PINNACLE: PINN Adaptive ColLocation & Experimental points selection

Gregory Kang Ruey Lau*[†][‡] GREGLAU@COMP.NUS.EDU
Apivich Hemachandra*[†] APIVICH@COMP.NUS.EDU
See-Kiong Ng[†] SEEKIONG@NUS.EDU.SG
Bryan Kian Hsiang Low[†] LOWKH@COMP.NUS.EDU

** Equal contribution*
[†] *Department of Computer Science, National University of Singapore, Singapore 117417*
[‡] *CNRS@CREATE, 1 Create Way, #08-01 Create Tower, Singapore 138602*

Abstract

Physics-Informed Neural Networks (PINNs), which incorporate PDEs as soft constraints, train with a composite loss function that contains multiple training point types: different types of *collocation points* chosen during training to enforce each PDE and initial/boundary conditions, and *experimental points* which are usually costly to obtain via experiments or simulations. Training PINNs using this loss function is challenging as it typically requires selecting large numbers of points of different types, each with different training dynamics. Unlike past works that focused on the selection of either collocation or experimental points, this work introduces PINN ADAPTIVE COLLOCATION AND EXPERIMENTAL POINTS SELECTION (PINNACLE), the first algorithm that *jointly optimizes the selection of all training point types, while automatically adjusting the proportion of collocation point types as training progresses.* PINNACLE uses information on the interaction among training point types, which had not been considered before, based on an analysis of PINN training dynamics via the Neural Tangent Kernel (NTK). We theoretically show that the criterion used by PINNACLE is related to the PINN generalization error, and empirically demonstrate that PINNACLE is able to outperform existing point selection methods for forward, inverse, and transfer learning problems.

Keywords: Physics-informed neural networks, Adaptive training points selection

1. Introduction

Deep learning (DL) successes in domains with massive datasets have led to questions on whether it can also be efficiently applied to the scientific domains where training data may be more limited. In these settings, domain knowledge such as partial differential equations (PDEs) could be used as inductive biases for DL training. Physics-informed neural networks (PINNs) are neural networks that incorporate PDEs and their initial/boundary conditions (IC/BCs) as soft constraints during training (Raissi et al., 2019), and have been applied to various problems (Cai et al., 2021; Rodríguez et al., 2023; Jagtap et al., 2022).

However, the training of PINNs is challenging. PINNs need a composite loss function that requires multiple training point types: different types of *collocation points* (CL points) chosen during training to enforce each PDE and IC/BCs, and *experimental points* (EXP points) obtained via experiments or simulations that are queries for ground truth solution values. These points have different training dynamics, making training difficult especially in problems with complex structure. In practice, getting accurate results from PINNs also require large numbers of CL and Exp points, both which lead to high costs – the former leads to high computational costs during training (Cho et al., 2023; Chiu et al., 2022), while the latter is generally costly to acquire experimentally or simulate. Past works have tried to separately address these individually by considering an adaptive selection of CL points (Wu et al., 2023; Peng et al., 2022; Tang et al., 2023), or Exp points selection through traditional active learning methods (Jiang et al., 2022; Cuomo et al., 2022). Some works have also proposed heuristics that adjust loss term weights of the various point types to try improve training dynamics, but do not consider point selection (Wang et al., 2022c).

In this paper, we introduce the algorithm PINN Adaptive ColLocation and Experimental data selection (PINNACLE) that is the first to jointly optimize the selection of all types of training points (e.g. PDE and IC/BC CL and Exp points) given a budget, *making use of cross information among the various types of points* that past works had not considered to provide significant PINNs training improvements.

2. Proposed Method

In this section, we first present the problem of joint point selection for improving PINN training, and a novel representation for the augmented input space for PINNs. We then analyze PINNs training dyanmics using the PINNs NTK eigenspectrum in this space, propose a new criterion related to PINN convergence and generalization bound, and introduce our novel point selection algorithm, PINNACLE.

Point Selection Problem Setup. Given a PDE $\mathcal{N}[u,\beta](x) = f(x)$ for $x \in \mathcal{X}$ with IC/BC $\mathcal{B}[u](x') = g(x')$ for $x' \in \partial\mathcal{X}$, we train a PINN $\hat{u}_\theta$ to model the true solution u by minimizing the PINN composite loss function (see (6) in Appendix for details), which comprises of separate loss terms corresponding to Exp ($X_s \subset \mathcal{X}$), PDE CL ($X_p \subset \mathcal{X}$), and multiple IC/BC CL ($X_b \subset \partial\mathcal{X}$) points[1]. The task involves selecting training sets X_s, X_p, and X_b given a fixed training budget, to achieve the best PINN performance. Due to high acquisition cost we consider a fixed Exp training budget $|X_s|$. We also consider a combined training budget for the various CL point types $|X_p| + |X_b| = k$, which is limited due to computational cost during training. Note that the algorithm is allowed to freely allocate the CL budget among the PDE and various IC/BC point types during training, unlike other PINNs algorithms where the user needs to manually fix the number of training points for each type. Also, while the Exp and CL points do not share a common budget, they can still be jointly optimized for better performance, i.e., Exp points selection could still benefit from information from CL points and vice versa.

Augmented Input Representation for PINNs. To analyze the interplay among the training dynamics of different point types, we define a new augmented space $\mathcal{Z} \subset \mathcal{X} \times \{\mathrm{s},\mathrm{p},\mathrm{b}\}$, containing training points of all types, as $\mathcal{Z} = \big\{(x,\mathrm{s}) : x \in \mathcal{X}\big\} \cup \big\{(x,\mathrm{p}) : x \in$

1. To compare with the active learning problem setting, Exp points can be viewed as queries to the ground truth oracle, and CL points as queries to oracles that advise whether $\hat{u}_\theta$ satisfies the PDE and IC/BC constraints. See Appendix C.1 for background on PINNs.

$\mathcal{X}\} \cup \{(x, \mathrm{b}) : x \in \partial\mathcal{X}\}$, where s, p and b are the indicators for the Exp points, PDE CL points and IC/BC CL points respectively, to specify which type of training point z is associated with[2]. We abuse notation by defining any function $h : \mathcal{X} \to \mathbb{R}$ to also be defined on $\mathcal{Z}$ by $h(z) = h(x, \mathrm{r}) \triangleq h(x)$ for all indicators $\mathrm{r} \in \{\mathrm{s}, \mathrm{p}, \mathrm{b}\}$, and also define a general prediction operator F that applies the appropriate operator depending on the indicator r, i.e., $F[h](x, \mathrm{s}) = h(x)$, $F[h](x, \mathrm{p}) = \mathcal{N}[h](x)$, and $F[h](x, \mathrm{b}) = \mathcal{B}[h](x)$. The residual of the network output at any training point z can then be expressed as $R_{\theta_t}(z) = F[\hat{u}_{\theta_t}](z) - F[u](z)$, and we can express the training loss on dataset Z as

$$\mathcal{L}(\hat{u}_\theta; Z) = \tfrac{1}{2}\textstyle\sum_{z \in Z} R_{\theta_t}(z)^2 = \tfrac{1}{2}\textstyle\sum_{z \in Z} \big(F[\hat{u}_\theta](z) - F[u](z)\big)^2 . \tag{1}$$

NTK Eigenbasis and Training Dynamics Intuition. With this PINN augmented space, we can now consider different basis that naturally encode cross information on the interactions among the various point types during training. The eNTK in the augmented space[3] is $\Theta_t(z, z') = \nabla_\theta F[\hat{u}_{\theta_t}](z)\ \nabla_\theta F[\hat{u}_{\theta_t}](z')^\top$. Since Θ_t is a continuous, positive semi-definite kernel, by Mercer's theorem (Schölkopf and Smola, 2002) we can decompose Θ_t to its eigenfunctions $\psi_{t,i}$ and corresponding eigenvalues $\lambda_{t,i} \geq 0$ (which are formally defined in Appendix C.4) as $\Theta_t(z, z') = \sum_{i=1}^{\infty} \lambda_{t,i}\ \psi_{t,i}(z)\psi_{t,i}(z')$. Under this representation, the eigenfunctions will naturally already capture the cross information between each type of training point. In Appendix D, we provide insights on the eNTK eigenfunctions as basis in the augmented space, and demonstrate that two concurrent processes occur during training: 1) the residual will decay fastest along the direction of the dominant eigenfunctions, and 2) the eigenfunctions will evolve such that they can better reconstruct the true solution.

Convergence Degree Criterion. With the eNTK eigenbasis, the change in residual at any point $z \in \mathcal{Z}$, given training set Z for a GD step with small enough η, is

$$\Delta R_{\theta_t}(z; Z) \triangleq R_{\theta_{t+1}}(z; Z) - R_{\theta_t}(z; Z) \approx -\eta \textstyle\sum_{i=1}^{\infty} \lambda_{t,i} \underbrace{\langle \psi_{t,i}(Z), R_{\theta_t}(Z) \rangle_{\mathcal{H}_\Theta}}_{a_{t,i}(Z)} \psi_{t,i}(z) . \tag{2}$$

To capture the convergence of the whole domain, we consider the function $\Delta R_{\theta_t}(\cdot; Z)$ in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}_{\Theta_t}$ of Θ_t, with its inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_{\Theta_t}}$ defined such that $\langle \psi_{t,i}, \psi_{t,j} \rangle_{\mathcal{H}_{\Theta_t}} = \delta_{ij}/\lambda_{t,i}$. Ignoring factors of η, we define the *convergence degree* $\alpha(Z)$ to be the RKHS norm of $\Delta R_{\theta_t}(\ \cdot\ ; Z)$, or

$$\alpha(Z) \triangleq \|\Delta R_{\theta_t}(\ \cdot\ ; Z)\|^2_{\mathcal{H}_{\Theta_t}} = \textstyle\sum_{i=0}^{\infty} \lambda_{t,i}^{-1} \langle \Delta R_{\theta_t}(\ \cdot\ ; Z), \psi_{t,i} \rangle^2_{\mathcal{H}_{\Theta_t}} = \sum_{i=0}^{\infty} \lambda_{t,i} a_{t,i}(Z)^2. \tag{3}$$

Training points that maximizes the convergence degree could speed up training convergence, which corresponds to Process 1 of the PINN training process. Hence, our point selection algorithm will be centered around choosing a training set Z that maximizes $\alpha(Z)$. In Appendix E, we show that maximizing $\alpha(Z)$ also leads to lower generalization error bounds.

In practice, we cannot compute the eigenfunctions of the eNTK exactly. Hence, we estimate the eNTK eigenfunctions and corresponding eigenvalues $\psi_{t,i}$ and $\lambda_{t,i}$ using the

2. Note that a PDE CL point $z = (x, \mathrm{p}) \in \mathcal{Z}$ is distinct from an Exp point $z = (x, \mathrm{s})$, even though both points have the same coordinate x.
3. For compactness, we will overload the notations with $R_{\theta_t}(Z)$ and $\Theta_t(Z, Z')$ for cases of multiple inputs, and use $\Theta_t(Z)$ to refer to $\Theta_t(Z, Z)$.

Nystrom approximation (Williams and Seeger, 2000; Schölkopf and Smola, 2002). Given a reference set Z_{ref} of size p, and the eigenvectors $\tilde{v}_{t,i}$ and eigenvalues $\tilde{\lambda}_{t,i}$ of $\Theta_t(Z_{\text{ref}})$, for each $i = 1, \ldots, p$, we can approximate $\psi_{t,i} \approx \hat{\psi}_{t,i}(z) \triangleq \tilde{\lambda}_{t,i}^{-1} \sqrt{|Z_{\text{ref}}|}\Theta_t(z, Z_{\text{ref}})\tilde{v}_{t,i}$ and $\lambda_{t,i} \approx \hat{\lambda}_{t,i} \triangleq \tilde{\lambda}_{t,i}|Z_{\text{ref}}|^{-1}$. Selecting $p \ll |Z_{\text{pool}}|$ is sufficient since NTK eigenvalues typically decay rapidly (Lee et al., 2019; Wang et al., 2022c). Given $\hat{\psi}_{t,i}$ and $\hat{\lambda}_{t,i}$, we can approximate $a_{t,i}(Z) \approx \hat{a}_{t,i}(Z) \triangleq \langle \hat{\psi}_{t,i}(Z), R_{\theta_t}(Z) \rangle = \sqrt{|Z_{\text{ref}}|}\ \tilde{\lambda}_{t,i}^{-1}\ \tilde{v}_{t,i}^{\top}\Theta_t(Z_{\text{ref}}, Z)R_{\theta_t}(Z)$ and the convergence degree (3) as

$$\alpha(Z) \approx \hat{\alpha}(Z) \triangleq \textstyle\sum_{i=0}^{p} \hat{\lambda}_{t,i}\ \hat{a}_{t,i}(Z)^2 = \sum_{i=1}^{p} \tilde{\lambda}_{t,i}^{-1} \big(\tilde{v}_{t,i}^{\top}\Theta_t(Z_{\text{ref}}, Z)R_{\theta_t}(Z)\big)^2\ . \tag{4}$$

In Appendix E.2, we show that the approximate convergence degree (4) provides a sufficiently good estimate for the true convergence degree.

PINNACLE Algorithm. We now introduce the PINNACLE algorithm (Algorithm 1 in the Appendix), which consists of two alternating phases: the point selection phase, and the training phase. In the point selection phase, we generate a random pool of candidate points Z_{pool} (Line 4), then aim to select a subset $Z \subset Z_{\text{pool}}$ which fits our constraints and maximizes $\hat{\alpha}(Z)$ (Line 6). However, this optimization problem is difficult to solve exactly. Instead, we propose two approaches to approximate the optimal set for $\hat{\alpha}$ (additional details are in Appendix F.1):

1. **Sampling (S)**: We sample a batch of points from Z_{pool} where the probability of selecting a point $z \in Z_{\text{pool}}$ is proportional to $\hat{\alpha}(z)$.

2. **K-Means++ (K)**: We embed each point by $z \mapsto \big(\hat{\lambda}_{t,i}^{1/2}\hat{a}_{t_i}(z)\big)_{i=1}^{p}$, and perform K-Means++ initialization on these embedded vectors to select a batch of points. K-Means++ initialization is a technique also used in previous batch-mode active learning algorithms (Ash et al., 2020) to promote diversity based on the embedding used.

In the training phase, we perform model training with GD (Line 10) for a fixed number of iterations or until the eNTK has change too much (Line 11), to account for the eNTK evolution during GD (Process 2 from above). Inspired by Lee et al. (2019), we quantify this change based on the Frobenius norm of the difference of the eNTK since the start of the phase. The point selection phase will then repeat with a re-computed eNTK. In practice, rather than replacing the entire training set during the new point selection phase, we randomly retain a portion of the old set and fill the remaining budget with new points selected based on the recomputed eNTK. We discuss these mechanisms further in Appendix F.2.

3. Experimental Results

Point Selection Behavior. We first interpret PINNACLE's automated point selection. We consider the forward problems for the 1D Advection and 1D Burgers time-dependent PDEs, where the goal is to estimate the PDE solution u given the PDEs with known β and IC/BCs but no Exp data. Figure 1 plots the training points of all types selected by PINNACLE-K overlaid on the NN output $\hat{u}_\theta$, at various training stages. First, PINNACLE automatically starts with more IC CL points and gradually shifts towards more PDE and BC CL points as training progresses. Intuitively, this helps as ICs provide more informative

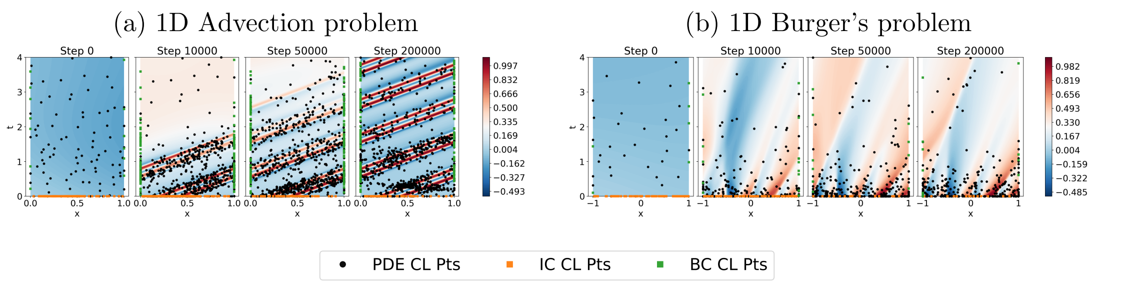


Figure 1: Example of points selected by PINNACLE-K during training of various forward problems. The points represent the various types of CL points selected by PINNACLE-K, whereas the patterns in the background represents the obtained NN solution.

