Supplementary Information for Automatic Network Structure Discovery of Physics Informed Neural Networks via Knowledge Distillation

Ziti Liu[1,2†], Yang Liu[2†], Xunshi Yan[3*], Wen Liu[2], Han Nie[2], Shuaiqi Guo[2], Chen-an Zhang[2*]

[1]School of Advanced Interdisciplinary Sciences, University of Chinese Academy of Sciences, Beijing, 100049, China.

[2*]State Key Laboratory of High Temperature Gas Dynamics, Institute of Mechanics, Chinese Academy of Sciences, Beijing, 100190, China.

[3*]Institute of Nuclear and New Energy Technology, Tsinghua University, Beijing, 100084, China.

*Corresponding author(s). E-mail(s): yanxs@tsinghua.edu.cn; zhch_a@imech.ac.cn;
Contributing authors: Liuziti22@mails.ucas.edu.cn; liuyang2@imech.ac.cn; lw@imech.ac.cn; niehan@imech.ac.cn; guoshuaiqi@imech.ac.cn;
†These authors contributed equally to this work.

1 Background knowledge

This section will introduce the basic principles of PINN and the fundamentals of distillation learning.

1.1 Physics-informed neural networks (PINNs)

PINN is a data-driven algorithm based on NNs for solving partial differential equations (PDEs) [1]. The fundamental idea of PINN is to approximate the solution of a PDE using a NN while ensuring that the solution satisfies the prior physical constraints of the PDE. The loss function of PINN typically consists of two components: the data fitting term (e.g., observed data points, boundary conditions, and initial conditions) and the physical constraint term (i.e., the governing equations of the physical scenario). The data fitting term ensures that the network's output aligns with the observed data, while the physical constraint term ensures that the network's output satisfies the PDE's physical constraints. By minimizing the loss function, one can obtain a solution that adheres to the PDE.

Ψ-NN's (Physics structure informed Neural Network, abbreviated as Ψ-NN, pronounced as 'Psi-NN') teacher model iteration method aligns with that of PINN [2]: the loss function for iteration is formulated based on the prior knowledge. Consider a PDE that includes the temporal coordinate t and spatial coordinates $\boldsymbol{x} \in \mathbb{R}^n$, with its solution denoted as $\boldsymbol{u}(t, \boldsymbol{x}) \in \mathbb{R}^k$: The PINN is defined as:

$$\mathcal{N}(\boldsymbol{x}, t; \boldsymbol{\theta}) = \tilde{\boldsymbol{u}} \tag{1.1}$$

where $\boldsymbol{\theta}$ is the trainable parameter vector of the NN, $\tilde{\boldsymbol{u}}$ is the prediction. In the PDEs' forward problem, the sources of the loss function stem from the governing equations $\mathcal{L}$, initial conditions $\mathcal{I}$ and boundary conditions $\mathcal{B}$:

$$\mathcal{L}(\boldsymbol{u}, t, \boldsymbol{x}) = 0 \quad t \in [0, T], \quad \boldsymbol{x} \in \Omega \tag{1.2}$$

$$\mathcal{I}(\boldsymbol{u}, t, \boldsymbol{x}) = 0 \tag{1.3}$$

$$\mathcal{B}(\boldsymbol{u}, t, \boldsymbol{x}) = 0 \tag{1.4}$$

The loss function is constructed using mean squared error (MSE). The specific components of the loss function for the forward problem are as follows:

$$MSE_f = \frac{1}{M_f}\sum_{i=1}^{M_f}\left|\mathcal{L}(\tilde{\boldsymbol{u}}^i, t^i, \boldsymbol{x}^i)\right|^2 \tag{1.5}$$

$$MSE_I = \frac{1}{M_I}\sum_{i=1}^{M_I}\left|\mathcal{I}(\tilde{\boldsymbol{u}}^i, t^i, \boldsymbol{x}^i)\right|^2 \tag{1.6}$$

$$MSE_B = \frac{1}{M_B}\sum_{i=1}^{M_B}\left|\mathcal{B}(\tilde{\boldsymbol{u}}^i, t^i, \boldsymbol{x}^i)\right|^2 \tag{1.7}$$

$$MSE_{\text{forward}} = \omega_f MSE_f + \omega_I MSE_I + \omega_B MSE_B. \tag{1.8}$$

where M is a set of given finite configuration points, with subscripts representing different types. ω is the normalization weight of the loss function. In the inverse problem, the data loss stems from M_D sample points $\check{\boldsymbol{u}}$, and Eq. (1.2) involves an undetermined parameter vector $\boldsymbol{\lambda}$. Therefore:

$$MSE_{f'} = \frac{1}{M_{f'}}\sum_{i=1}^{M_{f'}}\left|\mathcal{L}(\tilde{\boldsymbol{u}}^i, t^i, \boldsymbol{x}^i, \boldsymbol{\lambda}^i)\right|^2 \tag{1.9}$$

$$MSE_D = \frac{1}{M_D}\sum_{i=1}^{M_D}\left|\tilde{\boldsymbol{u}}^i - \check{\boldsymbol{u}}^i\right|^2 \tag{1.10}$$

$$MSE_{\text{inverse}} = \omega_{f'} MSE_{f'} + \omega_D MSE_D \tag{1.11}$$

where the undetermined parameter vector $\boldsymbol{\lambda} \in \mathbb{R}^{\boldsymbol{\lambda}}$ is concatenated with NN's origional trainable parameter vector $\boldsymbol{\theta}_{ori} \in \mathbb{R}^{\boldsymbol{\theta}_{ori}}$, forming a modified parameter vector $\boldsymbol{\theta}$ with a dimension of $\mathbb{R}^{\boldsymbol{\lambda}+\boldsymbol{\theta}}$.

In both forward and inverse problems, the NN utilizes the backpropagation (BP) algorithm to find the vector of trainable parameters $\boldsymbol{\theta}$ that minimizes the MSE, i.e.

$$\boldsymbol{\theta}_{\text{optimal}} = \arg\min_{\boldsymbol{\theta}} \text{MSE}(\boldsymbol{\theta}) \tag{1.12}$$

1.2 Distillation learning

In classification tasks, hidden knowledge gives rise to distillation learning methods [3]. Unlike the hard labels learned by the teacher network, the student network learns the outputs of the teacher network. These outputs contain information about the teacher network's understanding of the correlations among different categories of the problem, enabling information transfer. Consider a classification problem where the teacher and student networks are defined as:

$$\mathcal{N}_T(\boldsymbol{x}; \boldsymbol{\theta}_T) = \tilde{\boldsymbol{U}}_T \tag{1.13}$$

$$\mathcal{N}_S(\boldsymbol{x}; \boldsymbol{\theta}_S) = \tilde{\boldsymbol{U}}_S \tag{1.14}$$

where $\boldsymbol{\theta}$ is the vector of trainable parameters in the network, and the output $\boldsymbol{U}$ is the output vector matrix of the network, with subscripts denoting teacher T and student S. During teacher training, the target for fitting is a vector matrix composed of One-Hot vectors. In the distillation process, the softmax function smooths the teacher network's output to serve as the soft label for the student network, i.e., both outputs become:

$$\tilde{\boldsymbol{U}}_T = \text{softmax}(\mathcal{N}_T(\boldsymbol{x}; \boldsymbol{\theta}_T)/\tau) \tag{1.15}$$

$$\tilde{\boldsymbol{U}}_S = \text{softmax}(\mathcal{N}_S(\boldsymbol{x}; \boldsymbol{\theta}_S)/\tau) \tag{1.16}$$

where $\tau > 1$ is the temperature parameter used to further smooth the teacher network's output, preserving the category information in the teacher network's output while allowing the student network to gain additional dark knowledge. Thus, the student network can learn the teacher network's output by minimizing the KL divergence[4]:

$$\text{KL}(\tilde{\boldsymbol{U}}_T || \tilde{\boldsymbol{U}}_S) = \sum_i \tilde{\boldsymbol{U}}_T^i \log \frac{\tilde{\boldsymbol{U}}_T^i}{\tilde{\boldsymbol{U}}_S^i} \tag{1.17}$$

