A Limitations

In this study, we mainly focus on the cases where PDE parameters are the coefficients of various PDE terms. In particular, we train the proposed model on a set of benchmark PDEs (e.g., convection equations or reaction equations or Helmholtz equations). As a future direction of this study, we list the following potential extensions:

1. Extending our framework so that the set of basis vectors are parameter dependant to some extent, as we did to have adaptive rank (i.e, the rank structure is parameter dependant),
2. (Related to the above extension) Extending our framework to be equipped with a specialized optimizer, which alternately update the basis vectors and diagonal elements,
3. Extending our framework to more general settings, attempting to learn the solutions of parameterized PDEs where the PDE parameters define initial/boundary conditions.

Regarding the alternating solver: we expect that making the basis vector learnable in phase 2 would increase expressivity. However, in achieving low error, the difficulty is expected to come from the training algorithm, where the similar phenomena were observed in matrix decomposition methods. Consider finding a low-rank decomposition of matrix such that min with a norm induced from an inner product. In such a problem, updating all together in an interactive solver typically introduces more complexity. To avoid such difficulty, special solvers have been developed such as direct linear algebraic decompositions [38] which does not use a gradient-based optimization method, or notably, alternating minimization for matrix completion [39]. In the context of low-rank approximation of solutions of PDEs, similar alternating approaches (e.g., alternating least-squares or alternating energy minimization) have been shown to be more effective [40–42].

B Remark on our low-rank approximation in general deep learning

It had been reported that deep neural networks have low-rank biases for learned representations, which is the reason why over-parameterized neural networks do not always fail even when relatively small data is given [43–45]. One can consider that our proposed method imposes an effective learning bias, i.e., our adaptive low-rank approximation, on PINNs to make their learning process easier than before. Thus, we focus on the PINN's notorious failure modes and empirically prove the efficacy of our design.

C Reduced-Order Modeling (ROM)

In the following, we summarize the offline and the online phases of traditional linear-subspace reduced-order modeling approaches.

Offline phase

- Perform high-fidelity simulations on a training PDE parameter instances $\{\boldsymbol{\mu}_{\text{train}}^{(i)}\}_{i=1}^{n_{\text{train}}}$, i.e.,

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{u}; \boldsymbol{\mu}_{\text{train}}^{(i)})$$

- Collect solution snapshots from the high-fidelity simulation

$$U = [\boldsymbol{u}(t_1, \boldsymbol{\mu}_{\text{train}}^{(1)}), \boldsymbol{u}(t_2, \boldsymbol{\mu}_{\text{train}}^{(1)}), \cdots, \boldsymbol{u}(T, \boldsymbol{\mu}_{\text{train}}^{(1)}), \boldsymbol{u}(t_1, \boldsymbol{\mu}_{\text{train}}^{(2)}), \cdots, \boldsymbol{u}(T, \boldsymbol{\mu}_{\text{train}}^{(n_{\text{train}})})]$$

- Compute SVD on U: $U = \Psi D \Phi^{\mathsf{T}}$
- Truncate the series Ψ_p

Online phase

- Perform inexpensive simulation on unseen test PDE parameter instances $\{\boldsymbol{\mu}_{\text{test}}^{(i)}\}_{i=1}^{n_{\text{test}}}$,
- Represent a solution as $\tilde{u}(t, \boldsymbol{\mu}_{\text{test}}^{(i)}) = \Psi_p \boldsymbol{c}(t, \boldsymbol{\mu}_{\text{test}}^{(i)})$

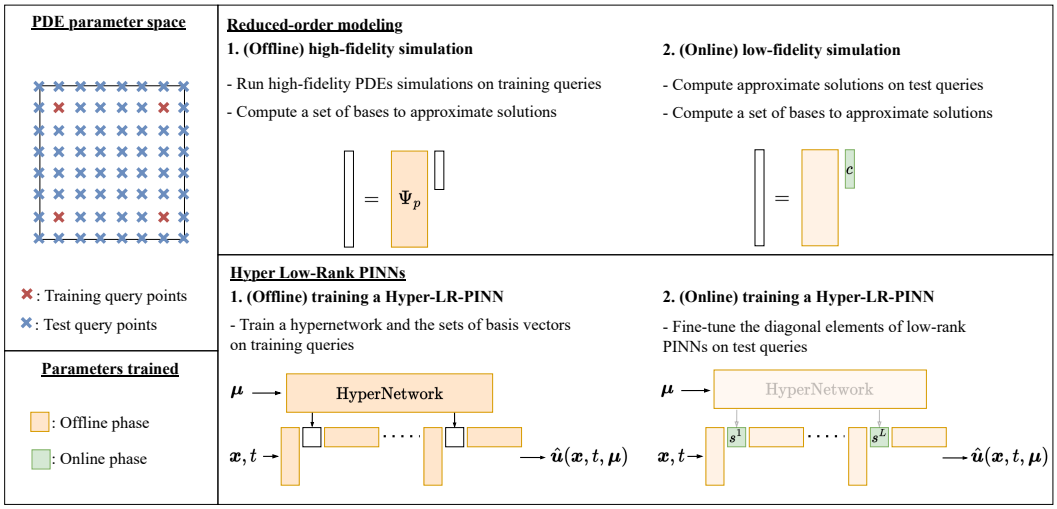


Figure 8: Graphical comparisons between ROMs and Hyper-LR-PINNs.

- Project the dynamical system into the low-dimensional space

$$\begin{aligned} \frac{\mathrm{d}(\Psi_p \boldsymbol{c})}{\mathrm{d}t} &= \boldsymbol{f}(\Psi_p \boldsymbol{c}; \boldsymbol{\mu}_{\text{test}}^{(i)}) \\ \Leftrightarrow \frac{\mathrm{d}(\Psi_p^\mathsf{T} \Psi_p \boldsymbol{c})}{\mathrm{d}t} &= \Psi_p^\mathsf{T} \boldsymbol{f}(\Psi_p \boldsymbol{c}; \boldsymbol{\mu}_{\text{test}}^{(i)}) \quad \text{(multiply } \Psi_p^\mathsf{T} \text{on both sides)} \\ \Leftrightarrow \frac{\mathrm{d}(\boldsymbol{c})}{\mathrm{d}t} &= \hat{\boldsymbol{f}}(\Psi_p \boldsymbol{c}; \boldsymbol{\mu}_{\text{test}}^{(i)}) \quad (\Psi_p^\mathsf{T} \Psi_p = I \text{ and } \hat{\boldsymbol{f}} = \Psi_p^\mathsf{T} \boldsymbol{f} \in \mathbb{R}^p) \end{aligned}$$

Analogy between ROMs and the proposed Hyper-LR-PINNs Figure 8 draws an analogy between ROMs and Hyper-LR-PINNs. It is highlighted that the both approaches perform heavy-lifting in the offline phase to make the online phase less expensive so that solutions at the large number of test PDE parameter instances can be evaluated rapidly.

Difference from reservoir computing We agree that there is a very high-level analogy that can be made to reservoir computing (RC) [46], since in the second phase of our two-phase training procedure only part of the parameters are trained. However, this is a feature shared with all models making use of encoding-decoding schemes. Moreover, an important distinguishing feature in RC is the recurrent neural network, whereas our neural network architecture is feedforward. The fact that we are specifically solving parametrized partial differential equations also makes our model more specialized. We believe these points makes our model much more similar to a ROM, where a dimension-reduced implicit representation is learned in the first phase. In the second phase, a family of functions is quickly approximated by training a few parameters.

D Preliminary experiments

D.1 Approximating trained PINNs' weights with low-rank matrices

To motivate our study, we perform some preliminary experiments on the canonical one-dimensional viscous Burgers' equation. We parameterize a PINN as a multi-layer perceptron (MLP) with 5 hidden layers and 40 neurons in each layer, followed by the TANH nonlinearity; the l-th layer of the MLP can be represented as $\boldsymbol{h}^{l+1} = \sigma(W^l \boldsymbol{h}^l + \boldsymbol{b}^l)$, where W and $\boldsymbol{b}$ denote the weight and the bias, respectively, σ denotes the nonlinear activation, and $\boldsymbol{h}$ denotes the post activation. Training with L-BFGS results in the solution accuracy measured in the relative L2 error (around .1% error, the black dashed line in Figure 9).

Figure 9: [Viscous Burgers' equation] The relative L2 errors of the low-rank reconstructed models and the low-rank retrained models.

After the training, we build a low-rank PINN in the following steps: we i) take the weight matrices of hidden layers (except the input and the output layers), ii) take the SVDs, and iii) assemble weight matrices by retaining the r largest singular values and corresponding singular vectors such that

$$W^l = U^l \Sigma^l V^{l\mathsf{T}}, \quad \text{and} \quad W_r^l = U_r^l \Sigma_r^l V_r^{l\mathsf{T}}, \tag{3}$$

which results in a low-rank PINN (LR-PINN) with a rank of r, which shares the same model parameters with the trained PINN, except the hidden layer weights obtained from the previous truncation step. We then vary the value of r from 1 to 40 and evaluate the resulting LR-PINNs on the test set ("reconstruction" in Figure 9); the blue line depicts the error measured on the test set for LR-PINNs with varying r. Even truncating 3–4 smallest singular values seems to yield significant accuracy degradation by an order of magnitude.