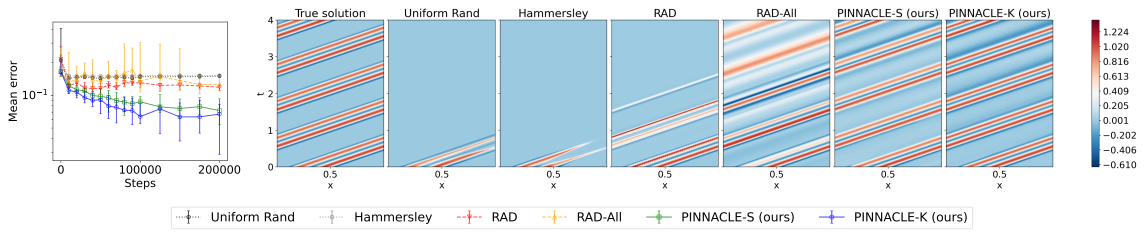


Figure 2: Results from the 1D Advection forward problem experiments. In each row, from left to right: plot of mean prediction error for each method, the true PDE solution, and the PINN output from different point selection methods. Equivalent plots for the 1D Burger's equation can be found in Figure 12 in the Appendix.

constraints for $\hat{u}_\theta$ in earlier training stages, while later on, PDE CL points then are used to "propagate" the solution across the domain interior.

Second, the specific choice of points are surprisingly interpretable. For the Advection equation experiment (shown in Figure 1a), we can see that PINNACLE focuses on PDE and BC CL points closer to the initial boundary and "propagates" the concentration of points upwards, similar to the process of causal learning to construct the PDE solution (Wang et al., 2022b; Krishnapriyan et al., 2021). The selected points also cluster around more informative regions, such as the stripes in the Advection solution. Similar observations can be made for Burger's equation (shown in Figure 1b), where points focus around regions with richer features near the IC and sharper features higher in the plot.

Forward Problems. We present the results for the 1D Advection (Figure 2) and 1D Burger's (Figure 12 in the Appendix) forward problem experiments[4]. Quantitatively, both variants of PINNACLE are able to achieve a lower predictive loss than other algorithms. The performance gap is also large enough to be seen qualitatively, as evident in, for example, 1D Advection experiments (Figure 2), where PINNACLE learn all stripes of the original solution while the benchmarks got stuck at only the first few closest to the ICs. Additional forward problem experiments and results can be found in Appendix H.2.

4. We provide details about the experiment setup in Appendix G.

Inverse problem. In the inverse problem, we estimate the unknown parameters of the PDE β, given the form of the PDE, IC/BC conditions, and some EXP points. We consider the 2D time-dependent incompressible Navier-Stokes equation, where we aim to find two unknown parameters in the PDE based on the queried EXP data. Figure 3 shows that PINNACLE can recover the correct inverse parameters, while the other algorithms had difficulties doing so. Further results for the inverse problem can be found in Appendix H.3.

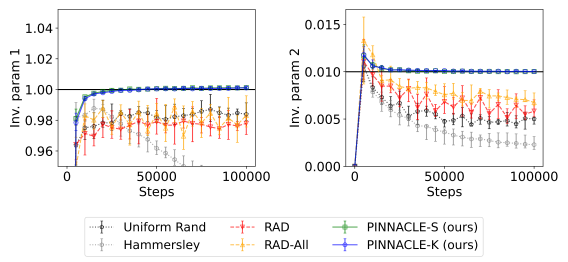


Figure 3: The two inverse parameters learned in the 2D Navier-Stokes equation problem. The black horziontal lines represents the true parameter values, while the other lines represent the predicted values of the inverse parameters during training.

Transfer Learning of PINNs with Perturbed IC. We also consider the setting where we already have a PINN trained for a given PDE and IC, but need to learn the solution for the same PDE with a perturbed IC. Different IC/BCs correspond to different PDE solutions, and hence can be viewed as different PINN tasks. To reduce cost, we consider fine-tuning the pre-trained PINN given a tighter budget ($5\times$ fewer points) compared to training another PINN from scratch. Figure 4 shows results for the 1D Advection equation. PINNACLE is able to exploit the pre-trained solution structure and more efficiently reconstruct the solution for the new IC than benchmarks. Additional transfer learning results can be found in Appendix H.4.

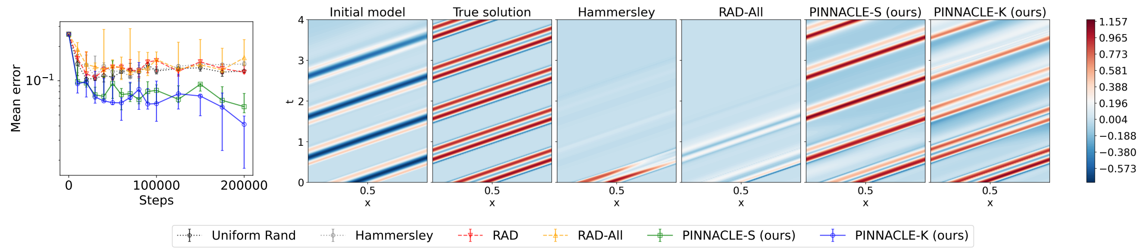


Figure 4: Predictive error for the 1D Advection transfer learning problem with perturbed IC. From left to right: plot of mean prediction error for each method, the true PDE solution, and the PINN output from different point selection methods.

4. Conclusion

We propose a novel algorithm, PINNACLE, that jointly optimizes the selection of all training point types for PINNs for more efficient training, while automatically adjusting the proportion of collocation point types during training. While this work is focused on PINNs, PINNACLE variants can also be applied to other deep learning problems which involve a composite loss function and input points from different domains as well. Future work could also extend PINNACLE to other settings such as deep operator learning.

Acknowledgments

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-PhD/2023-01-039J). This research is part of the programme DesCartes and is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-018).

References

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks. In *Proc. ICML*, pages 322–332, May 2019a.

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Proc. NeurIPS*, number 731, pages 8141–8150. December 2019b.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In *Proc. ICLR*, April 2020.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George E. Karniadakis. Physics-Informed Neural Networks for Heat Transfer Problems. *Journal of Heat Transfer*, 143(6):060801, June 2021. ISSN 0022-1481, 1528-8943.

Yuan Cao and Quanquan Gu. Generalization Bounds of Stochastic Gradient Descent for Wide and Deep Neural Networks. In *Proc. NeurIPS*, volume 32. Curran Associates, Inc., 2019.

Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395: 114909, May 2022. ISSN 0045-7825.

Junwoo Cho, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, and Eunbyung Park. Separable Physics-Informed Neural Networks. (arXiv:2306.15969), July 2023.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, 92, July 2022.

Zhongxiang Dai, Yao Shu, Bryan Kian Hsiang Low, and Patrick Jaillet. Sample-then-optimize batch neural Thompson sampling. In *Proc. NeurIPS*, 2022.

Zhongxiang Dai, Gregory Kang Ruey Lau, Arun Verma, Yao Shu, Bryan Kian Hsiang Low, and Patrick Jaillet. Quantum Bayesian Optimization. In *Proc. NeurIPS*, 2023a.

Zhongxiang Dai, Yao Shu, Arun Verma, Flint Xiaofeng Fan, Bryan Kian Hsiang Low, and Patrick Jaillet. Federated neural bandits. In *Proc. ICLR*, 2023b.

Tim De Ryck and Siddhartha Mishra. Generic bounds on the approximation error for physics-informed (and) operator learning. In *Proc. NeurIPS*, volume 35, pages 10945–10958, 2022.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian Active Learning with Image Data. In *Proc. ICML*, pages 1183–1192, July 2017.

Yihang Gao, Yiqi Gu, and Michael Ng. Gradient Descent Finds the Global Optima of Two-Layer Physics-Informed Neural Networks. In *Proc. ICML*, pages 10676–10707, July 2023.

Roman Garnett. *Bayesian Optimization.* Cambridge Univ. Press, 2022.

Alex Gittens. The spectral norm error of the naive Nystrom extension. (arXiv:1110.5305), October 2011.

Alex Gittens and Michael Mahoney. Revisiting the Nystrom method for improved large-scale machine learning. In *Proc. ICML*, volume 28, pages 567–575, June 2013.

Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. Bayesian Deep Ensembles via the Neural Tangent Kernel. In *Proc. NeurIPS*, volume 33, pages 1010–1022. Curran Associates, Inc., 2020.

Apivich Hemachandra, Zhongxiang Dai, Jasraj Singh, See-Kiong Ng, and Bryan Kian Hsiang Low. Training-free neural active learning with initialization-robustness guarantees. In *Proc. ICML*, pages 12931—12971, July 2023.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Proc. NeurIPS*, volume 31, 2018.

Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2239):20200334, July 2020a.

Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, March 2020b. ISSN 0021-9991.

Ameya D. Jagtap, Zhiping Mao, Nikolaus Adams, and George Em Karniadakis. Physics-informed neural networks for inverse problems in supersonic flows. *Journal of Computational Physics*, 466:111402, October 2022. ISSN 00219991.

Xinchao Jiang, Xin Wanga, Ziming Wena, Enying Li, and Hu Wang. An E-PINN assisted practical uncertainty quantification for inverse problems. (arXiv:2209.10195), September 2022.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. In *Proc. NeurIPS*, volume 32, 2019.

Dmitry Kopitkov and Vadim Indelman. Neural Spectrum Alignment: Empirical Study. *Artificial Neural Networks and Machine Learning*, 12397:168–179, 2020.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Proc. NeurIPS*, volume 34, pages 26548–26560, 2021.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124002, December 2019. ISSN 1742-5468.

Lu Lu, Xuhui Meng, Zhiping Mao, and George E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, January 2021. ISSN 0036-1445, 1095-7200.

Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating PDEs. (arXiv:2006.16144), September 2021.

Mohamad Amin Mohamadi, Wonho Bae, and Danica J. Sutherland. Making Look-Ahead Active Learning Strategies Feasible with Neural Tangent Kernels. In *Proc. NeurIPS*, 2022.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning (Second Edition)*. Adaptive Computation and Machine Learning series. MIT Press, 2018.

Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36(8):962–977, August 2021. ISSN 1093-9687, 1467-8667.

Wei Peng, Weien Zhou, Xiaoya Zhang, Wen Yao, and Zheliang Liu. RANG: A Residual-based Adaptive Node Generation Method for Physics-Informed Neural Networks. (arXiv:2205.01051), May 2022.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019. ISSN 0021-9991.

Hiranmayi Ranganathan, Hemanth Venkateswara, Shayok Chakraborty, and Sethuraman Panchanathan. Deep active learning for image classification. In *Proc. ICIP*, pages 3934–3938, Beijing, September 2017. ISBN 978-1-5090-2175-8.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A Survey of Deep Active Learning. *ACM Computing Surveys*, 54(9):180:1–180:40, October 2021. ISSN 0360-0300.

Simiao Ren, Yang Deng, Willie J. Padilla, Leslie Collins, and Jordan Malof. Deep Active Learning for Scientific Computing in the Wild. (arXiv:2302.00098), January 2023.

Alexander Rodríguez, Jiaming Cui, Naren Ramakrishnan, Bijaya Adhikari, and B. Aditya Prakash. Einns: Epidemiologically-informed neural networks. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Proc. AAAI*, pages 14453–14460, 2023.

Franz M. Rohrhofer, Stefan Posch, Clemens Gößnitzer, and Bernhard C. Geiger. On the Role of Fixed Points of Dynamical Systems in Training Physics-Informed Neural Networks. (arXiv:2203.13648), February 2023.

Bernhard Schölkopf and Alexander Johannes Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Adaptive Computation and Machine Learning Series. MIT Press, Cambridge, Mass., reprint. edition, 2002. ISBN 978-0-262-19475-4 978-0-262-53657-8.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *Proc. ICLR*, 2018.

Burr Settles. *Active Learning.* Springer International Publishing, Cham, 2012. ISBN 978-3-031-00432-2 978-3-031-01560-1.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

Yao Shu, Shaofeng Cai, Zhongxiang Dai, Beng Chin Ooi, and Bryan Kian Hsiang Low. NASI: Label- and Data-agnostic Neural Architecture Search at Initialization. In *Proc. ICLR*, 2022a.

Yao Shu, Zhongxiang Dai, Zhaoxuan Wu, and Bryan Kian Hsiang Low. Unifying and Boosting Gradient-Based Training-Free Neural Architecture Search. In *Proc. NeurIPS*, 2022b.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An Extensive Benchmark for Scientific Machine Learning. In *Proc. NeurIPS*, volume 35, pages 1596–1611, 2022.

Kejun Tang, Xiaoliang Wan, and Chao Yang. DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations. *Journal of Computational Physics*, 476:111868, March 2023. ISSN 0021-9991.

Sattar Vakili, Michael Bromberg, Da-Shan Shiu, and Alberto Bernacchia. Uniform Generalization Bounds for Overparameterized Neural Networks. *CoRR*, abs/2109.06099, 2021.

Haonan Wang, Wei Huang, Ziwei Wu, Hanghang Tong, Andrew J. Margenot, and Jingrui He. Deep Active Learning by Leveraging Training Dynamics. In *Proc. NeurIPS*, volume 35, pages 25171–25184, December 2022a.

Shusen Wang, Alex Gittens, and Michael W. Mahoney. Scalable Kernel K-Means Clustering with Nyström Approximation: Relative-Error Bounds. *Journal of Machine Learning Research*, 20(1):431–479, January 2019. ISSN 1532-4435.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, October 2021a. ISSN 00457825.

Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. (arXiv:2203.07404), March 2022b.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, January 2022c. ISSN 0021-9991.

Zheng Wang and Jieping Ye. Querying discriminative and representative samples for batch mode active learning. *ACM Transactions on Knowledge Discovery from Data*, 9(3), February 2015. ISSN 1556-4681.

Zhilei Wang, Pranjal Awasthi, Christoph Dann, Ayush Sekhari, and Claudio Gentile. Neural Active Learning with Performance Guarantees. In *Proc. NeurIPS*, volume 34, pages 7510–7521, 2021b.

Christopher Williams and Matthias Seeger. Using the Nyström Method to Speed Up Kernel Machines. In *Proc. NeurIPS*, volume 13, 2000.

Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023. ISSN 0045-7825.

Zhaoxuan Wu, Yao Shu, and Bryan Kian Hsiang Low. DAVINZ: Data Valuation using Deep Neural Networks at Initialization. In *Proc. ICML*, pages 24150–24176, June 2022.

Shaojie Zeng, Zong Zhang, and Qingsong Zou. Adaptive deep neural networks methods for high-dimensional partial differential equations. *Journal of Computational Physics*, 463: 111232, August 2022. ISSN 0021-9991.

Appendix A. Notations and Abbreviations

Table 1: List of notations used throughout the paper

Symbol	Meaning	Example
$\delta_{i,j}$	Dirac-Delta function, i.e. $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise	
$\mathbb{1}$	Identity matrix	
$\mathcal{N}$	PDE operator	(5)
$\mathcal{B}$	IC/BC operator	(5)
$\mathcal{X}$	Input space of PDE solution	(5)
$\partial\mathcal{X}$	Boundary of the input space of PDE solution	(5)
u	PDE true solution	(5)
$\hat{u}_\theta$	PINN used to approximate PDE solution	(6)
$\hat{u}_{\theta_t}$	PINN at timestep t of training	
$\mathcal{L}(\hat{u}_\theta; X)$	Loss of NN $\hat{u}_\theta$ w.r.t. set X	(6)
η	Learning rate	(9)
x	Input of PDE solution, i.e., elements in $\mathcal{X}$	
r	Generic training point label, i.e., $\mathrm{r} \in \{\mathrm{s}, \mathrm{p}, \mathrm{b}\}$	
$\mathrm{s}, \mathrm{p}, \mathrm{b}$	Specific training point label	
$\mathcal{Z}$	Augmented space	
Z	Set of elements from the augmented space	
F	General prediction operator	
$\Theta_t(\cdot, \cdot)$	eNTK of NN $\hat{u}_{\theta_t}$ at timestep t	
$\lambda_{t,i}$	ith largest eigenvalue of Θ_t	
$\psi_{t,i}$	Eigenfunction of Θ_t corresponding to eigenvalue $\lambda_{t,i}$	
$\nabla_\theta \hat{u}_{\theta_t}(x)$	Jacobian of $\hat{u}_\theta(x)$ with respect to θ at $\theta = \theta_t$	
$R_{\theta_t}(z)$	Residual at time t at point z	(15)
$\Delta R_{\theta_t}(z; Z)$	Change in residual at time t at point z when trained with Z	(15)
$a_{t,i}(Z)$	Residual expansion coefficient in dimension i at time t	(15)
$\langle \cdot, \cdot \rangle_{\mathcal{H}_\Theta}$	Inner product of RKHS space	(15)
Z_{ref}	Subset of $\mathcal{Z}$ for Nystrom approximation	
Z_{pool}	Subset of $\mathcal{Z}$ for point selection	Algorithm 1

Table 2: List of abbreviations used throughout the paper in alphabetical order

Abbreviation	Meaning
BC	boundary condition
CL	collocation (as in collocation points)
Exp	experimental (as in experimental points)
eNTK	empirical neural tangent kernel
GD	gradient descent
IC	initial condition
NTK	neural tangent kernel
PDE	partial differential equation
PINN	physics-informed neural network
RKHS	reproducing kernel Hilbert space

Appendix B. Related Works

Multiple works have examined the challenges of training PINNs, such as failure of PINNs to learn certain PDE solutions, particularly those with high frequency (Krishnapriyan et al., 2021; Rohrhofer et al., 2023). In response, there have been several proposals to adjust the training procedures of PINNs to mitigate this effect. This includes reweighing the loss function (Wang et al., 2022c), causal learning (Wang et al., 2022b) or curriculum learning approaches (Krishnapriyan et al., 2021). Most relevant to this paper are works that looked into training points selection to improve PINNs training performance, though they have focused only on better selection of PDE CL points rather than joint optimization of the selection of Exp points and all types of PDE and IC/BC CL points. These works either adopt a non-adaptive approach (Wu et al., 2023), or an adaptive approach involving a scoring function based on some quantity related to the magnitude of the residual at each training point (Lu et al., 2021; Nabian et al., 2021; Wu et al., 2023; Peng et al., 2022; Zeng et al., 2022; Tang et al., 2023).

