i, the two-dimensional incompressible Navier–Stokes momentum equations are

$$u_t + uu_x + vu_y = -\frac{1}{\rho}p_x + \nu_i\left(u_{xx} + u_{yy}\right), \tag{1}$$

$$v_t + uv_x + vv_y = -\frac{1}{\rho}p_y + \nu_i\left(v_{xx} + v_{yy}\right). \tag{2}$$

Under the nondimensional formulation used in this work, $\rho = 1$ and $\nu_i = 1/\mathrm{Re}_i$. All spatial and temporal derivatives are computed using automatic differentiation.

Although the tasks share the same governing equations, their viscous effects, velocity distributions, and local flow structures vary with Reynolds number. Independent models preserve task-specific characteristics but repeatedly learn shared physical laws, whereas a single shared model may fail to represent local differences across flow regimes. MF-PINN addresses this issue by combining shared physical representation learning, task-vector fusion, and task-specific adaptation, as illustrated in Fig. 2. First, the model jointly pretrains a shared network and fine-tunes it for each task to obtain task-specific updates. These updates are then extracted as task vectors and fused into a shared backbone. Finally, a lightweight physics-informed Adapter is attached to the shared backbone for each task and optimized with PDE-residual, boundary-condition, and flow-field data losses to recover regime-specific characteristics. The following subsections describe these three stages.

3.2 SHARED PRETRAINING, FINE-TUNING, AND TASK-VECTOR EXTRACTION

MF-PINN first jointly trains a shared PINN using data from all Reynolds-number tasks. Let θ_{pre} denote the parameters of the resulting pretrained model, which are obtained by minimizing the

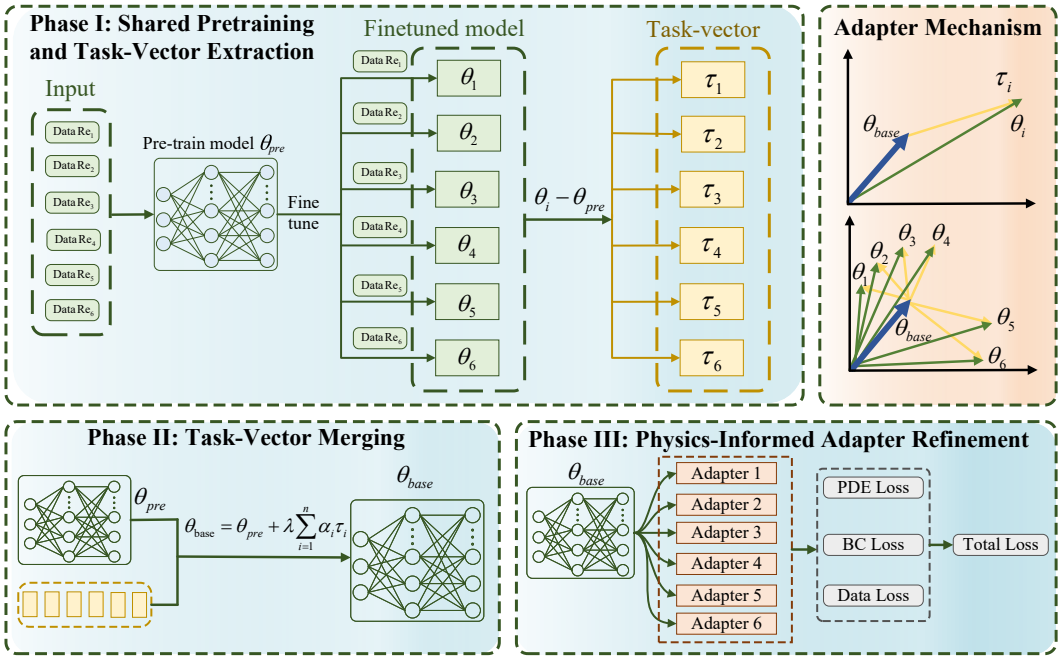


Figure 2: Overview of MF-PINN. Phase I jointly pretrains a shared model and fully fine-tunes it for each Reynolds-number task to extract task vectors. Phase II fuses the task vectors with learnable weights to construct a shared Base model. Phase III attaches a task-specific Adapter to the frozen Base model for each task and optimizes it using PDE, boundary-condition, and data losses.

average loss across tasks:

$$\theta_{\text{pre}} = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\theta; \text{Re}_i). \tag{3}$$

Here, $\mathcal{L}(\theta; \text{Re}_i)$ is the total physics-informed loss for task Re_i, consisting of equation, boundary, and data losses:

$$\mathcal{L}(\theta; \text{Re}_i) = \omega_{\text{eqn}} \mathcal{L}_{\text{eqn}} + \omega_{\text{bc}} \mathcal{L}_{\text{bc}} + \omega_{\text{data}} \mathcal{L}_{\text{data}}. \tag{4}$$

$\mathcal{L}_{\text{eqn}}$, $\mathcal{L}_{\text{bc}}$, and $\mathcal{L}_{\text{data}}$ denote the equation, boundary, and flow-field data losses, respectively, with corresponding weights ω_{eqn}, ω_{bc}, and ω_{data}. The equation loss enforces the momentum and continuity equations, while the boundary loss includes stream-function and velocity constraints. Joint pretraining captures shared physical structures, although a single model may not fully represent task-specific flow features.