Next, we investigate what happens if we further train LR-PINN from the model parameters obtained from the truncation Eq. (3). Instead of training all model parameters, U, Σ, V, we i) represent the hidden layer weights in a factored form $W_r^l = U_r^l \Sigma_r^l V_r^{l\mathsf{T}}$, ii) fix the basis matrices (U_r^l, V_r^l) and iii) make $\boldsymbol{s}_r$ only the trainable parameters, $\Sigma_r^l = \mathrm{diag}\left(\boldsymbol{s}_r^l\right)$. After training, we test the trained models on the same test data set. As shown by the red dashed line in Figure 9, its accuracy becomes comparable to that of the full model when $r \geq 20$, which shows that *the weights of hidden layers have low-rank structures.*

D.2 A study on the effect of rank and orthogonality of basis sets for varying PDE parameters

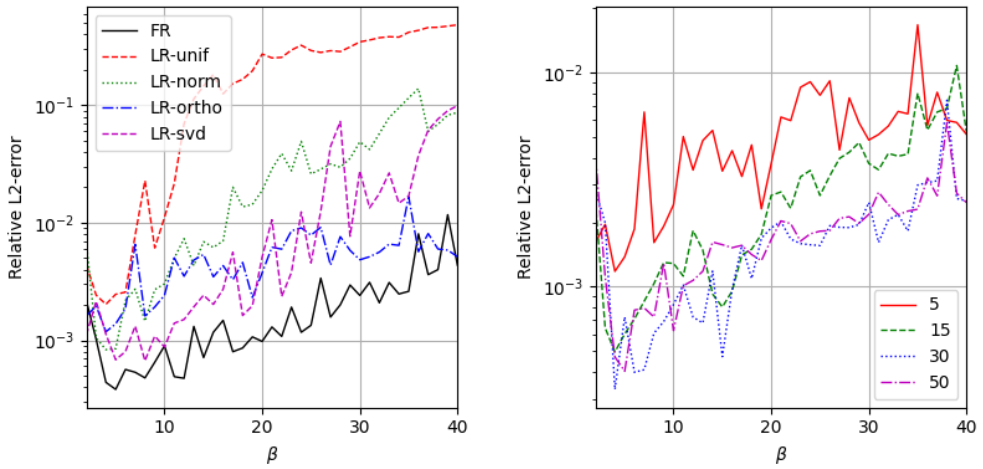
As an example parameterized PDE, we consider a one-dimensional convection-diffusion equations:

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

where $u(x, t; \boldsymbol{\mu})$ denotes the solution satisfying the equation, with $\boldsymbol{\mu} = (\beta, \nu)$ denote the convection and diffusion coefficients, respectively. The higher the ratio $\frac{\beta}{\nu}$ is, the more convection-dominated the problem is, which is considered to be a more challenging scenario for PINNs. In the following experiments, we set $\nu = 1$ fixed and vary β from 2 to 40 to control the difficulty of training LR-PINNs. For training, we use a curriculum-learning-type training approach proposed in [18]; in a high β regime (e.g., $\beta \geq 20$), training PINNs directly on the target β typically fails and the authors resolved this issue by curriculum learning which i) starts with a low β value (e.g., $\beta = 1$) and ii) over the course of training, gradually increases the value of β until it reaches to the target β.

Setting the basis vectors, U_r and V_r: We have tested four different schemes for setting the basis vectors: three schemes that are commonly used for initializing the weights of FC layers: HE-UNIFORM, HE-NORMAL [47], and ORTHOGONAL [48], and one scheme that extracts the left and right singular vectors from a PINN trained for $\beta = 1$ and use them for setting the basis vectors, which we denote by SVD-INIT.

Figure 10(a) shows the errors of approximate solutions obtained by all trained full-rank PINNs (FR) and LR-PINNs with $r = 5$ for varying β values. Among the LR-PINNs, the ones initialized with ORTHOGONAL (LR-ortho) and SVD-INIT (LR-svd) outperform the ones initialized with HE-UNIFORM (LR-unif) and HE-NORMAL (LR-norm). LR-ortho tends to perform better than LR-svd in



(a) Varying initializations

(b) Varying ranks

Figure 10: [Convection-diffusion equations] The relative L2 errors of full-rank PINNs (FR) and low-rank PINNs (LR): (a) LR-PINN with varying weight initialization schemes including orthogonal, He-uniform, He-normal, and extracted basis from the pre-trained PINN, and (b) LR-PINN with varying ranks ($r = 5, 15, 30, 50$) and the orthogonal initialization.

very high β regime ($\beta \geq 30$). Compared to LR-svd, LR-ortho has another advantage not requiring pretrainig of a PINN. An apparent observation is that *orthogonality in basis vectors helps achieving better accuracy* (Observation #1).

The effect of rank: Next, with LR-ortho, we vary the rank $r = \{5, 15, 30, 50\}$. Figure 10(b) depicts the errors of LR-PINNs for varying β and ranks. For lower β ($\beta < 20$), LR-PINNs with $r = 15, 30, 50$ produce comparable results. For higher β values ($\beta \geq 20$), it appears that a higher rank is required to achieve a certain level of accuracy, presumably due to the difficulty of the numerical optimization problem. From this result, the second observation we make is that *required ranks for varying $\boldsymbol{\mu}$ to achieve a certain level of accuracy could vary* (Observation #2) and, thus, a solution method that adaptively decides the ranks of each individual hidden layers is required.

E Proposed two-phase training algorithm

Algorithm 1 summarizes the two-phase algorithm we describe in Section 3.2.

F Reproducibility

Software and hardware environments We implement and conduct experiments with PYTHON 3.9.7 and PYTORCH 1.13.0, CUDA 11.6.1, NVIDIA Driver 470.74, i9 CPU, NVIDIA RTX A5000, and RTX 2080 TI. Our source code for the benchmark PDEs is mainly based on `https://github.com/a1k12/characterizing-pinns-failure-modes` (MIT License).

Hyperparameters We collect 256/100/1,000 points from initial/boundary/collocation points, and 1,000 test points for each CDR equation. For the 2D Hemlholtz equation, we collect 400/1,000/10,000 points from boundary/collocation/test points. The train/test sets contain non-overlapping spatial/temporal collocation points. The PINN baselines consist of 6 FC layers with 50 neurons. For our methods, we employ 3 hidden layers (L=M=3). The Adap optimizer is used with learning rate 1e-3 for PINN baselines, and 1e-3 and 2.5e-4 for Phase1 and Phase2 of meta-learning methods. These hyperparameters are common in all our experiments.

Algorithm 1 Two-phase training of the proposed method

/* Phase 1 */
Input: A set of training sampled points: $\{(\boldsymbol{x}, t, \boldsymbol{\mu}^{(i)})\}_{i=1}^{n_{p_1}}$, where n_{p_1} is number of PDE equations used in Phase 1.
Initialize weights and biases $\{(U^l, V^l, \boldsymbol{b}^l)\}_{l=1}^{L}$, $W^0, W^{L+1}, b^0, b^{L+1}$ of LR-PINN, and those $\{(W^{\text{emb},m}, \boldsymbol{b}^{\text{emb},m})\}_{m=1}^{M}$, $\{(W^{\text{hyper},l}, \boldsymbol{b}^{\text{hyper},l})\}_{l=1}^{L}$ of the hypernetwork.
for $epoch = 1$ to ep_1 **do**
  for $i = 1$ **to** n_{p_1} **do**
    $\{\boldsymbol{s}^l(\boldsymbol{\mu}^{(i)})\}_{l=1}^{L} = f^{\text{hyper}}(\boldsymbol{\mu}^{(i)})$
    Compute forward pass: $u_\Theta((\boldsymbol{x}, t); \{\boldsymbol{s}^l(\boldsymbol{\mu}^{(i)})\}_{l=1}^{L})$
    Compute PINN loss and update model parameters via backpropagation
  end for
end for

/* Phase 2 */
Input: A set of training sampled points: $\{(\boldsymbol{x}, t, \boldsymbol{\mu}^{\text{target}})\}$
Freeze $\{(U^l, V^l, \boldsymbol{b}^l)\}_{l=1}^{L}$, $\{(W^{\text{emb},m}, \boldsymbol{b}^{\text{emb},m})\}_{m=1}^{M}$, and $\{(W^{\text{hyper},l}, \boldsymbol{b}^{\text{hyper},l})\}_{l=1}^{L}$
Initialize $\{\boldsymbol{s}^l\}_{l=1}^{L} = f^{\text{hyper}}(\boldsymbol{\mu}^{\text{target}})$
for $epoch = 1$ to ep_2 **do**
  Compute forward pass: $u_\Theta((\boldsymbol{x}, t); \{\boldsymbol{s}^l(\boldsymbol{\mu}^{\text{target}})\}_{l=1}^{L})$
  Compute PINN loss and update model parameters via backpropagation
end for

G Meta-learning baselines: MAML and Reptile

We consider the two most representative optimization-based meta-learning algorithms: model-agnostic meta learning (MAML) [20] and Reptile [21]. In the parameterized PDEs setting, we can define a task, $\tau^{(i)}$, as a specific setting of PDE parameters, $\boldsymbol{\mu}^{(i)}$. Both MAML and Reptile seek an initial weights of a PINN, which can serve as a good starting point for gradient-based optimizers when a solution of a new unseen PDE parameters is sought. Both methods consist of an *inner* loop and an *outer* loop. The inner loop takes k optimization gradient descent steps to update model parameters from a current of the meta initial points θ_0^l given a training task $\tau^{(i)}$. Here, this k-step update is denoted as $\theta_k(\theta_0^l, \tau^{(i)})$. Then the outer loop updates the meta-learned initial points using the information obtained from the inner loop such that

$$\theta_0^{l+1} = \theta_0^l - \beta \nabla_\theta L(\theta_k(\theta, \tau^{(l)}))|_{\theta=\theta_0^l}, \qquad \text{(MAML)}$$
$$\theta_0^{l+1} = \theta_0^l - \beta(\theta_k(\theta_0^l, \tau^{(l)}) - \theta_0^l), \qquad \text{(Reptile)}$$

where Reptile has a simpler update rule, which does not require the second-order gradients.