Outside of PINNs, the most similar problem to training point selection for PINNs would be the problem of deep active learning, which is a well-studied problem (Settles, 2012; Ren et al., 2021), with proposed algorithms based on diversity measures (Sener and Savarese, 2018; Ash et al., 2020; Wang and Ye, 2015) and uncertainty measures (Gal et al., 2017; Kirsch et al., 2019). Active learning can be viewed as an exploration-focused variant of Bayesian optimization (BO), which also utilizes principled uncertainty measures provided by surrogate models to optimize a black-box function (Garnett, 2022; Dai et al., 2023a; Shahriari et al., 2016).

A number of works have also incorporated NTKs to estimate the NN training dynamics to better select points for the NN (Wang et al., 2021b, 2022a; Mohamadi et al., 2022; Hemachandra et al., 2023). However, most of these deep active learning methods are unable to directly translate to scientific machine learning settings (Ren et al., 2023). Nonetheless, there have been limited works regarding active learning for PINNs, such as methods using the uncertainty computed from an ensemble of PINNs (Jiang et al., 2022).

In this work, we extensively used NTKs to analyze our neural network traaining dynamics. NTKs are also useful other tasks such as data valuation (Wu et al., 2022), neural architecture search (Shu et al., 2022a,b), multi-armed bandits (Dai et al., 2023b), Bayesian optimization (Dai et al., 2022), and uncertainty quantification (He et al., 2020). NTKs have also been used to study NN generalization loss (Cao and Gu, 2019) and also in relating NN training with gradient descent and kernel ridge regression (Arora et al., 2019a; Vakili et al., 2021).

Appendix C. Background

In this section we provide relevant background information for our work.

C.1 Physics-Informed Neural Networks

Consider partial differential equations (PDEs) of the form

$$\mathcal{N}[u,\beta](x) = f(x) \quad \forall x \in \mathcal{X} \qquad \text{and} \qquad \mathcal{B}_i[u](x'_i) = g_i(x'_i) \quad \forall x'_i \in \partial\mathcal{X}_i \tag{5}$$

where $u(x)$ is the function of interest over a coordinate variable x defined on a bounded domain $\mathcal{X} \subset \mathbb{R}^d$ (where time could be a subcomponent), $\mathcal{N}$ is a PDE operator acting on $u(x)$ with parameters[5] β, and $\mathcal{B}_i[u]$ are initial/boundary conditions (IC/BCs)

Physics-Informed Neural Networks (PINNs) are neural networks (NN) $\hat{u}_\theta$ that approximates $u(x)$. They are trained on a dataset X that can be partitioned into X_s, X_p and X_b corresponding to the EXP points, PDE CL points and IC/BC CL points respectively. The PDE and IC/BCs (5) are added as regularization (typically mean-squared errors terms) in the loss function[6]

$$\mathcal{L}(\hat{u}_\theta; X) = \sum_{x \in X_s} \frac{\left(\hat{u}_\theta(x) - u(x)\right)^2}{2N_s} + \lambda_p \sum_{x \in X_p} \frac{\left(\mathcal{N}[\hat{u}_\theta](x) - f(x)\right)^2}{2N_p} + \lambda_b \sum_{x \in X_b} \frac{\left(\mathcal{B}[\hat{u}_\theta](x) - g(x)\right)^2}{2N_b} \tag{6}$$

where the components penalize the failure of $\hat{u}_\theta$ in satisfying ground truth labels $u(x)$, the PDE, and the IC/BCs constraints respectively, and λ_p and λ_b are positive scalar weights.

Rationale on Loss Function. In regular NN training, our goal is to train a model such that its output is as close to the true solution as possible. This can be quantified using mean squared error (MSE) loss between the true output $u(x)$ and NN output $\hat{u}_\theta(x)$ at some selected samples, which can simply be written as $\sum_x \left(\hat{u}_\theta(x) - u(x)\right)^2$, ignoring constant factors.

For PINNs, we also want to further constrain our solution such that they follow some PDE with IC/BC conditions such as (5). This can be seen as a constraint on the NN output, where we want the expected residual to be zero, i.e., considering only the PDE without the IC/BC, the constraint would be in the form $\mathcal{N}[\hat{u}_\theta](x) - f(x) = 0$. However, since enforcing a hard constraint, given by the PDE, on the NN is difficult to do, we instead incorporate a soft constraint that introduce a penalty depending on how far the NN diverges from a valid PDE solution. We would want this penalty (or regularization term in the loss function) to be of the form of $\mathbb{E}_x\left[\left(\mathcal{N}[\hat{u}_\theta](x) - f(x)\right)^2\right]$, where instead of performing the expectation exactly we sample some CL points to approximate the PDE loss term. This is a motivation behind using non-adaptive CL point selection method, which is seen as a way to approximate the expected residual, akin to the roles of quadrature points used in numerical integration.

Alternatively, some works have considered an adaptive sampling strategy where the CL points are selected where there is a higher residual. In this case, the expected residual plays a different role, which is to encourage the NN to learn regions of the solution with higher loss. for boundaries $\partial\mathcal{X}_i \subset \mathcal{X}$. For PINN training, we assume operators $\mathcal{N}$ and $\mathcal{B}$, and functions f and g are known, while β may or may not be known.

5. To simplify notation, we write $\mathcal{N}[u, \beta]$ as $\mathcal{N}[u]$, keeping the β dependence implicit, except for cases where the task is to learn β. Examples of PDEs, specifically those in our experiments, can be found in Appendix G.1.
6. For notational simplicity we will denote a single IC/BC $\mathcal{B}$ in subsequent equations, however the results can easily be generalized to include multiple ICs/BCs as well.

C.2 Neural Tangent Kernels

PINNACLE relies on analyzing NN training dynamics and convergence rates. To do so we require the use of Neural Tangent Kernels (NTKs), which is a tool which can be used to analyze general NNs as well as PINNs. The empirical NTK (eNTK) of a NN $\hat{u}_{\theta_t}$ at step t of the training process (Jacot et al., 2018),

$$\Theta_t(x, x') = \nabla_\theta \hat{u}_{\theta_t}(x) \ \nabla_\theta \hat{u}_{\theta_t}(x')^\top, \tag{7}$$

can be used to characterize its training behavior under gradient descent (GD). For a NN trained using GD with learning rate η on training set X, the NN parameters θ_t evolves as $\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\hat{u}_{\theta_t}; X)$ while the model output evolves as $\hat{u}_{\theta_{t+1}}(x) = \hat{u}_{\theta_{\theta_t}}(x) - \eta \Theta_t(x, X) \nabla_{\hat{u}_\theta} \mathcal{L}(\hat{u}_{\theta_t}; X)$. Given an MSE loss $\mathcal{L}(\hat{u}_\theta; X) = \sum_{x \in X} \big(\hat{u}_\theta(x) - u(x)\big)^2 / 2|X|$, the change in model output during training in this case can be written as

$$\Delta \hat{u}_{\theta_t}(x) = \hat{u}_{\theta_{t+1}}(x) - \hat{u}_{\theta_t}(x) = -\eta \Theta_t(x, X)\big(\hat{u}_{\theta_t}(X) - u(X)\big). \tag{8}$$

Past works (Arora et al., 2019b) had shown that in the infinite-width limit and GD with infinitesimal learning rate, the NN follows a linearlized regime and the eNTK converges to a fixed kernel Θ that is invariant over training step t. Even though these assumptions may not fully hold in practice, many works (e.g. Arora et al. (2019b,a); **?**); Shu et al. (2022b)) have shown that the NTK still provides useful insights to the training dynamics for finite-width NNs.

C.3 NTK of PINNs

The Neural Tangent Kernel (NTK) of PINNs can be expressed as a block matrix broken down into the various loss components in (6) (Wang et al., 2021a). To illustrate, a PINN trained with just the Exp points X_s and PDE CL points X_p using gradient descent (GD) with learning rate η has training dynamics that can be described by

$$\begin{bmatrix} \hat{u}_{\theta_{t+1}}(X_s) - \hat{u}_{\theta_t}(X_s) \\ \mathcal{N}[\hat{u}_{\theta_{t+1}}](X_p) - \mathcal{N}[\hat{u}_{\theta_t}](X_p) \end{bmatrix} = -\eta \begin{bmatrix} \Theta_{t,ss} & \Theta_{t,sp} \\ \Theta_{t,ps} & \Theta_{t,pp} \end{bmatrix} \begin{bmatrix} \hat{u}_{\theta_t}(X_s) - u(X_s) \\ \mathcal{N}[\hat{u}_{\theta_t}](X_p) - f(X_p) \end{bmatrix} \tag{9}$$

where $J_{t,s} = \nabla_\theta \hat{u}_{\theta_t}(X_s)$, $J_{t,p} = \nabla_\theta \mathcal{N}[\hat{u}_{\theta_t}](X_p)$, and $\Theta_{t,ss} = J_{t,s} J_{t,s}^\top$, $\Theta_{t,pp} = J_{t,p} J_{t,p}^\top$, and $\Theta_{t,sp} = \Theta_{t,ps}^\top = J_{t,p} J_{t,s}^\top$, are the submatrices of the PINN empirical NTK (eNTK). Past works have analyzed PINNs training dynamics using the eNTK (Wang et al., 2021a, 2022c; Gao et al., 2023).

C.4 Reproducing Kernel Hilbert Space

In this section, we provide a brief overview required on the reproducing kernel Hilbert space (RKHS) for our paper. This section adapted from Schölkopf and Smola (2002).

Consider a symmetric positive semi-definite kernel $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. We can define an integral operator $T_\mathcal{K}$ as

$$T_\mathcal{K}[f](\cdot) = \int_\mathcal{Z} f(x) \mathcal{K}(x, \cdot) d\mu(x). \tag{10}$$

We can compute the eigenfunctions and corresponding eigenvectors of $T_\mathcal{K}$, which are pairs of functions ψ_i and non-negative values λ_i such that $T_\mathcal{K}[\psi_i](\cdot) = \lambda_i \psi_i(\cdot)$. Typically, we also refer to the eigenvectors and eigenvalues of $T_{\mathcal{K}}$ as the eigenvectors and eigenvalues of $\mathcal{K}$. By Mercer's theorem, we can show that the kernel $\mathcal{K}$ can be written in terms of its eigenfunctions and eigenvalues as

$$\mathcal{K}(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(x'). \tag{11}$$

Based on this, we define the feature map as

$$\phi_i(\cdot) = \mathcal{K}(x, \cdot) = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(\cdot)$$

and consequently the inner product such that $\langle \psi_i, \psi_j \rangle_{\mathcal{H}_{\mathcal{K}}} = \delta_{i,j}/\lambda_i$. This is so that we are able to write

$$\langle \phi(x), \phi(x') \rangle_{\mathcal{H}_{\mathcal{K}}} = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \lambda_i \lambda_j \psi_i(x) \psi_j(x') \langle \psi_i(x), \psi_j(x') \rangle_{\mathcal{H}_{\mathcal{K}}} \tag{12}$$

$$= \sum_{i=1}^{\infty} \frac{\lambda_i^2 \psi_i(x) \psi_i(x')}{\lambda_i} = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(x') = \mathcal{K}(x, x'). \tag{13}$$

From this, the RKHS $\mathcal{H}_{\mathcal{K}}$ of a kernel $\mathcal{K}$ would be given by a set of function with finite RKHS norm, i.e.,

$$\mathcal{H}_{\mathcal{K}} = \left\{ f \in L_2(\mathcal{X}) : \langle f, f \rangle_{\mathcal{H}_{\mathcal{K}}} = \sum_{i=1}^{\infty} \frac{\langle f, \phi_i \rangle_{\mathcal{H}_{\mathcal{K}}}^2}{\lambda_i} \leq \infty \right\}. \tag{14}$$

Appendix D. PINN eNTK and Training Dynamics Intuition

In this section, we further discuss the PINNs eNTK eigenspectrum in the augmented space that we have introduced, and show how it relates to the training dynamics of PINNs.

The change in residual at any point $z \in \mathcal{Z}$, given training set Z for a GD step with small enough η, is

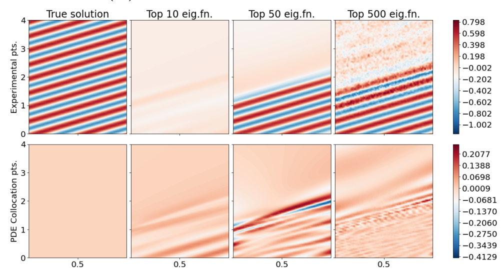
$$\Delta R_{\theta_t}(z; Z) \triangleq R_{\theta_{t+1}}(z; Z) - R_{\theta_t}(z; Z) \approx -\eta \sum_{i=1}^{\infty} \lambda_{t,i} \underbrace{\langle \psi_{t,i}(Z), R_{\theta_t}(Z) \rangle_{\mathcal{H}_\Theta}}_{a_{t,i}(Z)} \psi_{t,i}(z) \tag{15}$$

Note that $\Delta R_{\theta_t}(z; Z)$ describes how the residual *at any point* z (of any point type, and including points outside Z) is influenced by GD training on any training set Z that can consist of *multiple* point types. Hence, $\Delta R_{\theta_t}(z; Z)$ describes the overall training dynamics for all point types.

We can gain useful insights on PINN training dynamics by considering the augmented eNTK eigenbasis. First, we show that the *residual components that align to the more dominant eigenfunctions will decay faster.* During a training period where the eNTK remains constant, i.e., $\Theta_t(z, z') \approx \Theta_0(z, z')$, the PINN residuals will evolve according to

$$R_{\theta_t}(z; Z) \approx R_{\theta_0}(z) - \Theta_0(z, Z) \sum_{i=1}^{\infty} \frac{1 - e^{-\eta t \lambda_{0,i}}}{\lambda_{0,i}} \, a_{0,i}(Z) \, \psi_{0,i}(Z) \,. \tag{16}$$

(a) After 10k steps of GD



(b) After 100k steps of GD

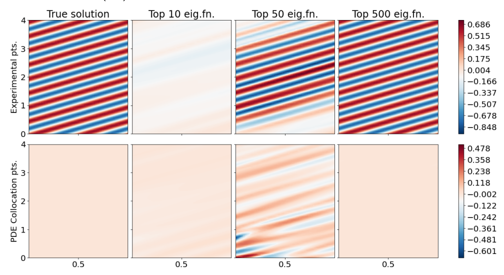


Figure 5: Reconstruction of the true PDE solution of the 1D Advection equation using eigenfunctions of Θ_t at different GD timesteps. For each time step, we plot the eigenfunction component both for (x, s) subspace (top row) and (x, p) subspace (bottom row).

From (16), we can see that the residual component that is aligned to the eigenfunction $\psi_{t,i}$ (given by $a_{t,i}(Z)$) will decay as training time t progresses at a rate proportional to $\eta\lambda_{0,i}$. For training set points $z \in Z$, the decay is exponential as (16) further reduces to $\sum_{i=1}^{\infty} e^{-\eta t \lambda_{0,i}}\ a_{t,i}(Z)\ \psi_{0,i}(Z)$. Interestingly, note that (16) applies to all $z \in \mathcal{Z}$, suggesting that choosing training point set Z with residual components more aligned to dominant eigenfunctions also results in lower residuals for the entire augmented space and faster overall convergence.