The final objective of the student network is:

$$\boldsymbol{\theta}_S = \arg\min_{\boldsymbol{\theta}_S} \text{MSE}(\tilde{\boldsymbol{U}}_S) + \lambda \text{KL}(\tilde{\boldsymbol{U}}_T || \tilde{\boldsymbol{U}}_S) \tag{1.18}$$

where $\text{MSE}(\tilde{\boldsymbol{U}}_S)$ represents the mean squared error between the student network's output and the hard labels, and λ is the weight parameter for the KL divergence.

2 Ablation study

To further evaluate the individual contributions of the three steps—distillation, structure extraction, and reconstruction—we conducted ablation studies. In these experiments, the teacher network was configured with three hidden layers and 16 nodes per layer, while the student network had three hidden layers and 8 nodes per layer. Both the teacher and student networks were trained for 8e4 iterations during distillation, and the same number of iterations was used in the comparative experiments. The Adam optimizer[5] was employed with a constant learning rate of 1e-4. To ensure reproducibility, the random seed was fixed at 1234. All experiments were performed on the same hardware (Intel 12400f CPU and RTX4080 GPU) and used the same Laplace example settings as described in 4. The ablation study groups were configured as follows:

1. Only the structure extraction and reconstruction steps are used, with the distillation step removed. Regularization-based structure extraction is applied to the teacher network in the Ψ-NN model, followed by network reconstruction. Iterative prediction is then performed on the case study.
2. Only the distillation and reconstruction steps are used, with the structure extraction step removed. The regularization-based structure extraction step for the student network is omitted in Ψ-NN; the network is directly identified and reconstructed. Iterative prediction is then performed on the case study.
3. Only the distillation and structure extraction steps are used, with the reconstruction step removed. After structure identification in the student network of Ψ-NN, iterative prediction is performed directly using the network results without reinitializing the trainable parameters.
4. Only the distillation step is used. The student network learns from the teacher network results via distillation.
5. Only the structure extraction step is used. Regularization-based structure extraction is applied to the teacher network in the Ψ-NN model, and iterative prediction is performed on the case study without parameter reinitialization.
6. Only the reconstruction step is used. The results of the teacher network in the Ψ-NN model are reconstructed, and iterative prediction is performed on the case study.

To more clearly present the effects of the ablation studies, the experiments are evaluated using two metrics: the emergence of physically meaningful network structures and the model accuracy. After the distillation-extraction-reconstruction steps, parameter clustering is performed on the student network to assess whether physically relevant structures are obtained. Subsequently, the L-2 accuracy of the entire computational domain using the resulting models (the reconstructed model if reconstruction is included, or the pre-reconstruction model otherwise) is compared. The results of the ablation studies are summarized in Table 1.

The loss function iteration results of the ablation studies are shown in Fig. 1. For a clearer comparison, parameter clustering analysis was performed for all experiments,, regardless of whether structure extraction or reconstruction was included, to illustrate the structural potential (see Fig. 2). In Experiment 1, after removing the distillation step and directly applying regularization and structure extraction to the teacher network, some structure was identified in the first hidden layer, but deeper layers remained disordered. In Experiment 2, after omitting structure extraction and directly reconstructing the student network post-distillation, the resulting structure was disorganized and similar to a fully connected network (with random parameters). Experiment 3, which is closest to Ψ-NN and includes both distillation and meaningful structure, failed to retain the structure in terms of accuracy due to the lack of structure embedding; this led to faster initial convergence but ultimately higher MSE as the process essentially restarted optimization. Experiment 4, with only the distillation step, allowed the student network to quickly fit the teacher's predictions, but provided limited improvement in final model performance. Experiment 5, with only structure extraction, effectively retained the regularized fully connected model and continued training without regularization; while the initial MSE was low, the final MSE was still higer than that of full model. Experiment 6, which only performed network reconstruction without structure extraction, could not preserve any structure (as shown in Fig. 2d); thus, the reconstruction alone had no effect on the MLP, and under the same initialization and random seed, the results of Experiment 6 and the baseline PINN were identical.

Currently, some distillation methods in PINN [6] improve accuracy by fusing original sampling data and teacher network predictions based on confidence weighting. However, the primary objective of the Ψ-NN student network is to discover network structure rather than to enhance fitting accuracy, making such fusion

Table 1: Ablation study settings and results. The first three columns show the settings, where ✓ indicates usage and ✗ indicates non-usage. The last two columns present the results, with "Yes" indicating that the experiment produced a structure and "No" indicating that it did not. The L-2 loss represents the average error across the entire computational field. Since Ablation-6 only includes the reconstruction step, it is essentially a re-initialization (resetting all trainable parameters of a trained model to new random values, i.e., discarding all previous training results and starting training from scratch.) of the PINN model, resulting in identical outcomes to PINN under the same random seed.

Items / Models	Distillation	Extraction	Reconstruction	Structure	L-2 error (1e-4)
Ψ-NN	✓	✓	✓	Yes	0.7422
Ablation-1	✗	✓	✓	No	6.361
Ablation-2	✓	✗	✓	No	8.396
Ablation-3	✓	✓	✗	Yes	14.32
Ablation-4	✓	✗	✗	No	4.659
Ablation-5	✗	✓	✗	No	5.227
Ablation-6	✗	✗	✓	No	11.59
PINN	✗	✗	✗	No	11.59

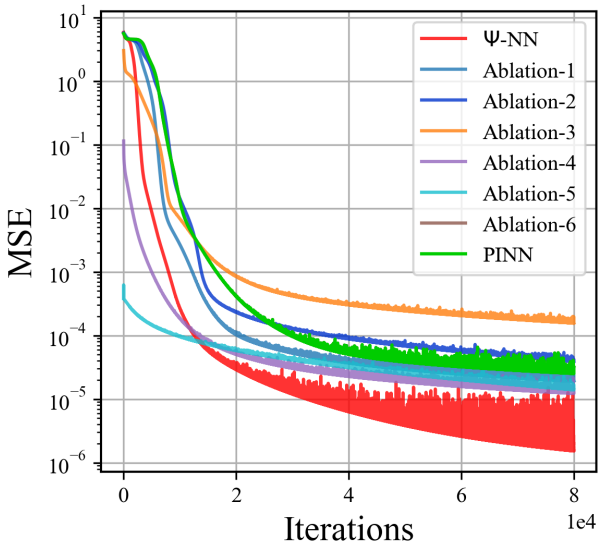


Fig. 1: Iteration results of the loss function for the ablation studies. The results of Ablation-1 to Ablation-6 correspond to the respective entries in Table 1. Ablation-6 only includes the reconstruction step, which is essentially a re-initialization of the PINN model; thus, under the same random seed, the results of Ablation-6 and PINN are identical, and the curve for this experiment overlaps with that of PINN.