H Comparisons of baselines and our method

Table 5 compares the baseline models with our method in three aspects: a target function being approximated, an initialization scheme, and the rank structure. For vanilla PINN and their variants, the function being approximated is $u(\boldsymbol{x}, t; \boldsymbol{\mu})|_{\boldsymbol{\mu}=\boldsymbol{\mu}^i}$, which is the solution of the parameterized PDE that is realized at a certain PDE parameter $\boldsymbol{\mu}^i$. On the other hand, PINN-P and our proposed method models the parameterized solution itself $u(\boldsymbol{x}, t; \boldsymbol{\mu})$. The initilization column indicates if meta-learning is used or not. The rank structure column indicates if the rank is adaptively chosen for different values of $\boldsymbol{\mu}$.

I Train and test datasets generation

Train/test datasets are collected at collocations points for varying PDE parameters of a specific PDE type; e.g., for convection equation, we only vary β, resulting in $\{\{(\boldsymbol{x}_k, t_k; \beta^{(i)})\}_{k=1}^{n_c^\ell}\}_{i=1}^{n_\beta}$, where n_c^ℓ denotes the number of collocation points in train ($\ell = 0$) and test ($\ell = 1$) sets, n_β denotes the number

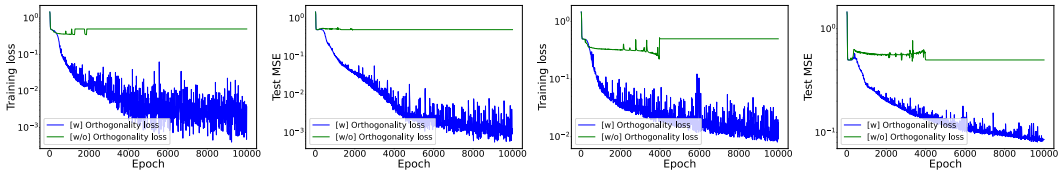
Table 5: Comparisons of baselines and our method

	Target function		Initialization		Rank structure	
	$u(\boldsymbol{x}, t; \boldsymbol{\mu})\|_{\boldsymbol{\mu}=\boldsymbol{\mu}^{(i)}}$	$u(\boldsymbol{x}, t; \boldsymbol{\mu})$	Random	Meta-learned	Fixed	Adaptive
PINN	✓		✓		✓	
PINN-R	✓		✓		✓	
PINN-S2S	✓		✓		✓	
Naïve LR-PINN	✓		✓		✓	
PINN-P		✓	✓		✓	
MAML	✓			✓	✓	
Reptile	✓			✓	✓	
HyperPINN	✓			✓	✓	
Hyper-LR-PINN		✓		✓		✓

of distinct β values. The test dataset, which also includes the reference solution is constructed by either analytically or numerically solving the CDR equations.

J Ablation Study

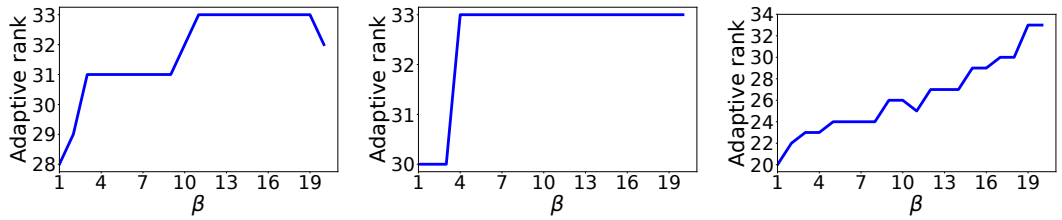
As an ablation study, we check training loss and test MSE with and without the orthogonality penalty Eq. (2) ($w_1 = w_2 = 1$). We observe that when the orthogonality penalty plays an important role in minimizing the train loss and learning the representative basis vectors of the solutions for varying range of parameter values.



(a) Training loss (b) Test MSE (c) Training loss (d) Test MSE

Figure 11: [Convection equations] training loss and test MSE with/without the orthogonality penalty (Eq. (2)) for the $\beta \in [1, 20]$ (a-b), and $\beta \in [1, 40]$ (c-d).

K Adaptive rank: learned rank structure of hidden layers



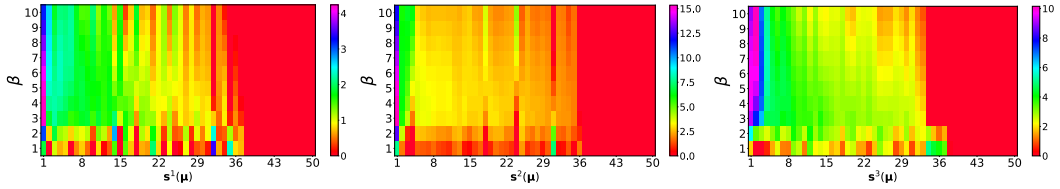
(a) $\boldsymbol{s}^1(\boldsymbol{\mu})$ (b) $\boldsymbol{s}^2(\boldsymbol{\mu})$ (c) $\boldsymbol{s}^3(\boldsymbol{\mu})$

Figure 12: [Convection equation] Learned rank structures of the three hidden layers. $\beta \in [1, 20]$

L Visualization of learned diagonal elements for varying PDE parameters

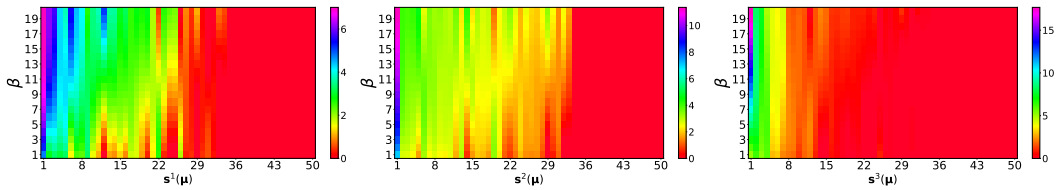
As we shown in the main manuscript, we visualize the magnitude of learned diagonal coefficients in the heatmap format. The results of the convection equations and reaction equations are plotted. One evident observation is that as we vary the PDE parameter in a wider range (e.g., $\beta \in [1, 30]$) (as opposed to a shorter range, e.g., $\beta \in [30, 40]$ or $\rho \in [1, 5]$), the adaptive way of learning

the rank results in more dynamic rank structure and, thus, will be more beneficial in terms of computational/memory efficiency. When the considered model parameters lie in a relative shorter range, the characteristics of the solutions are similar to each other and, thus, it does not require for the network to learn different rank structures (e.g., Figure 16, Figure 17).



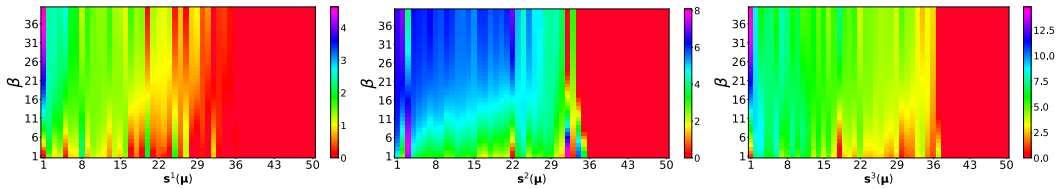
(a) Hypernetwork output: $\boldsymbol{s}^1(\boldsymbol{\mu})$ (b) Hypernetwork output: $\boldsymbol{s}^2(\boldsymbol{\mu})$ (c) Hypernetwork output: $\boldsymbol{s}^3(\boldsymbol{\mu})$

Figure 13: Convection equation ($\beta \in [1, 10]$)



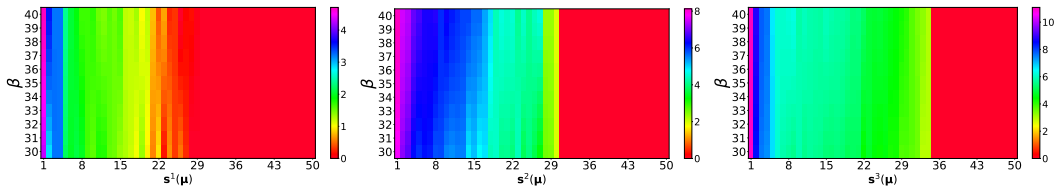
(a) Hypernetwork output: $\boldsymbol{s}^1(\boldsymbol{\mu})$ (b) Hypernetwork output: $\boldsymbol{s}^2(\boldsymbol{\mu})$ (c) Hypernetwork output: $\boldsymbol{s}^3(\boldsymbol{\mu})$