Each task is then fine-tuned from θ_{pre} by updating all PINN parameters:

$$\theta_i = \arg\min_{\theta} \mathcal{L}(\theta; \text{Re}_i). \tag{5}$$

The fine-tuned parameters θ_i retain the shared physical representation while incorporating task-specific updates for the corresponding flow condition. We define this update as the task vector $\boldsymbol{\tau}_i = \theta_i - \theta_{\text{pre}}$. Since all task-specific models share the same initialization, the task vectors lie in a common parameter space and can be fused in the following stage.

3.3 Task-Vector Fusion

After obtaining the task vectors, MF-PINN fuses them with the shared pretrained model to construct a unified shared Base model:

$$\theta_{\text{base}} = \theta_{\text{pre}} + \lambda \sum_{i=1}^{N} \alpha_i \boldsymbol{\tau}_i. \tag{6}$$

Here, α_i denotes the fusion weight of the task vector for task Re_i, with $\sum_{i=1}^{N} \alpha_i = 1$, and λ controls the overall magnitude of the fused update. This operation integrates the parameter updates learned

from multiple Reynolds-number tasks into the shared pretrained model, enabling θ_{base} to contain both cross-task shared information and task-specific information from different flow regimes.

Fusion is performed directly in parameter space and does not require retraining the complete model during the fusion stage. It therefore enables a shared Base model to reuse information from multiple tasks. However, task vectors associated with different Reynolds numbers may still have different directions or magnitudes in some parameter dimensions. Moreover, the physics-informed loss contains coupled equation, boundary, and data constraints. Consequently, the directly fused model may not be optimal for every task or fully preserve the physical consistency of each flow regime.

Therefore, θ_{base} serves as a shared foundation containing multi-task information rather than the final optimal model for every task. It is further corrected by task-specific Adapters in the next stage.

3.4 Physics-Informed Adapter Refinement

On the fused shared Base model θ_{base}, MF-PINN assigns a task-specific Adapter to each Reynolds-number task. Let $\mathbf{h}$ denote the shared feature produced by the SharedMLP. For task Re_i, the Adapter $A_{\boldsymbol{\phi}_i}$ is placed between the SharedMLP and the HeadMLP and applies a residual correction to the shared feature:

$$\widetilde{\mathbf{h}}_i = \mathbf{h} + A_{\boldsymbol{\phi}_i}(\mathbf{h}), \tag{7}$$

where $\boldsymbol{\phi}_i$ denotes trainable Adapter parameters and $\widetilde{\mathbf{h}}_i$ is corrected task-specific feature. The Adapter adopts a bottleneck structure with dimension r, which maps feature to a low-dimensional space and then back to its original dimension. It introduces a small number of task-specific parameters to compensate for the residual error of the shared model under current flow condition.

For task Re_i, the corresponding fine-tuned model θ_i can be recovered from the task vector and the shared pretrained parameters. We use its output as a reference to initialize Adapter through output matching, allowing the Adapter to acquire task-specific flow information before physics-informed optimization. This process does not directly copy the task vector into the Adapter parameters.

During refinement, θ_{base} remains frozen and only Adapter parameters for current task are optimized:

$$\boldsymbol{\phi}_i^{\star} = \arg\min_{\boldsymbol{\phi}_i} \mathcal{L}\left(\theta_{\text{base}}, \boldsymbol{\phi}_i; \text{Re}_i\right). \tag{8}$$

Here, $\mathcal{L}(\theta_{\text{base}}, \boldsymbol{\phi}_i; \text{Re}_i)$ is the total physics-informed loss, consisting of equation, boundary, and real-data losses. Freezing the Base model preserves the fused cross-task information, while the task-specific Adapter corrects local flow characteristics for the current Reynolds number. After training, the shared Base model and the corresponding Adapter jointly form the final prediction model for a specific Reynolds-number task.

Training proceeds by shared pretraining, task-wise fine-tuning and task-vector fusion, followed by frozen-base, task-specific Adapter refinement. Given (x, y, t, Re_i) at inference, the fused Base is paired with the Adapter for task Re_i to produce the prediction. The complete training and inference procedure is provided in Appendix A.3.

