Challenges in Training PINNs: A Loss Landscape Perspective

Pratik Rathore [1] **Weimu Lei** [2] **Zachary Frangella** [3] **Lu Lu** [4] **Madeleine Udell** [2 3]

Abstract

This paper explores challenges in training Physics-Informed Neural Networks (PINNs), emphasizing the role of the loss landscape in the training process. We examine difficulties in minimizing the PINN loss function, particularly due to ill-conditioning caused by differential operators in the residual term. We compare gradient-based optimizers Adam, L-BFGS, and their combination Adam+L-BFGS, showing the superiority of Adam+L-BFGS, and introduce a novel second-order optimizer, NysNewton-CG (NNCG), which significantly improves PINN performance. Theoretically, our work elucidates the connection between ill-conditioned differential operators and ill-conditioning in the PINN loss and shows the benefits of combining first- and second-order optimization methods. Our work presents valuable insights and more powerful optimization strategies for training PINNs, which could improve the utility of PINNs for solving difficult partial differential equations.

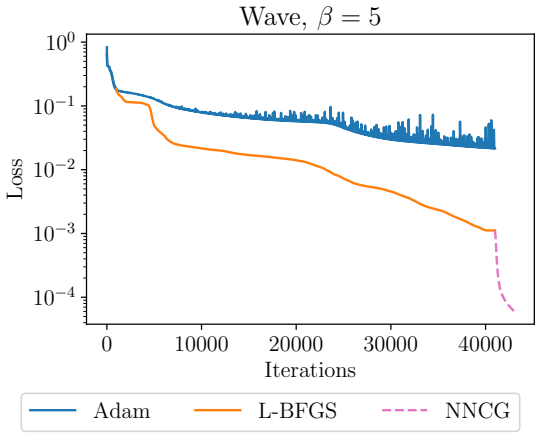

Figure 1. On the wave PDE, Adam converges slowly due to ill-conditioning and the combined Adam+L-BFGS optimizer stalls after about 40000 steps. Running NNCG (our method) after Adam+L-BFGS provides further improvement.

1. Introduction

The study of Partial Differential Equations (PDEs) grounds a wide variety of scientific and engineering fields, yet these fundamental physical equations are often difficult to solve numerically. Recently, neural network-based approaches including physics-informed neural networks (PINNs) have shown promise to solve both forward and inverse problems involving PDEs (Raissi et al., 2019; E & Yu, 2018; Lu et al., 2021a;b; Karniadakis et al., 2021; Cuomo et al., 2022). PINNs parameterize the solution to a PDE with a neural network, and are often fit by minimizing a least-squares loss involving the PDE residual, boundary condition(s), and initial condition(s). The promise of PINNs is the potential to obtain solutions to PDEs without discretizing or meshing the space, enabling scalable solutions to high-dimensional problems that currently require weeks on advanced supercomputers. This loss is typically minimized with gradient-based optimizers such as Adam (Kingma & Ba, 2014), L-BFGS (Liu & Nocedal, 1989), or a combination of both.

However, the challenge of optimizing PINNs restricts the application and development of these methods. Previous work has shown that the PINN loss is difficult to minimize (Krishnapriyan et al., 2021; Wang et al., 2021a; 2022b; De Ryck et al., 2023) even in simple settings. As a result, the PINN often fails to learn the solution. Furthermore, optimization challenges can obscure the effectiveness of new neural network architectures for PINNs, as an apparently inferior performance may stem from insufficient loss function optimization rather than inherent limitations of an architecture. A simple, reliable training paradigm is critical to enable wider adoption of PINNs.

This work explores the loss landscape of PINNs and the challenges this landscape poses for gradient-based optimization methods. We provide insights from optimization theory

[1]Department of Electrical Engineering, Stanford University, Stanford, CA, USA [2]ICME, Stanford University, Stanford, CA, USA [3]Department of Management Science & Engineering, Stanford University, Stanford, CA, USA [4]Department of Statistics and Data Science, Yale University, New Haven, CT, USA. Correspondence to: Pratik Rathore <pratikr@stanford.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

that explain slow convergence of first-order methods such as Adam and show how ill-conditioned differential operators make optimization difficult. We also use our theoretical insights to improve the PINN training pipeline by combining existing and new optimization methods.

The most closely related works to ours are Krishnapriyan et al. (2021); De Ryck et al. (2023), which both identify ill-conditioning in the PINN loss. Unlike Krishnapriyan et al. (2021), we empirically confirm the ill-conditioning of the loss by visualizing the spectrum of the Hessian and demonstrating how quasi-Newton methods improve the conditioning. Our theoretical results directly show how an ill-conditioned linear operator induces an ill-conditioned objective, in contrast to the approach in De Ryck et al. (2023) which relies on a linearization.

Contributions. We highlight contributions of this paper:

- We demonstrate that the loss landscape of PINNs is ill-conditioned due to differential operators in the residual term and show that quasi-Newton methods improve the conditioning by $1000\times$ or more (Section 5).
- We compare three optimizers frequently used for training PINNs: (i) Adam, (ii) L-BFGS, and (iii) Adam followed by L-BFGS (referred to as Adam+L-BFGS). We show that Adam+L-BFGS is superior across a variety of network sizes (Section 6).
- We show the PINN solution resembles the true PDE solution only for extremely small loss values (Section 4). However, we find that the loss returned by Adam+L-BFGS can be improved further, which also improves the PINN solution (Section 7).
- Motivated by the ill-conditioned loss landscape, we introduce a novel second-order optimizer, NysNewton-CG (NNCG). We show NNCG can significantly improve the solution returned by Adam+L-BFGS (Figure 1 and Section 7).
- We prove that ill-conditioned differential operators lead to an ill-conditioned PINN loss (Section 8). We also prove that combining first- and second-order methods (e.g., Adam+L-BFGS) leads to fast convergence, providing justification for the importance of the combined method (Section 8).

Notation. We denote the Euclidean norm by $\|\cdot\|_2$ and use $\|M\|$ to denote the operator norm of $M \in \mathbb{R}^{m\times n}$. For a smooth function $f : \mathbb{R}^p \to \mathbb{R}$, we denote its gradient at $w \in \mathbb{R}^p$ by $\nabla f(w)$ and its Hessian by $H_f(w)$. We write $\partial_{w_i} f$ for $\partial f/\partial w_i$. For $\Omega \subset \mathbb{R}^d$, we denote its boundary by $\partial\Omega$. For any $m \in \mathbb{N}$, we use I_m to denote the $m\times m$ identity matrix. Finally, we use $\preceq$ to denote the Loewner ordering on the convex cone of positive semidefinite matrices.

2. Problem Setup

This section introduces physics-informed neural networks as optimization problems and our experimental methodology.

2.1. Physics-informed Neural Networks

The goal of physics-informed neural networks is to solve partial differential equations. Similar to prior work (Lu et al., 2021b; Hao et al., 2023), we consider the following system of partial differential equations:

$$\mathcal{D}[u(x), x] = 0, \quad x \in \Omega, \tag{1a}$$

$$\mathcal{B}[u(x), x] = 0, \quad x \in \partial\Omega, \tag{1b}$$

where $\mathcal{D}$ is a differential operator defining the PDE, $\mathcal{B}$ is an operator associated with the boundary and/or initial conditions, and $\Omega \subseteq \mathbb{R}^d$. To solve (1), PINNs model u as a neural network $u(x; w)$ (often a multi-layer perceptron (MLP)) and approximate the true solution by the network whose weights solve the following non-linear least-squares problem:

$$\underset{w\in\mathbb{R}^p}{\text{minimize}}\ L(w) := \frac{1}{2n_{\text{res}}}\sum_{i=1}^{n_{\text{res}}}\left(\mathcal{D}[u(x_r^i; w), x_r^i]\right)^2 + \frac{1}{2n_{\text{bc}}}\sum_{i=1}^{n_{\text{bc}}}\left(\mathcal{B}[u(x_b^j; w), x_b^j]\right)^2. \tag{2}$$

Here $\{x_r^i\}_{i=1}^{n_{\text{res}}}$ are the residual points and $\{x_b^j\}_{j=1}^{n_{\text{bc}}}$ are the boundary/initial points. The first loss term measures how much $u(x; w)$ fails to satisfy the PDE, while the second term measures how much $u(x; w)$ fails to satisfy the boundary/initial conditions.

For this loss, $L(w) = 0$ means that $u(x; w)$ exactly satisfies the PDE and boundary/initial conditions at the training points. In deep learning, this condition is called *interpolation* (Zhang et al., 2021; Belkin, 2021). There is no noise in (1), so the true solution of the PDE would make (2) equal to zero. Hence a PINN approach should choose an architecture and an optimizer to achieve interpolation. Moreover, smaller training error corresponds to better generalization for PINNs (Mishra & Molinaro, 2023). Common optimizers for (2) include Adam, L-BFGS, and Adam+L-BFGS (Raissi et al., 2019; Krishnapriyan et al., 2021; Hao et al., 2023).

2.2. Experimental Methodology

We conduct experiments on optimizing PINNs for convection, wave PDEs, and a reaction ODE. These equations have been studied in previous works investigating difficulties in training PINNs; we use the formulations in Krishnapriyan et al. (2021); Wang et al. (2022b) for our experiments. The coefficient settings we use for these equations are considered challenging in the literature (Krishnapriyan et al., 2021; Wang et al., 2022b). Appendix A contains additional details.

We compare the performance of Adam, L-BFGS, and Adam+L-BFGS on training PINNs for all three classes of PDEs. For Adam, we tune the learning rate by a grid search on $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. For L-BFGS, we use the default learning rate 1.0, memory size 100, and strong Wolfe line search. For Adam+L-BFGS, we tune the learning rate for Adam as before, and also vary the switch from Adam to L-BFGS (after 1000, 11000, 31000 iterations). These correspond to Adam+L-BFGS (1k), Adam+L-BFGS (11k), and Adam+L-BFGS (31k) in our figures. All three methods are run for a total of 41000 iterations.

We use multilayer perceptrons (MLPs) with tanh activations and three hidden layers. These MLPs have widths 50, 100, 200, or 400. We initialize these networks with the Xavier normal initialization (Glorot & Bengio, 2010) and all biases equal to zero. Each combination of PDE, optimizer, and MLP architecture is run with 5 random seeds.

We use 10000 residual points randomly sampled from a 255×100 grid on the interior of the problem domain. We use 257 equally spaced points for the initial conditions and 101 equally spaced points for each boundary condition.

We assess the discrepancy between the PINN solution and the ground truth using ℓ_2 relative error (L2RE), a standard metric in the PINN literature. Let $y = (y_i)_{i=1}^n$ be the PINN prediction and $y' = (y'_i)_{i=1}^n$ the ground truth. Define

$$\text{L2RE} = \sqrt{\frac{\sum_{i=1}^n (y_i - y'_i)^2}{\sum_{i=1}^n y_i'^2}} = \sqrt{\frac{\|y - y'\|_2^2}{\|y'\|_2^2}}.$$

We compute the L2RE using all points in the 255×100 grid on the interior of the problem domain, along with the 257 and 101 points used for the initial and boundary conditions.

We develop our experiments in PyTorch 2.0.0 (Paszke et al., 2019) with Python 3.10.12. Each experiment is run on a single NVIDIA Titan V GPU using CUDA 11.8. The code for our experiments is available at https://github.com/pratikrathore8/opt_for_pinns.

3. Related Work

Here we review common approaches for solving PDEs with physics-informed machine learning and PINN training strategies proposed in the literature.

3.1. Physics-informed ML for Solving PDEs

A variety of ML-based methods for solving PDEs have been proposed, including PINNs (Raissi et al., 2019), the Fourier Neural Operator (FNO) (Li et al., 2021), and DeepONet (Lu et al., 2021a). The PINN approach solves the PDE by using the loss function to penalize deviations from the PDE residual, boundary, and initial conditions. Notably, PINNs do not require knowledge of the solution to solve the forward PDE problem. On the other hand, the FNO and DeepONet sample and learn from known solutions to a parameterized class of PDEs to solve PDEs with another fixed value of the parameter. However, these operator learning approaches may not produce predictions consistent with the underlying physical laws that produced the data, which has led to the development of hybrid approaches such as physics-informed DeepONet (Wang et al., 2021c). Our theory shows that the ill-conditioning issues we study in PINNs are unavoidable for any ML-based approach that penalizes deviations from the known physical laws.

3.2. Challenges in Training PINNs

The vanilla PINN (Raissi et al., 2019) can perform poorly when trying to solve high-dimensional, non-linear, and/or multi-scale PDEs. Researchers have proposed a variety of modifications to the vanilla PINN to address these issues, many of which attempt to make the optimization problem easier to solve. Wang et al. (2021a; 2022a;b); Nabian et al. (2021); Wu et al. (2023a;b) propose loss reweighting/resampling to balance different components of the loss, Yao et al. (2023); Müller & Zeinhofer (2023) propose scale-invariant and natural gradient-based optimizers for PINN training, Jagtap et al. (2020a;b); Wang et al. (2023) propose adaptive activation functions which can accelerate convergence of the optimizer, and Liu et al. (2024) propose an approach to precondition the PINN loss itself. Other approaches include innovative loss functions and regularizations (E & Yu, 2018; Lu et al., 2021c; Kharazmi et al., 2021; Khodayi-Mehr & Zavlanos, 2020; Yu et al., 2022) and new architectures (Jagtap et al., 2020c; Jagtap & Karniadakis, 2020; Li et al., 2020; Moseley et al., 2023). These strategies work with varying degrees of success, and no single strategy improves performance across all PDEs.

Our work attempts to understand and tame the ill-conditioning in the (vanilla) PINN loss directly. We expect our ideas to work well with many of the above training strategies for PINNs; none of these training strategies rid the objective of the differential operator that generates the ill-conditioning in the PINN loss (with the possible exception of Liu et al. (2024)). However, Liu et al. (2024) preconditions the PINN loss directly, which is equivalent to left preconditioning, while our work studies the effects of preconditioned optimization methods on the PINN loss, which is equivalent to right preconditioning (Appendix C.1). There is potential in combining the approach of Liu et al. (2024) and our approach to obtain a more reliable framework for training PINNs.

Our work analyzes the spectrum (eigenvalues) of the Hessian H_L of the loss. Previous work (Wang et al., 2022b) studies the conditioning of the loss using the neural tangent kernel (NTK), which requires an infinite-width assumption

on the neural network; our work studies the conditioning of the loss through the lens of the Hessian and yields useful results for finite-width PINN architectures. Several works have also studied the *spectral bias* of PINNs (Wang et al., 2021b; 2022b; Moseley et al., 2023), which refers to the inability of neural networks to learn high-frequency functions. Note that our paper uses the word spectrum to refer to the Hessian eigenvalues, not the spectrum of the PDE solution.