methods less applicable. To systematically assess the role of distillation learning in Ψ-NN, we introduce a fusion approach and conduct an ablation study. The results demonstrate that using only the teacher network's output (without sampling data) is more effective for structure extraction, confirming the necessity of the distillation learning component in Ψ-NN. The method is as follows:

$$\varphi = 1 - tanh(\kappa \cdot |(\tilde{\boldsymbol{u}}_T - \check{\boldsymbol{u}})|) \tag{2.1}$$

where κ is referred to as the fusion coefficient. Thus, at observational coordinate points, the loss function for the student network is expressed as a mean squared error (MSE):

$$MSE_S = \frac{1}{M_S} \sum_{i=1}^{M_S} \left| \tilde{\boldsymbol{u}}_S^i - \varphi \tilde{\boldsymbol{u}}_T^i - (1-\varphi)\check{\boldsymbol{u}}^i \right|^2 \tag{2.2}$$

The settings for this parameter ablation study are as follows: 1. At observational coordinate points, only observational data is used without the teacher network's predicted data, in which case $\kappa \to +\infty, \varphi \equiv 0$; 2.

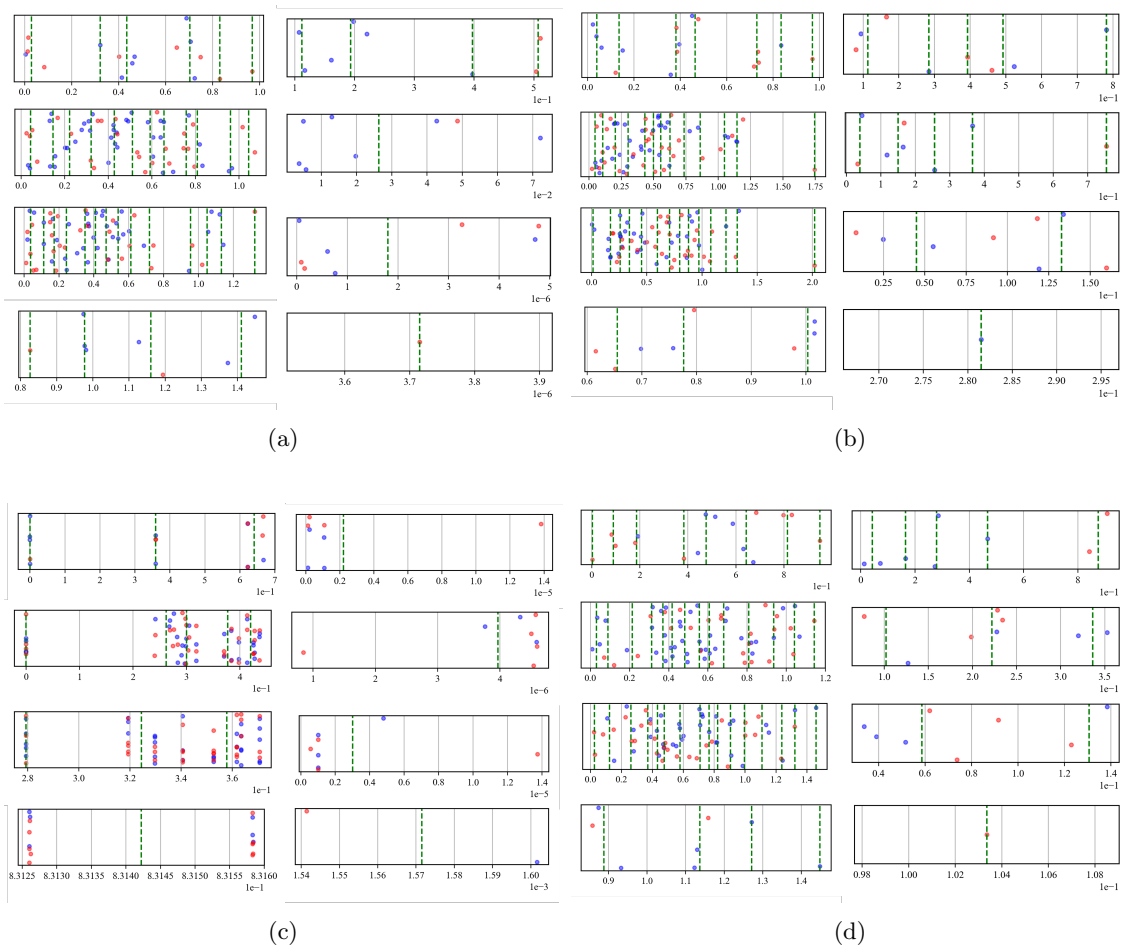


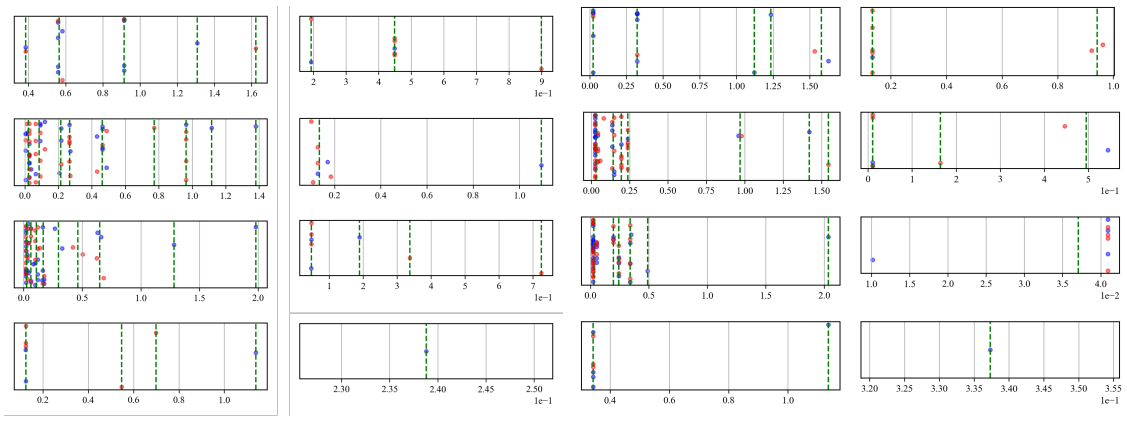
Fig. 2: Clustering results of the ablation studies. 2a, the structure clustering results for Ablation-1 and Ablation-5. 2b, the clustering results for Ablation-2 and Ablation-4. 2c, the clustering results for Ablation-3. and 2d, the clustering results for Ablation-6.