4 Experiments

4.1 Datasets and Evaluation Metric

We use the two-dimensional incompressible Navier–Stokes data from PDEBench (Takamoto et al., 2022), treating the flow regimes with $\text{Re} = 500, 1000, 1500, 2000, 2500$, and 3000 as six independent physical tasks. Although these tasks follow the same governing equations, they differ in viscous effects, velocity distributions, and local flow structures.

For each task, the data are divided into training and test sets. The training set is used for model training, task-vector construction, and Adapter optimization, whereas the test set is used only for final evaluation. All methods use the same data split and evaluation configuration. We use the total physics-informed loss $\mathcal{L}(\theta; \text{Re}_i)$ defined in Eq. equation 4 as the evaluation metric for each task and report its arithmetic mean across all tasks; lower values indicate better performance. The complete definition is provided in Appendix A.1.

Table 1: Comparison with multi-task learning, multi-task PINN, and model-fusion baselines.

Method	Re=500	Re=1000	Re=1500	Re=2000	Re=2500	Re=3000	Mean
General multi-task learning methods							
Adversarial MTL	0.8371	0.9831	1.6958	1.3810	1.8850	2.1738	1.4926
PCGrad	1.4628	1.7212	1.7917	2.5402	3.1638	2.2541	2.1556
FiLM	1.4380	1.5357	0.9227	1.4241	3.0369	2.1293	1.7478
Multi-task PINN methods							
MTO-PINN	0.9319	1.5645	1.6952	1.4097	2.8335	3.1939	1.9381
MultiPINN	0.5305	0.8166	1.4566	1.3346	1.5757	2.0533	1.2946
Parameterized PINN	2.4104	2.4709	2.7574	2.3780	3.0034	2.0869	2.5178
Multi-head UR-PINN	0.4900	0.6206	0.9307	1.4992	1.7340	1.8722	1.1911
MHPINN	0.4579	0.5553	0.7426	0.6187	0.9356	1.0795	0.7316
AW-EL-PINNs	1.4939	1.6717	1.8632	1.4201	3.0005	2.1179	1.9279
Model fusion methods							
TIES	0.3881	0.9862	0.5754	0.8362	0.9407	0.7451	0.7453
DARE-TIES	0.3937	0.8725	0.4834	0.8286	0.8711	0.6679	0.6862
Fisher-weighted	0.2337	0.9947	0.7207	0.9864	1.1605	0.9204	0.8361
AdaMerging	0.8449	1.0200	0.5138	0.9606	0.8829	0.8662	0.8481
MF-PINN (Ours)	**0.1104**	**0.1969**	**0.1260**	**0.1236**	**0.1078**	**0.1814**	**0.1410**

4.2 Compared Methods and Experimental Results

To evaluate different multi-task strategies for parameterized Navier–Stokes problems, we group the baselines into general multi-task learning methods, multi-task and parameterized PINNs, and model-merging methods. All methods use the same six Reynolds-number tasks, data split, sampling settings, and physics-informed test loss to ensure a consistent comparison.

• The general multi-task learning baselines include MTO-PINN, Adversarial MTL, PCGrad, and FiLM-PINN. These methods improve joint multi-task training through task weighting, gradient updates, or conditional feature modeling, and evaluate the ability of general multi-task strategies to accommodate differences among Reynolds-number tasks.

• The multi-task and parameterized PINN baselines include MultiPINN, Parameterized PINN, Multi-head UR-PINN, MHPINN, and AW-EL-PINNs. They represent multiple physical tasks through shared networks, task-specific prediction structures, or parameter conditioning, and provide structural baselines for evaluating shared representations across different flow regimes.

• The model-merging baselines include TIES, DARE-TIES, Fisher-weighted Fusion, and AdaMerging. These methods integrate information from task models or task vectors in parameter space. For a fair comparison, all model-merging baselines use the same task-specific Adapter refinement configuration after merging.

Table 1 reports the physics-informed test losses and mean values across the six Reynolds-number tasks; lower losses indicate better predictive performance and physical consistency. The general multi-task learning and multi-task PINN methods obtain relatively high losses overall, mainly because differences in viscous effects, velocity distributions, and local flow structures across Reynolds numbers induce competing parameter updates in the shared network. Task weighting, gradient correction, conditional modulation, and multi-head structures alleviate part of this conflict, but remain limited in recovering the task-specific flow characteristics required by each regime.

Among the model-fusion methods, DARE-TIES, TIES, Fisher-weighted Fusion, and AdaMerging obtain mean losses of 0.6862, 0.7453, 0.8361, and 0.8481, respectively, whereas MF-PINN achieves a mean loss of 0.1410. These results show that general fusion rules can reduce part of the parameter interference, but still have difficulty handling differences in the directions and magnitudes of task vectors. They also do not use the Navier–Stokes residuals, boundary conditions, or flow-field data to

physically refine the fused model. Building on task-vector fusion from a common pretrained model, MF-PINN further performs physics-informed optimization with task-specific Adapters. This design preserves shared cross-task information while recovering local flow characteristics under different Reynolds numbers, resulting in low and stable test losses across all six tasks.