Second, during PINN training, *the eNTK eigenfunctions will gradually align to the true target solution*. While many theoretical analysis of NTKs assumes a fixed kernel during training, this does not hold in practice (Kopitkov and Indelman, 2020). Figure 5 shows that the dominant eigenfunctions can better reconstruct the true PDE solution as training progresses, indicating that the eNTK evolves during training to further align to the target solution. Notice how the predictions on *both subspaces* needed to be considered for PINNs, which had not been explored before in past works. While the top 10 eigenfunctions replicated the (x, p)-subspace prediction better, more eigenfunctions were needed to reconstruct the full augmented space prediction (both (x, s) and (x, p) subspaces).

In the following subsections, we will provide verification for (15) and (16).

D.1 Calculations of (15)

From the gradient descent update formula with loss function given by (1), we can see that

$$\begin{aligned}
\theta_{t+1} - \theta_t &= -\eta \nabla_\theta \mathcal{L}(\hat{u}_{\theta_t}; Z) && (17)\\
&= -\eta \nabla_\theta F[\hat{u}_{\theta_t}](Z)^\top \nabla_{F[\hat{u}_\theta]} \mathcal{L}(\hat{u}_{\theta_t}; Z) \qquad \text{by chain rule} && (18)\\
&= -\eta \nabla_\theta F[\hat{u}_{\theta_t}](Z)^\top R_{\theta_t}(Z) && (19)
\end{aligned}$$

which, as $\eta \to 0$, can be rewritten with continuous time GD (where the difference between each time step is approximated as a derivative) as (**?**)

$$\partial_t \theta_t = -\eta \nabla_\theta F[\hat{u}_{\theta_t}](Z)^\top R_{\theta_t}(Z) \tag{20}$$

and so under continuous time GD, we can write the function output as

$$\begin{aligned}
\Delta R_{\theta_t}(z;Z) &\approx \partial_t R_{\theta_t}(z;Z) && \text{as } \eta \to 0 && (21)\\
&= \partial_t F[\hat{u}_{\theta_t}](z) && && (22)\\
&= \nabla_\theta F[\hat{u}_{\theta_t}](z)\ \partial_t \theta_t && \text{by chain rule} && (23)\\
&= -\eta \nabla_\theta F[\hat{u}_{\theta_t}](z)\ \nabla_\theta F[\hat{u}_{\theta_t}](Z)^\top\ R_{\theta_t}(Z) && && (24)\\
&= -\eta \Theta_t(z,Z)\ R_{\theta_t}(Z). && && (25)
\end{aligned}$$

D.2 Calculations of (16)

In this case, we consider the PINN under the linearized regime, where we approximate $F[\hat{u}_\theta]$ using the Taylor approximation around θ_0,

$$F^{\text{lin}}[\hat{u}_\theta](z) = F[\hat{u}_{\theta_0}](z) + \nabla_\theta F[\hat{u}_{\theta_0}](z)\ (\theta - \theta_0)\,, \tag{26}$$

and subsequently $R^{\text{lin}}_\theta(z) = F^{\text{lin}}[\hat{u}_\theta](z) - F[u](z)$. Under this regime, the eNTK remains constant at all t, i.e. $\Theta_t(z,z') = \Theta_0(z,z') = \nabla_\theta F[\hat{u}_{\theta_0}](z)\ \nabla_\theta F[\hat{u}_{\theta_0}](z')^\top$ (**?**), which also means that its eigenvectors and corresponding values will not evolve over time. It has been shown empirically both for normal NNs (**?**) and PINNs (Wang et al., 2022c) that this regime holds true for sufficiently wide neural networks.

For simplicity we let ψ_i and λ_i be the eigenfunctions and corresponding eigenvalues of Θ_0 (where we drop the subscripts t for simplicity). The change in residue of the training set can be estimated using continuous time GD $\Delta R^{\text{lin}}_{\theta_t}(Z;Z) \approx \partial_t R^{\text{lin}}_{\theta_t}(Z;Z)$, and based on (25) can be rewritten as

$$\partial_t R^{\text{lin}}_{\theta_t}(Z) = -\eta \Theta_0(Z)\ R^{\text{lin}}_{\theta_t}(Z) \tag{27}$$

which, when solved, gives

$$R^{\text{lin}}_{\theta_t}(Z) = e^{-\eta t \Theta_0(Z)} R^{\text{lin}}_{\theta_0}(Z). \tag{28}$$

To extrapolate this result to other values of $z \in \mathcal{Z}$, we can use the definition of $R^{\text{lin}}_{\theta_t}(Z)$ to see that

$$F^{\text{lin}}[\hat{u}_{\theta_t}](Z) - F[u](Z) = e^{-\eta t \Theta_0(Z)}\big(F^{\text{lin}}[\hat{u}_{\theta_0}](Z) - F[u](Z)\big) \tag{29}$$

and therefore

$$\begin{aligned}
F^{\text{lin}}[\hat{u}_{\theta_t}](Z) &= F[u](Z) + e^{-\eta t \Theta_0(Z)}\big(F^{\text{lin}}[\hat{u}_{\theta_0}](Z) - F[u](Z)\big) && (30)\\
&= F^{\text{lin}}[\hat{u}_{\theta_0}](Z) - \big(\mathbb{1} - e^{-\eta t \Theta_0(Z)}\big)\big(F^{\text{lin}}[\hat{u}_{\theta_0}](Z) - F[u](Z)\big) && (31)\\
&= F^{\text{lin}}[\hat{u}_{\theta_0}](Z) - \Theta_0(Z)\Theta_0(Z)^{-1}\big(\mathbb{1} - e^{-\eta t \Theta_0(Z)}\big) R^{\text{lin}}_{\theta_0}(Z) && (32)\\
&= F^{\text{lin}}[\hat{u}_{\theta_0}](Z) + \nabla_\theta F[\hat{u}_{\theta_0}](Z)\ \underbrace{\nabla_\theta F[\hat{u}_{\theta_0}](Z)^\top \Theta_0(Z)^{-1}\big(e^{-\eta t \Theta_0(Z)} - \mathbb{1}\big) R^{\text{lin}}_{\theta_0}(Z)}_{\theta_t - \theta_0} && (33)
\end{aligned}$$

which, through comparing the above with the form of the linearized PINN, gives us what $\theta_t - \theta_0$ is during training. Therefore, plugging this into (26),

$$F^{\text{lin}}[\hat{u}_{\theta_t}](z) = F^{\text{lin}}[\hat{u}_{\theta_0}](z) + \nabla_\theta F[\hat{u}_{\theta_0}](z)\nabla_\theta F[\hat{u}_{\theta_0}](Z)^\top \Theta_0(Z)^{-1}\big(e^{-\eta t \Theta_0(Z)} - \mathbb{1}\big) R^{\text{lin}}_{\theta_0}(Z) \tag{34}$$

$$= F^{\text{lin}}[\hat{u}_{\theta_0}](z) + \Theta_0(z, Z)\Theta_0(Z)^{-1}\big(e^{-\eta t \Theta_0(Z)} - \mathbb{1}\big) R^{\text{lin}}_{\theta_0}(Z) \tag{35}$$

$$R^{\text{lin}}_{\theta_t}(z; Z) = R^{\text{lin}}_{\theta_0}(z; Z) + \Theta_0(z, Z)\Theta_0(Z)^{-1}\big(e^{-\eta t \Theta_0(Z)} - \mathbb{1}\big) R^{\text{lin}}_{\theta_0}(Z). \tag{36}$$

We can then decompose (36) into

$$R^{\text{lin}}_{\theta_t}(z; Z) = R^{\text{lin}}_{\theta_0}(z; Z) + \Theta_0(z, Z) \sum_{i=1}^{\infty} \frac{e^{-\eta t \lambda_i} - 1}{\lambda_i} \psi_i(Z)\psi_i(Z)^\top R^{\text{lin}}_{\theta_0}(Z). \tag{37}$$

D.3 Eigenspectrum of the eNTK

In this subsection, we elaborate further about the eigenspectrum of the eNTK. In Figure 6, we plot out the eigenfunctions $\psi_{t,i}$ of the eNTK for the 1D Advection forward problem at GD training steps $t = 10\text{k}$ and $t = 100\text{k}$. We can see that in the case of $t = 100\text{k}$ (when the PINN output $\hat{u}_\theta$ is already able to reconstruct the PDE solution well), the eigenfunctions are able to reconstruct the patterns that correspond to the true solution well. Note that due to the tight coupling between the true NN output and the PDE residual, the components of the eigenfunction for (x, s) and (x, p) are correlated and may share similar patterns. The dominant eigenfunctions will then tend to either have $\psi_{t,i}(x, \text{s})$ which aligns well with the true solution, or have $\psi_{t,i}(x, \text{p})$ which is flat (corresponding to the true PDE solution whose residual should be flat). In the less dominant eigenfunctions, the patterns will align worse with the true solution. Meanwhile, in the $t = 10\text{k}$ case, we see that the eigenfunctions will share some pattern with the true solution, but will not be as refined as for $t = 100\text{k}$. Furthermore, the less dominant eigenfunctions will be even noisier.

Empirically, we also find that the PINN eNTK eigenvalues falls quickly, allowing the dominant eigenfunctions (i.e., those with the largest eigenvalues) to effectively form a basis for the NN output in augmented space. This is consistent with results from past eNTK works for vanilla NNs (Kopitkov and Indelman, 2020; Vakili et al., 2021).

In Figure 7, we show how the eigenfunctions can describe the model training at a given GD training step. We see that by using just the top dominant eigenfunctions we are able to reconstruct the NN output $\hat{u}_\theta$.

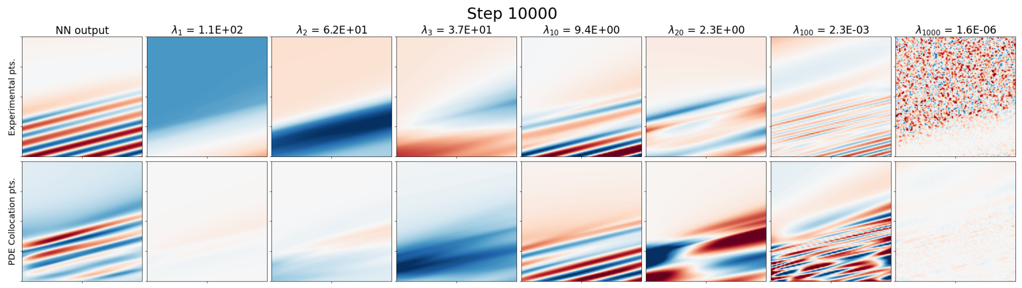
Appendix E. Theoretical Results

In this section, we will present two main theoretical results for the paper. This includes the relation between our criterion α and the generalization bound of PINNs, and the approximation error of the criterion $\hat{\alpha}$ compared to the true criterion value α.

E.1 Convergence Degree Criterion and Relations to Generalization Bounds

The change in residual $\Delta R_{\theta_t}(z; Z)$ (15) only captures the convergence at a single test point z. While it is possible to consider the average change in residual w.r.t. some validation

(a) 10k Steps



(b) 100k Steps

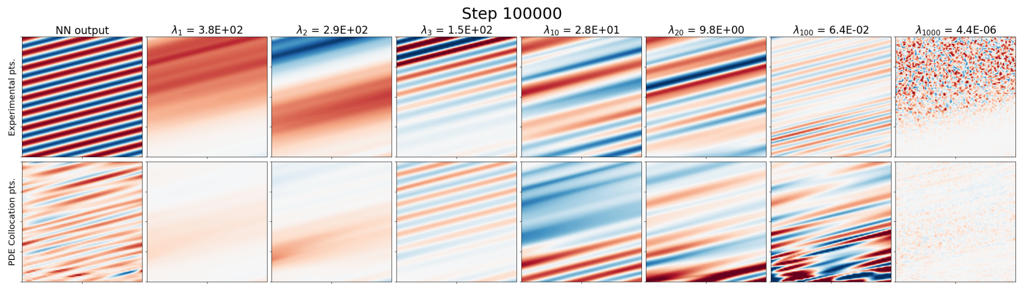
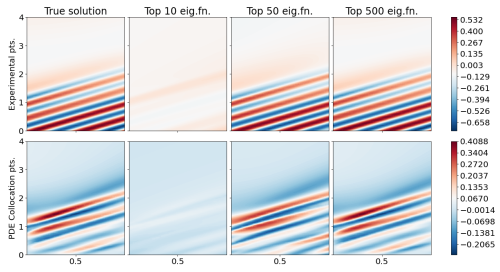


Figure 6: Eigenspectrum of Θ_t at various timesteps.

(a) After 10k steps of GD



(b) After 100k steps of GD

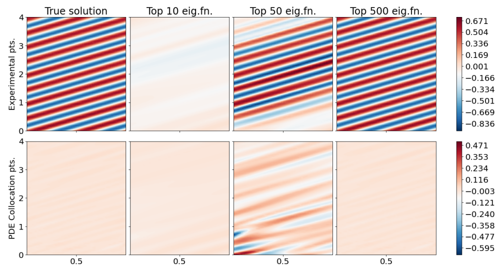


Figure 7: Reconstruction of the NN prediction using the eigenbasis of Θ_t.

set, the quality of this criterion would depend on the chosen set. Instead, to capture the convergence of the whole domain, we consider the function $\Delta R_{\theta_t}(\cdot; Z)$ in the reproducing kernel Hilbert Space (RKHS) $\mathcal{H}_{\Theta_t}$ of Θ_t, with its inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_{\Theta_t}}$ defined such that $\langle \psi_{t,i}, \psi_{t,j} \rangle_{\mathcal{H}_{\Theta_t}} = \delta_{ij}/\lambda_{t,i}$. Ignoring factors of η, we define the *convergence degree* $\alpha(Z)$ to be the RKHS norm of $\Delta R_{\theta_t}(\,\cdot\,; Z)$, or

$$\alpha(Z) \triangleq \|\Delta R_{\theta_t}(\,\cdot\,; Z)\|^2_{\mathcal{H}_{\Theta_t}} = \sum_{i=0}^{\infty} \lambda_{t,i}^{-1} \langle \Delta R_{\theta_t}(\,\cdot\,; Z), \psi_{t,i} \rangle^2_{\mathcal{H}_{\Theta_t}} = \sum_{i=0}^{\infty} \lambda_{t,i} a_{t,i}(Z)^2. \tag{38}$$

Selecting training points that maximizes the convergence degree could speed up training convergence, which corresponds to Process 1 of the PINN training process. Hence, our point selection algorithm will aim to choose a training set Z that maximizes $\alpha(Z)$.

In the remainder of this subsection, we will show that a set which maximizes α will also result in a low generalization error, as stated in Theorem 1. The proofs will follow closely to Shu et al. (2022b). In order to do so, we will first state some assumptions about the PDE being solved and the PINN used for the training process.

Assumption 1 (Stability of PDE) *Suppose we have a differential operator $\mathcal{N} : \mathcal{P} \to \mathcal{Y}$ where $\mathcal{P}$ is a closed subset of some Sobolev space and $\mathcal{Y}$ is a closed subset of some L^p space. For any $u, v \in \mathcal{P}$, let there exists some $C_{\mathrm{pde}} \geq 1$ such that*

$$\|\mathcal{N}[u] - \mathcal{N}[v]\|_{\mathcal{Y}} \leq C_{\mathrm{pde}} \|u - v\|_{\mathcal{P}}. \tag{39}$$

For simplicity, we can view the Sobolev space as a L^p space with a defined derivative up to a certain order. Note that while it is possible for $C_{\mathrm{pde}} < 1$, we assume otherwise for convenience.

We now provide an assumption about the NN we are training with.

Assumption 2 (On the Neural Network Training) *Assume some NN $\hat{u}_\theta : \mathcal{X} \to \mathbb{R}$ with L hidden layers each of width n_L. Suppose there exists constants ρ such that $\|\nabla_\theta \hat{u}_\theta(x)\| \leq \rho$ and $\|\nabla_\theta \mathcal{N}[\hat{u}_\theta](x)\| \leq \rho$. Also let $b_{\min}$ and $b_{\max}$ be the minimum and maximum of the eigenvalues of the eNTK matrix of the possible training set at initialization, i.e. for any training set S, $b_{\max} \geq \lambda_1(\Theta_0(S)) \geq \cdots \geq \lambda_{|S|}(\Theta_0(S)) \geq b_{\min}$.*

We will also make an assumption about the boundedness of the PDE solution u and the NN output $\hat{u}_\theta$. This is done for convenience of the proof – it is possible to add appropriate constants to the subsequent proofs to get a more general result.