4. Good Solutions Require Near-zero Loss

First, we show that PINNs must be trained to near-zero loss to obtain a reasonably low L2RE. This phenomenon can be observed in Figure 2, demonstrating that a lower loss generally corresponds to a lower L2RE. For example, on the convection PDE, a loss of 10^{-3} yields an L2RE around 10^{-1}, but decreasing the loss by a factor of 100 to 10^{-5} yields an L2RE around 10^{-2}, a $10\times$ improvement. This relationship between loss and L2RE in Figure 2 is typical of many PDEs (Lu et al., 2022).

The relationship in Figure 2 underscores that high-accuracy optimization is required for a useful PINN. There are instances (especially on the reaction ODE), where the PINN solution has a L2RE around 1, despite a near-zero loss; we provide insight into why this is occurring in Appendix B. In Sections 5 and 7, we show that ill-conditioning and under-optimization make reaching a solution with sufficient accuracy difficult.

5. The Loss Landscape is Ill-conditioned

We show empirically that the ill-conditioning of the PINN loss is mainly due to the residual loss, which contains the differential operator. We also show that quasi-Newton methods like L-BFGS improve the conditioning of the problem.

5.1. The PINN Loss is Ill-conditioned

The conditioning of the loss L plays a key role in the performance of first-order optimization methods (Nesterov, 2018). We can understand the conditioning of an optimization problem through the eigenvalues of the Hessian of the loss, H_L. Intuitively, the eigenvalues of H_L provide information about the local curvature of the loss function at a given point along different directions. The condition number is defined as the ratio of the largest magnitude's eigenvalue to the smallest magnitude's eigenvalue. A large condition number implies the loss is very steep in some directions and flat in others, making it difficult for first-order methods to make sufficient progress toward the minimum. When $H_L(w)$ has a large condition number (particularly, for w near the optimum), the loss L is called *ill-conditioned*. For example, the convergence rate of gradient descent (GD) depends on the condition number (Nesterov, 2018), which results in GD converging slowly on ill-conditioned problems.

To investigate the conditioning of the PINN loss L, we would like to examine the eigenvalues of the Hessian. For large matrices, it is convenient to visualize the set of eigenvalues via *spectral density*, which approximates the distribution of the eigenvalues. Fast approximation methods for the spectral density of the Hessian are available for deep neural networks (Ghorbani et al., 2019; Yao et al., 2020). Figure 3 shows the estimated Hessian spectral density (solid lines) of the PINN loss for the convection, reaction, and wave problems after training with Adam+L-BFGS. For all three problems, we observe large outlier eigenvalues ($> 10^4$ for convection, $> 10^3$ for reaction, and $> 10^5$ for wave) in the spectrum, and a significant spectral density near 0, implying that the loss L is ill-conditioned. The plots also show how the spectrum is improved by preconditioning (Section 5.3).

5.2. The Ill-conditioning is Due to the Residual Loss

We use the same method to study the conditioning of each component of the PINN loss. Figures 3 and 7 show the estimated spectral density of the Hessian of the residual, initial condition, and boundary condition components of the PINN loss for each problem after training with Adam+L-BFGS. We see residual loss, which contains the differential operator $\mathcal{D}$, is the most ill-conditioned among all components. Our theory (Section 8) shows this ill-conditioning is likely due to the ill-conditioning of $\mathcal{D}$.

5.3. L-BFGS Improves Problem Conditioning

Preconditioning is a popular technique for improving conditioning in optimization. A classic example is Newton's method, which uses second-order information (i.e., the Hessian) to (locally) transform an ill-conditioned loss landscape into a well-conditioned one. L-BFGS is a quasi-Newton method that improves conditioning without explicit access to the problem Hessian. To examine the effectiveness of quasi-Newton methods for optimizing L, we compute the spectral density of the Hessian after L-BFGS preconditioning. (For details of this computation and how L-BFGS preconditions, see Appendix C.) Figure 3 shows this preconditioned Hessian spectral density (dashed lines). For all three problems, the magnitude of eigenvalues and the condition number has been reduced by at least 10^3. In addition, the preconditioner improves the conditioning of each individual loss component of L (Figures 3 and 7). These observations offer clear evidence that quasi-Newton methods improve the conditioning of the loss, and show the importance of quasi-Newton methods in training PINNs, which we demonstrate in Section 6.

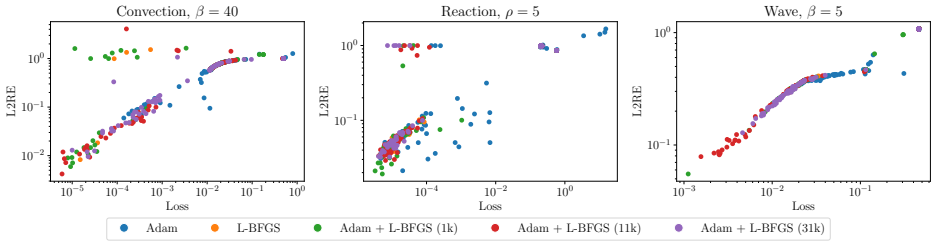

Figure 2. We plot the final L2RE against the final loss for each combination of network width, optimization strategy, and random seed. Across all three PDEs, a lower loss generally corresponds to a lower L2RE.

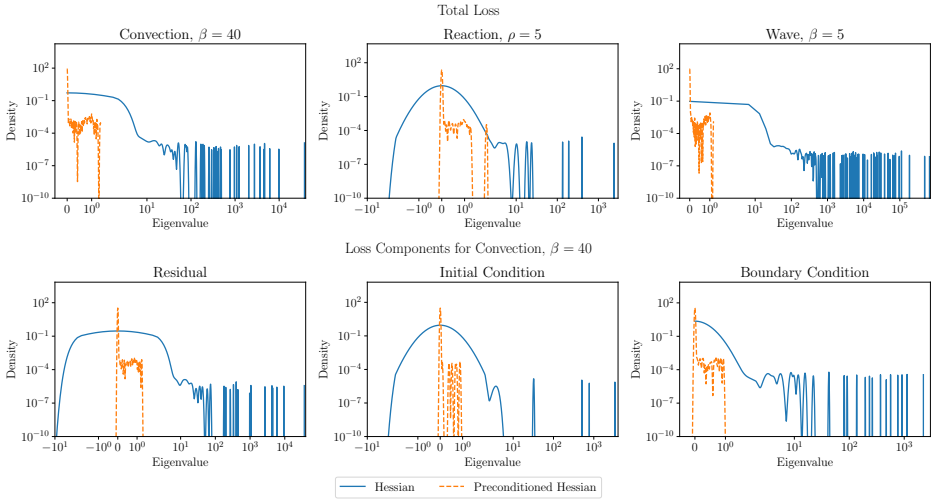

Figure 3. (Top) Spectral density of the Hessian and the preconditioned Hessian after 41000 iterations of Adam+L-BFGS. The plots show that the PINN loss is ill-conditioned and that L-BFGS improves the conditioning, reducing the top eigenvalue by 10^3 or more. (Bottom) Spectral density of the Hessian and the preconditioned Hessian of each loss component after 41000 iterations of Adam+L-BFGS for convection. The plots show that each component loss is ill-conditioned and that the conditioning is improved by L-BFGS.

6. Adam+L-BFGS Optimizes the Loss Better Than Other Methods

We demonstrate that the combined optimization method Adam+L-BFGS consistently provides a smaller loss and L2RE than using Adam or L-BFGS alone. We justify this finding using intuition from optimization theory.

6.1. Adam+L-BFGS vs Adam or L-BFGS

Figure 8 in Appendix D compares Adam+L-BFGS, Adam, and L-BFGS on the convection, reaction, and wave problems at difficult coefficient settings noted in the literature (Krishnapriyan et al., 2021; Wang et al., 2022b). Across each network width, the lowest loss and L2RE is always delivered by Adam+L-BFGS. Similarly, the lowest median loss and L2RE are almost always delivered by Adam+L-BFGS (Figure 8). The only exception is the reaction problem, where Adam outperforms Adam+L-BFGS on loss at width = 100 and L2RE at width = 200 (Figure 8).

Table 1. Lowest loss for Adam, L-BFGS, and Adam+L-BFGS across all network widths after hyperparameter tuning. Adam+L-BFGS attains both smaller loss and L2RE vs. Adam or L-BFGS.

Optimizer	Convection		Reaction		Wave	
	Loss	L2RE	Loss	L2RE	Loss	L2RE
Adam	1.40e-4	5.96e-2	4.73e-6	2.12e-2	2.03e-2	3.49e-1
L-BFGS	1.51e-5	8.26e-3	8.93e-6	3.83e-2	1.84e-2	3.35e-1
Adam+L-BFGS	**5.95e-6**	**4.19e-3**	**3.26e-6**	**1.92e-2**	**1.12e-3**	**5.52e-2**

Table 1 summarizes the best performance of each optimizer. Again, Adam+L-BFGS is better than running either Adam or L-BFGS alone. Notably, Adam+L-BFGS attains 14.2× smaller L2RE than Adam on the convection problem and 6.07× smaller L2RE than L-BFGS on the wave problem.

6.2. Intuition From Optimization Theory

The success of Adam+L-BFGS over Adam and L-BFGS can be explained by existing results in optimization theory. In neural networks, saddle points typically outnumber local

minima (Dauphin et al., 2014; Lee et al., 2019). A saddle point can never be a global minimum. We want to reach a global minimum when training PINNs.

Newton's method (which L-BFGS attempts to approximate) is attracted to saddle points (Dauphin et al., 2014), and quasi-Newton methods such as L-BFGS also converge to saddle points since they ignore negative curvature (Dauphin et al., 2014). On the other hand, first-order methods such as gradient descent and AdaGrad (Duchi et al., 2011) avoid saddle points (Lee et al., 2019; Antonakopoulos et al., 2022). We expect that (full-gradient) Adam also avoids saddles for similar reasons, although we are not aware of such a result.

Alas, first-order methods converge slowly when the problem is ill-conditioned. This result generalizes the well-known slow convergence of conjugate gradient (CG) for ill-conditioned linear systems: $\mathcal{O}(\sqrt{\kappa}\log(\frac{1}{\epsilon}))$ iterations to converge to an ϵ-approximate solution of a system with condition number κ. In optimization, an analogous notion of a condition number in a set $\mathcal{S}$ near a global minimum is given by $\kappa_f(\mathcal{S}) := \sup_{w\in\mathcal{S}} \|H_f(w)\|/\mu$, where μ is the PŁ*-constant (see Section 8). Then gradient descent requires $\mathcal{O}(\kappa_f(\mathcal{S})\log(\frac{1}{\epsilon}))$ iterations to converge to an ϵ-suboptimal point. For PINNs, the condition number near a solution is often $> 10^4$ (Figure 3), which leads to slow convergence of first-order methods. However, Newton's method and L-BFGS can significantly reduce the condition number (Figure 3), which yields faster convergence.

Adam+L-BFGS combines the best of both first- and second-order/quasi-Newton methods. By running Adam first, we avoid saddle points that would attract L-BFGS. By running L-BFGS after Adam, we can reduce the condition number of the problem, which leads to faster local convergence. Figure 1 exemplifies this, showing faster convergence of Adam+L-BFGS over Adam on the wave equation.

This intuition also explains why Adam sometimes performs as well as Adam+L-BFGS on the reaction problem. Figure 3 shows the largest eigenvalue of the reaction problem is around 10^3, while the largest eigenvalues of the convection and wave problems are around 10^4 and 10^5, suggesting the reaction problem is less ill-conditioned.

7. The Loss is Often Under-optimized

In Section 6, we show that Adam+L-BFGS improves on running Adam or L-BFGS alone. However, even Adam+L-BFGS does not reach a critical point of the loss: the loss is still under-optimized. We show that the loss and L2RE can be further improved by running a damped version of Newton's method.

7.1. Why is the Loss Under-optimized?

Figure 4 shows the run of Adam+L-BFGS with smallest L2RE for each PDE. For each run, L-BFGS stops making progress before reaching the maximum number of iterations. L-BFGS uses *strong Wolfe line search*, as it is needed to maintain the stability of L-BFGS (Nocedal & Wright, 2006). L-BFGS often terminates because it cannot find a positive step size satisfying these conditions—we have observed several instances where L-BFGS picks a step size of zero (Figure 9 in Appendix E), leading to early stopping. Perversely, L-BFGS stops in these cases without reaching a critical point: the gradient norm is around 10^{-2} or 10^{-3} (see the bottom row of Figure 4). The gradient still contains useful information for improving the loss.

7.2. NysNewton-CG (NNCG)

We can avoid premature termination by using a damped version of Newton's method with *Armijo line search*. The Armijo conditions use only a subset of the strong Wolfe conditions. Under only Armijo conditions, L-BFGS is unstable; we require a different approximation to the Hessian ($p \times p$ for a neural net with p parameters) that does not require storing ($\mathcal{O}(p^2)$) or inverting ($\mathcal{O}(p^3)$) the Hessian. Instead, we run a Newton-CG algorithm that solves for the Newton step using preconditioned conjugate gradient (PCG). This algorithm can be implemented efficiently with Hessian-vector products. These can be computed $\mathcal{O}\left((n_{\text{res}} + n_{\text{bc}})p\right)$ time (Pearlmutter, 1994). Section 5 shows that the Hessian is ill-conditioned with fast spectral decay, so CG without preconditioning will converge slowly. Hence we use NyströmPCG, a PCG method that is designed to solve linear systems with fast spectral decay (Frangella et al., 2023). The resulting algorithm is called NysNewton-CG (abbreviated NNCG); a full description of the algorithm appears in Appendix E.

7.3. Performance of NNCG

Figure 4 shows that NNCG significantly improves both the loss and gradient norm of the solution when applied after Adam+L-BFGS, while Figure 5 visualizes how NNCG improves the absolute error (pointwise) of the PINN solution when applied after Adam+L-BFGS. Furthermore, Table 2 shows that NNCG also improves the L2RE of the PINN solution. In contrast, applying gradient descent (GD) after Adam+L-BFGS improves neither the loss nor the L2RE. This result is unsurprising, as our theory predicts that NNCG will work better than GD for an ill-conditioned loss (Section 8).