4.3 Visualization Results

To qualitatively assess the spatial prediction quality of MF-PINN, Fig. 3 compares the ground-truth streamwise velocity field, the model prediction, and the corresponding signed error across the six Reynolds-number tasks. The predicted fields preserve the main flow structures and remain close to the reference solutions, while the error fields show that the remaining discrepancies are localized rather than uniformly distributed across the domain. This visualization provides a qualitative complement to the quantitative test-loss results.

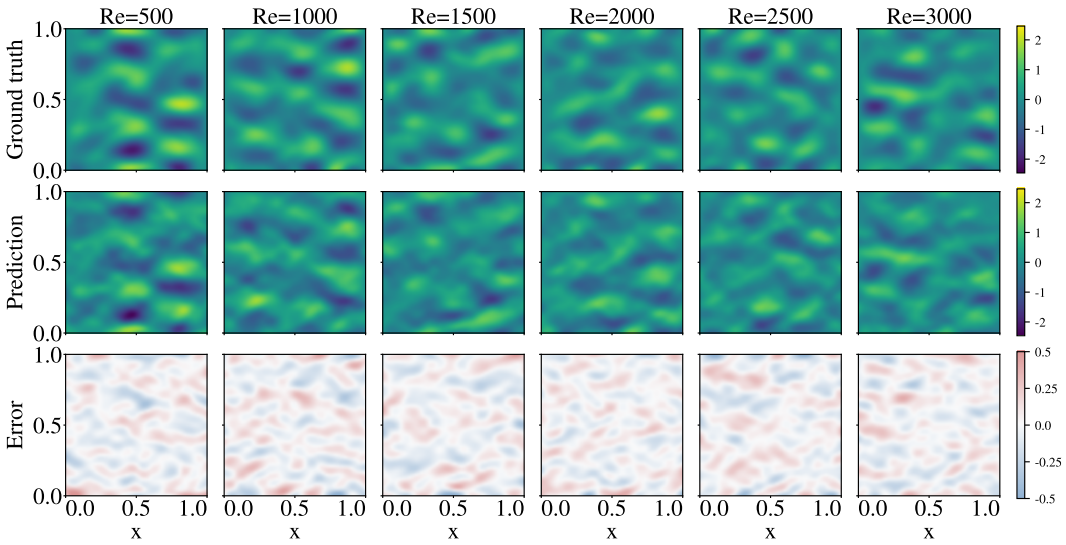


Figure 3: Visualization of the streamwise velocity field u for the six Reynolds-number tasks. The first row shows the ground-truth fields, the second row shows the predicted fields, and the third row presents the signed prediction errors.

Table 2: Pipeline ablation of MF-PINN on the six-Reynolds-number benchmark.

Method	Re=500	Re=1000	Re=1500	Re=2000	Re=2500	Re=3000	Mean
Single-task PINN	0.1525	0.7968	0.4781	0.4289	0.4278	0.6581	0.4904
Pretrained model (θ_{pre})	22.6681	32.5442	25.4003	31.7862	30.5021	37.8362	30.1229
Fine-tuned model (θ_i)	0.0980	0.1230	0.0914	0.1044	0.1860	0.1885	0.1319
Base model (θ_{base})	65.0724	71.2313	56.4126	71.5198	55.4925	56.7317	62.7434
MF-PINN w/o Fusion	0.8387	1.1398	0.9721	1.0443	1.2014	0.9024	1.0165
Random Adapter	0.2103	0.2960	0.2260	0.2238	0.2077	0.2812	0.2408
MF-PINN (Ours)	**0.1104**	**0.1969**	**0.1260**	**0.1236**	**0.1078**	**0.1814**	**0.1410**

4.4 Ablation Study

We ablate the main components and key hyperparameters; the results are summarized in Tables 2–3

• Table 2 reports the test losses of the main MF-PINN stages and variants. The shared pretrained and directly fused Base models obtain mean losses of 30.1229 and 62.7434, respectively, showing that neither a single shared model nor direct task-vector fusion is sufficient for these flow regimes.

Table 3: Effect of the task-vector fusion scale λ.

λ	Re=500	Re=1000	Re=1500	Re=2000	Re=2500	Re=3000	Mean
0.5	0.1130	0.2077	0.1259	0.1234	0.1080	0.1839	0.1437
1.0	0.1104	0.1969	0.1260	0.1236	0.1078	0.1814	0.1410
1.5	0.2099	0.2154	0.1335	0.1364	0.1232	0.1789	0.1662

Table 4: Effect of Adapter bottleneck capacity.