Figure 14: Convection equation ($\beta \in [1, 20]$)



(a) Hypernetwork output: $\boldsymbol{s}^1(\boldsymbol{\mu})$ (b) Hypernetwork output: $\boldsymbol{s}^2(\boldsymbol{\mu})$ (c) Hypernetwork output: $\boldsymbol{s}^3(\boldsymbol{\mu})$

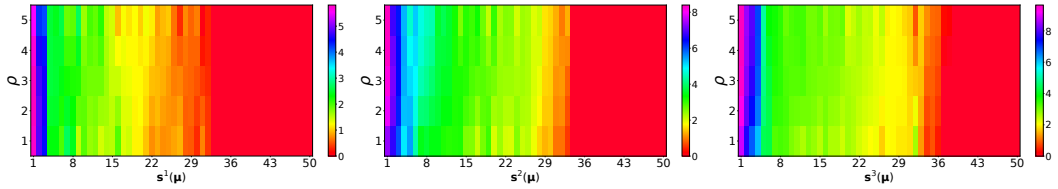
Figure 15: Convection equation ($\beta \in [1, 40]$)



(a) Hypernetwork output: $\boldsymbol{s}^1(\boldsymbol{\mu})$ (b) Hypernetwork output: $\boldsymbol{s}^2(\boldsymbol{\mu})$ (c) Hypernetwork output: $\boldsymbol{s}^3(\boldsymbol{\mu})$

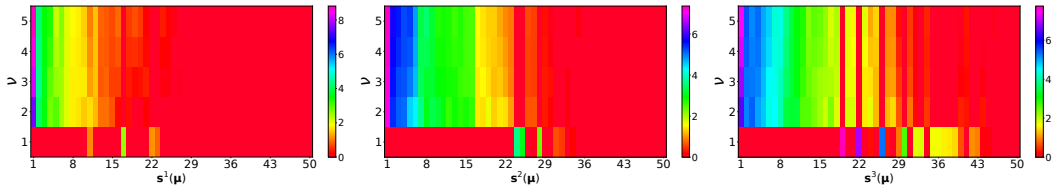
Figure 16: Convection equation ($\beta \in [30, 40]$)

Nevertheless, from the experiments of all considered PDEs, we observe the low-rank structures, which our model captures well and leading to the improved training process even compared to the well-known meta-learning algorithms (MAML and Reptile). Also, as we train our model by showing multiple PDEs describing similar physical phenomena, the proposed model overcomes the failure modes without any special learning approaches (i.e., curriculum or sequence-to-sequence).



(a) Hypernetwork output: $\boldsymbol{s}^1(\boldsymbol{\mu})$ (b) Hypernetwork output: $\boldsymbol{s}^2(\boldsymbol{\mu})$ (c) Hypernetwork output: $\boldsymbol{s}^3(\boldsymbol{\mu})$

Figure 17: Reaction equation ($\rho \in [1, 5]$)



(a) Hypernetwork output: $\boldsymbol{s}^1(\boldsymbol{\mu})$ (b) Hypernetwork output: $\boldsymbol{s}^2(\boldsymbol{\mu})$ (c) Hypernetwork output: $\boldsymbol{s}^3(\boldsymbol{\mu})$

Figure 18: Reaction-diffusion equation ($\nu \in [1, 5], \rho = 5$)

M More experimental results on general cases

To verify the performance of Hyper-LR-PINNs in the general cases, we measure the average performance of baselines and our method in various PDEs with $\beta, \nu, \rho \in [1, 5]$. The initial condition of each equation is Gaussian distribution $N(\pi, (\pi/2)^2)$.

Table 6: The average absolute and relative errors on general cases

PDE-type	Metric	No pre-training				Meta-learning		
		PINN	PINN-R	PINN-P	PINN-S2S	MAML	Reptile	Hyper-LR-PINN
Convection	Abs. err.	0.0183	0.0222	0.0112	0.1281	0.0579	0.0173	0.0038
	Rel. err.	0.0327	0.0381	0.0217	0.2160	0.1036	0.0347	0.0085
Diffusion	Abs. err.	0.1335	0.1665	0.1433	0.1987	0.0803	0.0844	0.1169
	Rel. err.	0.2733	0.3462	0.2920	0.4050	0.1673	0.1742	0.2458
Reaction	Abs. err.	0.3341	0.3336	0.1749	0.4714	0.0029	0.0033	0.0025
	Rel. err.	0.3907	0.3907	0.2024	0.5907	0.0057	0.0064	0.0045
Conv.-Diff.	Abs. err.	0.0610	0.0654	0.0733	0.0979	0.0354	0.0372	0.0331
	Rel. err.	0.1175	0.1289	0.1437	0.1950	0.0667	0.0713	0.0664
Reac.-Diff.	Abs. err.	0.1900	0.1876	0.2201	0.4201	0.0310	0.0250	0.0684
	Rel. err.	0.2702	0.2777	0.3179	0.5346	0.0537	0.0427	0.1168
C-D-R	Abs. err.	0.1676	0.1629	0.1704	0.4878	0.0090	0.0461	0.0201
	Rel. err.	0.2210	0.2149	0.2308	0.5983	0.0144	0.0701	0.0329

N Experimental results on failure modes

In this section, we present the results of additional experiments on the failure mode. We show comparisons with baselines using additional metrics for convection equations, reaction equations, and reaction-diffusion equations, such as maximum error and explained variance. The initial condition of convection equations are $1 + \sin(x)$, and the initial conditions of reaction equations and reaction-diffusion equations are Gaussian distribution $N(\pi, (\pi/4)^2)$. As shown in the following Tables, our Hyper-LR-PINNs overwhelmingly outperform other baselines.

N.1 Convection equation

For experiments on convection equations, meta-learning methods train convection equations in the range $\beta \in [30, 40]$.

Table 7: The absolute and relative errors of the solutions of convection equations with $\beta = \{30, 35, 40\}$

β	PINN		PINN-R		PINN-P		PINN-S2S	
	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.
30	0.4015	0.4033	0.5435	0.5219	0.3821	0.3889	0.6342	0.5831
35	0.4785	0.4701	0.5579	0.5309	0.1701	0.1621	0.6396	0.5868
40	0.5490	0.5219	0.5897	0.5558	0.4988	0.4861	0.7319	0.7300

Table 8: The max error and explained variance score of the solutions of convection equations with $\beta = \{30, 35, 40\}$

β	PINN		PINN-R		PINN-P		PINN-S2S	
	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.
30	1.0471	0.5431	1.1105	0.2453	1.0394	0.5747	1.1719	0.0564
35	1.0685	0.3435	1.0987	0.1707	0.4289	0.9216	1.3323	0.0092
40	1.1018	0.2136	1.1356	0.1175	1.0714	0.3160	1.5677	-0.0155

Table 9: The max error and explained variance score of the solutions of convection equations with $\beta = \{30, 35, 40\}$

β	Rank	[w/o] Pre-training		[w] Pre-training									
		Naïve-LR-PINN		Curriculum learning		MAML		Reptile		Hyper-LR-PINN (Full rank)		Hyper-LR-PINN (Adaptive rank)	
		Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.
30	10	1.1096	0.2067	1.0303	0.5321	1.3550	0.0050	1.1645	0.1445	0.1270	0.9959	0.1229	0.9957
	20	1.0961	0.2322	1.0188	0.5539	1.4006	0.0074	1.1810	0.1008				
	30	1.1901	0.1314	1.0417	0.5084	1.1445	0.1662	1.1780	0.1214				
	40	1.1140	0.2850	0.9798	0.6097	1.1580	0.1542	1.1240	0.1712				
	50	1.1114	0.2709	1.0346	0.5257	1.1670	0.1428	1.1430	0.1592				
35	10	1.1088	0.1592	1.1725	0.1177	1.4062	0.0031	1.1540	0.1260	0.1269	0.9941	0.1390	0.9936
	20	1.1025	0.1546	1.1867	0.0693	1.4412	-0.0047	1.1911	0.0966				
	30	1.1805	0.0784	1.1753	0.1134	1.1683	0.1172	1.1752	0.1125				
	40	1.0959	0.1949	1.1646	0.1084	1.1603	0.1255	1.1480	0.1276				
	50	1.0967	0.1999	1.1883	0.0900	1.1759	0.1119	1.1377	0.1358				
40	10	1.1718	0.1008	1.1671	0.0953	1.4451	0.0007	1.1741	0.0908	0.2437	0.9875	0.2672	0.9848
	20	1.1259	0.1225	1.1781	0.0469	1.4981	-0.0075	1.2091	0.0715				
	30	1.1853	0.0757	1.1859	0.0894	1.1878	0.0787	1.2095	0.0817				
	40	1.1133	0.1953	1.1864	0.0843	1.2176	0.0789	1.1717	0.0885				
	50	1.2450	0.0947	1.1880	0.0730	1.1987	0.0743	1.1791	0.0886				

N.2 Reaction equation

For experiments on reaction equations, meta-learning methods train reaction equations in the range $\rho \in [1, 5]$.