Assumption 3 (Boundedness of the NN Residual) *Suppose the PDE solution and the NN output was such that $|R_\theta(x, \mathrm{s})| \leq 1/C_{\mathrm{pde}}$. This would also imply that $|R_\theta(x, \mathrm{p})| \leq 1$.*

We will be working with the hypothesis spaces

$$\mathcal{U} \triangleq \{z \mapsto F[\hat{u}_{\theta_t}](z) : t > 0\} \tag{40}$$

and

$$\mathcal{U}^{\mathrm{lin}} \triangleq \{z \mapsto F[\hat{u}_{\theta_0}](z) + \nabla_\theta F[\hat{u}_{\theta_0}](z)\ (\theta_t - \theta_0) : t > 0\} \tag{41}$$

which is the linearization of the operated neural network $F[\hat{u}_\theta]$ around the initialization given by $F[\hat{u}_\theta](z) \approx F^{\mathrm{lin}}[\hat{u}_\theta](z) = F[\hat{u}_{\theta_0}](z) + \nabla_\theta F[\hat{u}_{\theta_0}](z)\ (\theta_t - \theta_0)$. Note that since F^{lin} performs a Taylor expansion with respect to θ, this means the operator can only be applied on functions approximated by a NN that is parametrized by parameters θ. This is unlike F which is an operator that can be applied on any function. Also note that the linear expansion is performed on $F[\hat{u}_\theta]$ (rather than on just the operand $\hat{u}_\theta$), and hence does not require the assumption that F is a linear operator. We can also define each components of $F^{\mathrm{lin}}[\hat{u}_\theta]$ separately as

$$\hat{u}^{\mathrm{lin}}_\theta(x) \triangleq F^{\mathrm{lin}}[\hat{u}_\theta](x, \mathrm{s}) = \hat{u}_{\theta_0}(x) + \nabla_\theta \hat{u}_{\theta_0}(x)\ (\theta_t - \theta_0), \tag{42}$$

and

$$\mathcal{N}^{\mathrm{lin}}[\hat{u}_\theta](x) \triangleq F^{\mathrm{lin}}[\hat{u}_\theta](x, \mathrm{p}) = \mathcal{N}[\hat{u}_{\theta_0}](x) + \nabla_\theta \mathcal{N}[\hat{u}_{\theta_0}](x)\ (\theta_t - \theta_0). \tag{43}$$

First we will state the result for training the NN where the loss function is based on the linearized operator, or

$$\mathcal{L}^{\text{lin}}(\hat{u}_\theta; Z) = \frac{1}{2}\sum_{z\in Z}\big(F^{\text{lin}}[\hat{u}_\theta](x) - F[u](x)\big)^2. \tag{44}$$

We will show that under the linearized regime, the weights of the NN will not change too much compared to its initialization.

Lemma 1 *Consider a linearized neural network $\hat{u}_\theta^{\text{lin}}$ following Assumption 2 is trained on training set $Z \subset \mathcal{Z}$ with loss in the form* (44) *with GD with learning rate $\eta < 1/\lambda_{\max}(\Theta_0(Z))$. Then, for any $t > 0$,*

$$\|\theta_t - \theta_0\| \leq \|\theta_\infty - \theta_0\| = \sqrt{R_{\theta_0}(Z)^\top \Theta_0(Z)^{-1} R_{\theta_0}(Z)}. \tag{45}$$

Proof Let $R_\theta^{\text{lin}}(z) = F^{\text{lin}}[\hat{u}_\theta](z) - F[u](z)$ be the residual of the linearized operated network. Note that $R_{\theta_0}^{\text{lin}}(z) = R_{\theta_0}(z)$. Based on the update rule of GD, we see that

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta F[\hat{u}_{\theta_0}](Z)^\top R_{\theta_t}^{\text{lin}}(Z) \tag{46}$$

and so

$$\begin{aligned}
F^{\text{lin}}[\hat{u}_{\theta_{t+1}}](Z) &= F[\hat{u}_{\theta_0}](Z) + \nabla_\theta F[\hat{u}_{\theta_0}](Z)(\theta_{t+1} - \theta_0) && (47)\\
&= F[\hat{u}_{\theta_0}](Z) + \nabla_\theta F[\hat{u}_{\theta_0}](Z)(\theta_t - \eta \cdot \nabla_\theta F[\hat{u}_{\theta_0}](Z)^\top R_{\theta_t}^{\text{lin}}(Z) - \theta_0) && (48)\\
&= F^{\text{lin}}[\hat{u}_{\theta_t}](Z) - \eta\Theta_0 R_{\theta_t}^{\text{lin}}(Z) && (49)\\
&= \big(\mathbb{1} - \eta\Theta_0\big) F^{\text{lin}}[\hat{u}_{\theta_t}](Z) + \eta\Theta_0 F[u](Z) && (50)\\
&= \big(\mathbb{1} - \eta\Theta_0\big)^{t+1} R_{\theta_0}^{\text{lin}}(Z) + F[u](Z) && (51)
\end{aligned}$$

where $\mathbb{1}$ is the identity matrix and the shorthand $\Theta_0 = \Theta_0(Z)$. We note that (51) arrives from applying (50) recursively $t+1$ times and using the fact that we set $\eta < 1/\lambda_{\max}(\Theta_0(Z))$ to simplify a corresponding geometric series. This gives

$$\begin{aligned}
\theta_t - \theta_0 &= \sum_{k=0}^{t}(\theta_{k+1} - \theta_k) && (52)\\
&= \eta \cdot \nabla_\theta F[\hat{u}_{\theta_0}](Z)\ \sum_{k=0}^{t} R_{\theta_k}^{\text{lin}}(Z) && (53)\\
&= \eta \cdot \nabla_\theta F[\hat{u}_{\theta_0}](Z)\ \sum_{k=0}^{t} \big(\mathbb{1} - \eta\Theta_0\big)^{t+1} R_{\theta_0}^{\text{lin}}(Z). && (54)
\end{aligned}$$

The remainder of the proof follows directly with Lemma A.5 from Shu et al. (2022b). ■

Corollary 1 *Given theorem 1,*

$$\|\theta_t - \theta_0\| \leq \|R_{\theta_t}(S)\|^2 \cdot \sqrt{\frac{\lambda_{\max}(\Theta_0(S))}{\alpha(S)\cdot\lambda_{\min}(\Theta_0(S))}} \tag{55}$$

Proof Notice that we can rewrite $\alpha(S)$ as

$$\alpha(S) = \left\langle \sum_{i=1}^{\infty} \lambda_{t,i} \langle \psi_{t,i}(S), R_{\theta_t}(S) \rangle \psi_{t,i}(S), \sum_{i=1}^{\infty} \lambda_{t,i} \langle \psi_{t,i}(S), R_{\theta_t}(S) \rangle \psi_{t,i}(S) \right\rangle_{\mathcal{H}_{\Theta_t}} \tag{56}$$

$$= \sum_{i=1}^{\infty} \lambda_{t,i} \langle \psi_{t,i}(S), R_{\theta_t}(S) \rangle^2 \tag{57}$$

$$= R_{\theta_t}(S)^\top \left(\sum_{i=1}^{\infty} \lambda_{t,i} \psi_{t,i}(S) \psi_{t,i}(S)^\top \right) R_{\theta_t}(S) \tag{58}$$

$$= R_{\theta_t}(S)^\top \Theta_t(S) R_{\theta_t}(S) \tag{59}$$

$$= R_{\theta_t}(S)^\top \Theta_0(S) R_{\theta_t}(S) \tag{60}$$

where we use $\Theta_t = \Theta_0$ due to the linearized NN assumption. Let $A = \Theta_0(S)$, $y = R_{\theta_t}(S)$ and $\kappa = \|R_{\theta_t}(S)\|$. Note that A is a PSD matrix, and therefore we can define some $A^{1/2}$ such that $A = (A^{1/2})^\top(A^{1/2})$, and also some $A^{-1/2}$ such that $A^{-1} = (A^{-1/2})^\top(A^{-1/2})$.

$$(y^\top A y)(y^\top A^{-1} y) = \|A^{1/2} y\|_2^2 \cdot \|A^{-1/2} y\|_2^2 \tag{61}$$

$$\leq \kappa^2 \|A^{1/2}\|_2^2 \cdot \kappa^2 \|A^{-1/2}\|_2^2 \tag{62}$$

$$= \kappa^4 \cdot \lambda_{\max}(A^{1/2})^2 \cdot \lambda_{\max}(A^{-1/2})^2 \tag{63}$$

$$= \kappa^4 \cdot \lambda_{\max}(A) \cdot \lambda_{\max}(A^{-1}) \tag{64}$$

$$= \kappa^4 \cdot \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \tag{65}$$

which gives

$$\sqrt{R_{\theta_0}(S)^\top \Theta_0(S)^{-1} R_{\theta_0}(S)} \leq \|R_{\theta_t}(S)\|^2 \cdot \sqrt{\frac{\lambda_{\max}(\Theta_0(S))}{\alpha(S) \cdot \lambda_{\min}(\Theta_0(S))}}. \tag{66}$$

■

We now will link the linearized operated NN to the non-linearized version. We first demonstrate that the difference of the two versions of the NN can be bounded.

Lemma 2 *Suppose we have a neural network $\hat{u}_\theta$ such that Assumption 2 holds. Then, there exists some constants $c, c', c'', N_L, \eta_0 > 0$ such that if $n_L > N_L$, then when applying GD with learning rate $\eta < \eta_0$,*

$$\sup_{t \geq 0} \|\hat{u}_{\theta_t}^{\text{lin}} - \hat{u}_{\theta_t}\| \leq \frac{c}{\sqrt{n_L}} \tag{67}$$

and

$$\sup_{t \geq 0} \|\mathcal{N}^{\text{lin}}[\hat{u}_{\theta_t}] - \mathcal{N}[\hat{u}_{\theta_t}]\| \leq \frac{c \cdot C_{\text{pde}}}{\sqrt{n_L}} + c' \cdot \|R_{\theta_t}(S)\|^2 \cdot \sqrt{\frac{\lambda_{\max}(\Theta_0(S))}{\alpha(S) \cdot \lambda_{\min}(\Theta_0(S))}} \tag{68}$$

with probability at least $1 - \delta$ over the random network initialization.

Proof The first part follows from Theorem H.1 of **?**. The second part follows the fact that

$$
\begin{aligned}
&\|\mathcal{N}^{\text{lin}}[\hat{u}_{\theta_t}](x) - \mathcal{N}[\hat{u}_{\theta_t}](x)]\| && (69)\\
&= \|\mathcal{N}[\hat{u}_{\theta_0}](x) + \nabla_\theta \mathcal{N}[\hat{u}_{\theta_0}](x)^\top(\theta_t - \theta_0) - \mathcal{N}[\hat{u}_{\theta_t}](x)\| && (70)\\
&\le \|\mathcal{N}[\hat{u}_{\theta_0}](x) - \mathcal{N}[\hat{u}_{\theta_t}](x)\| + \|\nabla_\theta \mathcal{N}[\hat{u}_{\theta_0}](x)^\top(\theta_t - \theta_0)\| && (71)\\
&\le C_{\text{pde}}\|\hat{u}_{\theta_0}(x) - \hat{u}_{\theta_t}(x)\| + \|\nabla_\theta \mathcal{N}[\hat{u}_{\theta_0}](x)^\top(\theta_t - \theta_0)\| && (72)\\
&\le C_{\text{pde}}\big(\|\hat{u}_{\theta_0}(x) - \hat{u}^{\text{lin}}_{\theta_t}(x)\| + \|\hat{u}^{\text{lin}}_{\theta_t}(x) - \hat{u}_{\theta_t}(x)\|\big) + \|\nabla_\theta \mathcal{N}[\hat{u}_{\theta_0}](x)^\top(\theta_t - \theta_0)\| && (73)\\
&\le C_{\text{pde}}\big(\|\nabla_\theta \hat{u}_{\theta_0}(x)^\top(\theta_t - \theta_0)\| + \|\hat{u}^{\text{lin}}_{\theta_t}(x) - \hat{u}_{\theta_t}(x)\|\big) + \|\nabla_\theta \mathcal{N}[\hat{u}_{\theta_0}](x)^\top(\theta_t - \theta_0)\| && (74)\\
&\le C_{\text{pde}}\|\nabla_\theta \mathcal{N}[\hat{u}_{\theta_0}](x)\ (\theta_t - \theta_0)\| + \|\nabla_\theta \mathcal{N}[\hat{u}_{\theta_0}](x)^\top(\theta_t - \theta_0)\| + C_{\text{pde}}\|\hat{u}^{\text{lin}}_{\theta_t}(x) - \hat{u}_{\theta_t}(x)\| && (75)\\
&\le \big(C_{\text{pde}}\|\nabla_\theta \mathcal{N}[\hat{u}_{\theta_0}](x)\| + \|\nabla_\theta \mathcal{N}[\hat{u}_{\theta_0}](x)\|\big)\|\theta_t - \theta_0\| + C_{\text{pde}} \cdot \frac{c}{\sqrt{n_L}} && (76)\\
&\le (1 + C_{\text{pde}})\rho\|R_{\theta_t}(S)\|^2 \cdot \sqrt{\frac{\lambda_{\max}(\Theta_0(S))}{\alpha(S)\cdot\lambda_{\min}(\Theta_0(S))}} + C_{\text{pde}} \cdot \frac{c}{\sqrt{n_L}} && (77)
\end{aligned}
$$

which proves the second part of the claim. Note that the bound from (68) could be used to bound (67) as well. ■

We now use results above to compute the Rademacher complexity of $\mathcal{U}$. We first compute the Rademacher complexity of $\mathcal{U}^{\text{lin}}$ in theorem 3, then relate it to the Rademacher complexity of $\mathcal{U}$ in theorem 4. As a reference, the empirical Rademacher of a hypothesis class $\mathcal{G}$ over a set S is given by (Mohri et al., 2018)

$$\hat{\mathcal{R}}_S(\mathcal{G}) \triangleq \mathbb{E}_{\epsilon\in\{\pm1\}^{|S|}}\left[\sup_{g\in\mathcal{G}} \frac{\epsilon^\top g(S)}{|S|}\right]. \tag{78}$$

Lemma 3 *Let $\hat{\mathcal{R}}_S(\mathcal{G})$ be the Rademacher complexity of a hypothesis class $\mathcal{G}$ over some dataset S with size N_S. Then, there exists some $c > 0$ such that with probability at least $1-\delta$ over the random initialization,*

$$\hat{\mathcal{R}}_S(\mathcal{U}^{\text{lin}}) \le \frac{\rho}{N_S^{3/2}} \cdot \|R_{\theta_t}(S)\|^2 \sqrt{\frac{\lambda_{\max}(\Theta_0(S))}{\alpha(S)\cdot\lambda_{\min}(\Theta_0(S))}}. \tag{79}$$

Proof We apply the same technique from Shu et al. (2022b). We see that

$$
\begin{aligned}
\hat{\mathcal{R}}_S(\mathcal{U}^{\text{lin}}) &= \mathbb{E}_{\epsilon\sim\{\pm1\}^{N_S}} \sum_{i=1}^{N_S} \frac{\epsilon_i F[\hat{u}_{\theta_0}](z_i)}{N_S} + \sup_{t\ge0}\left[\mathbb{E}_{\epsilon\sim\{\pm1\}^{N_S}} \sum_{i=1}^{N_S} \frac{\epsilon_i \nabla_\theta F[\hat{u}_{\theta_0}](z_i)\ (\theta_t - \theta_0)}{N_S}\right] && (80)\\
&\le \sup_{t\ge0} \frac{\|\theta_t - \theta_0\|_2 \|\nabla_\theta F[\hat{u}_{\theta_0}](S)\|_2}{N_S} && (81)\\
&\le \frac{1}{N_S} \cdot \|R_{\theta_t}(S)\|^2 \sqrt{\frac{\lambda_{\max}(\Theta_0(S))}{\alpha(S)\cdot\lambda_{\min}(\Theta_0(S))}} \cdot \frac{\rho}{\sqrt{N_S}}. && (82)
\end{aligned}
$$

In the first equality, we use the definition of $\hat{\mathcal{R}}$. The first part of the first equality is independent of t, and will cancel to 0 due to the summation over ϵ. The second inequality then follows from Cauchy-Schwartz inequality. The third inequality uses theorem 1 and the assumption regarding $\|\nabla_\theta F[\hat{u}_{\theta_0}](z)\|$. ■

Lemma 4 *The Rademacher complexity of $\mathcal{U}$ is given by*

$$\hat{\mathcal{R}}_S(\mathcal{U}) \leq \frac{c \cdot C_{\text{pde}}}{\sqrt{n_L}} + \left(\frac{\rho}{N_S^{3/2}} + c'\right) \cdot \|R_{\theta_t}(S)\|^2 \sqrt{\frac{\lambda_{\max}(\Theta_0(S))}{\alpha(S) \cdot \lambda_{\min}(\Theta_0(S))}} \tag{83}$$

for some c and c'.