Bottleneck	Re=500	Re=1000	Re=1500	Re=2000	Re=2500	Re=3000	Mean
8	0.1135	0.2152	0.1295	0.1332	0.1206	0.1820	0.1490
16	0.1123	0.1980	0.1280	0.1258	0.1097	0.1832	0.1428
32	0.1104	0.1969	0.1260	0.1236	0.1078	0.1814	0.1410
64	0.1097	0.1952	0.1257	0.1236	0.1075	0.1797	0.1402

Full task-wise fine-tuning achieves the lowest mean loss (0.1319), but requires retaining a complete model for each task. MF-PINN attains a comparable mean loss of 0.1410 while outperforming the Single-task PINN (0.4904), MF-PINN without fusion (1.0165), and the Random Adapter variant (0.2408). The latter comparison shows that task-informed Adapter initialization improves refinement from the fused Base.

• Table 3 shows the effect of the task-vector fusion scale λ. The mean loss is 0.1437 for $\lambda = 0.5$, decreases to 0.1410 for $\lambda = 1.0$, and increases to 0.1662 for $\lambda = 1.5$. This indicates that $\lambda = 1.0$ provides a suitable balance between combining task-specific updates and avoiding excessively large fused updates;we therefore use it in subsequent experiments.

• Table 4 examines the Adapter bottleneck capacity. The mean loss decreases from 0.1490 at dimension 8 to 0.1428, 0.1410, and 0.1402 at dimensions 16, 32, and 64, respectively, indicating stable performance once sufficient capacity is available.

4.5 Limitations

This work has several limitations. First, the experiments are mainly conducted on six Reynolds-number tasks from PDEBench, covering a limited range of flow conditions and flow types. The generalization of MF-PINN to other geometries, boundary conditions, or higher Reynolds numbers therefore remains to be investigated. Second, task-vector fusion assumes that the task models share the same architecture and parameter space; fusion across heterogeneous architectures is not considered. Finally, although Adapters reduce the number of task-specific parameters that must be updated and stored, a separate Adapter is still maintained for each Reynolds number. The management cost of these Adapters may become more significant as the number of tasks increases.

5 Conclusion

This paper presents MF-PINN, a model-fusion framework for two-dimensional incompressible Navier–Stokes problems across multiple Reynolds numbers. The method first extracts task vectors through shared pretraining and task-specific fine-tuning, then fuses these vectors into a shared base model, and finally uses lightweight task-specific Adapters to correct local flow characteristics under different regimes. The experimental results show that a shared model or task-vector fusion alone is insufficient to adapt to all Reynolds-number tasks. Compared with general model-fusion methods, MF-PINN achieves lower and more stable test physics-informed losses across the six tasks, with a mean loss of 0.1410. The ablation results further demonstrate that task-vector fusion and task-specific Adapter refinement provide complementary benefits. By preserving task adaptation capability while reducing the number of parameters that must be updated and stored for each task, MF-PINN provides a practical approach to efficient PINN deployment across multiple flow regimes.

AI USE STATEMENT

We used generative AI tools to refine the research idea and experimental design, assist with code implementation and result interpretation, search related literature, format figures and references, and draft and polish parts of this paper. Synthetic data generation and mathematical proof generation were not used in this work. We reviewed all AI-assisted content and take full responsibility for the final manuscript, including its text, claims, and artifacts.

ETHICS STATEMENT

This work studies a computational method for solving two-dimensional incompressible Navier–Stokes equations across multiple Reynolds numbers. The experiments use publicly available PDEBench flow data and involve no human subjects, personal data, or personally identifiable information. We do not identify foreseeable ethical risks associated with the present methodology.

REPRODUCIBILITY STATEMENT

To support reproducibility, the main text specifies the problem formulation, MF-PINN architecture, governing equations, loss functions, and task-vector fusion procedure. The Experiments section describes the PDEBench data source, the six Reynolds-number tasks, data splitting, sampling configuration, evaluation metric, and comparison settings. Appendix A.2 provides the derivation of the network outputs and the incompressibility constraint, while Appendix A.3 gives the complete training and inference procedure and the corresponding algorithm. The two-dimensional incompressible Navier–Stokes data are publicly available through PDEBench (Takamoto et al., 2022). Source code and training configurations will be released upon acceptance.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023.

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. doi: 10.1007/s10409-021-01148-1.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734.

Woojin Cho, Minju Jo, Haksoo Lim, Kookjin Lee, Dongeun Lee, Sanghyun Hong, and Noseong Park. Parameterized physics-informed neural networks for parameterized PDEs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8510–8533. PMLR, 21–27 Jul 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. doi: 10.1007/s10915-022-01939-z.

Hamidreza Eivazi, Mojtaba Tahani, Philipp Schlatter, and Ricardo Vinuesa. Physics-informed neural networks for solving reynolds-averaged navier–stokes equations. *Physics of Fluids*, 34(7), 2022. doi: 10.1063/5.0095270.