Table 10: The absolute and relative errors of the solutions of reaction equations with $\rho = \{4, 5, 6, 7\}$

ρ	PINN		PINN-R		PINN-P		PINN-S2S	
	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.
4	0.0242	0.0750	0.0260	0.0720	0.0208	0.0614	0.4085	0.8280
5	0.5501	0.9862	0.0334	0.1017	0.4981	0.9349	0.4497	0.8205
6	0.5987	0.9913	0.0403	0.1200	0.5981	0.9906	0.5434	0.9068
7	0.6431	0.9950	0.0275	0.0848	0.6459	0.9976	0.6285	0.9794

Table 11: The absolute and relative errors of the solutions of reaction equations with $\rho = \{4, 5, 6, 7\}$

ρ	Rank	[w/o] Pre-training		[w] Pre-training									
		Naïve-LR-PINN		Curriculum learning		MAML		Reptile		Hyper-LR-PINN (Full rank)		Hyper-LR-PINN (Adaptive rank)	
		Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.
4	10	0.0448	0.1221	0.1287	0.2897	0.1509	0.3396	0.1426	0.3108	0.0126	0.0395	0.0151	0.0460
	20	0.0498	0.1362	0.1333	0.2977	0.1476	0.3305	0.1243	0.3017				
	30	0.0511	0.1494	0.1203	0.2683	0.0949	0.2237	0.0570	0.1534				
	40	0.0425	0.1247	0.1182	0.2629	0.1218	0.2947	0.1076	0.2528				
	50	0.0551	0.1498	0.1195	0.2644	0.0749	0.1913	0.0741	0.1912				
5	10	0.5381	0.9647	0.1349	0.3031	0.5166	0.9273	0.5472	0.9806	0.0157	0.0518	0.0211	0.0655
	20	0.5471	0.9810	0.1372	0.3049	0.5489	0.9816	0.5483	0.9845				
	30	0.5385	0.9664	0.1209	0.2724	0.5378	0.9662	0.5454	0.9778				
	40	0.5413	0.9696	0.1208	0.2701	0.5321	0.9538	0.5450	0.9785				
	50	0.5475	0.9822	0.1186	0.2648	0.5478	0.9815	0.5536	0.9906				
6	10	0.5987	0.9912	0.1355	0.2957	0.5926	0.9806	0.5969	0.9882	0.0167	0.0552	0.0183	0.0572
	20	0.5981	0.9905	0.1303	0.2822	0.5988	0.9906	0.5989	0.9919				
	30	0.5959	0.9863	0.1306	0.2854	0.5947	0.9853	0.5992	0.9931				
	40	0.5988	0.9917	0.1196	0.2613	0.5951	0.9854	0.6001	0.9936				
	50	0.5989	0.9919	0.1194	0.2586	0.6004	0.9930	0.6022	0.9970				
7	10	0.6431	0.9949	0.1448	0.3011	0.6394	0.9890	0.6413	0.9922	0.0180	0.0575	0.0203	0.0456
	20	0.6425	0.9942	0.1363	0.2837	0.6410	0.9914	0.6423	0.9941				
	30	0.6424	0.9938	0.1285	0.2764	0.6401	0.9903	0.6431	0.9955				
	40	0.6430	0.9949	0.1145	0.2445	0.6401	0.9901	0.6436	0.9958				
	50	0.6428	0.9948	0.1140	0.2418	0.6417	0.9927	0.6437	0.9962				

Table 12: The max error and explained variance score of the solutions of reaction equations with $\rho = \{4, 5, 6, 7\}$

ρ	PINN		PINN-R		PINN-P		PINN-S2S	
	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.
4	0.2282	0.9881	0.2055	0.9881	0.1681	0.9921	0.9895	0.0514
5	0.9961	0.0288	0.3297	0.9745	1.0077	-0.0375	0.9746	-0.0890
6	0.9982	0.0163	0.4422	0.9596	0.9983	0.0158	1.0877	-0.1455
7	0.9996	0.0088	0.4038	0.9750	1.0042	0.0139	1.0267	-0.0224

Table 13: The max error and explained variance score of the solutions of reaction equations with $\rho = \{4, 5, 6, 7\}$

ρ	Rank	[w/o] Pre-training		[w] Pre-training									
		Naïve-LR-PINN		Curriculum learning		MAML		Reptile		Hyper-LR-PINN (Full rank)		Hyper-LR-PINN (Adaptive rank)	
		Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.
4	10	0.3395	0.9699	0.6069	0.8453	0.6653	0.8064	0.5896	0.8443	0.1356	0.9964	0.1524	0.9953
	20	0.3618	0.9649	0.5921	0.8352	0.6820	0.8162	0.6349	0.8420				
	30	0.4223	0.9556	0.5635	0.8770	0.5249	0.9131	0.3748	0.9559				
	40	0.3601	0.9680	0.5438	0.8825	0.6538	0.8423	0.5659	0.8874				
	50	0.3900	0.9576	0.5507	0.8805	0.4316	0.9350	0.4767	0.9320				
5	10	0.9828	0.0690	0.6997	0.8061	0.9639	0.1355	0.9951	0.0418	0.1967	0.9929	0.2287	0.9891
	20	0.9909	0.0371	0.6969	0.8030	0.9910	0.0482	0.9951	0.0261				
	30	0.9850	0.0626	0.6576	0.8472	0.9883	0.0591	0.9865	0.0452				
	40	0.9824	0.0640	0.6464	0.8525	0.9776	0.0917	0.9935	0.0377				
	50	0.9899	0.0333	0.6405	0.8568	0.9928	0.0399	0.9980	0.0279				
6	10	1.0073	0.0163	0.7566	0.7896	0.9942	0.0402	0.9989	0.0223	0.2405	0.9905	0.2418	0.9899
	20	0.9986	0.0165	0.7293	0.8042	0.9968	0.0223	1.0113	0.0136				
	30	0.9967	0.0272	0.7319	0.8111	0.9985	0.0253	1.0118	0.0081				
	40	1.0089	0.0135	0.6944	0.8407	0.9965	0.0278	1.0064	0.0118				
	50	1.0071	0.0131	0.6918	0.8448	1.0089	0.0178	1.0231	0.0049				
7	10	1.0115	0.0084	0.7502	0.7513	0.9991	0.0225	1.0014	0.0142	0.2922	0.9882	0.2150	0.9920
	20	1.0048	0.0092	0.7136	0.7820	0.9996	0.0180	1.0064	0.0080				
	30	1.0057	0.0116	0.7468	0.7961	1.0001	0.0183	1.0126	0.0044				
	40	1.0110	0.0075	0.6827	0.8394	0.9994	0.0193	1.0061	0.0071				
	50	1.0090	0.0077	0.6765	0.8443	1.0008	0.0137	1.0142	0.0046				

N.3 Reaction-diffusion equation

For experiments on reaction-diffusion equations, meta-learning methods train reaction equations in the range $\nu \in [1, 5], \rho = 5$. To learn reaction-diffusion equations, PINN baselines employ 2000 epochs, but meta-learning methods train only 10 epochs, after pre-training.

Table 14: The absolute and relative errors of the solutions of reaction-diffusion equations with $\nu = \{4, 5, 6, 7\}, \rho = 5$

(ν, ρ)	PINN		PINN-R		PINN-P		PINN-S2S	
	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.
(4, 5)	0.2909	0.4902	0.2909	0.4914	0.2804	0.4744	0.7265	0.9950
(5, 5)	0.3186	0.5134	0.3364	0.5400	0.3380	0.5456	0.7383	1.0037
(6, 5)	0.3183	0.4891	0.3810	0.5884	0.4255	0.6648	0.7265	0.9881
(7, 5)	0.4860	0.7273	0.4126	0.6273	0.5229	0.7925	0.6630	0.9213

Table 15: The absolute and relative errors of the solutions of reaction-diffusion equations with $\nu = \{4, 5, 6, 7\}, \rho = 5$

(ν, ρ)	Rank	[w/o] Pre-training		[w] Pre-training									
		Naive-LR-PINN		Curriculum learning		MAML		Reptile		Hyper-LR-PINN (Full rank)		Hyper-LR-PINN (Adaptive rank)	
		Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.
(4, 5)	10	0.0333	0.0631	0.6862	0.9744	0.0522	0.0873	0.7284	0.9882	0.0439	0.0850	0.0440	0.0851
	20	0.0470	0.0791	0.6633	1.0030	0.6976	0.9408	0.7443	1.0108				
	30	0.0175	0.0312	0.6628	0.9630	0.0481	0.0817	0.7088	0.9597				
	40	0.0658	0.1128	0.6667	0.9709	0.0516	0.0874	0.7428	1.0068				
	50	0.0794	0.1331	0.6693	0.9718	0.0458	0.0767	0.7269	0.9856				
(5, 5)	10	0.1036	0.1688	0.7156	0.9985	0.0604	0.0998	0.7407	0.9964	0.0518	0.0934	0.0519	0.0936
	20	0.1001	0.1633	0.7084	1.0030	0.7106	0.9511	0.7567	1.0190				
	30	0.0731	0.1161	0.7079	0.9928	0.0824	0.1258	0.7198	0.9667				
	40	0.0946	0.1569	0.7073	0.9968	0.0678	0.1116	0.7563	1.0164				
	50	0.0875	0.1416	0.7126	1.0036	0.0733	0.1136	0.7390	0.9935				
(6, 5)	10	0.1009	0.1593	0.6936	0.9542	0.1100	0.1626	0.7512	1.0045	0.0665	0.1106	0.0665	0.1104
	20	0.1024	0.1616	0.6370	0.8897	0.7125	0.9499	0.7669	1.0265				
	30	0.0938	0.1454	0.6962	0.9589	0.1354	0.1903	0.7291	0.9735				
	40	0.0983	0.1561	0.6470	0.8983	0.1145	0.1707	0.7672	1.0250				
	50	0.1125	0.1804	0.7022	0.9694	0.1298	0.1822	0.7492	1.0014				
(7, 5)	10	0.0926	0.1430	0.2754	0.4140	0.1676	0.2312	0.7607	1.0127	0.0796	0.1262	0.0800	0.1269
	20	0.1158	0.1785	0.2935	0.4473	0.7135	0.9485	0.7760	1.0341				
	30	0.0907	0.1381	0.2463	0.3696	0.1679	0.2306	0.7379	0.9811				
	40	0.1065	0.1666	0.2583	0.3926	0.1571	0.2218	0.7763	1.0327				
	50	0.1417	0.2231	0.2560	0.3837	0.1674	0.2287	0.7583	1.0090				