Proof From theorem 2, we know that there will exist some constant c and c' such that for any $\epsilon_i \in \{\pm 1\}$,

$$\epsilon_i F[\hat{u}_{\theta_t}](z_i) \leq \epsilon_i F^{\text{lin}}[\hat{u}_{\theta_t}](z_i) + \frac{c \cdot C_{\text{pde}}}{\sqrt{n_L}} + c' \cdot \|R_{\theta_t}(S)\|^2 \cdot \sqrt{\frac{\lambda_{\max}(\Theta_0(S))}{\alpha(S) \cdot \lambda_{\min}(\Theta_0(S))}} \tag{84}$$

which gives

$$\hat{\mathcal{R}}_S(\mathcal{U}) \leq \hat{\mathcal{R}}_S(\mathcal{U}^{\text{lin}}) + \frac{c \cdot C_{\text{pde}}}{\sqrt{n_L}} + c' \cdot \|R_{\theta_t}(S)\|^2 \cdot \sqrt{\frac{\lambda_{\max}(\Theta_0(S))}{\alpha(S) \cdot \lambda_{\min}(\Theta_0(S))}}. \tag{85}$$

Applying theorem 3 to the above completes the lemma. ■

We now use the Rademacher complexity to relate back to the overall residual. For simplicity, we let $r(z) = |R_{\theta_\infty}(z)|$ and let $\bar{r}_{\mathcal{D}} = \mathbb{E}_{z\sim\mathcal{D}}[r(z)]$ be the expected absolute residual given test samples drawn from a distribution $\mathcal{D}$.

Lemma 5 *Suppose $S \sim \mathcal{D}_S$ is an i.i.d. sample from $\mathcal{Z}$ according to some distribution $\mathcal{D}_S$ of size N_S. Then, there exists some $c, c', N_L > 0$ such that for $n_L \geq N_L$, with probability at least $1 - \delta$,*

$$\bar{r}_{\mathcal{D}_S} \leq \frac{1}{N_S}\|R_{\theta_\infty}(S)\|_1 + \frac{2cC_{\text{pde}}}{\sqrt{n_L}} + 2\left(\frac{\rho}{N_S^{3/2}} + c'\right)\|R_{\theta_t}(S)\|^2 \sqrt{\frac{\lambda_{\max}(\Theta_0(S))}{\alpha(S) \cdot \lambda_{\min}(\Theta_0(S))}} + 3\sqrt{\frac{\log(2/\delta)}{2N_S}} \tag{86}$$

Proof Note that based on Assumption 3, we have $|R_{\theta_\infty}(z)| \leq 1$. The result then follows from directly applying theorem 4 to Thm. 3.3 from Mohri et al. (2018). ■

We finally prove the required generalization bound.

Theorem 1 (Generalization Bound for PINNs) *Consider a PDE $\mathcal{N}[u,\beta](x) = f(x)$ which follows Assumption 1. Let $\mathcal{Z} = [0,1]^d \times \{\mathrm{s},\mathrm{p}\}$. Let $\hat{u}_\theta$ be a NN which follows Assumption 2. Let the NN residual follows Assumption 3. Let $\mathcal{D}_S$ be a probability distribution over $\mathcal{Z}$ with p.d.f. $p(z)$ such that $\mu\big(\{z \in \mathcal{Z} : p(z) < 1/2\}\big) \leq s$, where $\mu(W)$ is the Lebesgue measure of set W*[7]*. Let $\hat{u}_\theta$ be trained on a dataset $S \subset \mathcal{Z}$, where $S \sim \mathcal{D}_S$ uniformly at random with $|S| = N_S$, by GD with learning rate $\eta \leq 1/\lambda_{\max}(\Theta_0(S))$. Then, there exists some c such that with probability $1 - 2\delta$ over the random model initialization and set S,*

$$\mathbb{E}_{x\sim\mathcal{X}}[|\hat{u}_{\theta_\infty}(x) - u(x)|] \leq c_1 \|R_{\theta_\infty}(S)\|_1 + \frac{c_2}{\sqrt{\alpha(S)}} + c_3, \tag{87}$$

where

$$c_1 = \frac{2}{N_S}, \tag{88}$$

$$c_2 = 4\rho\sqrt{\frac{b_{\max}}{b_{\min}}}\bigg(\frac{1}{N_S^{3/2}} + (1 + C_{\mathrm{pde}})\sqrt{\frac{b_{\max}}{b_{\min}}}\bigg), \tag{89}$$

$$c_3 = \frac{4c \cdot C_{\mathrm{pde}}}{\sqrt{n_L}} + 6\sqrt{\frac{\log(2/\delta)}{2N_S}} + 2s. \tag{90}$$

Proof Let $\mathcal{D}_\mathcal{Z}$ refer to the uniform distribution over $\mathcal{Z}$. The p.d.f. of the uniform distribution q is given by $q(z) = 1/2$. Notice that we can then write

$$\bar{r}_{\mathcal{D}_\mathcal{Z}} = \frac{1}{2}\mathbb{E}_{x\sim\mathcal{X}}[r(x)] + \frac{1}{2}\mathbb{E}_{x\sim\mathcal{X}}[r(x)] \geq \frac{1}{2}\mathbb{E}_{x\sim\mathcal{X}}[r(x)]. \tag{91}$$

Let $\mathcal{Z}_\uparrow = \{z \in \mathcal{Z} : p(z) \geq 1/2\}$ be a set of input with a "high" probability of being chosen by the point selection algorithm, and $\mathcal{Z}_\downarrow = \mathcal{Z} \setminus \mathcal{Z}_\uparrow$. Then, abusing the integral notation to allow summation over discrete spaces, we see that

$$\begin{aligned}
\bar{r}_{\mathcal{D}_\mathcal{Z}} &= \mathbb{E}_{z\sim\mathcal{D}_\mathcal{Z}}[r(z)] && (92)\\
&= \int_{\mathcal{Z}_\uparrow} r(z)dz + \int_{\mathcal{Z}_\downarrow} r(z)dz && (93)\\
&\leq \int_{\mathcal{Z}_\uparrow} r(z)dp(z) + \int_{\mathcal{Z}_\downarrow} r(z)dz && (94)\\
&\leq \int_{\mathcal{Z}} r(z)dp(z) + \int_{\mathcal{Z}_\downarrow} dz && (95)\\
&= \bar{r}_{\mathcal{D}_S} + \mu(\mathcal{Z}_\downarrow) && (96)\\
&\leq \bar{r}_{\mathcal{D}_S} + s && (97)
\end{aligned}$$

and so

$$\mathbb{E}_{x\sim\mathcal{X}}[r(x)] \leq 2(\bar{r}_{\mathcal{D}_S} + s). \tag{98}$$

7. We use a slight abuse of notation to work with the augmented space $\mathcal{Z}$. Given $\mathcal{X}$ is defined as a Cartesian product of intervals, its Lebesgue measure is naturally defined. Then, formally, for $W \subseteq \mathcal{Z}$, we define $\mu(W) = \frac{1}{2}\mu(\{x \in \mathcal{X} : (x,\mathrm{s}) \in W\}) + \frac{1}{2}\mu(\{x \in \mathcal{X} : (x,\mathrm{p}) \in W\})$.

Applying results from theorem 5 to bound $\bar{r}_{\mathcal{D}_S}$ completes the proof. ■

Our derivation takes into account the PINN architecture (which is captured by the eNTK) and the gradient-based training method, unlike past works (Mishra and Molinaro, 2021; De Ryck and Mishra, 2022). The term $\|R_{\theta_\infty}(S)\|_1$ is the training error at convergence of the training set, and can be arbitrarily small (Gao et al., 2023) which we also observe empirically. This leaves the generalization error to be mainly dependent on $\alpha(Z)$. Specifically, the generalization bound is minimized when $\alpha(Z)$ is maximized. This suggests that a point selection algorithm that maximizes the convergence degree should also lead to better generalization error.

E.2 Approximation Error of Criterion

In this section, we want to show that $\hat{\alpha}$ provides a good approximation for α. We will drop the subscript t from the eNTK Θ_t for convenience.

We first make the following observation about the quantity $\hat{\alpha}(Z)$ in (4). The quantity can be rewritten as

$$\hat{\alpha} = \sum_{i=1}^{p} \hat{\lambda}_{t,i}^{-1} \left(\hat{v}_i^\top \Theta(Z_{\text{ref}}, Z) R_{\theta_t}(Z)\right)^2 \tag{99}$$

$$= R_{\theta_t}(Z)^\top \Theta(Z, Z_{\text{ref}}) \left(\sum_{i=1}^{p} \frac{\hat{v}_i \hat{v}_i^\top}{\lambda_{t,i}} \right) \Theta(Z_{\text{ref}}, Z) R_{\theta_t}(Z) \tag{100}$$

$$= R_{\theta_t}(Z)^\top \underbrace{\Theta(Z, Z_{\text{ref}}) \Theta(Z_{\text{ref}})^{-1} \Theta(Z_{\text{ref}}, Z)}_{\approx \Theta(Z)} R_{\theta_t}(Z). \tag{101}$$

Notice that the highlighted part can be seen as a Nystrom approximation of $\Theta(Z)$, where the approximation is done using the inputs Z_{ref}. We will first analyze the highlighted part, by comparing its difference from the full-rank matrix $\Theta(Z)$. This result is a direct consequence of the approximation error bounds from previous works on the Nystrom approximation method.

Proposition 1 *Let N_{pool} be the size of set Z_{pool}. Let $\varepsilon \in (0,1)$ be a number. Suppose we write $\Theta(Z_{\text{pool}}) = U \Lambda U^\top$ using the eigenvalue decomposition such that columns of U are eigenfunctions of $\Theta(Z_{\text{pool}})$ and diagonals of Λ are their corresponding eigenvalues. Let U_k be the first k columns of U, corresponding to the top k eigenvalues of $\Theta(Z_{\text{pool}})$, and let $\tau \triangleq (N_{\text{pool}}/k) \max_i \|(U_k)_i\|^2$ be the coherence of U_k. Let $k \le N_{\text{pool}}$ be some integer. Then, if a set Z_{ref} of size p where*

$$p \ge \frac{2\tau k \log(k/\delta)}{(1-\varepsilon)^2} \tag{102}$$

is sampled uniformly randomly from Z_{pool}, then with probability greater than $1-\delta$, for any $Z \subseteq Z_{\text{pool}}$,

$$\left\|\Theta(Z) - \Theta(Z, Z_{\text{ref}}) \Theta(Z_{\text{ref}})^{-1} \Theta(Z_{\text{ref}}, Z)\right\|_2 \le \lambda_{k+1}(\Theta(Z_{\text{pool}})) \cdot \left(1 + \frac{N_{\text{pool}}}{\varepsilon p}\right) \triangleq \hat{c} \tag{103}$$

where $\lambda_k(\Theta(Z_{\text{pool}}))$ is the kth largest eigenvalue of $\Theta(Z_{\text{pool}})$.

Proof Let $D(Z) = \Theta(Z) - \Theta(Z, Z_{\text{ref}})\Theta(Z_{\text{ref}})^{-1}\Theta(Z_{\text{ref}}, Z)$. Consider the case where $Z = Z_{\text{pool}}$. Then, based directly from Thm. 2 in Gittens (2011), we can show that $\|D(Z)\|_2 \leq \lambda_{\min}(\Theta(Z_{\text{pool}})) \cdot (1 + N_{\text{pool}}/\varepsilon p)$.

To apply this to every Z, note that since $Z \subseteq Z_{\text{pool}}$, it means $D(Z)$ will be a submatrix of $D(Z_{\text{pool}})$, and hence it is the case that $\|D(Z)\|_2 \leq \|D(Z_{\text{pool}})\|_2$, which completes the proof. ■

A few remarks are in order.

Remark 6 *The bound in Proposition 1 depends on the* k*th smallest eigenvalue of* $\Theta(Z_{\text{pool}})$. *From both empirical (Kopitkov and Indelman, 2020; Wang et al., 2022c) and theoretical (Vakili et al., 2021) studies of eNTKs, we find that the eigenvalue of eNTKs tend to decay quickly, meaning as long as our pool* Z_{pool} *is large enough and we let* k *be a large enough value, we would be able to achieve an approximation error which is arbitrarily small.*

Remark 7 *In Proposition 1 (and throughout the paper) we assume that the set* Z_{ref} *is constructed by uniform random sample. As a result, the number of points* c *that the bound requires is higher than what would be needed under other sampling schemes (Gittens and Mahoney, 2013; Wang et al., 2019), which we will not cover here. The version of Proposition 1 only serves to show that as long as we use enough points in* Z_{ref}*, the Nystrom approximation can achieve an arbitrarily small error in the eNTK approximation.*

We now attempt to link the difference in Frobenius norm proven in Proposition 1 to the difference between $\alpha(Z)$ and $\hat{\alpha}(Z)$.

Proposition 2 *Let the conditions in Proposition 1. For a given* $Z \subseteq Z_{\text{pool}}$, $|\alpha(Z) - \hat{\alpha}(Z)| \leq \hat{c}\|R_{\theta_t}(Z)\|_2^2$.

Proof It applies directly from Proposition 1 that

$$\begin{aligned} |\alpha(Z) - \alpha'(Z)| &= \left|R_{\theta_t}(Z)^\top \left(\Theta(Z) - \Theta(Z, Z_{\text{ref}})\Theta(Z_{\text{ref}})^{-1}\Theta(Z_{\text{ref}}, Z)\right) R_{\theta_t}(Z)\right| && (104) \\ &\leq \|R_{\theta_t}(Z)\|_2^2 \left\|\Theta(Z) - \Theta(Z, Z_{\text{ref}})\Theta(Z_{\text{ref}})^{-1}\Theta(Z_{\text{ref}}, Z)\right\|_2^2 && (105) \\ &\leq \hat{c}\|R_{\theta_t}(Z)\|_2^2 && (106) \end{aligned}$$

where the second inequality follows from the definition of Frobenius norm. ■

Appendix F. Extended Discussion on PINNACLE Algorithm

F.1 Discussion of the Point Selection Methods

PINNACLE-S. Theoretically, PINNACLE-S is better motivated than PINNACLE-K. This is because PINNACLE-S involves sampling from an implicit distribution (whose p.d.f. is proportional to $\hat{\alpha}$), and therefore is more directly compatible with assumptions required in Theorem 1, where S is assumed to be i.i.d. samples from some unknown distribution.

Algorithm 1 PINNACLE

```
1: **Input:** PINN $\hat{u}_\theta$, learning rate $\eta$, number of iterations $T$, eNTK approx. error $\delta$.
2: **repeat**
3:     // Point selection phase
4:     Randomly sample candidates $Z_{\text{pool}}$ from $\mathcal{Z}$
5:     Compute $\Theta_t$ using Nystrom approximation
6:     Select subset $Z \subset Z_{\text{pool}}$ to fit constraint using Sampling or K-Means++
7:     // Training phase
8:     Compute $\bar{\Theta} = \Theta_t(Z_{\text{pool}})$
9:     **for** $t' = t, \dots, t+T$ **do**
10:        $\theta_{t'+1} \leftarrow \theta_{t'} - \eta \nabla_\theta \mathcal{L}(\hat{u}_{\theta_{t'}}; Z)$
11:        Exit training if $\|\bar{\Theta} - \Theta_{t'}(Z_{\text{pool}})\| \geq \delta \|\bar{\Theta}\|$
12: **until** training converges or budget exhausted
```

In theory, PINNACLE-S is also sensitive to the ratio of the pool of points. For example, regions with many duplicate points will be more likely to be selected during sampling due to more of such points in the pool. This, however, is an inherent characteristic of many algorithms which selects a batch of points using sampling or greedy selection methods. Despite this, in our experiments we select a pool size that is around $4\times$ as large as the selected points budget in each round, which encourages enough diversity for the algorithm, while still allowing the algorithm to focus in on particular regions that may help with training. Future work could be done to make PINNACLE-S more robust to pool size, through using sampling techniques such as Gibbs sampling which would make the method independent of pool size.

PINNACLE-K. For the K-Means++ sampling mechanism, we use Euclidean distance for the embedded vectors. This is similar to the method used in BADGE Ash et al. (2020). Each dimension of the embedded vector will represent a certain eigenfunction direction (where a larger value represents a stronger alignment in this direction). Therefore, points that are selected in this method will tend to be pointing in different directions (i.e., act on different eigenfunction direction from one another), and also encourage selecting point that have high magnitude (i.e., high change in residual, far away from other points that has less effect on residual change).