At observational coordinate points, only the teacher network's predicted data is used without observational data, in which case $\kappa = 0, \varphi \equiv 1$; 3. At observational coordinate points, both the teacher network's predicted data and observational data are fused, in which case $\kappa \in [0, +\infty], \varphi \in [0, 1]$. The results are shown in Table 2. Inverse problem experiments were conducted on the Burgers equation with these three settings, and the

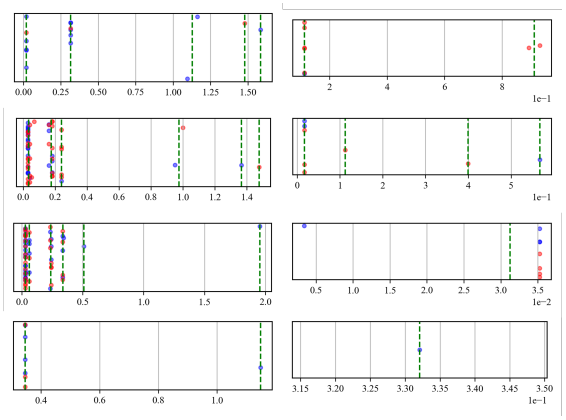
Table 2: κ Ablation study settings and results. The first three columns are the settings, and the last two columns are the results. In the settings, ✓ indicates usage, while ✗ indicates non-usage. To distinguish from the settings, "Yes" in the results indicates that a structure is produced, while "No" indicates that no structure is produced.

κ \ Value	Teacher model	Sample points	Structure
$+\infty$	✗	✓	No
$[0, +\infty]$	✓	✓	Yes
0	✓	✗	Yes

extracted structure results are as follows:



(a) (b)



(c)

Fig. 3: Clustering results of the distillation learning for the Burgers equation under different κ values. 3a, $\kappa \to +\infty(\varphi \equiv 0)$, where the student network relies entirely on sampling data at sampling points, resulting in poor structure extraction. 3b, $\kappa = 0.8(\varphi \in [0,1])$, where the student network's supervision value fuses the teacher network's output and sampling data, leading to clearer structure extraction. 3c, $\kappa = 0(\varphi \equiv 1)$, where the student network extracts structure solely based on the teacher network's output, resulting in clear clustering results.

This result reinforces the importance of distillation learning: the higher the value of κ, meaning the greater the weight of the sample data used, the more detrimental it is to structure extraction. A lower κ value (e.g., 0.8) has a minimal impact on clustering structure, while $\kappa = 0$ (using only teacher network prediction data) produces clear structures. Therefore, the case studies validate the necessity of distillation learning in the Ψ-NN method, and the fusion method is not essential; in the main text's case studies, we set $\kappa = 0$ (using only teacher network prediction data).

3 Pseudocode

Algorithm 1 Ψ-NN Method Pseudocode

Input: Sample data $\mathcal{D} = \{(\check{\boldsymbol{x}}_i, \check{t}_i, \check{\boldsymbol{u}}_i)\}_{i=1}^{N}$;
Governing PDE $\mathcal{L}$, initial conditions $\mathcal{I}$, boundary conditions $\mathcal{B}$.
Hyperparameters: Learning rates η_T, η_S, maximum iterations N_T, N_S, node number N, layer number L, regularization weights ω_{rgl}, clustering threshold δ.
Output: Structured physics-informed neural network Ψ-NN with constrained parameters $\hat{\boldsymbol{\theta}}$.
Step 1.1: Teacher Network Training
    Initialize teacher network $\mathcal{N}_T(\boldsymbol{x}, t; \boldsymbol{\theta}_T)$.
for $k = 1$ **to** N_T **do**

1. Compute teacher loss MSE_T (Eq. (1.8) for forward problem or (1.11) for inverse problem) without regularization.
2. Update $\boldsymbol{\theta}_T \leftarrow \boldsymbol{\theta}_T - \eta_T \nabla_{\boldsymbol{\theta}_T} MSE_T$.
if converged **then**
break
end if
end for
Obtain teacher predictions $\widetilde{\boldsymbol{u}}_T$ on domain.
Step 1.2: Student Network Training
Initialize student network $\mathcal{N}_S(\boldsymbol{x}, t; \boldsymbol{\theta}_S)$.
for $k = 1$ **to** N_S **do**
1. Compute student loss MSE_S (Eq. (2.2)), including regularization $\Omega(\boldsymbol{\theta}_S)$(Eq. (3.2)).
2. Update $\boldsymbol{\theta}_S \leftarrow \boldsymbol{\theta}_S - \eta_S \nabla_{\boldsymbol{\theta}_S}(MSE_S + \Omega)$.
if converged **then**
break
end if
end for
Obtain trained parameters $\boldsymbol{\theta}_S$.
Step 2: Structure Extraction
for each layer $l = 1, \ldots, L$ **do**
1. Extract weight matrix $\boldsymbol{W}_S^{(l)}$ and bias $\boldsymbol{b}_S^{(l)}$ from $\boldsymbol{\theta}_S$.
2. Compute absolute values: $|\boldsymbol{W}_S^{(l)}|$ and $|\boldsymbol{b}_S^{(l)}|$.
3. Apply hierarchical clustering (HAC) with threshold δ to $|\boldsymbol{W}_S^{(l)}|$ and $|\boldsymbol{b}_S^{(l)}|$ to obtain cluster centers $\boldsymbol{c}_W^{(l)}$ and $\boldsymbol{c}_b^{(l)}$.
4. Replace weights and biases in same cluster by corresponding center: $\boldsymbol{W}_S^{(l)} \leftarrow \boldsymbol{c}_W^{(l)}$ and $\boldsymbol{b}_S^{(l)} \leftarrow \boldsymbol{c}_b^{(l)}$.
end for
5. Analyze inter-layer parameter relations, construct relation matrices $\boldsymbol{R}_{(l)}$.
Step 3: Structured Network Reconstruction
1. Build new network $\mathcal{N}_\Psi$ with structure defined by $\{\boldsymbol{R}_{(l)}\}$ and trainable parameters $\hat{\boldsymbol{\theta}}$.
2. Embed parameter constraints: for each layer l, enforce $\boldsymbol{W}_\Psi^{(l)} = f(\hat{\boldsymbol{\theta}}, \boldsymbol{R}_{(l)})$.
3. Output final structured physics-informed network Ψ-NN.
Optional: Train Ψ-NN using same loss as Step 1 for examination.

The loss functions used in the Ψ-NN method are defined as follows:

$$MSE_{Tp} = \frac{1}{M_T} \sum_{i=1}^{M_T} \left| \tilde{\boldsymbol{u}}_T^i - \tilde{\boldsymbol{u}}_S^i \right|^2 \tag{3.1}$$

$$MSE_{rgl} = \frac{1}{N} \sum_{i=1}^{N} \left| \boldsymbol{\theta}_S^i \right|^2 \tag{3.2}$$

$$MSE_S = \omega_{Tp} MSE_{Tp} + \omega_{rgl} MSE_{rgl} \tag{3.3}$$

4 Case settings

This section provides the details of the case settings in the numerical experiments. We choose to use *tanh* as the activation function and the Adam algorithm [5], which is a first-order gradient optimization algorithm for stochastic objective functions, to optimize the loss function MSE. All code is run on an Intel 12400f CPU and RTX4080 GPU. To ensure reproducibility, the random seed is fixed at 1234. Hyperparameter settings for each problem are shown in Table 3.