7.4. Why Not Use NNCG Directly After Adam?

Since NNCG improves the PINN solution and uses simpler line search conditions than L-BFGS, it is tempting to

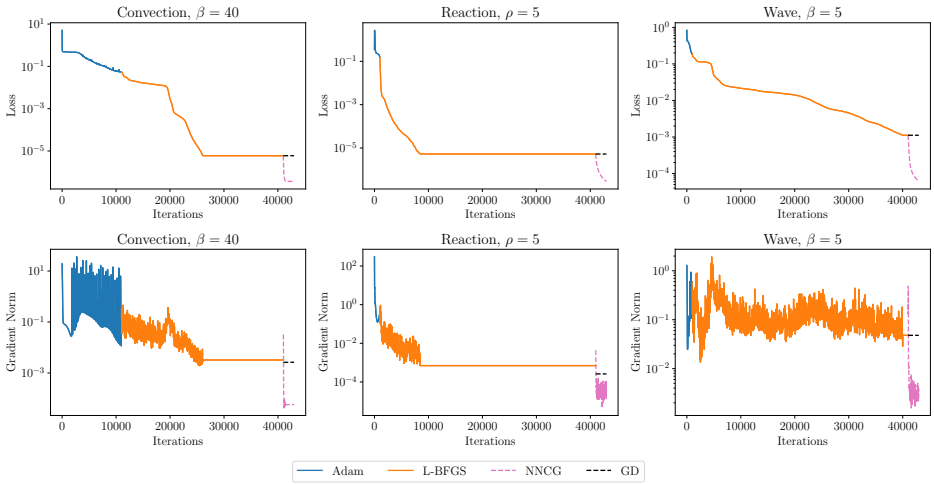

Figure 4. Performance of NNCG and GD after Adam+L-BFGS. (Top) NNCG reduces the loss by a factor greater than 10 in all instances, while GD fails to make progress. (Bottom) Furthermore, NNCG significantly reduces the gradient norm on the convection and wave problems, while GD fails to do so.

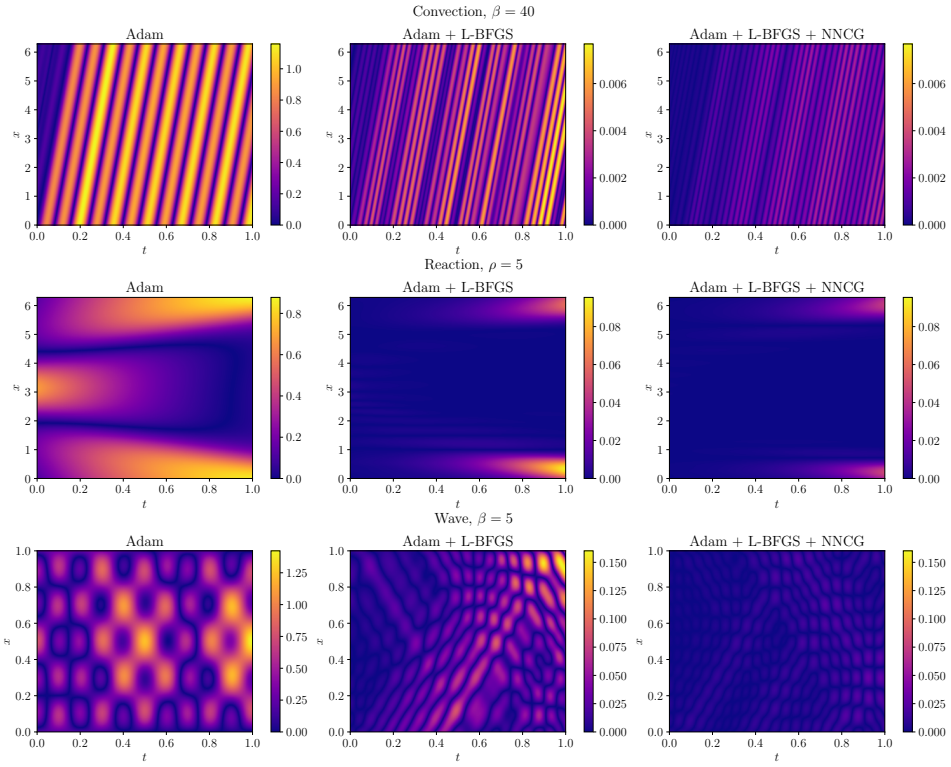

Figure 5. Absolute errors of the PINN solution at optimizer switch points. The first column shows errors after Adam, the second column shows errors after running L-BFGS following Adam, and the third column shows the errors after running NNCG folllowing Adam+L-BFGS. L-BFGS improves the solution obtained from first running Adam, and NNCG further improves the solution even after Adam+L-BFGS stops making progress. Note that Adam solution errors (left-most column) are presented at separate scales as these solutions are far off from the exact solutions.

Table 2. Loss and L2RE after fine-tuning by NNCG and GD. NNCG outperforms both GD and the original Adam+L-BFGS results.

Optimizer	Convection		Reaction		Wave	
	Loss	L2RE	Loss	L2RE	Loss	L2RE
Adam+L-BFGS	5.95e-6	4.19e-3	5.26e-6	1.92e-2	1.12e-3	5.52e-2
Adam+L-BFGS+NNCG	**3.63e-7**	**1.94e-3**	**2.89e-7**	**9.92e-3**	**6.13e-5**	**1.27e-2**
Adam+L-BFGS+GD	5.95e-6	4.19e-3	5.26e-6	1.92e-2	1.12e-3	5.52e-2

replace L-BFGS with NNCG entirely. However, NNCG is slower than L-BFGS: the L-BFGS update can be computed in $\mathcal{O}(mp)$ time, where m is the memory parameter, while just a single Hessian-vector product for computing the NNCG update requires $\mathcal{O}\left((n_{\text{res}} + n_{\text{bc}})p\right)$ time. Table 3 shows NNCG takes 5, 20, and 322 more times per-iteration as L-BFGS on convection, reaction, and wave respectively. Consequently, we should run Adam+L-BFGS to make as much progress as possible before switching to NNCG.

8. Theory

We relate the conditioning of the differential operator to the conditioning of the PINN loss function (2) in Theorem 8.4. When the differential operator is ill-conditioned, gradient descent takes many iterations to reach a high-precision solution. As a result, first-order methods alone may not deliver sufficient accuracy.

Algorithm 1 Gradient-Damped Newton Descent (GDND)

input # of gradient descent iterations K_{GD}, gradient descent learning rate η_{GD}, # of damped Newton iterations K_{DN}, damped Newton learning rate η_{DN}, damping parameter γ
  Phase I: **Gradient descent**
  for $k = 0, \ldots, K_{\text{GD}} - 1$ **do**
    $w_{k+1} = w_k - \eta_{\text{GD}} \nabla L(w_k)$
  end for
  Phase II: **Damped Newton**
  Set $\tilde{w}_0 = w_{K_{\text{GD}}}$
  for $k = 0, \ldots, K_{\text{DN}} - 1$ **do**
    $\tilde{w}_{k+1} = \tilde{w}_k - \eta_{\text{DN}} \left(H_L(\tilde{w}_k) + \gamma I\right)^{-1} \nabla L(\tilde{w}_k)$
  end for
output approximate solution $\tilde{w}_{K_{\text{DN}}}$

To address this issue, we develop and analyze a hybrid algorithm, Gradient Damped Newton Descent (GDND, Algorithm 1), that switches from gradient descent to damped Newton's method after a fixed number of iterations. We show that GDND gives fast linear convergence independent of the condition number. This theory supports our empirical results, which show that the best performance is obtained by running Adam and switching to L-BFGS. Moreover, it provides a theoretical basis for using Adam+L-BFGS+NNCG to achieve the best performance.

GDND differs from Adam+L-BFGS+NNCG, the algorithm we recommend in practice. We analyze GD instead of Adam because existing analyses of Adam (Défossez et al., 2022; Zhang et al., 2022) do not mirror its empirical performance. The reason we run both L-BFGS and damped Newton is to maximize computational efficiency (Section 7.4).

8.1. Preliminaries

We begin with the main assumption for our analysis.

Assumption 8.1 (Interpolation)**.** Let $\mathcal{W}_\star$ denote the set of minimizers of (2). We assume that

$$L(w_\star) = 0, \quad \text{for all } w_\star \in \mathcal{W}_\star,$$

i.e., the model perfectly fits the training data.

From a theoretical standpoint, Assumption 8.1 is natural in light of various universal approximation theorems (Cybenko, 1989; Hornik et al., 1990; De Ryck et al., 2021), which show neural networks are capable of approximating any continuous function to arbitrary accuracy. Moreover, interpolation in neural networks is common in practice (Zhang et al., 2021; Belkin, 2021).

PŁ*-condition. In modern neural network optimization, the PŁ*-condition (Liu et al., 2022; 2023) is key to showing convergence of gradient-based optimizers. It is a local version of the celebrated Polyak-Łojasiewicz condition (Polyak, 1963; Karimi et al., 2016), specialized to interpolation.

Definition 8.2 (PŁ*-condition)**.** Suppose L satisfies Assumption 8.1. Let $\mathcal{S} \subset \mathbb{R}^p$. Then L is μ-PŁ*in $\mathcal{S}$ if

$$\frac{\|\nabla L(w)\|^2}{2\mu} \geq L(w), \qquad \forall w \in \mathcal{S}.$$

The PŁ*-condition relates the gradient norm to the loss and implies that any minimizer in $\mathcal{S}$ is a global minimizer. Importantly, the PŁ*-condition can hold for non-convex losses and is known to hold, with high probability, for sufficiently wide neural nets with the least-squares loss (Liu et al., 2022).

Definition 8.3 (Condition number for PŁ* loss functions)**.** Let $\mathcal{S}$ be a set for which L is μ-PŁ*. Then the condition number of L over $\mathcal{S}$ is given by

$$\kappa_L(\mathcal{S}) = \frac{\sup_{w \in \mathcal{S}} \|H_L(w)\|}{\mu},$$

where $H_L(w)$ is the Hessian matrix of the loss function.

Gradient descent over $\mathcal{S}$ converges to ϵ-suboptimality in $\mathcal{O}\left(\kappa_L(\mathcal{S}) \log\left(\frac{1}{\epsilon}\right)\right)$ iterations (Liu et al., 2022).

8.2. Ill-conditioned Differential Operators Lead to Challenging Optimization

Here, we show that when the differential operator defining the PDE is *linear* and ill-conditioned, the condition number of the PINN objective (in the sense of Definition 8.3) is large. Our analysis in this regard is inspired by the recent work of De Ryck et al. (2023), who prove a similar result for the population PINN residual loss. However, De Ryck et al. (2023)'s analysis is based on the *lazy training regime*, which assumes the NTK is approximately constant. This regime does not accurately capture the behavior of practical neural networks (Allen-Zhu & Li, 2019; Chizat et al., 2019; Ghorbani et al., 2020; 2021). Moreover, gradient descent can converge even with a non-constant NTK (Liu et al., 2020). Our theoretical result is more closely aligned with deep learning practice as it does not assume lazy training and pertains to the empirical loss rather than the population loss.

Theorem 8.4 provides an informal version of our result in Appendix F that shows that ill-conditioned differential operators induce ill-conditioning in the loss (2). The theorem statement involves a kernel integral operator, $\mathcal{K}_\infty$ (defined in (6) in Appendix F), evaluated at the optimum $w_\star$.

Theorem 8.4 (Informal)**.** *Suppose Assumption 8.1 holds and $p \geq n_{\text{res}} + n_{\text{bc}}$. Fix $w_\star \in \mathcal{W}_\star$ and set $\mathcal{A} = \mathcal{D}^*\mathcal{D}$. For some $\alpha > 1/2$, suppose the eigenvalues of $\mathcal{A} \circ \mathcal{K}_\infty(w_\star)$ satisfy $\lambda_j(\mathcal{A} \circ \mathcal{K}_\infty(w_\star)) = \mathcal{O}\left(j^{-2\alpha}\right)$. If $\sqrt{n_{\text{res}}} = \Omega\left(\log\left(\frac{1}{\delta}\right)\right)$, then for any set $\mathcal{S}$ that contains $w_\star$ and for which L is μ-PŁ*,*

$$\kappa_L(\mathcal{S}) = \Omega\left(n_{\text{res}}^{\alpha}\right), \qquad \textit{with probability} \geq 1 - \delta.$$

Theorem 8.4 relates the conditioning of the PINN optimization problem to the conditioning of the operator $\mathcal{A} \circ \mathcal{K}_\infty(w_\star)$, where $\mathcal{A}$ is the Hermitian square of $\mathcal{D}$. If the spectrum of $\mathcal{A} \circ \mathcal{K}_\infty(w_\star)$ decays polynomially, then, with high probability, the condition number grows with n_{res}. As n_{res} typically ranges from 10^3 to 10^4, Theorem 8.4 shows the condition number of the PINN problem is generally large, and so first-order methods will be slow to converge to the optimum. Figure 10 in Appendix F.5 empirically verifies the claim of Theorem 8.4 for the convection equation.

8.3. Efficient High-precision Solutions via GDND

We now analyze the convergence behavior of Algorithm 1. Theorem 8.5 provides an informal version of our result in Appendix G.

Theorem 8.5 (Informal)**.** *Suppose $L(w)$ satisfies the μ-PŁ*-condition in a certain ball about w_0. Then there exists $\eta_{\text{GD}} > 0$ and $K_{\text{GD}} < \infty$ such that Phase I of Algorithm 1 outputs a point $w_{K_{\text{GD}}}$, for which Phase II of Algorithm 1 with $\eta_{\text{DN}} = 5/6$ and appropriate damping $\gamma > 0$, satisfies*

$$L(\tilde{w}_k) \leq \left(\frac{2}{3}\right)^k L(w_{K_{\text{GD}}}).$$

Hence after $K_{\text{DN}} \geq 3 \log\left(\frac{L(w_{K_{\text{GD}}})}{\epsilon}\right)$ iterations, Phase II of Algorithm 1 outputs a point satisfying $L(\tilde{w}_{K_{\text{DN}}}) \leq \epsilon$.

Theorem 8.5 shows only a fixed number of gradient descent iterations are needed before Algorithm 1 can switch to damped Newton's method and enjoy linear convergence independent of the condition number. As the convergence rate of Phase II with damped Newton is independent of the condition number, Algorithm 1 produces a highly accurate solution to (2).

Note that Theorem 8.5 is local; Algorithm 1 must find a point sufficiently close to a minimizer with gradient descent before switching to damped Newton's method and achieving rapid convergence. It is not possible to develop a second-order method with a fast rate that does not require a good initialization, as in the worst-case, global convergence of second-order methods may fail to improve over first-order methods (Cartis et al., 2010; Arjevani et al., 2019). Moreover, Theorem 8.5 is consistent with our experiments, which show L-BFGS is inferior to Adam+L-BFGS.

9. Conclusion

In this work, we explore the challenges posed by the loss landscape of PINNs for gradient-based optimizers. We demonstrate ill-conditioning in the PINN loss and show it hinders effective training of PINNs. By comparing Adam, L-BFGS, and Adam+L-BFGS, and introducing NNCG, we have demonstrated several approaches to improve the training process. Our theory supports our experimental findings: we connect ill-conditioned differential operators to ill-conditioning in the PINN loss and prove the benefits of second-order methods over first-order methods for PINNs.