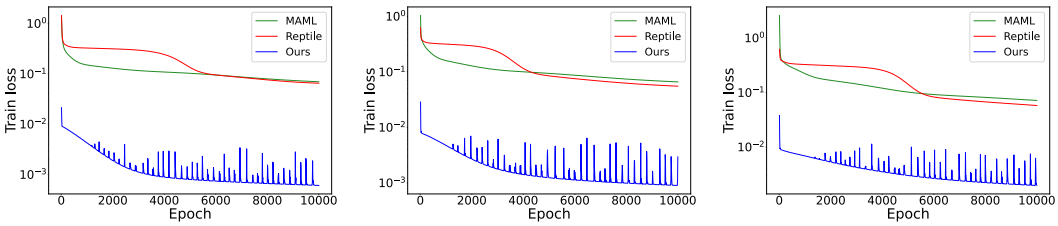
Table 16: The max error and explained variance score of the solutions of reaction-diffusion equations with $\nu = \{4, 5, 6, 7\}, \rho = 5$

(ν, ρ)	PINN		PINN-R		PINN-P		PINN-S2S	
	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.
(4, 5)	0.9624	0.0038	0.9697	-0.0073	0.9400	0.0542	1.0706	-0.0136
(5, 5)	0.9452	-0.0278	0.9782	-0.1213	1.0000	-0.1643	1.1295	-0.0634
(6, 5)	0.8441	0.8441	0.9931	-0.2086	1.1412	-0.6138	1.1584	-0.1617
(7, 5)	1.1444	-0.6415	1.0150	-0.348	-0.6138	-0.6138	1.2363	-0.4552

Table 17: The max error and explained variance score of the solutions of reaction-diffusion equations with $\nu = \{4, 5, 6, 7\}, \rho = 5$

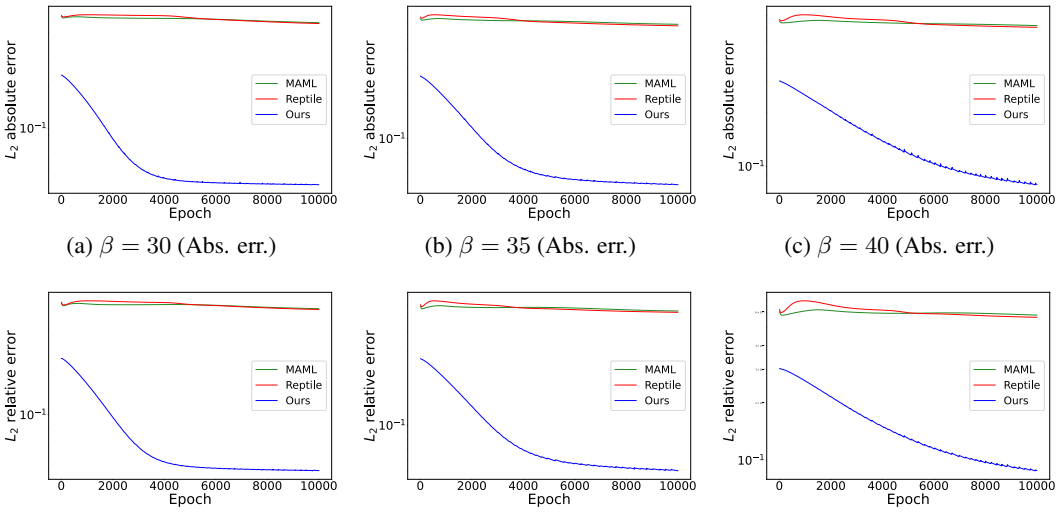
(ν, ρ)	Rank	[w/o] Pre-training		[w] Pre-training									
		Naïve-LR-PINN		Curriculum learning		MAML		Reptile		Hyper-LR-PINN (Full rank)		Hyper-LR-PINN (Adaptive rank)	
		Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.
(4, 5)	10	0.1859	0.9745	1.3388	-0.5457	0.2907	0.9450	1.0065	0.1604	0.2189	0.9588	0.2189	0.9588
	20	0.1967	0.9719	1.4119	-0.9933	1.0161	0.3333	1.0145	0.1023				
	30	0.1668	0.9934	1.3659	-0.8454	0.3037	0.9530	1.0199	0.2398				
	40	0.2753	0.9439	1.3921	-0.9100	0.2747	0.9402	1.0225	0.1462				
	50	0.3133	0.9265	1.3897	-0.8704	0.3118	0.9613	1.0200	0.1754				
(5, 5)	10	0.3998	0.8848	1.2749	-0.5216	0.2794	0.9345	1.0176	0.1469	0.2317	0.9523	0.2317	0.9523
	20	0.3778	0.8922	1.3262	-0.7929	1.0325	0.3102	1.0250	0.0883				
	30	0.2498	0.9470	1.2717	-0.5948	0.3300	0.9435	1.0310	0.2275				
	40	0.3739	0.8953	1.2902	-0.6908	0.2792	0.9250	1.0352	0.1322				
	50	0.3216	0.9213	1.3184	-0.7047	0.3214	0.9503	1.0314	0.1628				
(6, 5)	10	0.3527	0.9007	1.1203	-0.2824	0.3264	0.9240	1.0264	0.1345	0.2570	0.9430	0.2578	0.9429
	20	0.3580	0.8992	1.1396	-0.3386	1.0354	0.2814	1.0345	0.0753				
	30	0.3095	0.9235	1.1204	-0.3159	0.3742	0.9327	1.0397	0.2153				
	40	0.3493	0.9037	1.1130	-0.2780	0.3305	0.9126	1.0457	0.1182				
	50	0.4116	0.8689	1.1388	-0.3871	0.3679	0.9397	1.0407	0.1501				
(7, 5)	10	0.3048	0.9225	0.7000	0.4155	0.3745	0.9149	1.0334	0.1238	0.2827	0.9284	0.2833	0.9286
	20	0.3802	0.8820	0.7686	0.2740	1.0365	0.2589	1.0418	0.0635				
	30	0.2823	0.9318	0.6453	0.5257	0.3956	0.9202	1.0467	0.2049				
	40	0.3676	0.8915	0.6842	0.4392	0.3789	0.8998	1.0537	0.1066				
	50	0.5088	0.8037	0.6560	0.4977	0.3944	0.9276	1.0482	0.1398				

O Loss curves of meta-learning methods in Phase 2



(a) $\beta = 30$ (b) $\beta = 35$ (c) $\beta = 40$

Figure 19: Training loss curve in phase 2. We follow the experimental setting of Figure 5. $\beta = \{30, 35, 40\}$

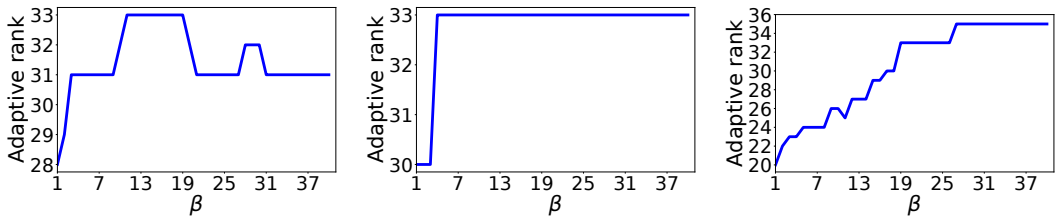


(a) $\beta = 30$ (Abs. err.) (b) $\beta = 35$ (Abs. err.) (c) $\beta = 40$ (Abs. err.)

(d) $\beta = 30$ (Rel. err.) (e) $\beta = 35$ (Rel. err.) (f) $\beta = 40$ (Rel. err.)