The comparative experiments are conducted by training the structure-embedded network automatically discovered by Ψ-NN (with trainable parameter relationships fixed by the relation matrix $\boldsymbol{R}$) and comparing its performance with PINN and PINN-post on the same case studies. The parameter settings for all experiments are shown in Table 4. The network width of the Ψ-NN structure is determined by the dimension d of its trainable parameter submatrices. Both PINN and PINN-post adopt a three-hidden-layer MLP architecture. The parameter sharing feature of the structure-embedded network leads to a different number of trainable parameters, but in order to ensure fairness and control variables, the Ψ-NN structure uses the same or fewer trainable parameters in each layer compared to the baseline models. Take Laplace case as an example, detailed (layer-wise) hyperparameter settings for each model are provided in Table 5.

Table 3: Model parameter settings for teacher model (T) and student model (S). In Poisson equation case, there is no teacher model since the low-frequency solution is directly used. All models use the Adam optimizer[5].

Case \ Item	Layer		Nodes		Iterations (1e5)		Step(1e-4)	
	T	S	T	S	T	S	T	S
Laplace	3	3	16	8	0.8	0.8	1	1
Burgers	3	3	16	8	1	2	1	1
Poisson	\	3	\	8	\	1	\	10

Table 4: Model parameter settings for comparison experiments. d is the parameter submatrix dimension of Ψ-NN structure and controls the network width. PINN and PINN-post use a three-hidden-layer MLP.

Case \ Item	Width (d & MLP layer)		Iterations (1e5)	Step(1e-4)
Laplace	8	[8,16,8]	0.8	1
Burgers	10	[10,20,10]	1	1
Poisson	20	[40,40,20]	0.3	1
Flow	20	[40,40,40]	2	10

Table 5: Trainable parameter dimensions of each layer (In Laplace question case for example). The parameter d denotes the dimension of a trainable parameter submatrix, thereby controlling the width of the network. MLP layer setting in PINN and PINN-post also convert to d expression for comparison. In Ψ-NN, the parameters between groups are connected via the learned relation matrix $\boldsymbol{R}$ (which is fixed and thus not counted as trainable parameters), whereas the baseline models do not include this relation matrix. In each layer, Ψ-NN uses the same or fewer trainable parameters as the baseline.

Layer \ Model	PINN		PINN-post		Ψ-NN structure	
	Weight	Bias	Weight	Bias	Weight	Bias
1	$4d$	$2d$	$4d$	$2d$	$4d$	$2d$
2	$8d^2$	$4d$	$8d^2$	$4d$	$4d^2$	$2d$
3	$8d^2$	$2d$	$8d^2$	$2d$	$8d^2$	$2d$
4	$2d$	1	$2d$	1	$2d$	1
Sum	$16d^2 + 8d + 1$ $6d$		$16d^2 + 8d + 1$ $6d$		$12d^2 + 6d + 1$ $6d$	

4.1 Laplace equation

The MSE and MSE_f, MSE_D are constructed as

$$MSE = \omega_f MSE_f + \omega_B MSE_B$$

where

$$\omega_f = \omega_B = 1$$

$$MSE_f = \frac{1}{M_f}\sum_{i=1}^{M_f}\left|\mathcal{L}(\tilde{u}^i, \lambda, \boldsymbol{x}^i)\right|^2$$

$$MSE_B = \frac{1}{M_B}\sum_{i=1}^{M_B}\left|\mathcal{B}(\tilde{u}^i, \boldsymbol{x}^i)\right|^2$$

where M_f is the number of configuration points, and M_B is the number of sample points (101 points evenly distributed on each side). $\tilde{u}$ is the network output, and $\breve{u}$ is the observed data.

In this case study, the teacher network is a multi-layer perceptron (MLP) with 3 hidden layers and 16 nodes, while the student network is a MLP with 3 hidden layers and 8 nodes. The learning rate for both networks is set to 1e-4, and both the teacher and student networks are trained for 8e4 iterations. The true value of the problem is $u = x^3 - 3xy^2$, which exhibits spatial symmetry.

In the control group, we use a standard PINN and a hard-mapping PINN to demonstrate the structural performance advantages of Ψ-NN. The mapping function of the hard-mapping network is represented by the spatial symmetry of x_2:

$$g : \boldsymbol{u}(x_1, x_2) \to \boldsymbol{u}(x_1, x_2) + \boldsymbol{u}(x_1, -x_2) \tag{4.1}$$

The symmetry is given by:

$$\mathcal{T} : (x_1, x_2) \mapsto (x_1, -x_2), \qquad \boldsymbol{u}(\mathcal{T}(x_1, x_2)) = \boldsymbol{u}(x_1, x_2) \tag{4.2}$$

The optimization speed is defined as the local or average reduction rate of the loss function, and the final L2 error results indicate the model's accuracy. The L2 error for the entire field is defined as:

$$L2 = \frac{1}{M} \sum_{i=1}^{M} \left| \tilde{u}^i - u^i \right|^2$$

where M is the number of grid points.

In the comparative experiments, the number of iterations for PINN, PINN-post, and Ψ-NN structures is set to 8e4. The dimension d of the trainable parameter submatrices in Ψ-NN is set to 4, while PINN and PINN-post use a three-hidden-layer MLP structure with [8,16,8] nodes. The learning rate is set to 1e-4.

4.2 Burgers equation

The MSE and $MSE_{f'}$, MSE_D that make up the loss function are defined as:

$$MSE = \omega_{f'} MSE_{f'} + \omega_D MSE_D$$

where

$$\begin{aligned}
&\omega_{f'} = \omega_D = 1 \\
&MSE_{f'} = \frac{1}{M_{f'}} \sum_{i=1}^{M_{f'}} \left| \tilde{u}_t^i - \tilde{u}^i \tilde{u}_x^i - \lambda_1 \tilde{u}_{xx}^i \right|^2 \\
&MSE_D = \frac{1}{M_D} \sum_{i=1}^{M_D} \left| \tilde{u}^i - \breve{u}^i \right|^2
\end{aligned}$$

In this case study, the teacher network is a MLP with 3 hidden layers and 16 nodes, while the student network is a MLP with 3 hidden layers and 8 nodes. The learning rate for both networks is set to 1e-4. The teacher network is trained for 1e5 iterations, while the student network is trained for 2e5 iterations. The ground truth is generated by the 4th-order Runge-Kutta method, employing a computational grid size of $1e3$ and a time step of $\Delta t = 2e{-}4$ s. Sampling points are randomly generated using Latin hypercube sampling and distributed within the computational domain. The baseline includes standard PINN, hard-mapping PINN, and the structure extracted by Ψ-NN. The mapping function for PINN-post is:

$$g : \boldsymbol{u}(\boldsymbol{x}, t) \to \boldsymbol{u}(\boldsymbol{x}, t) - \boldsymbol{u}(-\boldsymbol{x}, t) \tag{4.3}$$

The boundary conditions of the problem can be viewed as:

$$\mathcal{B} := u = 0 \quad x \in -1, 1 \tag{4.4}$$

The initial condition of the problem can be viewed as:

$$\mathcal{I} := u = -sin(\pi x), \quad t = 0 \tag{4.5}$$

The symmetry is given by:

$$\mathcal{T}:(x_1,x_2)\mapsto(x_1,-x_2),\qquad \boldsymbol{u}(\mathcal{T}(x_1,x_2))=-\boldsymbol{u}(x_1,x_2) \tag{4.6}$$

In the comparative experiments, the number of iterations for PINN, PINN-post, and Ψ-NN structures is set to 1e5. The dimension d of the trainable parameter submatrices in Ψ-NN is set to 5, while PINN and PINN-post use a three-hidden-layer MLP structure with [10,20,10] nodes. The learning rate is set to 1e-4.