Acknowledgements

We would like to acknowledge helpful comments from the anonymous reviewers and area chairs, which have improved this submission. MU, PR, WL, and ZF gratefully acknowledge support from the National Science Foundation (NSF) Award IIS-2233762, the Office of Naval Research (ONR) Award N000142212825 and N000142312203, and the Alfred P. Sloan Foundation. LL gratefully acknowledges support from the U.S. Department of Energy [DE-SC0022953].

Impact Statement

This paper presents work whose goal is to advance the field of scientific machine learning. There are many potential

societal consequences of our work, none which we feel must be specifically highlighted here.

References

Allen-Zhu, Z. and Li, Y. What Can ResNet Learn Efficiently, Going Beyond Kernels? In *Advances in Neural Information Processing Systems*, 2019.

Antonakopoulos, K., Mertikopoulos, P., Piliouras, G., and Wang, X. AdaGrad Avoids Saddle Points. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Arjevani, Y., Shamir, O., and Shiff, R. Oracle complexity of second-order methods for smooth convex optimization. *Mathematical Programming*, 178:327–360, 2019.

Bach, F. Sharp analysis of low-rank kernel matrix approximations. In *Conference on learning theory*, 2013.

Belkin, M. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numerica*, 30:203–248, 2021.

Cartis, C., Gould, I. N., and Toint, P. L. On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 20(6):2833–2852, 2010.

Chizat, L., Oyallon, E., and Bach, F. On Lazy Training in Differentiable Programming. In *Advances in Neural Information Processing Systems*, 2019.

Cohen, M. B., Musco, C., and Musco, C. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2017.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. Scientific Machine Learning Through Physics–Informed Neural Networks: Where We Are and What's Next. *J. Sci. Comput.*, 92(3), 2022.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, 2014.

De Ryck, T., Lanthaler, S., and Mishra, S. On the approximation of functions by tanh neural networks. *Neural Networks*, 143:732–750, 2021.

De Ryck, T., Bonnet, F., Mishra, S., and de Bézenac, E. An operator preconditioning perspective on training in physics-informed machine learning. *arXiv preprint arXiv:2310.05801*, 2023.

Défossez, A., Bottou, L., Bach, F., and Usunier, N. A simple convergence proof of Adam and Adagrad. *Transactions on Machine Learning Research*, 2022.

Duchi, J., Hazan, E., and Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61): 2121–2159, 2011.

E, W. and Yu, B. The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. *Communications in Mathematics and Statistics*, 6 (1):1–12, 2018.

Frangella, Z., Tropp, J. A., and Udell, M. Randomized Nyström Preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 44(2):718–752, 2023.

Ghorbani, B., Krishnan, S., and Xiao, Y. An Investigation into Neural Net Optimization via Hessian Eigenvalue Density. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. When Do Neural Networks Outperform Kernel Methods? In *Advances in Neural Information Processing Systems*, 2020.

Ghorbani, B., Mei, S., Misiakiewicz, T., and Montanari, A. Linearized two-layers neural networks in high dimension. *The Annals of Statistics*, 49(2):1029–1054, 2021.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

Golub, G. H. and Meurant, G. *Matrices, moments and quadrature with applications*, volume 30. Princeton University Press, 2009.

Hao, Z., Yao, J., Su, C., Su, H., Wang, Z., Lu, F., Xia, Z., Zhang, Y., Liu, S., Lu, L., and Zhu, J. PINNacle: A Comprehensive Benchmark of Physics-Informed Neural Networks for Solving PDEs. *arXiv preprint arXiv:2306.08827*, 2023.

Horn, R. A. and Johnson, C. R. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2012.

Hornik, K., Stinchcombe, M., and White, H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560, 1990.

Jagtap, A. D. and Karniadakis, G. E. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020.

Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020a.

Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E. 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*, 2020b.

Jagtap, A. D., Kharazmi, E., and Karniadakis, G. E. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020c.

Karimi, H., Nutini, J., and Schmidt, M. Linear Convergence of Gradient and Proximal-Gradient Methods under the Polyak-Łojasiewicz Condition. In *Machine Learning and Knowledge Discovery in Databases*, 2016.

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

Kharazmi, E., Zhang, Z., and Karniadakis, G. E. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021.

Khodayi-Mehr, R. and Zavlanos, M. VarNet: Variational Neural Networks for the Solution of Partial Differential Equations. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, pp. 298–307, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, 2021.

Lee, J. D., Panageas, I., Piliouras, G., Simchowitz, M., Jordan, M. I., and Recht, B. First-order methods almost always avoid strict saddle points. *Mathematical Programming*, 176(1):311–337, 2019.

Li, K., Tang, K., Wu, T., and Liao, Q. D3M: A Deep Domain Decomposition Method for Partial Differential Equations. *IEEE Access*, 8:5283–5294, 2020.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, 2021.

Lin, L., Saad, Y., and Yang, C. Approximating spectral densities of large matrices. *SIAM review*, 58(1):34–65, 2016.

Liu, C., Zhu, L., and Belkin, M. On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 2020.

Liu, C., Zhu, L., and Belkin, M. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

Liu, C., Drusvyatskiy, D., Belkin, M., Davis, D., and Ma, Y.-A. Aiming towards the minimizers: fast convergence of SGD for overparametrized problems. *arXiv preprint arXiv:2306.02601*, 2023.

Liu, D. C. and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

Liu, S., Su, C., Yao, J., Hao, Z., Su, H., Wu, Y., and Zhu, J. Preconditioning for physics-informed neural networks, 2024.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021a.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. DeepXDE: A Deep Learning Library for Solving Differential Equations. *SIAM Review*, 63(1):208–228, 2021b.

Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., and Johnson, S. G. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021c.

Lu, L., Pestourie, R., Johnson, S. G., and Romano, G. Multifidelity deep neural operators for efficient learning of partial differential equations with application to fast inverse design of nanoscale heat transport. *Physical Review Research*, 4(2):023210, 2022.

Mishra, S. and Molinaro, R. Estimates on the generalization error of physics-informed neural networks for approximating pdes. *IMA Journal of Numerical Analysis*, 43(1): 1–43, 2023.

Moseley, B., Markham, A., and Nissen-Meyer, T. Finite basis physics-informed neural networks (FBPINNs): a scalable domain decomposition approach for solving differential equations. *Advances in Computational Mathematics*, 49(4):62, 2023.

Müller, J. and Zeinhofer, M. Achieving High Accuracy with PINNs via Energy Natural Gradient Descent. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Nabian, M. A., Gladstone, R. J., and Meidani, H. Efficient training of physics-informed neural networks via importance sampling. *Comput.-Aided Civ. Infrastruct. Eng.*, 36 (8):962–977, 2021.

Nesterov, Y. *Lectures on Convex Optimization*. Springer Publishing Company, Incorporated, 2nd edition, 2018.

Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer, 2nd edition, 2006.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv preprint arXiv:1912.01703*, 2019.

Pearlmutter, B. A. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Polyak, B. T. Gradient methods for minimizing functionals. *Zhurnal vychislitel'noi matematiki i matematicheskoi fiziki*, 3(4):643–653, 1963.

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

Rohrhofer, F. M., Posch, S., Gößnitzer, C., and Geiger, B. C. On the Role of Fixed Points of Dynamical Systems in Training Physics-Informed Neural Networks. *Transactions on Machine Learning Research*, 2023.

Rudi, A., Carratino, L., and Rosasco, L. FALKON: An Optimal Large Scale Kernel Method. In *Advances in Neural Information Processing Systems*, 2017.

Tropp, J. A. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8 (1-2):1–230, 2015.

Wang, H., Lu, L., Song, S., and Huang, G. Learning Specialized Activation Functions for Physics-Informed Neural Networks. *Communications in Computational Physics*, 34(4):869–906, 2023.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a.

Wang, S., Wang, H., and Perdikaris, P. 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, 2021b.

Wang, S., Wang, H., and Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40): eabi8605, 2021c.

Wang, S., Sankaran, S., and Perdikaris, P. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*, 2022a.

Wang, S., Yu, X., and Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022b.

Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. 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, 2023a.

Wu, W., Daneker, M., Jolley, M. A., Turner, K. T., and Lu, L. Effective data sampling strategies and boundary condition constraints of physics-informed neural networks for identifying material properties in solid mechanics. *Applied mathematics and mechanics*, 44(7):1039–1068, 2023b.

Yao, J., Su, C., Hao, Z., Liu, S., Su, H., and Zhu, J. MultiAdam: Parameter-wise Scale-invariant Optimizer for Multiscale Training of Physics-informed Neural Networks. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. PyHessian: Neural Networks Through the Lens of the Hessian. In *2020 IEEE International Conference on Big Data (Big Data)*, 2020.

Yu, J., Lu, L., Meng, X., and Karniadakis, G. E. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Zhang, Y., Chen, C., Shi, N., Sun, R., and Luo, Z.-Q. Adam Can Converge Without Any Modification On Update Rules. In *Advances in Neural Information Processing Systems*, 2022.

A. Additional Details on Problem Setup

Here we present the differential equations that we study in our experiments.

A.1. Convection

The one-dimensional convection problem is a hyperbolic PDE that can be used to model fluid flow, heat transfer, and biological processes. The convection PDE we study is

$$\begin{aligned} \frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} &= 0, \quad x \in (0, 2\pi), t \in (0, 1), \\ u(x, 0) &= \sin(x), \quad x \in [0, 2\pi], \\ u(0, t) &= u(2\pi, t), \quad t \in [0, 1]. \end{aligned}$$

The analytical solution to this PDE is $u(x, t) = \sin(x - \beta t)$. We set $\beta = 40$ in our experiments.

A.2. Reaction

The one-dimensional reaction problem is a non-linear ODE which can be used to model chemical reactions. The reaction ODE we study is

$$\begin{aligned} \frac{\partial u}{\partial t} - \rho u(1 - u) &= 0, \quad x \in (0, 2\pi), t \in (0, 1) \\ u(x, 0) &= \exp\left(-\frac{(x - \pi)^2}{2(\pi/4)^2}\right), \quad x \in [0, 2\pi], \\ u(0, t) &= u(2\pi, t), \quad t \in [0, 1]. \end{aligned}$$

The analytical solution to this ODE is $u(x, t) = \frac{h(x)e^{\rho t}}{h(x)e^{\rho t} + 1 - h(x)}$, where $h(x) = \exp\left(-\frac{(x-\pi)^2}{2(\pi/4)^2}\right)$. We set $\rho = 5$ in our experiments.

A.3. Wave

The one-dimensional wave problem is a hyperbolic PDE that often arises in acoustics, electromagnetism, and fluid dynamics. The wave PDE we study is

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} - 4\frac{\partial^2 u}{\partial x^2} &= 0, \quad x \in (0, 1), t \in (0, 1), \\ u(x, 0) &= \sin(\pi x) + \frac{1}{2}\sin(\beta \pi x), \quad x \in [0, 1], \\ \frac{\partial u(x, 0)}{\partial t} &= 0, \quad x \in [0, 1], \\ u(0, t) &= u(1, t) = 0, \quad t \in [0, 1]. \end{aligned}$$

The analytical solution to this PDE is $u(x, t) = \sin(\pi x)\cos(2\pi t) + \frac{1}{2}\sin(\beta \pi x)\cos(2\beta \pi t)$. We set $\beta = 5$ in our experiments.

B. Why can Low Losses Correspond to Large L2RE?

In Figure 2, there are several instances on the convection PDE and reaction ODE where the PINN loss is close to 0, but the L2RE of the PINN solution is close to 1. Rohrhofer et al. (2023) demonstrate that PINNs can be attracted to points in the loss landscape that minimize the residual portion of the PINN loss, $\frac{1}{2n_{\text{res}}}\sum_{i=1}^{n_{\text{res}}}\left(\mathcal{D}[u(x_r^i; w), x_r^i]\right)^2$, to 0. However, these can correspond to trivial solutions: for the convection PDE, the residual portion is equal to 0 for any constant function u; for the reaction ODE, the residual portion is equal to 0 for constant $u = 0$ or $u = 1$.

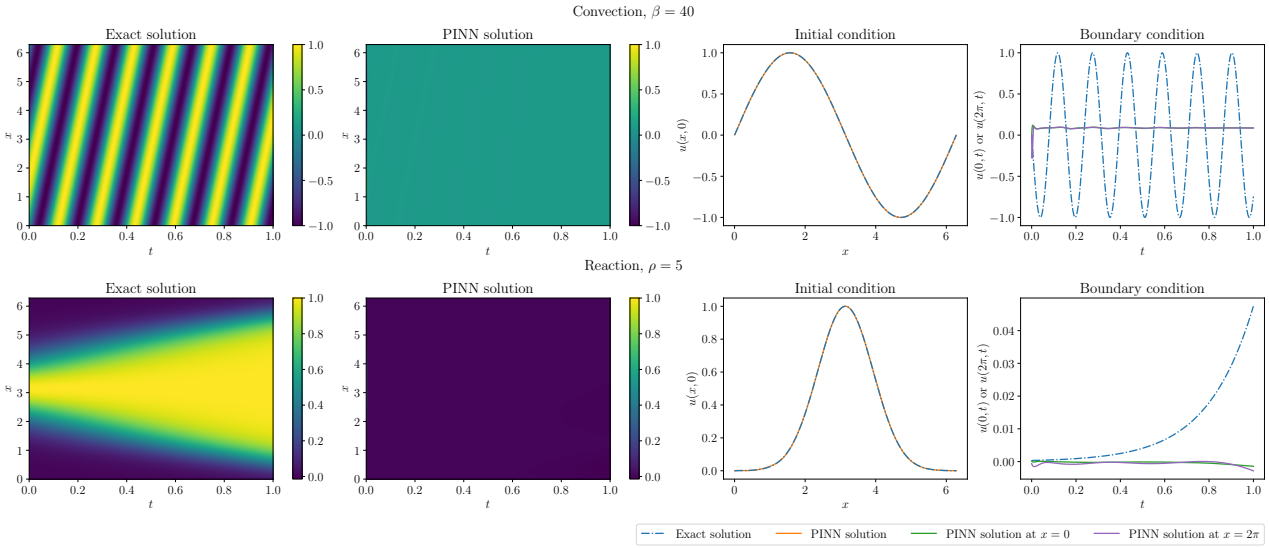

Figure 6. The first two columns from the left display the exact solutions and PINN solutions. The PINN fails to learn the exact solution, which leads to large L2RE. Moreover, the PINN solutions are effectively constant over the domain. The third and fourth columns from the left display the PINN solutions at the initial time ($t = 0$) and the boundaries ($x = 0$ and $x = 2\pi$). The PINN solutions learn the initial conditions, but they do not learn the boundary conditions.