Hidayaturrahman, Agung Trisetyarso, Iman Herwidiana Kartowisastro, and Widodo Budiharto. Adversarial multitask learning for domain adaptation through domain adapter. *IEEE Access*, 2024. doi: 10.1109/ACCESS.2024.3512544.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for

NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.

Xiaowei Jin, Shengze Cai, Hui Li, and George E. Karniadakis. NSFnets: Physics-informed neural networks for the incompressible navier–stokes equations. *Journal of Computational Physics*, 426: 109951, 2021. doi: 10.1016/j.jcp.2020.109951.

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. doi: 10.1038/s42254-021-00314-5.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491, 2018. doi: 10.1109/CVPR.2018.00781.

Georgios Kissas, Yibo Yang, Eileen Hwuang, Walter R. T. Witschey, John A. Detre, and Paris Perdikaris. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358:112623, 2020. doi: 10.1016/j.cma.2019.112623.

Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M. Kirby, and Michael W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26548–26560, 2021.

Chuandong Li and Runtian Zeng. Aw-el-pinns: A multi-task learning physics-informed neural network for euler-lagrange systems in optimal control problems. *Neural Networks*, 199:108694, 2026. doi: 10.1016/j.neunet.2026.108694.

Kangjie Li. Multipinn: Multi-head enriched physics-informed neural networks for differential equations solving. *Neural Computing and Applications*, 36(19):11371–11395, 2024. doi: 10.1007/s00521-024-09766-z.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021. doi: 10.1137/19M1274067.

Zhiping Mao, Ameya D. Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020. doi: 10.1016/j.cma.2019.112789.

Michael S. Matena and Colin A. Raffel. Merging models with fisher-weighted averaging of neural network parameters. In *Advances in Neural Information Processing Systems 35*, pp. 17703–17716, 2022. doi: 10.52202/068431-1287.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11671.

Jonas Pfeiffer, Aishwarya Kamath, Andreas R"uckl'e, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 487–503, 2021. doi: 10.18653/v1/2021.eacl-main.39.

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. doi: 10.1016/j.jcp.2018.10.045.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020. doi: 10.1126/science.aaw4741.

Alexandre Rame, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Leon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28656–28679. PMLR, 23–29 Jul 2023.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. ZipIt! merging models from different tasks without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15865–15874, 2023.

Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020. doi: 10.1016/j.cma.2019.112732.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pfl"uger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. In *Advances in Neural Information Processing Systems 35*, pp. 1596–1611, 2022. doi: 10.52202/068431-0117.

Pedro Tarancón-Álvarez, Pablo Tejerina-Pérez, Raul Jimenez, and Pavlos Protopapas. Efficient pinns via multi-head unimodular regularization of the solutions space. *Communications Physics*, 8, 2025. doi: 10.1038/s42005-025-02248-1.

Bo Wang, A. K. Qin, Sajjad Shafiei, Hussein Dia, Adriana-Simona Mihaita, and Hanna Grzybowska. Training physics-informed neural networks via multi-task optimization for traffic density prediction. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2023. doi: 10.1109/IJCNN54540.2023.10191301.

Sifan Wang, Xiaoqing Yu, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021. doi: 10.1137/20M1318043.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. doi: 10.1016/j.jcp.2021.110768.

Yizheng Wang, Jinshuai Bai, Mohammad Sadegh Eshaghi, Cosmin Anitescu, Xiaoying Zhuang, Timon Rabczuk, and Yinghua Liu. Transfer learning in physics-informed neural networks: Full fine-tuning, lightweight fine-tuning, and low-rank adaptation. *International Journal of Mechanical System Dynamics*, 5(2):212–235, 2025. doi: 10.1002/msd2.70030.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 17–23 Jul 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems 36*, pp. 7093–7115, 2023. doi: 10.52202/075280-0310.

Enneng Yang, Zhenyi Wang, Lin Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. AdaMerging: Adaptive model merging for multi-task learning. In *International Conference on Learning Representations*, 2024.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*, 2024.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836, 2020.

Zongren Zou and George Em Karniadakis. Multi-head physics-informed neural networks for learning functional priors and uncertainty quantification. *Journal of Computational Physics*, 531: 113947, 2025. doi: 10.1016/j.jcp.2025.113947.

A Supplementary Materials

A.1 Evaluation Metric

For the task set $\mathcal{R} = \{\mathrm{Re}_1, \ldots, \mathrm{Re}_N\}$, we report the arithmetic mean of the physics-informed test loss across all tasks:

$$\overline{\mathcal{L}} := \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\theta; \mathrm{Re}_i). \tag{9}$$

Lower values indicate better performance.