After the change in viscosity parameter, the time step of the R-K method is modified to $\Delta t = 7e{-}5$ s.

4.3 Poisson equation

Poisson's equation includes homogeneous Dirichlet boundary conditions:

$$\mathcal{B} := u|_{\boldsymbol{x}} = 0,\quad \boldsymbol{x}\in\partial\Omega \tag{4.7}$$

where $\Omega=[0,1]$. The loss function is defined as:

$$MSE = \omega_f MSE_f + \omega_D MSE_D$$

where

$$\begin{aligned}
&\omega_f = \omega_D = 1\\
&MSE_f = \frac{1}{M_f}\sum_{i=1}^{M_f}\left|\nabla^2\tilde{u}^i - \tilde{f}^i\right|^2\\
&MSE_D = \frac{1}{M_D}\sum_{i=1}^{M_D}\left|\tilde{u}^i - \check{u}^i\right|^2
\end{aligned}$$

where M_f and M_D are the number of sampling points for the equation loss and data loss, respectively. f is the source term. Inside the computational domain, we use $1e2\times 1e2$ sampling points, and on each boundary, 101 points are uniformly sampled. In this case study, the student network is a MLP with 3 hidden layers and 8 nodes. The initial learning rate is set to 1e-3, and the student network is trained for 2e5 iterations. The learning rate is reduced to 1e-1 at iterations 5e4 and 15e4. Since the Poisson equation uses a low-frequency solution function to supervise the student network, there is no teacher model. In the comparative experiments, the control group includes standard PINN, PINN-post, and the structure extracted by Ψ-NN. In the comparative experiments, the number of iterations for PINN, PINN-post, and Ψ-NN structures is set to 3e4. The mapping function for PINN-post is:

$$g:\boldsymbol{u}(x_1,x_2)\to\boldsymbol{u}(x_1,x_2)+\boldsymbol{u}(x_2,x_1) \tag{4.8}$$

The symmetry is given by:

$$\mathcal{T}:(x_1,x_2)\mapsto(x_2,x_1),\qquad \boldsymbol{u}(\mathcal{T}(x_1,x_2))=\boldsymbol{u}(x_1,x_2) \tag{4.9}$$

In the comparative experiments, the number of iterations for PINN, PINN-post, and Ψ-NN structures is set to 3e4. The dimension d of the trainable parameter submatrices in Ψ-NN is set to 20, while PINN and PINN-post use a three-hidden-layer MLP structure with [40,40,20] nodes. The learning rate is set to 1e-4.

4.4 Steady flow around a cylinder

The case conditions are: dynamic viscosity $\mu = 0.02kg/(m\cdot s)$, density $\rho = 1.0kg/m^3$, and the maximum flow rate limit is $U_{max} = 1.0m/s$.

The MSE and MSE_f, MSE_B are constructed as

$$MSE = \omega_f MSE_f + \omega_B MSE_B$$

where

$$\omega_f = 1 \quad \omega_B = 2$$

$$MSE_f = \frac{1}{M_f}\sum_{i=1}^{M_f}\left[\left|\mathcal{L}_1(\tilde{u}^i, x^i)\right|^2 + \left|\mathcal{L}_2(\tilde{u}^i, x^i)\right|^2 + \left|\mathcal{L}_3(\tilde{u}^i, x^i)\right|^2\right]$$

$$MSE_B = \frac{1}{M_B}\sum_{i=1}^{M_B}\left|\mathcal{B}(\tilde{u}^i, x^i)\right|^2$$

The mapping function used by PINN-post for the output $[p, u, v]^T$ is defined by the transformation set $\mathcal{G} = g_1, g_2$:

$$g_1 : \boldsymbol{u}(\boldsymbol{x}) \to \boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{u}(-\boldsymbol{x}) \tag{4.10}$$

$$g_2 : \boldsymbol{u}(\boldsymbol{x}) \to \boldsymbol{u}(\boldsymbol{x}) - \boldsymbol{u}(-\boldsymbol{x}) \tag{4.11}$$

$$\mathbb{G} = \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_1 & 0 \\ 0 & 0 & g_2 \end{bmatrix} \tag{4.12}$$

Ψ-NN incorporates the network structures discovered in the Laplace and Burgers case studies as sub-networks: the spatial symmetry from the Laplace case corresponds to the outputs of pressure p and x-velocity u, while the structure from the Burgers case corresponds to the output of y-velocity v. Trainable weights are applied to the outputs of these two sub-network structures, thereby unifying both structures within a parent network. The parameter matrix dimension d for the sub-networks is set to 20. Both PINN and PINN-post utilize a three-hidden-layer MLP with 60 nodes per layer. The number of training iterations is set to 2e5, with an initial learning rate of 1e-3, which is reduced by a factor of 1e-1 at iterations 5e4 and 1.5e5.

5 Hierarchical Agglomerative Clustering

Hierarchical Agglomerative Clustering (HAC) is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types: Agglomerative and Divisive. Agglomerative is a "bottom-up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. Divisive is a "top-down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In this work, we use the Agglomerative method to cluster the weights of the NN. The algorithm follows these steps:

1. Compute the distance matrix of the weights.
2. Merge the two closest weights into a cluster.
3. Update the distance matrix.
4. Repeat steps 2 and 3 until all weights are merged into one cluster.
5. Cut the tree at a certain height to get the clusters.
6. Replace the weights in each cluster with the cluster center.
7. Update the weights of the NN.

The distance matrix is calculated by the Euclidean distance between the weights. The distance between two clusters is calculated by the average distance between the weights in the two clusters.

6 Proof of Theorem 1 and Theorem 2

6.1 Proof of Theorem 1

Proof. Consider n parameters $\theta_1, \theta_2, \ldots, \theta_n$ in the same hidden layer, which play equivalent roles in the network. This can be reformulated using the action of the symmetric group S_n, where S_n is the permutation group of n elements:

$$S_n(\theta_1, \theta_2, \ldots, \theta_n) = (\theta_{\sigma(1)}, \theta_{\sigma(2)}, \ldots, \theta_{\sigma(n)}) \tag{6.1}$$

where σ is any permutation. Due to the equivalence of the parameters, these trainable parameters are invariant under the action of S_n:

$$\mathcal{N}(\theta_1, \theta_2, \ldots, \theta_n) = \mathcal{N}(S_n(\theta_1, \theta_2, \ldots, \theta_n)) \tag{6.2}$$

For the minimization under the L2 regularization term:

$$\min_{\theta_1,\theta_2,\ldots,\theta_n} \quad \text{subject to} \quad \sum_{i=1}^{n} \theta_i = C \tag{6.3}$$

where C is a constant representing implicit constraints from other loss terms.