To show that the PINN is indeed learning a trivial solution, we visualize two solutions with small residual loss but large L2RE in Figure 6. The second column of Figure 6 shows the PINN solutions are close to 0 almost everywhere in the domain. Interestingly, the PINN solutions correctly learn the initial condition. However, the PINN solutions for the convection PDE and reaction ODE do not match the exact solution at the boundaries. One approach for alleviating this training issue would be to (adaptively) reweight the residual, initial condition, and boundary condition terms in the PINN loss (Wang et al., 2021a; 2022b).

C. Computing the Spectral Density of the L-BFGS-preconditioned Hessian

C.1. How L-BFGS Preconditions

To minimize (2), L-BFGS uses the update

$$w_{k+1} = w_k - \eta H_k \nabla L(w_k), \tag{3}$$

where H_k is a matrix approximating the inverse Hessian. We now show how (3) is equivalent to preconditioning the objective (2). Define the coordinate transformation $w = H_k^{1/2} z$. By the chain rule, $\nabla L(z) = H_k^{1/2} \nabla L(w)$ and $H_L(z) = H_k^{1/2} H_L(w) H_k^{1/2}$. Thus, (3) is equivalent to

$$\begin{aligned} z_{k+1} &= z_k - \eta \nabla L(z_k), \\ w_{k+1} &= H_k^{1/2} z_{k+1}. \end{aligned} \tag{4}$$

Equation (4) reveals how L-BFGS preconditions (2). L-BFGS first takes a step in the *preconditioned* z-space, where the conditioning is determined by $H_L(z)$, the preconditioned Hessian. Since H_k approximates $H_L^{-1}(w)$, $H_k^{1/2} H_L(w) H_k^{1/2} \approx I_p$, so the condition number of $H_L(z)$ is much smaller than that of $H_L(w)$. Consequently, L-BFGS can take a step that makes more progress than a method like gradient descent, which performs no preconditioning at all. In the second phase, L-BFGS maps the progress in the preconditioned space back to the original space. Thus, L-BFGS is able to make superior progress by transforming (2) to another space where the conditioning is more favorable, which enables it to compute an update that better reduces the loss in (2).

C.2. Preconditioned Spectral Density Computation

Here we discuss how to compute the spectral density of the Hessian after preconditioning by L-BFGS. This is the procedure we use to generate the figures in Section 5.3.

L-BFGS stores a set of vector pairs given by the difference in consecutive iterates and gradients from most recent m iterations (we use $m = 100$ in our experiments). To compute the update direction $H_k \nabla f_k$, L-BFGS combines the stored vector pairs with a recursive scheme (Nocedal & Wright, 2006). Defining

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f_{k+1} - \nabla f_k, \quad \rho_k = \frac{1}{y_k^T s_k}, \quad \gamma_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}, \quad V_k = I - \rho_k y_k s_k^T, \quad H_k^0 = \gamma_k I,$$

the formula for H_k can be written as

$$H_k = (V_{k-1}^T V_{k-m}^T) H_k^0 (V_{k-m} V_{k-1}) + \sum_{l=2}^{m} \rho_{k-l} (V_{k-1}^T \cdots V_{k-l+1}^T) s_{k-l} s_{k-l}^T (V_{k-l+1} \cdots V_{k-1}) + \rho_{k-1} s_{k-1} s_{k-1}^T.$$

Expanding the terms, we have for $j \in \{1, 2, \ldots, i\}$,

$$V_{k-i} \cdots V_{k-1} = I - \sum_{j=1}^{i} \rho_{k-j} y_{k-j} \tilde{v}_{k-j}^T \quad \text{where} \quad \tilde{v}_{k-j} = s_{k-j} - \sum_{l=1}^{j-1} (\rho_{k-l} y_{k-l}^T s_{k-j}) \tilde{v}_{k-l}.$$

It follows that

$$H_k = (I - \tilde{Y}\tilde{V}^T)^T \gamma_k I (I - \tilde{Y}\tilde{V}^T) + \tilde{S}\tilde{S}^T = \begin{bmatrix} \sqrt{\gamma_k}(I - \tilde{Y}\tilde{V}^T)^T & \tilde{S} \end{bmatrix} \begin{bmatrix} \sqrt{\gamma_k}(I - \tilde{Y}\tilde{V}^T) \\ \tilde{S}^T. \end{bmatrix} = \tilde{H}_k \tilde{H}_k^T,$$

where

$$\tilde{Y} = \begin{bmatrix} | & & | \\ \rho_{k-1} y_{k-1} & \cdots & \rho_{k-m} y_{k-m} \\ | & & | \end{bmatrix},$$

$$\tilde{V} = \begin{bmatrix} | & & | \\ \tilde{v}_{k-1} & \cdots & \tilde{v}_{k-m} \\ | & & | \end{bmatrix},$$

$$\tilde{S} = \begin{bmatrix} | & & | \\ \tilde{s}_{k-1} & \cdots & \tilde{s}_{k-m} \\ | & & | \end{bmatrix}, \quad \tilde{s}_{k-1} = \sqrt{\rho_{k-1}} s_{k-1}, \ \tilde{s}_{k-l} = \sqrt{\rho_{k-l}} (V_{k-1}^T \cdots V_{k-l+1}^T) s_{k-l} \text{ for } 2 \le l \le m.$$

We now apply Algorithm 2 to unroll the above recurrence relations to compute columns of $\tilde{Y}$, $\tilde{S}$ and $\tilde{V}$.

Algorithm 2 Unrolling the L-BFGS Update

input saved directions $\{y_i\}_{i=k-1}^{k-m}$, saved steps $\{s_i\}_{i=k-1}^{k-m}$, saved inverse of inner products $\{\rho_i\}_{i=k-1}^{k-m}$
  $\tilde{y}_{k-1} = \rho_{k-1} y_{k-1}$
  $\tilde{v}_{k-1} = s_{k-1}$
  $\tilde{s}_{k-1} = \sqrt{\rho_{k-1}} s_{k-1}$
  for $i = k-2, \ldots, k-m$ **do**
    $\tilde{y}_i = \rho_i y_i$
    Set $\alpha = 0$
    for $j = k-1, \ldots, i+1$ **do**
      $\alpha = \alpha + (\tilde{y}_j^T s_i) \tilde{v}_j$
    end for
    $\tilde{v}_i = s_i - \alpha$
    $\tilde{s}_i = \sqrt{\rho_i}(s_i - \alpha)$
  end for
output vectors $\{\tilde{y}_i, \tilde{v}_i, \tilde{s}_i\}_{i=k-1}^{k-m}$

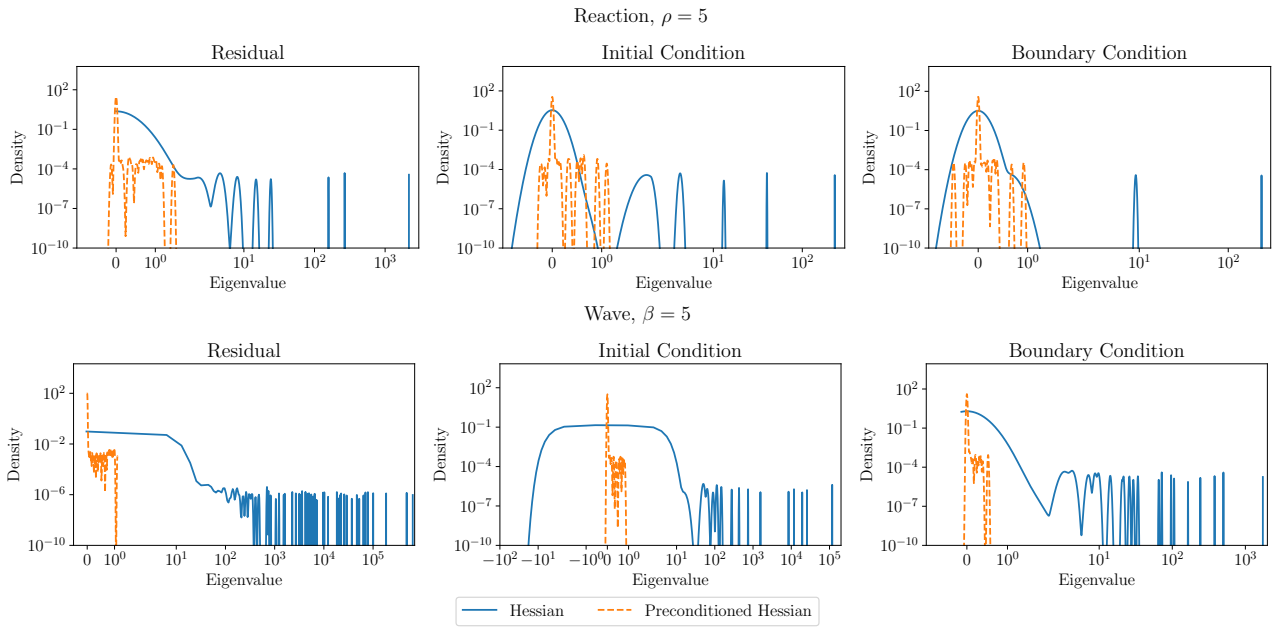

Figure 7. Spectral density of the Hessian and the preconditioned Hessian of each loss component after 41000 iterations of Adam+L-BFGS for the reaction and wave problems. The plots show the loss landscape of each component is ill-conditioned, and the conditioning of each loss component is improved by L-BFGS.

Since (non-zero) eigenvalues of $\tilde{H}_k^T H_L(w) \tilde{H}_k$ equal the eigenvalues of the preconditioned Hessian $H_k H_L(w) = \tilde{H}_k \tilde{H}_k^T H_L(w)$ (Theorem 1.3.22 of Horn & Johnson (2012)), we can analyze the spectrum of $\tilde{H}_k^T H_L(w) \tilde{H}_k$ instead. This is advantageous since methods for calculating the spectral density of neural network Hessians are only compatible with symmetric matrices.

Since $\tilde{H}_k^T H_L(w) \tilde{H}_k$ is symmetric, we can use stochastic Lanczos quadrature (SLQ) (Golub & Meurant, 2009; Lin et al., 2016) to compute spectral density of this matrix. SLQ only requires matrix-vector products with $\tilde{H}_k$ and Hessian-vector products, the latter of which may be efficiently computed via automatic differentiation; this is precisely what PyHessian does to compute spectral densities (Yao et al., 2020).

Algorithm 3 Performing matrix-vector product

input matrices $\tilde{Y}$, $\tilde{V}$, $\tilde{S}$ formed from resulting vectors from unrolling, vector v, and saved scaling factor for initializing diagonal matrix γ_k
Split vector v of length size$(w) + m$ into v_1 of size size(w) and v_2 of size m
$v' = \sqrt{\gamma_k}(v_1 - \tilde{V}\tilde{Y}^T v_1) + \tilde{S} v_2$
Perform Hessian-vector-product on v', and obtain v''
Stack $\sqrt{\gamma_k}(v'' - \tilde{Y}\tilde{V}^T v'')$ and $\tilde{S}^T v''$, and obtain v'''
output resulting vector v'''

By combining the matrix-vector product procedure described in Algorithm 3 with the Hessian-vector product operation, we are able to obtain spectral information of the preconditioned Hessian.

D. Adam+L-BFGS Generally Gives the Best Performance

Figure 8 shows that Adam+L-BFGS typically yields the best performance on both loss and L2RE across network widths.

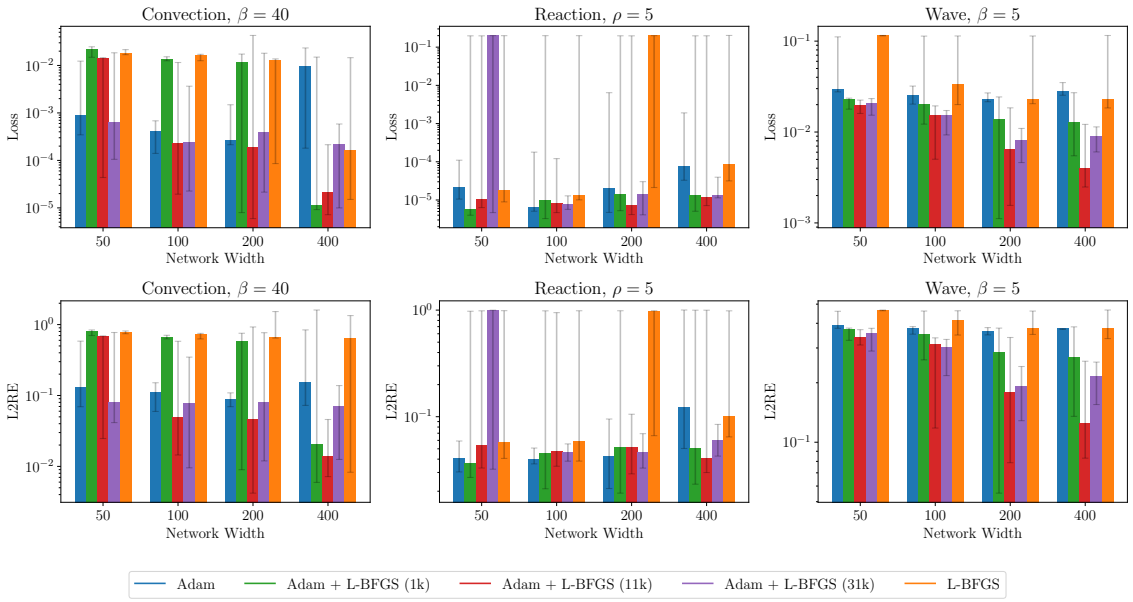

Figure 8. Performance of Adam, L-BFGS, and Adam+L-BFGS after tuning. We find the learning rate $\eta^\star$ for each network width and optimization strategy that attains the lowest loss (L2RE) across all random seeds. The min, median, and max loss (L2RE) are calculated by taking the min, median, and max of the losses (L2REs) for learning rate η^* across all random seeds. Each bar on the plot corresponds to the median, while the top and bottom error bars correspond to the max and min, respectively. The smallest min loss and L2RE are always attained by one of the Adam+L-BFGS strategies; the smallest median loss and L2RE are nearly always attained by one of the Adam+L-BFGS strategies.

E. Additional details on Under-optimization

E.1. Early Termination of L-BFGS

Figure 9 explains why L-BFGS terminates early for the convection, reaction, and wave problems. We evaluate the loss at 10^4 uniformly spaced points in the interval $[0, 1]$. The orange stars in Figure 9 are step sizes that satisfy the strong Wolfe conditions and the red dots are step sizes that L-BFGS examines during the line search.