F.2 Training Point Retention Mechanism

We briefly describe how training points chosen in previous rounds can be handled during training. While it is possible to retain all of the training points ever selected, this can lead to high computational cost as the training set grows. Furthermore, past results (Wu et al., 2023) have also shown that retaining all of the training point could lead to worsen performances.

Alternatively, we can have a case where none of the training points are retained throughout training. This is done in the case of RAD algorithm (Wu et al., 2023). However, in practice, we find that this can lead to catastrophic forgetting, where the PINN forgets regions where training used to be at. As a result, in PINNACLE, we decide to retain a number of training points from the previous round when choosing the next training set. In Figures 8 and 9, we see that the performance of PINNACLE consistently drops when this mechanism is not used, demonstrating the usefulness of retaining past training points.

F.3 Training Termination via eNTK Discrepancy

In PINNACLE, we also check the eNTK and perform training point selection again when the eNTK changes too much. This corresponds to when the chosen set of training points are no longer optimal given the PINN training dynamics. In Figures 8 and 9, we show that auto-triggering of AL phase can help improve model training in some cases (in particular, for the Advection equation with PINNACLE-K as seen in Figure 9a). However, in practice, we also find that this is often less useful in practice, since as training progresses, the eNTK does not change by much and so the AL process will unlikely be triggered before the maximum training duration T anyway.

F.4 Pseudo-Outputs used for Exp Point Selection

The residual term $R_{\theta_t}(z)$ can be computed exactly in the case for CL points since the PDE and IC/BC conditions are completely known (except for the inverse problem where the PDE constants β will need to be the estimated values at that point in training), and is independent of the true solution. However, in the case of Exp points, $R_{\theta_t}(x)$ will depend on the true output label, which is unknown during training. To solve this problem, we approximate the true solution u by training the NN at the time of the point selection $\hat{u}_{\theta_t}$ for T steps beyond that timestep, and use the trained network $\hat{u}_{\theta_{t+T}}$ as the pseudo-output. This works well in practice as the magnitude of the residual term roughly reflects how much error the model may have at that particular point based on its training dynamics. In our experiments, we set $T = 100$.

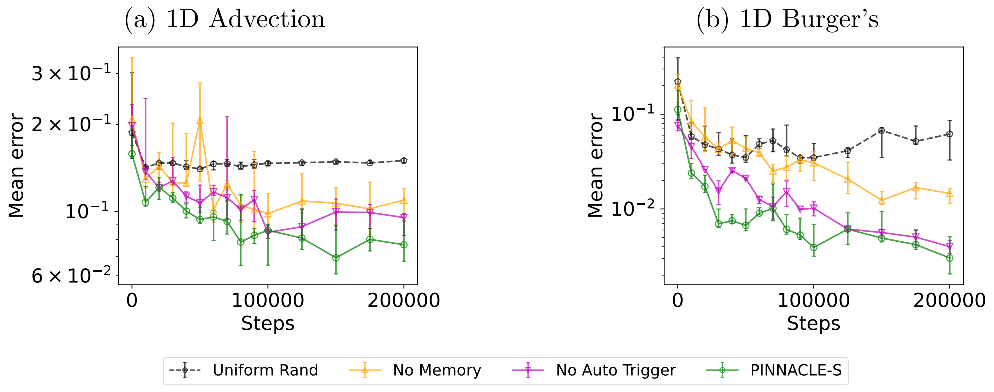


Figure 8: Performance of PINNACLE-S on the various forward problems when different components of the algorithm are removed.

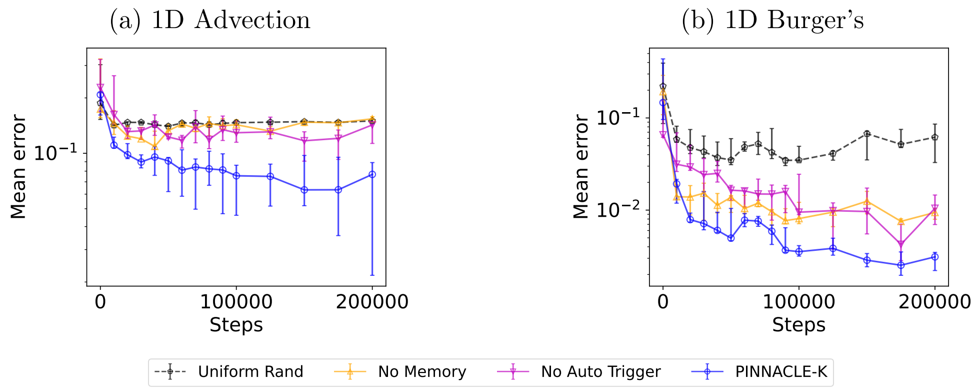


Figure 9: Performance of PINNACLE-K on the various forward problems when different components of the algorithm are removed.

Appendix G. Experimental Setup

G.1 Description of PDEs Used in the Experiments

In this section, we further describe the PDEs used in our benchmarks.

- **1D Advection equation.** The Advection PDE describes the movement of a substance through some fluid. This is a simple linear PDE which is often used to verify performance of PINNs in a simple and intuitive manner due to the easily interpretable solution. The 1D Advection PDE is given by

$$\frac{\partial u}{\partial t} - \beta \frac{\partial u}{\partial x} = 0, \quad x \in (0, L),\ t \in (0, T] \tag{107}$$

  with initial condition $u(x, 0) = u_0(x)$ and periodic BC $u(0, t) = u(L, t)$ is used. The closed form solution is given by $u(x, t) = u_0(x - \beta t)$. To compare with the PDE from (5), we have $\mathcal{N}[u, \beta] = \frac{\partial u}{\partial t} - \beta \frac{\partial u}{\partial x}$ and $f(x, t) = 0$. Meanwhile, for the boundary conditions, the IC can be written as $\mathcal{B}_1[u] = u$ and $g_1(x, t) = u_0(x)$, while the BC can be written as $\mathcal{B}_2[u](x, t) = u(0, t) - u(L, t)$ and $g_2(x, t) = 0$.

  In our forward problem experiments, we use $\beta = 1$ with a irregularly shaped IC from PDEBench (Takamoto et al., 2022). In the timing experiment (in Figure 17a), we use $\beta = 5$ with a sinusoidal initial condition, as past works have highlighted that the 1D Advection problem with high β values is a challenging problem setting (Krishnapriyan et al., 2021), which is possible for the timing experiments given that we provide more CL points to the benchmark algorithms.

- **1D Burger's equation.** Burger's equation is an equation that captures the process of convection and diffusion of fluids. The solution sometimes can exhibit regions of discontinuity and can be difficult to model. The 1D Burger's equation is given by

$$\frac{\partial u}{\partial t} + \frac{1}{2} \frac{\partial u^2}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad x \in (0, L),\ t \in (0, T] \tag{108}$$

with initial condition $u(x,0) = u_0(x)$ and periodic BC $u(0,t) = u(L,t)$ is applied. In our experiments, we used the dataset from Takamoto et al. (2022) with $\nu = 0.02$.

- **2D Navier-Stokes Equation.** This is a PDE with important industry applications which is used to describe motion of a viscous fluid. In this paper, we consider the incompressible variant of the Navier-Stokes equation based on the inverse problem from Raissi et al. (2019), where we consider the fluid flow through a cylinder. The output exhibits a vortex pattern which is periodic in the steady state. The problem consists of the velocities $u(x,y,t)$ and $v(x,y,t)$ in the two spatial directions, and the pressure $p(x,y,t)$. The three functions are related through a system of PDEs given by

$$\frac{\partial u}{\partial t} + \lambda_1\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = -\frac{\partial p}{\partial x} + \lambda_2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{109}$$

$$\frac{\partial v}{\partial t} + \lambda_1\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) = -\frac{\partial p}{\partial y} + \lambda_2\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right). \tag{110}$$

  In the inverse problem experiments, we assume we are able to observe u and v, and our goal is to compute the constants λ_1 and λ_2, as well as the pressure $p(x,y,u)$. There are no IC/BC conditions to follow.

The budgets allocated for each types of problems are as listed below.

Equation (Problem Type)	Training steps	CL points budget	Exp points query rate
1D Advection (Fwd)	200k	1000	-
1D Burger's (Fwd)	200k	300	-
2D Fluid Dynamics (Inv)	100k	1000	30 per 1k steps
1D Advection (FT)	200k	200	-
1D Burger's (FT)	200k	200	-

G.2 Detailed Experimental Setup

All code were implemented in Jax (Bradbury et al., 2018). This is done so due to its efficiency in performing auto-differentiation. To make DeepXDE and PDEBench compatible with Jax, we made modifications to the experimental code. While DeepXDE does have some support for Jax, it remains incomplete for our experiments and we needed to add Jax code components for NN training and training point selection.

In each experiment, we use a multi-layer perceptron with the number of hidden layers and width dependent on the specific setting. Each experiment uses tanh activation, with some also using LAAF (Jagtap et al., 2020b). The models are trained with the Adam optimizer, with learning rate of 10^{-4} for Advection and Burger's equation problem settings, and 10^{-3} for the others. We list the NN architectures used in each of the problem setting below.

Problem	Hidden layers	Hidden width	LAAF?
1D Advection (PDEBench, $\beta = 1$)	8	128	-
1D Advection ($\beta = 5$)	6	64	Yes
1D Burger's	4	128	-
2D Fluid Dynamics	6	64	Yes

The forward and inverse problem experiments in the main paper were repeated 10 rounds each, while the rest of the experiments are repeated 5 rounds each. In each graph shown in this paper, we report the median across these trials, and the error bars report the 20th and 80th percentile values.

For experiments where computational time is not measured, we performed model training on a compute cluster with varying GPU configurations. In the experiment where we measure the algorithms' runtime in forward problem (i.e., experiments ran corresponding to Figure 17a), we have trained each NN using one NVIDIA GeForce RTX 3080 GPU, and with Intel Xeon Gold 6326 CPU @ 2.90GHz, while for the inverse problem (i.e., experiments ran corresponding to Figure 17b), we trained each NN using one NVIDIA RTX A5000 GPU and with AMD EPYC 7543 32-Core Processor CPU.

G.3 Description of Benchmark Algorithms

Below we list some of the algorithms that we ran experiments on. Note that due to space constraints, some of these experiments are not presented in the main paper, but only in the appendix.

We first list the non-adaptive point selection methods. All of these methods were designed to only deal with the selection of PDE CL points – hence IC/BC CL points and Exp points are all selected uniformly at random from their respective domains. The empirical performance of these methods have also been studied in Wu et al. (2023)

- Uniform Random. Points are selected uniformly at random.
- Hammersley. Points are generated based on the Hammersley sequence.
- Sobol. Points are generated based on the Sobol sequence.

The latter two non-adaptive point selection methods are said to be *low-discrepancy sequences*, which are often preferred over purely uniform random points since they will be able to better span the domain of interest. In the main paper we only report the results for Hammersley since it is the best pseudo-random point generation sequence as experimented by Wu et al. (2023).

We now list the adaptive point selection methods that we have used.

- RAD (Wu et al., 2023). The PDE CL points are sampled such that each point has probability of being sampled proportional to the PDE residual squared. The remaining CL points are sampled uniformly at random. This is the variant of the adaptive point selection algorithm reported by (Wu et al., 2023) which performs the best.
- RAD-All. This is the method that we extended from RAD, where the IC/BC CL points are also selected such that they are also sampled at probability proportional

to the square of the residual. Exp points are selected using the same technique described in Appendix F.4, which is comparable to expected model output change (EMOC) criterion used in active learning (Ranganathan et al., 2017).

- PINNACLE-S and PINNACLE-K. This is our proposed algorithm as described in Sec. 2.

In cases where there are multiple types of CL points, for all of the methods that do not use dynamic allocation of CL point budget, we run the experiments where we fix the PDE CL points at 0.5, 0.8 or 0.95 of the total CL points budget, and divide the remaining budget equally amongst the IC/BC CL points. In the main paper, the presented results are for when the ratio is fixed to 0.8.

Appendix H. Additional Experimental Details and Results

Dataset. For comparability with other works, we conducted experiments with open-sourced data (Takamoto et al., 2022; Lu et al., 2021), and experimental setups that matches past work (Raissi et al., 2019). The specific PDEs studied and details are in Appendix G.1.

NN Architecture and Implementation. Similar to other PINNs work, we used fully-connected NNs of varying NN depth and width with tanh activation, both with and without LAAF (Jagtap et al., 2020a). We adapted the DeepXDE framework (Lu et al., 2021) for training PINNs and handling input data, by implementing it in Jax (Bradbury et al., 2018) for more efficient computation.

Algorithms Benchmarked. We benchmarked PINNACLE with a suite of non-adaptive and adaptive CL point selection methods. For our main results, apart from a baseline that randomly selects training points, we compare PINNACLE with the Hammersley sequence sampler and RAD algorithm, the best performing non-adaptive and adaptive PDE CL point selection method based on (Wu et al., 2023). We also extended RAD, named RAD-All, where IC/BC points are also adaptively sampled in the same way as the PDE CL points. We provide further details and results in Appendix G.3.

H.1 Point Selection Dynamics of PINNACLE

In this section, we further consider the dynamics of the point selection process between different algorithms. Figure 10, we plot the point selection dynamics of various point selection algorithms in the 1D Advection equation problem. We can see that Hammersley will select points which best spans the overall space, however this is does not necessarily correspond to points which helps training the best. Meanwhile, Rad-All tends to focus its points on the non-smooth stripes of the solution, which corresponds to regions of higher residual. While this allows Rad-All to gradually build up the stripes, it does so in a less efficient manner than PINNACLE. In the variants of PINNACLE, we can see that the selected CL points gradually build upwards, which allows the stripe to be learnt more efficiently. We can also see that PINNACLE-K selects points that are more diverse than PINNACLE-S, which can be seen from the larger spread of CL points throughout training, with PDE CL points even attempting to avoid the initial boundary in Step 200k.

In Figure 11, we plot the point selection dynamics of various point selection algorithms in the 1D Burger's equation problem. We can see that Hammersley peforms similar as in the

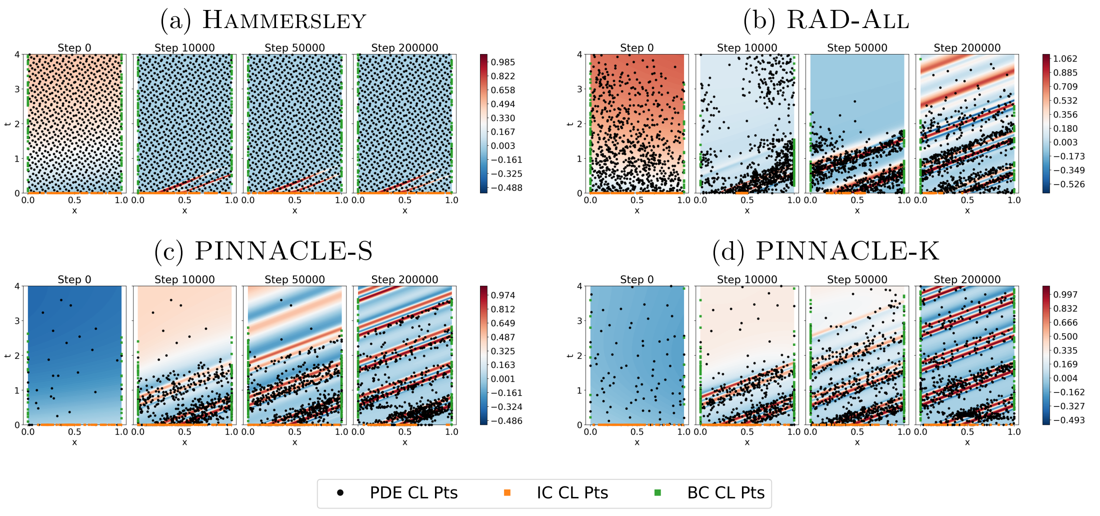


Figure 10: Training point selection dynamics for PINNACLE in the 1D Advection equation forward problem experiments.

Advection case, where it is able to span the space well but not able to guide model training well. Meanwhile, Rad-All selects points that are more spread out (that has high residual), but focuses less effort in regions near to the initial boundary or along the discontinuity in the solution, unlike PINNACLE. The points selection being less concentrated along these regions may explain why PINNACLE is able to achieve a better solution overall.