Solving with the Lagrange multiplier method, the optimal solution is:

$$\theta_1 = \theta_2 = \cdots = \theta_n = \frac{C}{n} \tag{6.4}$$

In this case, the sum of squares $\sum_{i=1}^{n} \theta_i^2 = \frac{C^2}{n}$ is minimized, while any asymmetric allocation $\theta_i \neq \theta_j$ leads to a larger regularization penalty. Therefore, these parameters tend to become equal under regularization constraints. □

6.2 Proof of Theorem 2

Proof. Consider the symmetry of trainable parameter values in the same hidden layer, i.e.:

$$\frac{\partial \mathcal{N}}{\partial |\theta_1|} = \frac{\partial \mathcal{N}}{\partial |\theta_2|} = \cdots = \frac{\partial \mathcal{N}}{\partial |\theta_n|} \tag{6.5}$$

The update rule for parameter θ_i is:

$$\theta_i^{\text{new}} = \theta_i - \eta(\frac{\partial \mathcal{L}}{\partial \theta_i} + \lambda \theta_i) \tag{6.6}$$

where η is the learning rate and λ is the regularization weight.

The update rule for the parameter difference $\delta = |\theta_i| - |\theta_j|$ of another parameter θ_j satisfies:

$$\Delta^{(t+1)} = \Delta^{(t)} - \eta\lambda\Delta^{(t)} = (1 - \eta\lambda)\Delta^{(t)} \tag{6.7}$$

When $0 < \eta\lambda < 1$, the parameter difference Δ decays exponentially to 0, meaning that $|\theta_i|$ and $|\theta_j|$ approach equality. □

7 Conflict between structure extraction methods and physics-informed method

The trainable parameters of PINN are insensitive to parameter decay operations [7], and the introduction of structure extraction methods leads to a decline in network performance. In the Laplace example, a comparison between PINN and PINN with L1 regularization, L2 regularization, and the GrOWL [8] structure extraction method yields the results shown in main text (Fig. 1). All regularization techniques significantly impact the fitting accuracy of PINN, which aligns with the intuition that regularization prevents overfitting.

Among them, L1 regularization has the greatest impact on accuracy. This method can automatically perform feature selection and is suitable for scenarios with high-dimensional feature redundancy, but may randomly select only one feature when multiple features are correlated [9], resulting in a significant drop in model accuracy. The GrOWL method can select relevant features in the network and encourages the coefficients of each layer that are strongly correlated with the previous layer to be nearly or exactly equal [8]. Although this accelerates model iteration in the initial $1e1$ steps, it is detrimental to further accuracy improvement. In contrast, L2 regularization, as a weight smoothing technique, has the least impact on accuracy, but even so, the loss of PINN increases from $1e-5$ to $1e-2$, indicating a substantial negative effect.

Moreover, the resulting structures from these methods are not extractable. As shown in Fig. 5, the clustering results of the trainable parameter matrices are highly scattered across different hidden layers, and the extracted cluster centers are random. For intuitive comparison, Fig. 4 shows the changes in the weights of the third hidden layer under different regularization treatments.

L-1 regularization causes most network parameters to cluster around 0, with significant contributions from trainable parameters being clustered separately. This greatly reduces multi-feature correlation, meaning that there is no parameter linkage or common feature representation, which contradicts the goal of structure extraction. Under the influence of L2 regularization, the overall values of trainable parameters decrease and become more concentrated, leading to denser cluster centers, but still without obvious structural features.

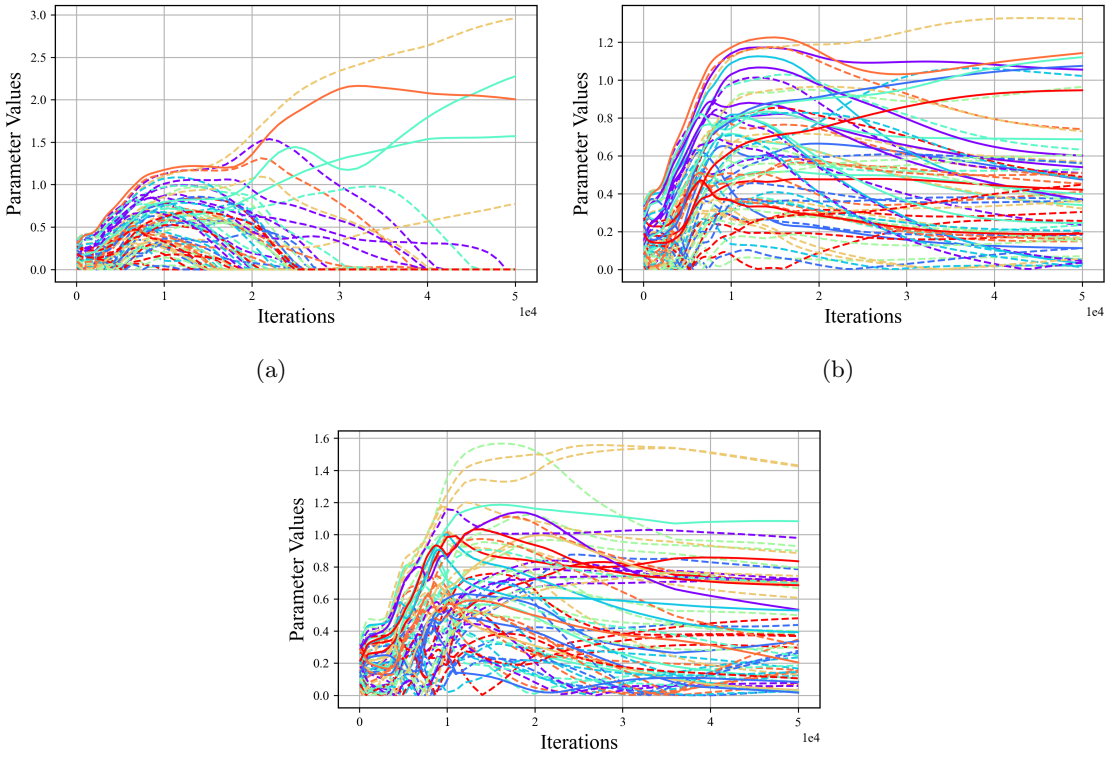


Fig. 4: Parameter evolution under different regularization methods. 4a, Parameter tendency after L-1 regularization. 4b, Parameter tendency after L-2 regularization. 4c, Parameter tendency after GrOWL regularization. The weights tendency of the third hidden layer in PINN with different structure extraction methods. Each line represents the weights of a neuron in the hidden layer, and the negative values are shown as imaginary lines.

Finally, the GrOWL method does not impose numerical penalties on parameters but encourages parameter linkage. However, the random clustering it induces results in more dispersed cluster centers, leading to the failure of structure extraction.

As a comparison, we provide the clustering results of Ψ-NN in Fig. 6. The Ψ-NN method exhibits clear structural features.