A.2 Network Output and Incompressibility Constraint Derivation

Given network parameters θ, the MF-PINN output is defined as

$$\boldsymbol{o}_\theta(\boldsymbol{z}) = [\psi_\theta(\boldsymbol{z}), p_\theta(\boldsymbol{z})], \tag{10}$$

where ψ_θ denotes the stream function and p_θ denotes the pressure.

Following the stream-function representation introduced in the main text, the velocity components are given by

$$u = \frac{\partial \psi}{\partial y}, \qquad v = -\frac{\partial \psi}{\partial x}. \tag{11}$$

Substituting these definitions into the continuity equation gives

$$u_x + v_y = \frac{\partial}{\partial x}\left(\frac{\partial \psi}{\partial y}\right) + \frac{\partial}{\partial y}\left(-\frac{\partial \psi}{\partial x}\right). \tag{12}$$

Therefore,

$$u_x + v_y = \frac{\partial^2 \psi}{\partial x \partial y} - \frac{\partial^2 \psi}{\partial y \partial x} = 0. \tag{13}$$

When the stream function ψ is sufficiently continuous and smooth, the mixed partial derivatives commute. Hence, the velocity field derived from the stream function naturally satisfies the incompressible continuity constraint.

A.3 Training and Inference Procedure

The training procedure of MF-PINN follows the three stages shown in the framework. The model sequentially performs shared representation learning, task-information fusion, and task-specific refinement. The complete procedure is summarized in Algorithm 1.

During inference, given a spatio-temporal coordinate and a Reynolds-number input $\boldsymbol{z} = (x, y, t, \mathrm{Re}_i)$, the model selects the task-specific Adapter associated with Re_i and uses it together

Algorithm 1 Training Procedure of MF-PINN

Require: Task set $\mathcal{R}$ and training data $\{\mathcal{D}_i\}_{i=1}^{N}$
Ensure: θ_{base} and $\{\phi_i^\star\}_{i=1}^{N}$

function TRAINMF-PINN($\mathcal{R}, \{\mathcal{D}_i\}_{i=1}^{N}$)
  $\theta_{\text{pre}} \leftarrow \text{JOINTTRAIN}(\mathcal{R})$
  for $i = 1$ to N **do**
    $\theta_i \leftarrow \text{FINETUNE}(\theta_{\text{pre}}, \mathcal{D}_i)$
    $\boldsymbol{\tau}_i \leftarrow \theta_i - \theta_{\text{pre}}$
  end for
  $\Delta\theta \leftarrow \lambda \sum_{i=1}^{N} \alpha_i \boldsymbol{\tau}_i$
  $\theta_{\text{base}} \leftarrow \theta_{\text{pre}} + \Delta\theta$
  for $i = 1$ to N **do**
    $\phi_i \leftarrow \text{INITIALIZEADAPTER}(\theta_{\text{base}}, \theta_i)$
    $\phi_i^\star \leftarrow \text{TRAINADAPTER}(\theta_{\text{base}}, \phi_i, \mathcal{D}_i)$
  end for
  return $\theta_{\text{base}}, \{\phi_i^\star\}_{i=1}^{N}$
end function

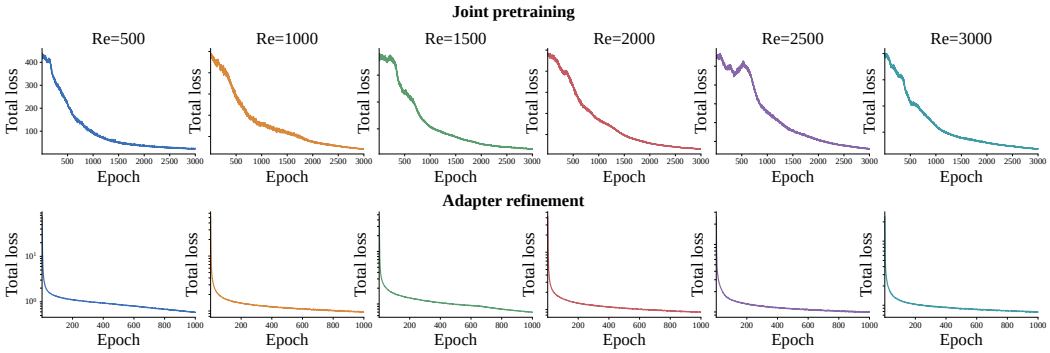


Figure 4: Training loss curves for the six Reynolds-number tasks. The top row shows joint pretraining, while the bottom row shows task-specific Adapter refinement.

with the shared base model for prediction. The network outputs the stream function and pressure, while the velocity components are recovered from spatial derivatives of the stream function. Different tasks share the same Fusion Base while retaining task-specific flow features through their respective Adapters, allowing the model to adapt to flow states at different Reynolds numbers.