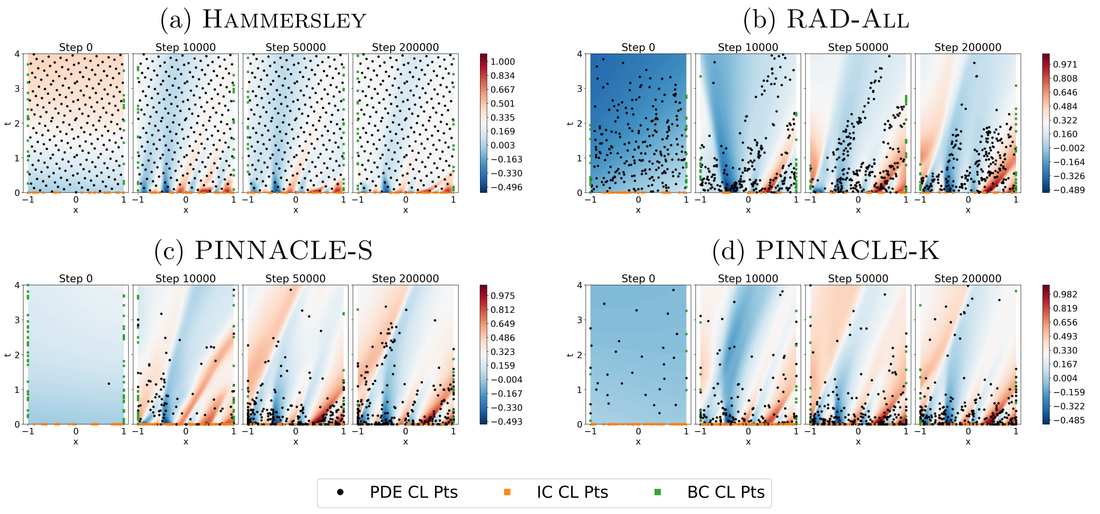


Figure 11: Training point selection dynamics for PINNACLE in the 1D Burger's equation forward problem experiments.

H.2 Forward Problems

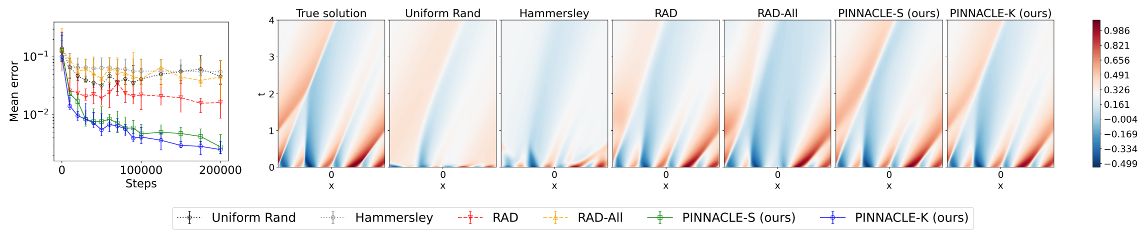


Figure 12: Results from the 1D Burger's equation forward problem experiments. In each row, from left to right: plot of mean prediction error for each method, the true PDE solution, and the PINN output from different point selection methods.

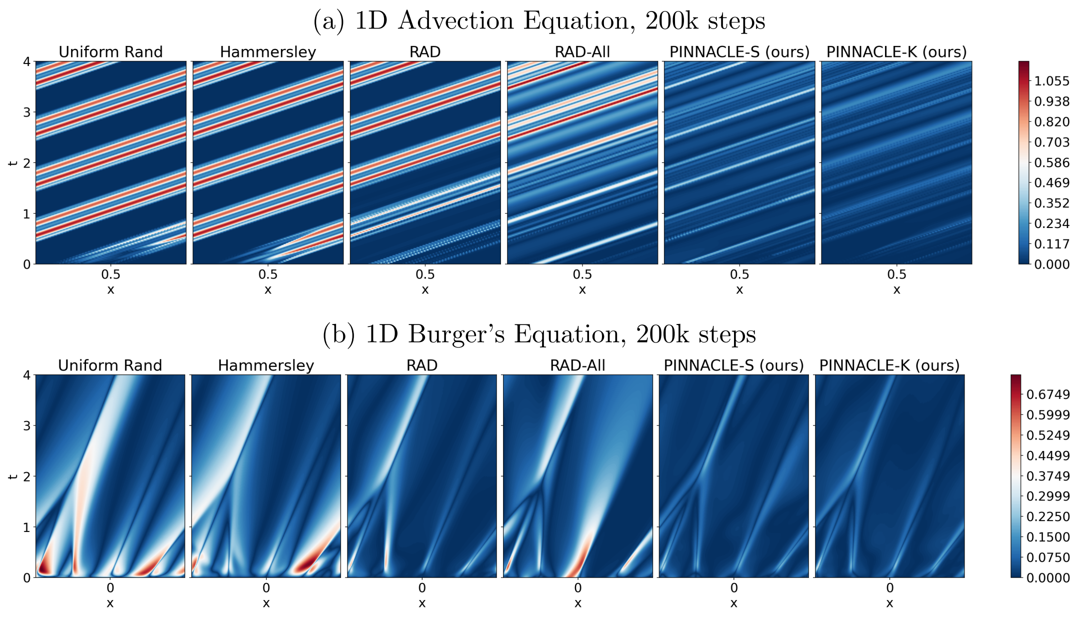


Figure 13: Predictive error for the forward problem experiments in **??**.

We plot the predictive error for experiments ran in Figures 2 and 12 in Figure 13. Additionally, we plot the 1D Advection and 1D Burger's equation experiments with additional point selection algorithms in Figures 14 and 15 respectively. Some observations can be made from these data. First, adaptive point selection methods outperform non-adaptive point selection methods. This is to be expected since adaptive methods allow for training points to be adjusted according to the current solution landscape, and respond better to the current training dynamics.

Second, for all the algorithms which do not perform dynamic CL point budget allocation, choosing whatever fixed budget at the beginning makes little difference in the training outcome. This can be seen that in each experiment, choosing the point allocation at 0.5, 0.8 or 0.95 shows little difference in performance, and is unable to outperform PINNACLE.

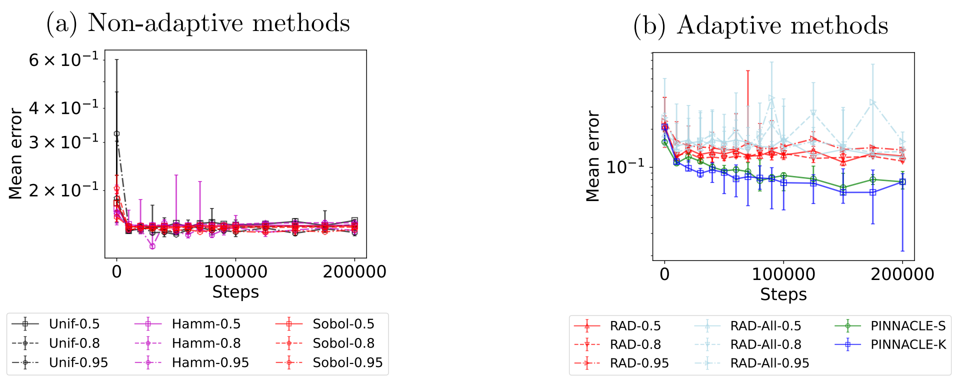


Figure 14: Further experimental results for 1D Advection equation forward problem.

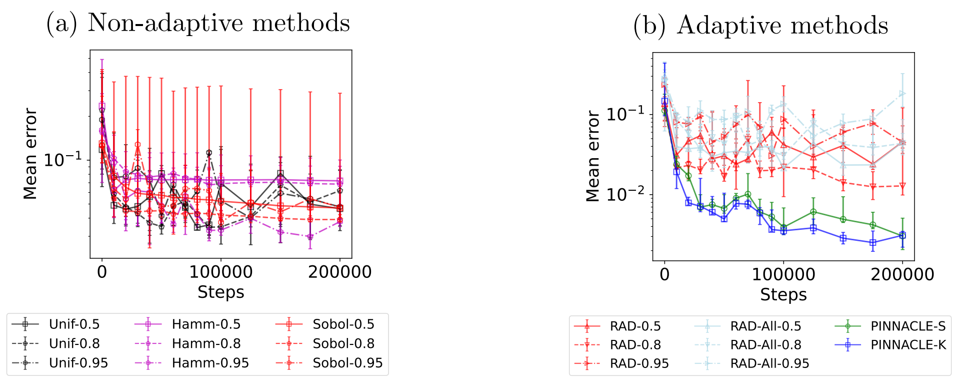


Figure 15: Further experimental results for 1D Burger's equation forward problem.

Meanwhile, the variants of PINNACLE are able to exploit the dynamic budgeting to prioritize the correct types of CL points in different stages of training.

Third, while PINNACLE-K outperforms PINNACLE-S, it is often not by a large margin. This is an interesting result, since it may suggest that selecting a diverse set of points may not be required for training a PINN well.

Additionally, in Figure 17a, we also measure the runtime of different algorithms to train a 1D Advection problem with $\beta = 5$ until it reaches a mean loss of 10^{-2}. We select this particular forward problem since it is known that Advection equation with a high β is a difficult problem for PINN training (Krishnapriyan et al., 2021). We can see that while some algorithms are able to converge with higher number of training points, they will end up also incurring a larger runtime due to the higher number of training points. Meanwhile, PINNACLE-K is able to achieve a low test error with much fewer points than pseudo-random sampling at a much quicker pace. This shows that while it is possible to train PINNs with more CL points, this often come with computation cost of having to compute the gradient of more training data. PINNACLE provides a good way to reduce the number of training points needed for training, which reduces this training cost problem.

H.3 Inverse Problems

In Figure 16, we present further experimental results for the Navier-Stokes equation experiments. We see that for the unseen components of the PDE solution (i.e., p, λ_1 and λ_2), PINNACLE is able to get much lower error values than the other algorithms (with exception of PINNACLE-K which is unable to recover p as accurately). This shows that the point selection method is able to select points to obtain the most information. For the prediction alone, all algorithms are able to perform about the same since they are able to get the information from the experimental points and interpolate accordingly. For the unknown quantities, however, a simple point selection mechanism fails whereas PINNACLE is able to perform better.

Furthermore, in Figure 17b, we measure the running time for the Navier-Stokes inverse problem where we aim to compute the PDE constants for the Navier-Stokes equation, and measure the time required until the estimated PDE constants are within 5% of the true value. We can see that using 10× fewer points, PINNACLE-K outperforms the pseudo-random benchmark by being able to achieve the correct answer at a faster rate. Furthermore, since PINNACLE-K is able to converge with fewer steps, it means that in reality it would be much more efficient also since it requires less query for the actual solution, which saves experimental costs.

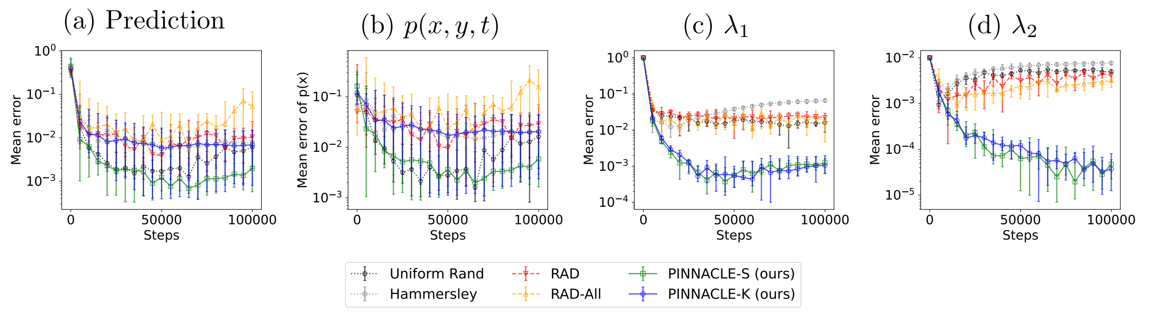


Figure 16: Errors of predicted value from the Navier-Stokes equation experiments. The graphs show, from left to right, the errors of overall prediction, the pressure p, λ_1, and λ_2.

H.4 Transfer Learning of PINNs

In Figure 18, we compare the point selection dynamics for each point selection algorithms for the experiments from Figure 4. We can see that for PINNACLE, the CL points are selected in such a way that the existing solution structure still remains throughout training. This makes PINNACLE more efficient for transfer learning since it does not have to relearn the solution again from scratch. This is in contrast with the other methods, where the stripes eventually gets unlearned which makes the training inefficient.

In Figure 19, we present more results from the transfer learning experiments for the Burger's equation. As we can see, again, the variants of PINNACLE are able to better reconstruct the solution with the alternate IC than other methods.

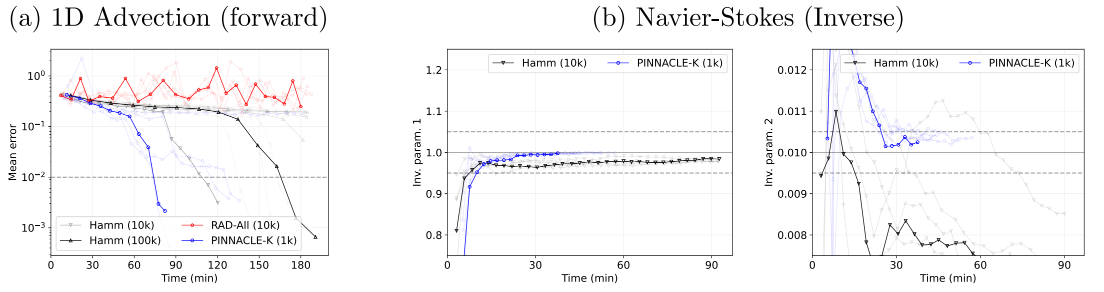


Figure 17: Experiments involving runtime of various point selection algorithms. The highlighted lines in each plot represent the example which is able to achieve the required accuracy and sustain the accuracy value for 5k additional steps within the shortest time. In each example, the dotted lines represent the threshold considered for acceptable accuracy (for forward problem, this marks the 10^{-2} accuracy point, while for inverse problem, this marks the region where the estimated constant are within 5% of the true value.

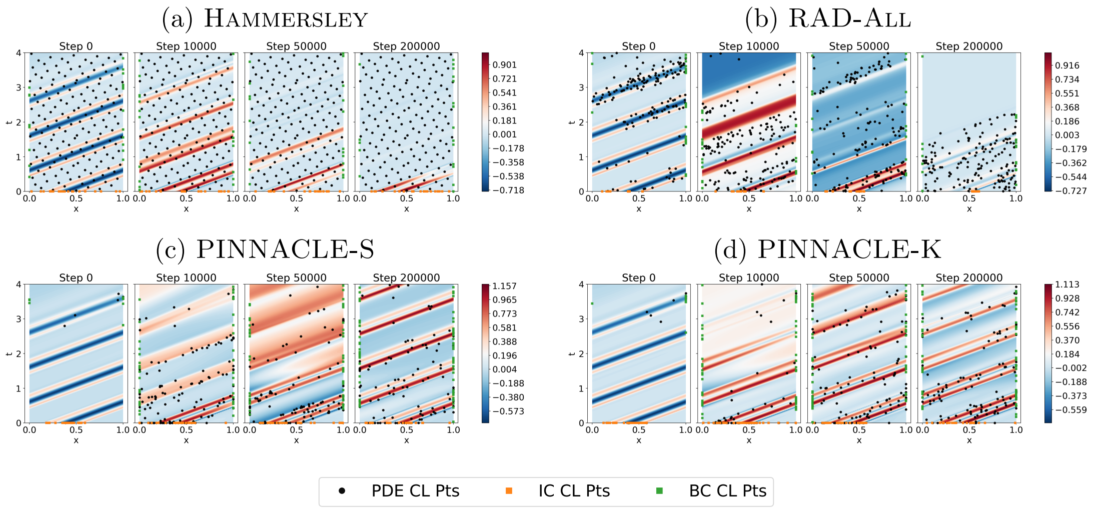


Figure 18: Training point selection dynamics for PINNACLE in the 1D Advection equation transfer learning problem experiments.

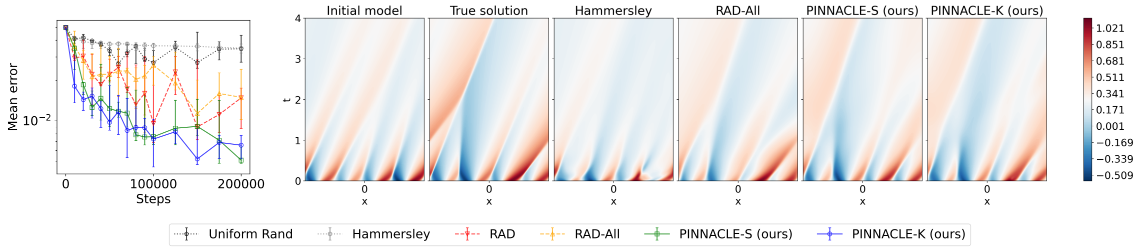


Figure 19: Predictive error for the 1D Burger's equation transfer learning problem with perturbed IC.