8 Further parameter compression

To further compress the network, explicit structure merging is performed. The parameter vector $\boldsymbol{w} = [w_1, \cdots, w_n]^T$ describes n cluster centers, and the original parameter matrix can be expressed as the tensor product of n symbol matrices and the parameter vector:

$$\boldsymbol{\Theta}_{0-1} = [\boldsymbol{S}_1, \cdots, \boldsymbol{S}_n] \otimes \boldsymbol{w} \tag{8.1}$$

where the symbol block matrix is formed by three independent symbol matrices $\boldsymbol{S}_i \in \mathbb{R}^{\dim\boldsymbol{\theta}_l \times \dim\boldsymbol{\theta}_{l-1} \times n}$ concatenated along the third dimension, where $\dim\boldsymbol{\theta}_{l-1}$ and $\dim\boldsymbol{\theta}_l$ are the dimensions of trainable parameters in the $(l-1)$th and lth layers, respectively. The elements are $\{+1, -1, 0\}$, indicating the sign pattern corresponding to the parameter $\boldsymbol{w}$ dimension. Taking the input layer to the first hidden layer parameter $\boldsymbol{\Theta}_{0-1}$ in the Laplace example as an example, the arranged symbol matrix is:

$$\boldsymbol{S}_1 = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{8.2}$$

$$\boldsymbol{S}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 1 & 1 & 0 \end{bmatrix}^T \tag{8.3}$$

$$\boldsymbol{S}_3 = \begin{bmatrix} 0 & -1 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}^T \tag{8.4}$$

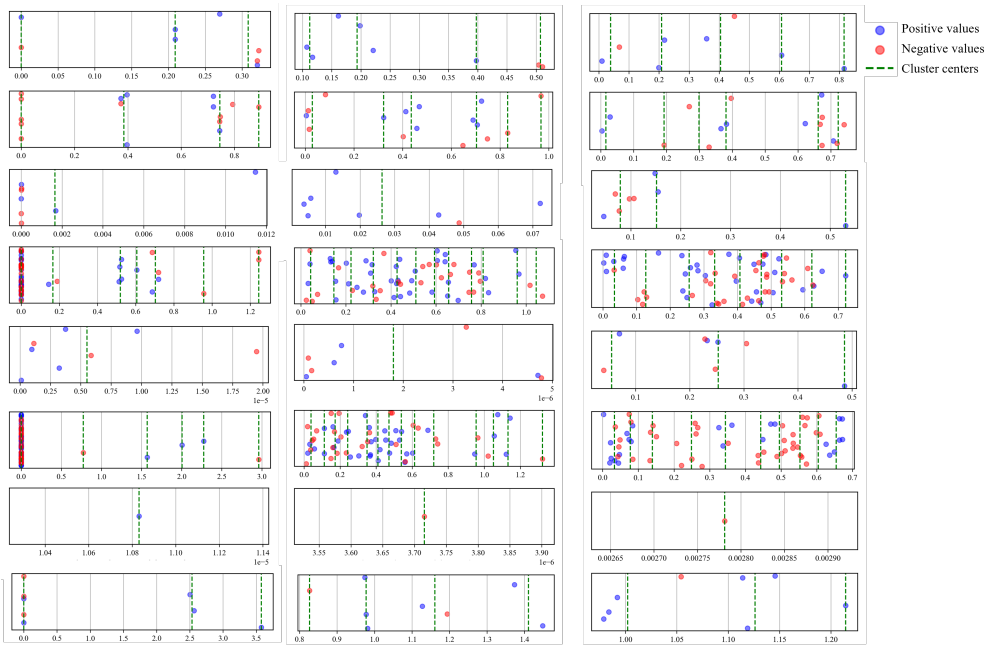


Fig. 5: The clustering results of the weights in PINN with other methods. The odd rows show the distribution of biases, while the even rows show the distribution of weights. The rows are arranged in order from input to output layer.

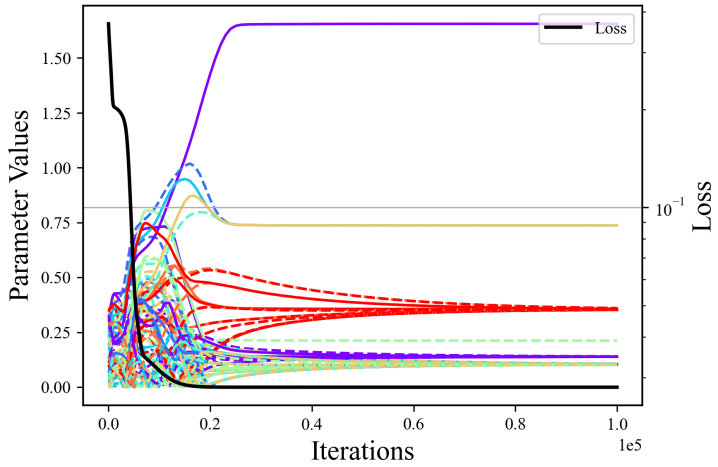


Fig. 6: The evolution of the third hidden layer weight parameters and the loss function of the Ψ-NN student network. As the loss function decreases and stabilizes, the weight parameters gradually cluster into multiple groups, exhibiting clear structural features.

In the symbol matrix S, the parameters in the row indices $S_{[2:4]}$ and $S_{[5:7]}$ exhibit opposite reuse characteristics in terms of sign patterns. Therefore, the original matrix can be further compressed by replacing it with submatrices:

$$\boldsymbol{S}_1 = \begin{bmatrix} \boldsymbol{W}_{11} & \boldsymbol{W}_{12} & -\boldsymbol{W}_{12} \end{bmatrix}^T \tag{8.5}$$

$$\boldsymbol{S}_2 = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{W}_{22} & -\boldsymbol{W}_{22} \end{bmatrix}^T \tag{8.6}$$

$$\boldsymbol{S}_3 = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{W}_{32} & -\boldsymbol{W}_{32} \end{bmatrix}^T \tag{8.7}$$

Here, a simple reuse replacement idea is provided, but it is worth noting that excessive compression can lead to the disappearance of the network structure, resulting in a decline in network performance. For example, the most extreme compression, where all parameters are represented by the same submatrix, directly leads

to no difference between the network and a standard DNN. Therefore, the degree of compression needs to be adjusted according to specific examples.

References

[1] Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. Nature Reviews Physics **3**(6), 422–440 (2021)

[2] Raissi, M., Perdikaris, P., Karniadakis, G.E.: 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)

[3] Hinton, G., Vinyals, O., Dean, J.: Distilling the Knowledge in a Neural Network. arXiv. arXiv:1503.02531 (2015)

[4] Kullback, S., Leibler, R.A.: On information and sufficiency. Annals of Mathematical Statistics **22**(1), 79–86 (1951)

[5] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv. arXiv:1412.6980, 2014. (2017)

[6] Li, Y., Yang, J., Wang, D.: Self-knowledge distillation enhanced universal framework for physics-informed neural networks. Nonlinear Dynamics (2025)

[7] Fuhg, J.N., Jones, R.E., Bouklas, N.: Extreme sparsification of physics-augmented neural networks for interpretable model discovery in mechanics. Computer Methods in Applied Mechanics and Engineering **426**, 116973 (2024)

[8] Zhang, D., Wang, H., Figueiredo, M., Balzano, L.: Learning to share: Simultaneous parameter tying and sparsification in deep learning. In: International Conference on Learning Representations (2018)

[9] Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological) **58**(1), 267–288 (1996)