E.2. NysNewton-CG (NNCG)

Here we present the NNCG algorithm (Algorithm 4) introduced in Section 7.2 and its associated subroutines Randomized-NyströmApproximation (Algorithm 5), NyströmPCG (Algorithm 6), and Armijo (Algorithm 7). At each iteration, NNCG first checks whether the Nyström preconditioner (stored in U and $\hat{\Lambda}$) for the NyströmPCG method needs to be updated. If so, the preconditioner is recomputed using the RandomizedNyströmApproximation subroutine. From here, the Newton step d_k is computed using NyströmPCG; we warm start the PCG algorithm using the Newton step d_{k-1} from the previous iteration. After computing the Newton step, we compute the step size η_k using Armijo line search — this guarantees that the loss will decrease when we update the parameters. Finally, we update the parameters using η_k and d_k.

In our experiments, we set $\eta = 1, K = 2000, s = 60, F = 20, \epsilon = 10^{-16}, M = 1000, \alpha = 0.1$, and $\beta = 0.5$. We tune $\mu \in [10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$; we find that $\mu = 10^{-2}, 10^{-1}$ work best in practice. Figures 1 and 4 show the NNCG run that attains the lowest loss after tuning μ.

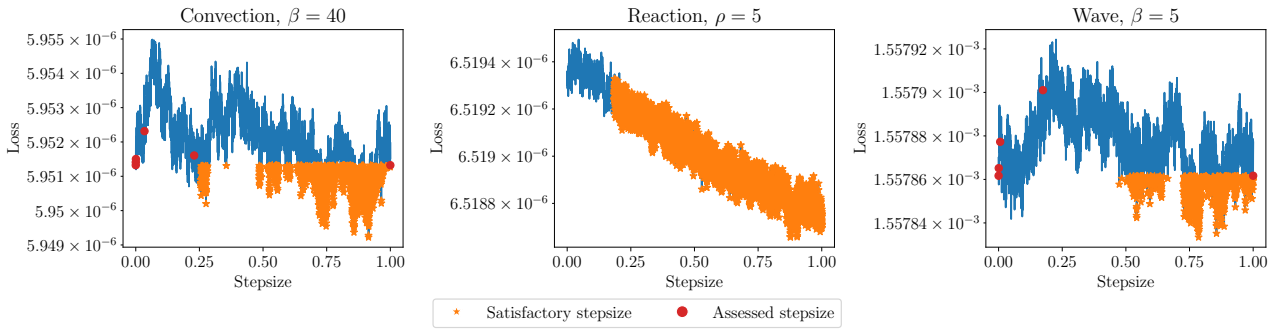

Figure 9. Loss evaluated along the L-BFGS search direction at different stepsizes after 41000 iterations of Adam+L-BFGS. For convection and wave, the line search does not find a stepsize that satisfies the strong Wolfe conditions, even though there are plenty of such points. For reaction, the slope of the objective used in the line search procedure at the current iterate is less than a pre-defined threshold 10^{-9}, so L-BFGS terminates without performing any line-search.

Algorithm 4 NysNewton-CG (NNCG)

input Initialization w_0, max. learning rate η, number of iterations K, preconditioner sketch size s, preconditioner update frequency F, damping parameter μ, CG tolerance ϵ, CG max. iterations M, backtracking parameters α, β
  $d_{-1} = 0$
  for $k = 0, \ldots, K-1$ **do**
    if k is a multiple of F **then**
      $[U, \hat{\Lambda}] = \text{RandomizedNyströmApproximation}(H_L(w_k), s)$ ▷ Update Nyström preconditioner every F iterations
    end if
    $d_k = \text{NyströmPCG}(H_L(w_k), \nabla L(w_k), d_{k-1}, U, \hat{\Lambda}, s, \mu, \epsilon, M)$ ▷ Damped Newton step $(H_L(w_k) + \mu I)^{-1} \nabla L(w_k)$
    $\eta_k = \text{Armijo}(L, w_k, \nabla L(w_k), -d_k, \eta)$ ▷ Compute step size via line search
    $w_{k+1} = w_k - \eta_k d_k$ ▷ Update parameters
  end for

The RandomizedNyströmApproximation subroutine (Algorithm 5) is used in NNCG to compute the preconditioner for NyströmPCG. The algorithm returns the top-s approximate eigenvectors and eigenvalues of the input matrix M. Within NNCG, the sketch computation $Y = MQ$ is implemented using Hessian-vector products. The portion in red is a fail-safe that allows for the preconditioner to be computed when H is an indefinite matrix. For further details, please see Frangella et al. (2023).

Algorithm 5 RandomizedNyströmApproximation

input Symmetric matrix M, sketch size s
  $S = \text{randn}(p, s)$ ▷ Generate test matrix
  $Q = \text{qr_econ}(S)$
  $Y = MQ$ ▷ Compute sketch
  $\nu = \sqrt{p}\text{eps}(\text{norm}(Y, 2))$ ▷ Compute shift
  $Y_\nu = Y + \nu Q$ ▷ Add shift for stability
  $\lambda = 0$ ▷ Additional shift may be required for positive definiteness
  $C = \text{chol}(Q^T Y_\nu)$ ▷ Cholesky decomposition: $C^T C = Q^T Y_\nu$
  if chol fails **then**
    Compute $[W, \Gamma] = \text{eig}(Q^T Y_\nu)$ ▷ $Q^T Y_\nu$ is small and square
    Set $\lambda = \lambda_{\min}(Q^T Y_\nu)$
    $R = W(\Gamma + |\lambda| I)^{-1/2} W^T$
    $B = YR$ ▷ R is psd
  else
    $B = YC^{-1}$ ▷ Triangular solve
  end if
  $[\hat{V}, \Sigma, \sim] = \text{svd}(B, 0)$ ▷ Thin SVD
  $\hat{\Lambda} = \max\{0, \Sigma^2 - (\nu + |\lambda| I)\}$ ▷ Compute eigs, and remove shift with element-wise max
  Return: $\hat{V}, \hat{\Lambda}$

The NyströmPCG subroutine (Algorithm 6) is used in NNCG to compute the damped Newton step. The preconditioner P and its inverse P^{-1} are given by

$$P = \frac{1}{\hat{\lambda}_s + \mu} U(\hat{\Lambda} + \mu I)U^T + (I - UU^T)$$
$$P^{-1} = (\hat{\lambda}_s + \mu)U(\hat{\Lambda} + \mu I)^{-1}U^T + (I - UU^T).$$

Within NNCG, the matrix-vector product involving the Hessian (i.e., $A = H_L(w_k)$) is implemented using Hessian-vector products. For further details, please see Frangella et al. (2023).

Algorithm 6 NyströmPCG

input Psd matrix A, right-hand side b, initial guess x_0, approx. eigenvectors U, approx. eigenvalues $\hat{\Lambda}$, sketch size s, damping parameter μ, CG tolerance ϵ, CG max. iterations M
  $r_0 = b - (A + \mu I)x_0$
  $z_0 = P^{-1} r_0$
  $p_0 = z_0$
  $k = 0$ ▷ Iteration counter
  while $\|r_0\|_2 \geq \varepsilon$ and $k < M$ **do**
    $v = (A + \mu I)p_0$
    $\alpha = (r_0^T z_0)/(p_0^T v_0)$ ▷ Compute step size
    $x = x_0 + \alpha p_0$ ▷ Update solution
    $r = r_0 - \alpha v$ ▷ Update residual
    $z = P^{-1} r$
    $\beta = (r^T z)/(r_0^T z_0)$
    $x_0 \leftarrow x, r_0 \leftarrow r, p_0 \leftarrow z + \beta p_0, z_0 \leftarrow z, k \leftarrow k + 1$
  end while
  Return: x

The Armijo subroutine (Algorithm 7) is used in NNCG to guarantee that the loss decreases at every iteration. The function oracle is implemented in PyTorch using a *closure*. At each iteration, the subroutine checks whether the *sufficient decrease condition* has been met; if not, it shrinks the step size by a factor of β. For further details, please see Nocedal & Wright (2006).

Table 3. Per-iteration times (in seconds) of L-BFGS and NNCG on each PDE.

Optimizer	Convection	Reaction	Wave
L-BFGS	4.6e-2	3.6e-2	9.0e-2
NNCG	2.5e-1	7.2e-1	2.9e1
Time Ratio	5.43	20	322.22

Algorithm 7 Armijo

input Function oracle f, current iterate x, current gradient $\nabla f(x)$, search direction d, initial step size t, backtracking parameters α, β
  while $f(x + td) > f(x) + \alpha t(\nabla f(x)^T d)$ **do**
    $t \leftarrow \beta t$ ▷ Shrink step size
  end while
  Return: t

E.3. Wall-clock Times for L-BFGS and NNCG

Table 3 summarizes the per-iteration wall-clock times of L-BFGS and NNCG on each PDE. The large gap on wave (compared to reaction and convection) is because NNCG has to compute hessian-vector products involving second derivatives, while this is not the case for the two other PDEs.

F. Ill-conditioned Differential Operators Lead to Difficult Optimization Problems

In this section, we state and prove the formal version of Theorem 8.4. The overall structure of the proof is based on showing the conditioning of the Gauss-Newton matrix of the population PINN loss is controlled by the conditioning of the differential operator. We then show the empirical Gauss-Newton matrix is close to its population counterpart by using matrix concentration techniques. Finally, as the conditioning of H_L at a minimizer is controlled by the empirical Gauss-Newton matrix, we obtain the desired result.

F.1. Preliminaries

Similar to De Ryck et al. (2023), we consider a general linear PDE with Dirichlet boundary conditions:

$$\begin{aligned} &\mathcal{D}[u](x) = f(x), \quad x \in \Omega, \\ &u(x) = g(x), \quad x \in \partial\Omega, \end{aligned}$$

where $u : \mathbb{R}^d \mapsto \mathbb{R}$, $f : \mathbb{R}^d \mapsto \mathbb{R}$ and Ω is a bounded subset of $\mathbb{R}^d$. The "population" PINN objective for this PDE is

$$L_\infty(w) = \frac{1}{2}\int_\Omega \left(\mathcal{D}[u(x;w)] - f(x)\right)^2 d\mu(x) + \frac{\lambda}{2}\int_{\partial\Omega} \left(u(x;w) - g(x)\right)^2 d\sigma(x).$$

λ can be any positive real number; we set $\lambda = 1$ in our experiments. Here μ and σ are probability measures on Ω and $\partial\Omega$ respectively, from which the data is sampled. The empirical PINN objective is given by

$$L(w) = \frac{1}{2n_{\text{res}}}\sum_{i=1}^{n_{\text{res}}} \left(\mathcal{D}[u(x_r^i;w)] - f(x_i)\right)^2 + \frac{\lambda}{2n_{\text{bc}}}\sum_{j=1}^{n_{\text{bc}}} \left(u(x_b^j;w) - g(x_j)\right)^2.$$

Moreover, throughout this section we use the notation $\langle f, g\rangle_{L^2(\Omega)}$ to denote the standard L^2-inner product on Ω:

$$\langle f, g\rangle_{L^2(\Omega)} = \int_\Omega fg d\mu(x).$$

Lemma F.1. *The Hessian of the $L_\infty(w)$ is given by*

$$H_{L_\infty}(w) = \int_\Omega \mathcal{D}[\nabla_w u(x;w)]\mathcal{D}[\nabla_w u(x;w)]^T d\mu(x) + \int_\Omega \mathcal{D}[\nabla_w^2 u(x;w)]\left(\mathcal{D}[\nabla_w u(x;w)] - f(x)\right) d\mu(x)$$
$$+ \lambda \int_{\partial\Omega} \nabla_w u(x;w)\nabla_w u(x;w)^T d\sigma(x) + \lambda \int_{\partial\Omega} \nabla_w^2 u(x;w)\left(u(x;w) - g(x)\right) d\sigma(x).$$

The Hessian of $L(w)$ is given by

$$H_L(w) = \frac{1}{n_{\text{res}}}\sum_{i=1}^{n_{\text{res}}} \mathcal{D}[\nabla_w u(x_r^i;w)]\mathcal{D}[\nabla_w u(x_r^i;w)]^T + \frac{1}{n_{\text{res}}}\sum_{i=1}^{n_{\text{res}}} \mathcal{D}[\nabla_w^2 u(x_i^r;w)]\left(\mathcal{D}[\nabla_w u(x_r^i;w)] - f(x_r^i)\right) \tag{5}$$
$$+ \frac{\lambda}{n_{\text{bc}}}\sum_{j=1}^{n_{\text{bc}}} \nabla_w u(x_b^j;w)\nabla_w u(x_b^j;w)^T + \frac{\lambda}{n_{\text{bc}}}\sum_{j=1}^{n_{\text{bc}}} \nabla_w^2 u(x_b^j;w)\left(u(x_b^j;w) - g(x_j)\right).$$

In particular, for $w_\star \in \mathcal{W}_\star$

$$H_L(w_\star) = G_r(w) + G_b(w).$$

Here

$$G_r(w) \coloneqq \frac{1}{n_{\text{res}}}\sum_{i=1}^{n_{\text{res}}} \mathcal{D}[\nabla_w u(x_i;w_\star)]\mathcal{D}[\nabla_w u(x_i;w_\star)]^T, \quad G_b(w) = \frac{\lambda}{n_{\text{bc}}}\sum_{j=1}^{n_{\text{bc}}} \nabla_w u(x_b^j;w_\star)\nabla_w u(x_b^j;w_\star)^T.$$

Define the maps $\mathcal{F}_{\text{res}}(w) = \begin{bmatrix} \mathcal{D}[u(x_r^1;w)] \\ \vdots \\ \mathcal{D}[u(x_r^{n_{\text{res}}};w)] \end{bmatrix}$, and $\mathcal{F}_{\text{bc}}(w) = \begin{bmatrix} u(x_b^1;w) \\ \vdots \\ u(x_b^{n_{\text{bc}}};w)] \end{bmatrix}$. We have the following important lemma, which follows via routine calculation.