Figure 20: Curves of L_2 absolute error (top row) and L_2 relative error (bottom row) in phase2. We follow the experimental setting of Figure 5. $\beta = \{30, 35, 40\}$

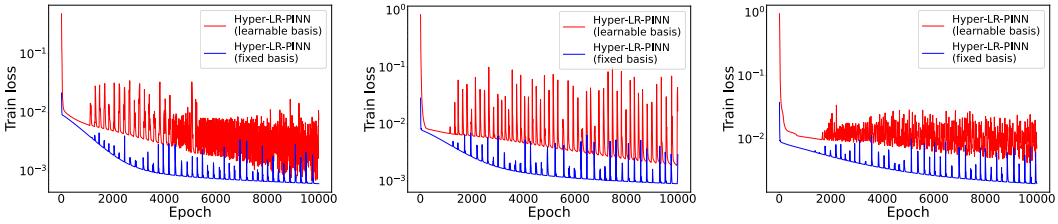
P Adaptive rank: β outside of the range in training phase 1



(a) $\boldsymbol{s}^1(\boldsymbol{\mu})$ (b) $\boldsymbol{s}^2(\boldsymbol{\mu})$ (c) $\boldsymbol{s}^3(\boldsymbol{\mu})$

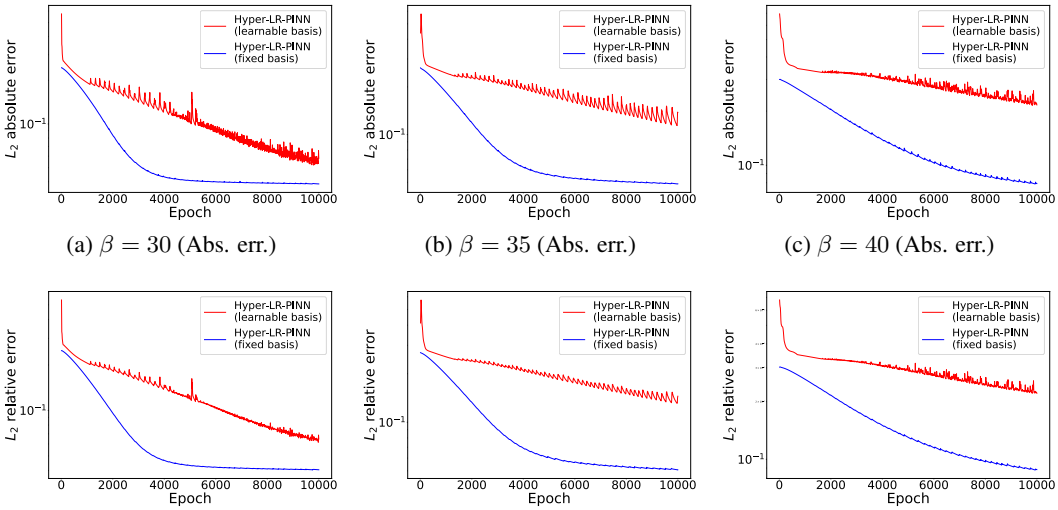
Figure 21: [Convection equation] Learned rank structures of the three hidden layers. $\beta \in [1, 20]$ are used in phase 1.

Q Comparison of learnable basis and fixed basis on Hyper-LR-PINN



(a) $\beta = 30$ (b) $\beta = 35$ (c) $\beta = 40$

Figure 22: Training loss curve on convection equations in phase 2. $\beta = \{30, 35, 40\}$



(a) $\beta = 30$ (Abs. err.) (b) $\beta = 35$ (Abs. err.) (c) $\beta = 40$ (Abs. err.)

(d) $\beta = 30$ (Rel. err.) (e) $\beta = 35$ (Rel. err.) (f) $\beta = 40$ (Rel. err.)

Figure 23: Curves of L_2 absolute error (top row) and L_2 relative error (bottom row) on convection equations in phase 2. $\beta = \{30, 35, 40\}$

Table 18: Experimental results of learnable basis and fixed basis. $\beta = \{30, 35, 40\}$

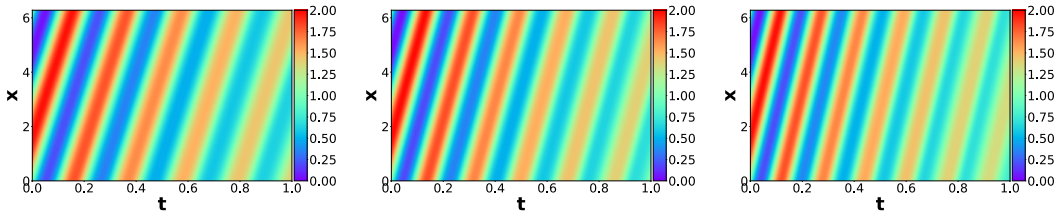
β	Learnable basis		Fixed basis	
	Abs. err.	Rel. err.	Abs. err.	Rel. err.
30	0.0567	0.0656	0.0386	0.0405
35	0.1344	0.1454	0.0490	0.0509
40	0.2169	0.2208	0.0790	0.0883

Table 19: Number of learnable parameters in Phase2

	$\beta = 30$	$\beta = 35$	$\beta = 40$
Learnable basis	9,392	9,594	9,594
Fixed basis	292	294	293

To compare fixed basis and learnable basis, we provide training loss curves in Figure 22 and curves of absolute error (Abs.err.) and relative error (Rel.err.) in Figure 23. In all cases, fixed basis exhibits more stable learning and superior performance compared to learnable basis. Additionally, as can be seen in Table 19, fixed basis is also significantly more efficient with a model size over 30 times smaller.

R Visualization of the results in phase 1 and phase 2



(a) $\beta = 30$ (phase 1) (b) $\beta = 35$ (phase 1) (c) $\beta = 40$ (phase 1)

(d) $\beta = 30$ (phase 2) (e) $\beta = 35$ (phase 2) (f) $\beta = 40$ (phase 2)

Figure 24: The figures in the top row are the results of phase 1, and the figures in the bottom row are the results of phase 2. We follow the experimental setting of Table 2. $\beta = \{30, 35, 40\}$

S Other experiments

S.1 Model size study for PINNs

In Table 20, we report the performance by the model size, i.e., the number of model parameters.

Table 20: PINNs performance by varying the model size. The numbers inside parentheses are model parameter numbers.

β	dim=50 (10,401)		dim=40 (6,721)		dim=30 (3,841)		dim=20 (1,761)		dim=10 (481)		dim=8 (321)	
	Abs.err.	Rel.err.	Abs.err.	Rel.err.	Abs.err.	Rel.err.	Abs.err.	Rel.err.	Abs.err.	Rel.err.	Abs.err.	Rel.err.
30	0.4015	0.4033	0.4077	0.4114	0.4576	0.4524	0.4365	0.4369	0.4398	0.4409	0.4479	0.4478
35	0.4785	0.4701	0.4815	0.4736	0.5132	0.4971	0.4491	0.4464	0.4823	0.4763	0.5023	0.4903
40	0.5490	0.5219	0.5599	0.5333	0.5792	0.5467	0.5302	0.5097	0.5088	0.4966	0.5410	0.5207

Table 21: Comparison of full rank and adaptive rank in Phase2

β	**Phase1**		**Phase2 (Full rank)**		**Phase2 (Adaptive rank)**	
	Abs.err.	Rel.err.	Abs.err.	Rel.err.	Abs.err.	Rel.err.
30	0.2294	0.2346	0.0360	0.0379	0.0375	0.0389
35	0.2443	0.2534	0.0428	0.0443	0.0448	0.0461
40	0.2443	0.2534	0.0603	0.0655	0.0656	0.0722

S.2 Performance comparisons on full-rank and adaptive-rank in Hyper-LR-PINNs

In Table 21, we compare the cases where we fix weights to be of full-rank and where we make them adaptive (i.e., some singular values are truncated by ReLU). The adaptive rank approach still achieve the comparable performance.

S.3 Performance comparisons against Hyper-PINNs

Table 22: Comparison of HyperPINN and Hyper-LR-PINN

β	**HyperPINN**		**Hyper-LR-PINN**	
	Abs.err.	Rel.err.	Abs.err.	Rel.err.
30	0.9491	0.9856	0.0375	0.0389
35	0.9639	0.9831	0.0448	0.0461
40	0.9773	0.9989	0.0656	0.0722

We provide the results of the comparison between HyperPINN and Hyper-LR-PINN (Adaptive rank) in Table 22. In all cases, Hyper-LR-PINN demonstrates overwhelmingly superior performance. All experimental settings are the same as in Table 2.

S.4 Performance on extrapolation in the PDE parameter domain

Table 23: Detailed results of the experiment in Figure 5

	$\beta = 27$	$\beta = 28$	$\beta = 29$	$\beta = 41$	$\beta = 42$	$\beta = 43$
Abs. err.	0.0311	0.0351	0.0358	0.1002	0.1089	0.1490
Rel. err.	0.0318	0.0366	0.0377	0.1174	0.1238	0.1624