Therefore, MF-PINN preserves task-specific adaptation capability while reducing the parameter redundancy caused by training a complete model separately for each Reynolds-number task. It provides an efficient modeling approach for shared learning and task-specific correction in multi-regime flow problems.

A.4 TRAINING DYNAMICS

Figure 4 shows the training loss curves during joint pretraining and task-specific Adapter refinement. The loss decreases consistently across all Reynolds-number tasks, indicating stable optimization during both stages.

A.5 PARAMETER EFFICIENCY

Table 5 compares the number of parameters required to store the models across six Reynolds-number tasks. MF-PINN requires 68,130 parameters in total, compared with 655,560 parameters for six independently stored Single-task or Fine-tuned PINNs, reducing task-specific storage by approximately 89.6%.

Table 5: Parameter-count comparison across six Reynolds-number tasks.

Method	**Total parameters**
Single-task PINN $\times 6$	655,560
Fine-tuned PINN $\times 6$	655,560
Fusion Base	18,210
MF-PINN (Fusion Base + 6 Adapters)	68,130

Table 6: Cross-task physics-informed losses obtained by applying each Fine-tuned model to the six Reynolds-number tasks. Diagonal entries denote the corresponding target-task results, while off-diagonal entries measure cross-task transferability. Lower values indicate better performance.

Source model	Re=500	Re=1000	Re=1500	Re=2000	Re=2500	Re=3000
Re=500	**0.0980**	148.3511	422.8203	301.2371	256.4910	426.9428
Re=1000	106.0517	**0.1230**	173.9150	127.7807	241.3110	191.3064
Re=1500	331.1738	238.2029	**0.0914**	401.2055	557.5407	195.4294
Re=2000	778.0578	462.1782	183.2582	**0.1044**	140.1366	189.2826
Re=2500	241.1039	557.5670	228.6513	224.7394	**0.1860**	75.2980
Re=3000	339.5770	884.1703	816.1722	431.3462	157.0215	**0.1885**

Table 7: Cross-task physics-informed losses obtained by applying each task-specific Adapter to the shared Base model. The diagonal entries are replaced by the corresponding MF-PINN target-task results. Lower values indicate better performance.

Adapter	Re=500	Re=1000	Re=1500	Re=2000	Re=2500	Re=3000
Adapter Re=500	**0.1104**	175.7418	222.2083	213.7456	351.3517	162.3516
Adapter Re=1000	105.6357	**0.1969**	172.6831	301.5940	146.9823	142.2372
Adapter Re=1500	352.9831	370.5446	**0.1260**	201.6127	305.1705	375.4881
Adapter Re=2000	776.5245	232.7822	91.0590	**0.1236**	78.1445	286.6250
Adapter Re=2500	635.8307	399.1821	189.1093	71.1480	**0.1078**	92.4174
Adapter Re=3000	174.1101	592.6895	458.3234	163.4272	180.3320	**0.1814**

A.6 Cross-Task Generalization Analysis

To further examine task-specific models across flow regimes, we evaluate each task-specific Adapter and Fine-tuned model on all six Reynolds-number tasks. Rows indicate the source task of a model, while columns indicate the task used for evaluation. The diagonal entries are replaced by the corresponding MF-PINN target-task results.

Table 7 presents the cross-task losses obtained by applying each task-specific Adapter to the shared Base model. Because the diagonal entries are replaced by the corresponding MF-PINN target-task results, the low diagonal values serve as target-task references, whereas the substantially larger off-diagonal values reveal the transfer behavior of each Adapter. For example, the Adapter associated with $\mathrm{Re} = 500$ produces much larger losses when evaluated on the other Reynolds-number tasks. Similar patterns are observed for the remaining Adapters, indicating that each Adapter learns flow characteristics specific to its target Reynolds number and that direct cross-task reuse causes a clear performance degradation. Thus, the Adapters are not intended to be universally transferable modules; they complement the shared Base model by recovering task-specific capabilities that are missing for the corresponding flow regime.

Table 6 reports the cross-task losses of the Fine-tuned models. Since the diagonal entries are omitted, the off-diagonal values provide the main evidence of cross-task transferability. These losses are generally much larger than the corresponding target-task results reported in Table 2, demonstrating that the Fine-tuned models encode strong task-specific information. For example, the model associ-

ated with $\mathrm{Re} = 3000$ produces losses of 884.1703 and 816.1722 on the $\mathrm{Re} = 1000$ and $\mathrm{Re} = 1500$ tasks, respectively.

Compared with the Fine-tuned models, the Adapters contain far fewer task-specific parameters while retaining similar task specificity. The low diagonal losses and high off-diagonal losses in the Adapter table show that the learned Adapters mainly operate on their corresponding Reynolds-number tasks rather than serving as universally transferable correction modules. Together, the two tables indicate that the combination of a shared Base model and task-specific Adapters balances parameter reuse with task adaptation.