Lemma F.2. *Let $n = n_{\text{res}} + n_{\text{bc}}$. Define the map $\mathcal{F} : \mathbb{R}^p \to \mathbb{R}^n$, by stacking $\mathcal{F}_{\text{res}}(w), \mathcal{F}_{\text{bc}}(w)$. Then, the Jacobian of $\mathcal{F}$ is given by*

$$J_{\mathcal{F}}(w) = \begin{bmatrix} J_{\mathcal{F}_{\text{res}}}(w) \\ J_{\mathcal{F}_{\text{bc}}}(w). \end{bmatrix}$$

Moreover, the tangent kernel $K_{\mathcal{F}}(w) = J_{\mathcal{F}}(w)J_{\mathcal{F}}(w)^T$ is given by

$$K_{\mathcal{F}}(w) = \begin{bmatrix} J_{\mathcal{F}_{\text{res}}}(w)J_{\mathcal{F}_{\text{res}}}(w)^T & J_{\mathcal{F}_{\text{res}}}(w)J_{\mathcal{F}_{\text{bc}}}(w)^T \\ J_{\mathcal{F}_{\text{bc}}}(w)J_{\mathcal{F}_{\text{res}}}(w)^T & J_{\mathcal{F}_{\text{bc}}}(w)J_{\mathcal{F}_{\text{bc}}}(w)^T \end{bmatrix} = \begin{bmatrix} K_{\mathcal{F}_{\text{res}}}(w) & J_{\mathcal{F}_{\text{res}}}(w)J_{\mathcal{F}_{\text{bc}}}(w)^T \\ J_{\mathcal{F}_{\text{bc}}}(w)J_{\mathcal{F}_{\text{res}}}(w)^T & K_{\mathcal{F}_{\text{bc}}}(w) \end{bmatrix}.$$

F.2. Relating $G_\infty(w)$ to $\mathcal{D}$

Isolate the population Gauss-Newton matrix for the residual:

$$G_\infty(w) = \int_\Omega \mathcal{D}[\nabla_w u(x;w)]\mathcal{D}[\nabla_w u(x;w)]^T d\mu(x).$$

Analogous to De Ryck et al. (2023) we define the functions $\phi_i(x;w) = \partial_{w_i} u(x;w)$ for $i \in \{1\ldots,p\}$. From this and the definition of $G_\infty(w)$, it follows that $(G_\infty(w))_{ij} = \langle \mathcal{D}[\phi_i], \mathcal{D}[\phi_j]\rangle_{L^2(\Omega)}$.

Similar to De Ryck et al. (2023) we can associate each $w \in \mathbb{R}^p$ with a space of functions $\mathcal{H}(w) = \text{span}\left(\phi_1(x;w),\ldots,\phi_p(x;w)\right) \subset L^2(\Omega)$. We also define two linear maps associated with $\mathcal{H}(w)$:

$$T(w)v = \sum_{i=1}^p v_i\phi_i(x;w),$$

$$T^*(w)f = \left(\langle f,\phi_1\rangle_{L^2(\Omega)},\ldots,\langle f,\phi_p\rangle_{L^2(\Omega)}\right).$$

From these definitions, we establish the following lemma.

Lemma F.3 (Characterizing $G_\infty(w)$)**.** *Define $\mathcal{A} = \mathcal{D}^*\mathcal{D}$. Then the matrix $G_\infty(w)$ satisfies*

$$G_\infty(w) = T^*(w)\mathcal{A}T(w).$$

Proof. Let e_i and e_j denote the ith and jth standard basis vectors in $\mathbb{R}^p$. Then,

$$\begin{aligned}(G_\infty(w))_{ij} &= \langle \mathcal{D}[\phi_i](w), \mathcal{D}[\phi_j](w)\rangle_{L^2(\Omega)} = \langle \phi_i(w), \mathcal{D}^*\mathcal{D}[\phi_j(w)]\rangle_{L^2(\Omega)} = \langle Te_i, \mathcal{D}^*\mathcal{D}[Te_j]\rangle_{L^2(\Omega)} \\ &= \langle e_i, (T^*\mathcal{D}^*\mathcal{D}T)[e_j]\rangle_{L^2(\Omega)},\end{aligned}$$

where the second equality follows from the definition of the adjoint. Hence, using $\mathcal{A} = \mathcal{D}^*\mathcal{D}$, we conclude $G_\infty(w) = T^*(w)\mathcal{A}T(w)$. □

Define the kernel integral operator $\mathcal{K}_\infty(w) : L^2(\Omega) \to \mathcal{H}$ by

$$\mathcal{K}_\infty(w)[f](x) = T(w)T^*(w)f = \sum_{i=1}^{p} \langle f, \phi_i(x;w)\rangle \phi_i(x;w), \tag{6}$$

and the kernel matrix $A(w)$ with entries $A_{ij}(w) = \langle \phi_i(x;w), \phi_j(x;w)\rangle_{L^2(\Omega)}$.

Using Lemma F.3 and applying the same logic as in the proof of Theorem 2.4 in De Ryck et al. (2023), we obtain the following theorem.

Theorem F.4. *Suppose that the matrix $A(w)$ is invertible. Then the eigenvalues of $G_\infty(w)$ satisfy*

$$\lambda_j(G_\infty(w)) = \lambda_j(\mathcal{A} \circ \mathcal{K}_\infty(w)), \quad \textit{for all } j \in [p].$$

F.3. $G_r(w)$ Concentrates Around $G_\infty(w)$

In order to relate the conditioning of the population objective to the empirical objective, we must relate the population Gauss-Newton residual matrix to its empirical counterpart. We accomplish this by showing $G_r(w)$ concentrates around $G_\infty(w)$. To this end, we recall the following variant of the intrinsic dimension matrix Bernstein inequality from Tropp (2015).

Theorem F.5 (Intrinsic Dimension Matrix Bernstein)**.** *Let $\{X_i\}_{i\in[n]}$ be a sequence of independent mean zero random matrices of the same size. Suppose that the following conditions hold:*

$$\|X_i\| \le B, \ \sum_{i=1}^{n} \mathbb{E}[X_i X_i^T] \preceq V_1, \ \sum_{i=1}^{n} \mathbb{E}[X_i^T X_i] \preceq V_2.$$

Define

$$\mathcal{V} = \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix}, \ \varsigma^2 = \max\{\|V_1\|, \|V_2\|\},$$

and the intrinsic dimension $d_{\text{int}} = \frac{\text{trace}(\mathcal{V})}{\|\mathcal{V}\|}$.
Then for all $t \ge \varsigma + \frac{B}{3}$,

$$\mathbb{P}\left(\left\|\sum_{i=1}^{n} X_i\right\| \ge t\right) \le 4 d_{\text{int}} \exp\left(-\frac{3}{8}\min\left\{\frac{t^2}{\varsigma^2}, \frac{t}{B}\right\}\right).$$

Next, we recall two key concepts from the kernel ridge regression literature and approximation via sampling literature: *γ-effective dimension* and *γ-ridge leverage coherence* (Bach, 2013; Cohen et al., 2017; Rudi et al., 2017).

Definition F.6 (γ-Effective dimension and γ-ridge leverage coherence)**.** Let $\gamma > 0$. Then the γ-effective dimension of $G_\infty(w)$ is given by

$$d_{\text{eff}}^{\gamma}(G_\infty(w)) = \text{trace}\left(G_\infty(w)(G_\infty(w) + \gamma I)^{-1}\right).$$

The γ-ridge leverage coherence is given by

$$\chi^\gamma(G_\infty(w)) = \sup_{x\in\Omega} \frac{\left\|(G_\infty(w) + \gamma I)^{-1/2}\mathcal{D}[\nabla_w u(x;w)]\right\|^2}{\mathbb{E}_{x\sim\mu}\left\|(G_\infty(w) + \gamma I)^{-1/2}\mathcal{D}[\nabla_w u(x;w)]\right\|^2} = \frac{\sup_{x\in\Omega}\left\|(G_\infty(w) + \gamma I)^{-1/2}\mathcal{D}[\nabla_w u(x;w)]\right\|^2}{d_{\text{eff}}^{\gamma}(G_\infty(w))}.$$

Observe that $d_{\text{eff}}^{\gamma}(G_\infty(w))$ only depends upon γ and w, while $\chi^\gamma(G_\infty(w))$ only depends upon γ, w, and Ω. Moreover, $\chi^\gamma(G_\infty(w)) < \infty$ as Ω is bounded.

We prove the following lemma using the γ-effective dimension and γ-ridge leverage coherence in conjunction with Theorem F.5.

Lemma F.7 (Finite-sample approximation)**.** *Let* $0 < \gamma < \lambda_1(G_\infty(w))$. *If* $n_{\text{res}} \geq 40\chi^\gamma(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w)) \log\left(\frac{8d_{\text{eff}}^{\gamma}(G_\infty(w))}{\delta}\right)$*, then with probability at least* $1-\delta$

$$\frac{1}{2}\left[G_\infty(w) - \gamma I\right] \preceq G_r(w) \preceq \frac{1}{2}\left[3G_\infty(w) + \gamma I.\right]$$

Proof. Let $x_i = (G_\infty(w)+\gamma I)^{-1/2}\mathcal{D}[\nabla_w u(x_i;w)]$, and $X_i = \frac{1}{n_{\text{res}}}\left(x_i x_i^T - D_\gamma\right)$, where $D_\gamma = G_\infty(w)\left(G_\infty(w)+\gamma I\right)^{-1}$. Clearly, $\mathbb{E}[X_i] = 0$. Moreover, the X_i's are bounded as

$$\begin{aligned}\|X_i\| &= \max\left\{\frac{\lambda_{\max}(X_i)}{n_{\text{res}}}, -\frac{\lambda_{\min}(X_i)}{n_{\text{res}}}\right\} \leq \max\left\{\frac{\|x_i\|^2}{n_{\text{res}}}, \frac{\lambda_{\max}(-X_i)}{n_{\text{res}}}\right\} \leq \max\left\{\frac{\chi^\gamma(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w))}{n_{\text{res}}}, \frac{1}{n_{\text{res}}}\right\} \\ &= \frac{\chi^\gamma(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w))}{n_{\text{res}}}.\end{aligned}$$

Thus, it remains to verify the variance condition. We have

$$\begin{aligned}\sum_{i=1}^{n_{\text{res}}}\mathbb{E}[X_i X_i^T] = n_{\text{res}}\mathbb{E}[X_1^2] &= n_{\text{res}} \times \frac{1}{n_{\text{res}}^2}\mathbb{E}[(x_1x_1^T - D_\gamma)^2] \preceq \frac{1}{n_{\text{res}}}\mathbb{E}[\|x_1\|^2 x_1 x_1^T] \\ &\preceq \frac{\chi^\gamma(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w))}{n_{\text{res}}}D_\gamma.\end{aligned}$$

Hence, the conditions of Theorem F.5 hold with $B = \frac{\chi^\gamma(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w))}{n_{\text{res}}}$ and $V_1 = V_2 = \frac{\chi^\gamma(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w))}{n_{\text{res}}}D_\gamma$. Now $1/2 \leq \|\mathcal{V}\| \leq 1$ as $n_{\text{res}} \geq \chi^\gamma(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w))$ and $\gamma \leq \lambda_1\left(G_\infty(w)\right)$. Moreover, as $V_1 = V_2$ we have $d_{\text{int}} \leq 4d_{\text{eff}}^{\gamma}(G_\infty(w))$. So, setting

$$t = \sqrt{\frac{8\chi^\gamma(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w))\log\left(\frac{8d_{\text{eff}}^{\gamma}(G_\infty(w))}{\delta}\right)}{3n_{\text{res}}}} + \frac{8\chi^\gamma(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w))\log\left(\frac{8d_{\text{eff}}^{\gamma}(G_\infty(w))}{\delta}\right)}{3n_{\text{res}}}$$

and using $n_{\text{res}} \geq 40\chi^\gamma(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w))\log\left(\frac{8d_{\text{eff}}^{\gamma}(G_\infty(w))}{\delta}\right)$, we conclude

$$\mathbb{P}\left(\left\|\sum_{i=1}^{n_{\text{res}}} X_i\right\| \geq \frac{1}{2}\right) \leq \delta.$$

Now, $\|\sum_{i=1}^{n_{\text{res}}} X_i\| \leq \frac{1}{2}$ implies

$$-\frac{1}{2}\left[G_\infty(w) + \gamma I\right] \preceq G_r(w) - G_\infty(w) \preceq \frac{1}{2}\left[G_\infty(w) + \gamma I\right].$$

The claim now follows by rearrangement. □

By combining Theorem F.4 and Lemma F.7, we show that if the spectrum of $\mathcal{A} \circ \mathcal{K}_\infty(w)$ decays, then the spectrum of the empirical Gauss-Newton matrix also decays with high probability.

Proposition F.8 (Spectrum of empirical Gauss-Newton matrix decays fast)**.** *Suppose the eigenvalues of* $\mathcal{A} \circ \mathcal{K}_\infty(w)$ *satisfy* $\lambda_j(\mathcal{A} \circ \mathcal{K}_\infty(w)) \leq Cj^{-2\alpha}$*, where* $\alpha > 1/2$ *and* $C > 0$ *is some absolute constant. Then if* $\sqrt{n_{\text{res}}} \geq 40C_1\chi^\gamma(G_\infty(w))\log\left(\frac{1}{\delta}\right)$*, for some absolute constant* C_1*, it holds that*

$$\lambda_{n_{\text{res}}}(G_r(w)) \leq n_{\text{res}}^{-\alpha}$$

with probability at least $1-\delta$*.*

Proof. The hypotheses on the decay of the eigenvalues implies $d_{\text{eff}}^{\gamma}(G_\infty(w)) \le C_1\gamma^{-\frac{1}{2\alpha}}$ (see Appendix C of Bach (2013)). Consequently, given $\gamma = n_{\text{res}}^{-\alpha}$, we have $d_{\text{eff}}^{\gamma}(G_\infty(w)) \le C_1 n_{\text{res}}^{\frac{1}{2}}$. Combining this with our hypotheses on n_{res}, it follows $n_{\text{res}} \ge 40C_1\chi^{\gamma}(G_\infty(w))d_{\text{eff}}^{\gamma}(G_\infty(w))\log\left(\frac{8d_{\text{eff}}^{\gamma}(G_\infty(w))}{\delta}\right)$. Hence Lemma F.7 implies with probability at least $1-\delta$ that

$$G_r(w) \preceq \frac{1}{2}\left(3G_\infty(w) + \gamma I\right),$$

which yields for any $1 \le r \le n$

$$\lambda_{n_{\text{res}}}(G_r(w)) \le \frac{1}{2}\left(3\lambda_r(G_\infty(w)) + \gamma\right).$$

Combining the last display with $n_{\text{res}} \ge 3d_{\text{eff}}^{\gamma}(G_\infty(w))$, Lemma 5.4 of Frangella et al. (2023) guarantees $\lambda_r(G_\infty(w)) \le \gamma/3$, and so

$$\lambda_{n_{\text{res}}}(G_r(w)) \le \frac{1}{2}\left(3\lambda_r(G_\infty(w)) + \gamma\right) \le \gamma \le n_{\text{res}}^{-\alpha}.$$

□

F.4. Formal Statement of Theorem 8.4 and Proof

Theorem F.9 (An ill-conditioned differential operator leads to hard optimization)**.** *Fix $w_\star \in \mathcal{W}_\star$, and let $\mathcal{S}$ be a set containing $w_\star$ for which $\mathcal{S}$ is μ-PŁ*. Let $\alpha > 1/2$. If the eigenvalues of $\mathcal{A}\circ\mathcal{K}_\infty(w_\star)$ satisfy $\lambda_j(\mathcal{A}\circ\mathcal{K}_\infty(w_\star)) \le Cj^{-2\alpha}$ and $\sqrt{n_{\text{res}}} \ge 40C_1\chi^{\gamma}(G_\infty(w_\star))\log\left(\frac{1}{\delta}\right)$, then*

$$\kappa_L(\mathcal{S}) \ge C_2 n_{\text{res}}^{\alpha},$$

with probability at least $1-\delta$. Here C, C_1, and C_2 are absolute constants.