T 2D-Helmholtz equation

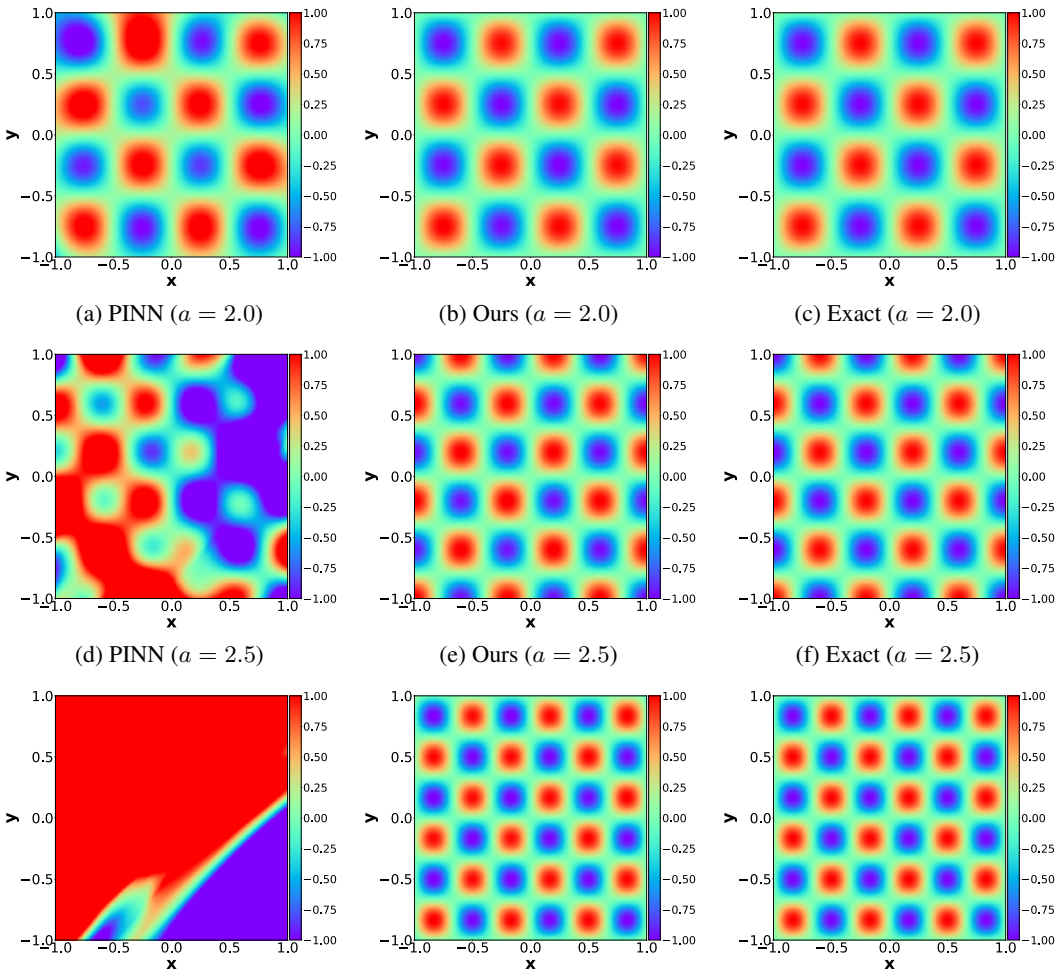
We present the results of applying Hyper-LR-PINN to compute solutions of parameterized 2D Helmholtz equations:

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} + k^2 u(x,y) - q(x,y) = 0,$$
$$q(x,y) = (-(a_1\pi)^2 - (a_2\pi)^2 + k^2)\sin(a_1\pi x)\sin(a_2\pi y),$$

where the forcing term q is chosen such that the analytical expression for the solutions are available:

$$u(x,y) = k^2 \sin(a_1\pi x)\sin(a_2\pi y).$$

Figure 25 shows the solution snapshots of 2D-Helmholtz equations for varying parameters and demonstrates that the proposed approach outperforms the vanilla PINNs.



(a) PINN ($a = 2.0$) (b) Ours ($a = 2.0$) (c) Exact ($a = 2.0$)

(d) PINN ($a = 2.5$) (e) Ours ($a = 2.5$) (f) Exact ($a = 2.5$)

(g) PINN ($a = 3.0$) (h) Ours ($a = 3.0$) (i) Exact ($a = 3.0$)

Figure 25: [2D-Helmholtz equation] Solution snapshots

The comparison results of the absolute and relative errors between PINN and Hyper-LR-PINN can be observed in Table 24. In all cases, Hyper-LR-PINN consistently exhibits significantly smaller errors compared to PINN.

Table 24: [2D-Helmholtz equation] The absolute and relative errors of the solutions of 2D-Helmholtz equations with $a = \{2.0, 2.5, 3.0\}$

	$a = 2.0$		$a = 2.5$		$a = 3.0$	
	Abs.err.	Rel.err.	Abs.err.	Rel.err.	Abs.err.	Rel.err.
PINN	0.1698±0.0553	0.4258±0.0939	0.7403±0.1050	1.8232±0.2464	2.0537±0.4904	4.8401±0.7366
Hyper-LR-PINN	0.0137±0.0021	0.0330±0.0032	0.0285±0.0029	0.0611±0.0071	0.0153±0.0028	0.0389±0.0060

U Standard deviation

Table 2 presents the mean error of the experimental results and additionally, we provide the standard deviation for the evaluation metrics in Tables 25 and 26.

Table 25: Standard deviation of the evaluation metrics (Abs. err. and Rel. err.) $\beta = \{30, 35, 40\}$

β	Rank	[w/o] Pre-training		[w] Pre-training									
		Naïve-LR-PINN		Curriculum learning		MAML		Reptile		Hyper-LR-PINN (Full rank)		Hyper-LR-PINN (Adaptive rank)	
		Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.	Abs. err.	Rel. err.
30	10	±0.0526	±0.0397	±0.0000	±0.0000	±0.0172	±0.0247	±0.0521	±0.0474	±0.0023	±0.0023	±0.0025	±0.0014
	20	±0.0374	±0.0300	±0.0000	±0.0000	±0.0164	±0.0243	±0.0260	±0.0217				
	30	±0.0522	±0.0423	±0.0000	±0.0000	±0.0508	±0.0446	±0.0135	±0.0125				
	40	±0.0446	±0.0339	±0.0015	±0.0005	±0.0780	±0.0870	±0.0024	±0.0021				
	50	±0.0385	±0.0298	±0.0001	±0.0003	±0.0610	±0.0647	±0.0055	±0.0047				
35	10	±0.0484	±0.0382	±0.0016	±0.0006	±0.0243	±0.0353	±0.0545	±0.0498	±0.0029	±0.0028	±0.0025	±0.0027
	20	±0.0254	±0.0201	±0.0000	±0.0000	±0.0267	±0.0414	±0.0218	±0.0163				
	30	±0.0143	±0.0151	±0.0000	±0.0000	±0.0483	±0.0437	±0.0069	±0.0051				
	40	±0.0303	±0.0224	±0.0001	±0.0002	±0.0788	±0.0942	±0.0016	±0.0012				
	50	±0.0307	±0.0228	±0.0001	±0.0000	±0.0734	±0.0848	±0.0041	±0.0028				
40	10	±0.0331	±0.0273	±0.0014	±0.0005	±0.0418	±0.0648	±0.0506	±0.0537	±0.0038	±0.0043	±0.0043	±0.0042
	20	±0.0215	±0.0188	±0.0000	±0.0000	±0.0410	±0.0589	±0.0233	±0.0214				
	30	±0.0107	±0.0116	±0.0000	±0.0000	±0.0468	±0.0510	±0.0091	±0.0109				
	40	±0.0363	±0.0305	±0.0001	±0.0001	±0.0703	±0.0899	±0.0005	±0.0009				
	50	±0.0065	±0.0080	±0.0004	±0.0005	±0.0800	±0.1027	±0.0086	±0.0071				

Table 26: Standard deviation of the evaluation metrics (Max. err. and Exp. var.) $\beta = \{30, 35, 40\}$

β	Rank	[w/o] Pre-training		[w] Pre-training									
		Naïve-LR-PINN		Curriculum learning		MAML		Reptile		Hyper-LR-PINN (Full rank)		Hyper-LR-PINN (Adaptive rank)	
		Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.	Max. err.	Exp. var.
30	10	±0.0629	±0.1001	±0.0000	±0.0000	±0.0802	±0.0126	±0.1296	±0.0803	±0.0025	±0.0015	±0.0009	±0.0015
	20	±0.0525	±0.0760	±0.0000	±0.0000	±0.0572	±0.0089	±0.0349	±0.0512				
	30	±0.0198	±0.0332	±0.0000	±0.0000	±0.1220	±0.0842	±0.0180	±0.0222				
	40	±0.0362	±0.0943	±0.0004	±0.0026	±0.1935	±0.0999	±0.0078	±0.0056				
	50	±0.0402	±0.0809	±0.0017	±0.0004	±0.1697	±0.0815	±0.0223	±0.0137				
35	10	±0.1273	±0.0817	±0.0023	±0.0019	±0.0571	±0.0082	±0.1590	±0.0742	±0.0024	±0.0019	±0.0057	±0.0018
	20	±0.0640	±0.0463	±0.0000	±0.0000	±0.0879	±0.0135	±0.0349	±0.0435				
	30	±0.0771	±0.0219	±0.0000	±0.0000	±0.1478	±0.0686	±0.0192	±0.0115				
	40	±0.0511	±0.0603	±0.0014	±0.0002	±0.2414	±0.0882	±0.0036	±0.0026				
	50	±0.0556	±0.0602	±0.0001	±0.0001	±0.2254	±0.0763	±0.0152	±0.0063				
40	10	±0.0672	±0.0446	±0.0010	±0.0019	±0.1190	±0.0165	±0.1741	±0.0574	±0.0106	±0.0011	±0.0105	±0.0004
	20	±0.0645	±0.0381	±0.0000	±0.0000	±0.1143	±0.0205	±0.0456	±0.0578				
	30	±0.0365	±0.0143	±0.0000	±0.0000	±0.1748	±0.0564	±0.0269	±0.0093				
	40	±0.0539	±0.0744	±0.0016	±0.0001	±0.2441	±0.0668	±0.0047	±0.0006				
	50	±0.0249	±0.0156	±0.0025	±0.0001	±0.2586	±0.0626	±0.0211	±0.0133				