Proof. By the assumption on n_{res}, the conditions of Proposition F.8 are met, so,

$$\lambda_{n_{\text{res}}}(G_r(w_\star)) \le n_{\text{res}}^{-\alpha}.$$

with probability at least $1-\delta$. By definition $G_r(w_\star) = J_{\mathcal{F}_{\text{res}}}(w_\star)^T J_{\mathcal{F}_{\text{res}}}(w_\star)$, consequently,

$$\lambda_{n_{\text{res}}}(K_{\mathcal{F}_{\text{res}}}(w_\star)) = \lambda_{n_{\text{res}}}(G_r(w_\star)) \le n_{\text{res}}^{-\alpha}.$$

Now, the PŁ*-constant for $\mathcal{S}$, satisfies $\mu = \inf_{w\in\mathcal{S}}\lambda_n(K_{\mathcal{F}}(w))$ (Liu et al., 2022). Combining this with the expression for $K_{\mathcal{F}}(w_\star)$ in Lemma F.2, we reach

$$\mu \le \lambda_n(K_{\mathcal{F}}(w_\star)) \le \lambda_{n_{\text{res}}}(K_{\mathcal{F}_{\text{res}}}(w_\star)) \le n_{\text{res}}^{-\alpha},$$

where the second inequality follows from Cauchy's Interlacing theorem. Recalling that $\kappa_L(\mathcal{S}) = \frac{\sup_{w\in\mathcal{S}}\|H_L(w)\|}{\mu}$, and $H_L(w_\star)$ is symmetric psd, we reach

$$\kappa_L(\mathcal{S}) \ge \frac{\lambda_1(H_L(w_\star))}{\mu} \overset{(1)}{\ge} \frac{\lambda_1(G_r(w_\star)) + \lambda_p(G_b(w_\star))}{\mu} \overset{(2)}{=} \frac{\lambda_1(G_r(w_\star))}{\mu} \overset{(3)}{\ge} C_3\lambda_1(G_\infty(w_\star))n_{\text{res}}^{\alpha}.$$

Here (1) uses $H_L(w_\star) = G_r(w_\star) + G_b(w_\star)$ and Weyl's inequalities, (2) uses $p \ge n_{\text{res}} + n_{\text{bc}}$, so that $\lambda_p(G_b(w_\star)) = 0$. Inequality (3) uses the upper bound on μ and the lower bound on $G_r(w)$ given in Lemma F.7. Hence, the claim follows with $C_2 = C_3\lambda_1(G_\infty(w_\star))$. □

F.5. κ Grows with the Number of Residual Points

Figure 10 plots the ratio $\lambda_1(H_L)/\lambda_{129}(H_L)$ near a minimizer $w_\star$. This ratio is a lower bound for the condition number of H_L, and is computationally tractable to compute. We see that the estimate of the κ grows polynomially with n_{res}, which provides empirical verification for Theorem 8.4.

G. Convergence of GDND (Algorithm 1)

In this section, we provide the formal version of Theorem 8.5 and its proof. However, this is delayed till Appendix G.4, as the theorem is a consequence of a series of results. Before jumping to the theorem, we recommend reading the statements in the preceding subsections to understand the statement and corresponding proof.

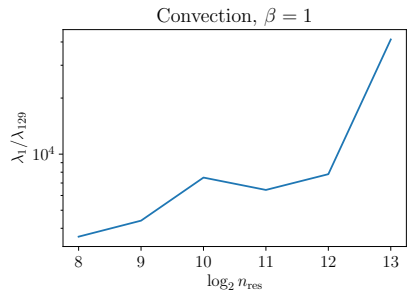

Figure 10. Estimated condition number after 41000 iterations of Adam+L-BFGS with different number of residual points from 255×100 grid on the interior. Here λ_i denotes the ith largest eigenvalue of the Hessian. The model has 2 layers and the hidden layer has width 32. The plot shows κ_L grows polynomially in the number of residual points.

G.1. Overview and Notation

Recall, we are interested in minimizing the objective in (2):

$$L(w) = \frac{1}{2n_{\text{res}}} \sum_{i=1}^{n_{\text{res}}} \left(\mathcal{D}[u(x_r^i; w)]\right)^2 + \frac{1}{2n_{\text{bc}}} \sum_{j=1}^{n_{\text{bc}}} \left(\mathcal{B}[u(x_b^j; w)]\right)^2,$$

where $\mathcal{D}$ is the differential operator defining the PDE and $\mathcal{B}$ is the operator defining the boundary conditions. Define

$$\mathcal{F}(w) = \begin{bmatrix} \frac{1}{\sqrt{n_{\text{res}}}}\mathcal{D}[u(x_r^1; w)] \\ \vdots \\ \frac{1}{\sqrt{n_{\text{res}}}}\mathcal{D}[u(x_r^{n_{\text{res}}}; w)] \\ \frac{1}{\sqrt{n_{\text{bc}}}}\mathcal{B}[u(x_b^1; w)] \\ \vdots \\ \frac{1}{\sqrt{n_{\text{bc}}}}\mathcal{B}[u(x_b^{n_{\text{bc}}}; w)] \end{bmatrix}, \; y = 0$$

Using the preceding definitions, our objective may be rewritten as:

$$L(w) = \frac{1}{2}\|\mathcal{F}(w) - y\|^2.$$

Throughout the appendix, we work with the condensed expression for the loss given above. We denote the $(n_{\text{res}} + n_{\text{bc}}) \times p$ Jacobian matrix of $\mathcal{F}$ by $J_{\mathcal{F}}(w)$. The tangent kernel at w is given by the $n \times n$ matrix $K_{\mathcal{F}}(w) = J_{\mathcal{F}}(w)J_{\mathcal{F}}(w)^T$. The closely related Gauss-Newton matrix is given by $G(w) = J_{\mathcal{F}}(w)^T J_{\mathcal{F}}(w)$.

G.2. Global Behavior: Reaching a Small Ball About a Minimizer

We begin by showing that under appropriate conditions, gradient descent outputs a point close to a minimizer after a fixed number of iterations. We first start with the following assumption which is common in the neural network literature (Liu et al., 2022; 2023).

Assumption G.1. The mapping $\mathcal{F}(w)$ is $\mathcal{L}_{\mathcal{F}}$-Lipschitz, and the loss $L(w)$ is β_L-smooth.

Under Assumption G.1 and a PŁ*-condition, we have the following theorem of Liu et al. (2022), which shows gradient descent converges linearly.

Theorem G.2. *Let w_0 denote the network weights at initialization. Suppose Assumption G.1 holds, and that $L(w)$ is μ-PŁ* in $B(w_0, 2R)$ with $R = \frac{2\sqrt{2\beta_L L(w_0)}}{\mu}$. Then the following statements hold:*

1. *The intersection $B(w_0, R) \cap \mathcal{W}_\star$ is non-empty.*

2. *Gradient descent with step size $\eta = 1/\beta_L$ satisfies:*

$$w_{k+1} = w_k - \eta \nabla L(w_k) \in B(w_0, R) \text{ for all } k \geq 0,$$
$$L(w_k) \leq \left(1 - \frac{\mu}{\beta_L}\right)^k L(w_0).$$

For wide neural neural networks, it is known that the μ-PŁ*condition in Theorem G.2 hold with high probability, see Liu et al. (2022) for details.

We also recall the following lemma from Liu et al. (2023).

Lemma G.3 (Descent Principle)**.** *Let $L : \mathbb{R}^p \mapsto [0, \infty)$ be differentiable and μ-PŁ*in the ball $B(w, r)$. Suppose $L(w) < \frac{1}{2}\mu r^2$. Then the intersection $B(w, r) \cap \mathcal{W}_\star$ is non-empty, and*

$$\frac{\mu}{2}\text{dist}^2(w, \mathcal{W}_\star) \leq L(w).$$

Let $\mathcal{L}_{H_L}$ be the Hessian Lipschitz constant in $B(w_0, 2R)$, and $\mathcal{L}_{J_\mathcal{F}} = \sup_{w \in B(w_0, 2R)} \|H_\mathcal{F}(w)\|$, where $\|H_\mathcal{F}(w)\| = \max_{i \in [n]} \|H_{\mathcal{F}_i}(w)\|$. Define $M = \max\{\mathcal{L}_{H_L}, \mathcal{L}_{J_\mathcal{F}}, \mathcal{L}_\mathcal{F}\mathcal{L}_{J_\mathcal{F}}, 1\}$, $\varepsilon_{\text{loc}} = \frac{\varepsilon \mu^{3/2}}{4M}$, where $\varepsilon \in (0, 1)$. By combining Theorem G.2 and Lemma G.3, we are able to establish the following important corollary, which shows gradient descent outputs a point close to a minimizer.

Corollary G.4 (Getting close to a minimizer)**.** *Set $\rho = \min\left\{\frac{\varepsilon_{\text{loc}}}{19\sqrt{\frac{\beta_L}{\mu}}}, \sqrt{\mu}R, R\right\}$. Run gradient descent for $k = \frac{\beta_L}{\mu} \log\left(\frac{4 \max\{2\beta_L, 1\} L(w_0)}{\mu \rho^2}\right)$ iterations, gradient descent outputs a point w_{loc} satisfying*

$$L(w_{\text{loc}}) \leq \frac{\mu \rho^2}{4} \min\left\{1, \frac{1}{2\beta_L}\right\},$$

$$\|w_{\text{loc}} - w_\star\|_{H_L(w_\star) + \mu I} \leq \rho, \text{ for some } w_\star \in \mathcal{W}_\star.$$

Proof. The first claim about $L(w_{\text{loc}})$ is an immediate consequence of Theorem G.2. For the second claim, consider the ball $B(w_{\text{loc}}, \rho)$. Observe that $B(w_{\text{loc}}, \rho) \subset B(w_0, 2R)$, so L is μ-PŁ* in $B(w_{\text{loc}}, \rho)$. Combining this with $L(w_{\text{loc}}) \leq \frac{\mu \rho^2}{4}$, Lemma G.3 guarantees the existence of $w_\star \in B(w_{\text{loc}}, \rho) \cap \mathcal{W}_\star$, with $\|w_{\text{loc}} - w_\star\| \leq \sqrt{\frac{2}{\mu} L(w_{\text{loc}})}$. Hence Cauchy-Schwarz yields

$$\begin{aligned}\|w_{\text{loc}} - w_\star\|_{H_L(w_\star) + \mu I} &\leq \sqrt{\beta_L + \mu}\|w_{\text{loc}} - w_\star\| \leq \sqrt{2\beta_L}\|w_{\text{loc}} - w_\star\| \\ &\leq 2\sqrt{\frac{\beta_L}{\mu} L(w_{\text{loc}})} \leq 2 \times \sqrt{\frac{\beta_L}{\mu} \frac{\mu \rho^2}{8\beta_L}} \leq \rho,\end{aligned}$$

which proves the claim. □

G.3. Fast Local Convergence of Damped Newton's Method

In this section, we show damped Newton's method with fixed stepsize exhibits fast linear convergence in an appropriate region about the minimizer $w_\star$ from Corollary G.4. Fix $\varepsilon \in (0, 1)$, then the region of local convergence is given by:

$$\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star) = \left\{w \in \mathbb{R}^p : \|w - w_\star\|_{H_L(w_\star) + \mu I} \leq \varepsilon_{\text{loc}}\right\},$$

where $\varepsilon_{\text{loc}} = \frac{\varepsilon \mu^{3/2}}{4M}$ as above. Note that $w_{\text{loc}} \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$.

We now prove several lemmas, that are essential to the argument. We begin with the following elementary technical result, which shall be used repeatedly below.

Lemma G.5 (Sandwich lemma)**.** *Let A be a symmetric matrix and B be a symmetric positive-definite matrix. Suppose that A and B satisfy $\|A - B\| \leq \varepsilon\lambda_{\text{min}}(B)$ where $\varepsilon \in (0, 1)$. Then*

$$(1-\varepsilon)B \preceq A \preceq (1+\varepsilon)B.$$

Proof. By hypothesis, it holds that

$$-\varepsilon\lambda_{\text{min}}(B)I \preceq A - B \preceq \varepsilon\lambda_{\text{min}}(B)I.$$

So using $B \succeq \lambda_{\text{min}}(B)I$, and adding B to both sides, we reach

$$(1-\varepsilon)B \preceq A \preceq (1+\varepsilon)B.$$

□

The next result describes the behavior of the damped Hessian in $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$.

Lemma G.6 (Damped Hessian in $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$)**.** *Suppose that $\gamma \geq \mu$ and $\varepsilon \in (0, 1)$.*

1. *(Positive-definiteness of damped Hessian in $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$) For any $w \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$,*

$$H_L(w) + \gamma I \succeq \left(1 - \frac{\varepsilon}{4}\right)\gamma I.$$

2. *(Damped Hessians stay close in $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$) For any $w, w' \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$,*

$$(1-\varepsilon)\left[H_L(w) + \gamma I\right] \preceq H_L(w') + \gamma I \preceq (1+\varepsilon)\left[H_L(w) + \gamma I\right].$$

Proof. We begin by observing that the damped Hessian at $w_\star$ satisfies

$$\begin{aligned} H_L(w_\star) + \gamma I &= G(w_\star) + \gamma I + \frac{1}{n}\sum_{i=1}^{n}\left[\mathcal{F}(w_\star) - y\right]_i H_{\mathcal{F}_i}(w_\star) \\ &= G(w_\star) + \gamma I \succeq \gamma I. \end{aligned}$$

Thus, $H_L(w_\star) + \gamma I$ is positive definite. Now, for any $w \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$, it follows from Lipschitzness of H_L that

$$\|(H_L(w) + \gamma I) - (H_L(w_\star) + \gamma I)\| \leq \mathcal{L}_{H_L}\|w - w_\star\| \leq \frac{\mathcal{L}_{H_L}}{\sqrt{\gamma}}\|w - w_\star\|_{H_L(w_\star)+\gamma I} \leq \frac{\varepsilon\mu}{4}.$$

As $\lambda_{\text{min}}\left(H_L(w_\star) + \gamma I\right) \geq \gamma > \mu$, we may invoke Lemma G.5 to reach

$$\left(1 - \frac{\varepsilon}{4}\right)\left[H_L(w_\star) + \gamma I\right] \preceq H_L(w) + \gamma I \preceq \left(1 + \frac{\varepsilon}{4}\right)\left[H_L(w_\star) + \gamma I\right].$$

This immediately yields

$$\lambda_{\text{min}}\left(H_L(w) + \gamma I\right) \geq \left(1 - \frac{\varepsilon}{4}\right)\gamma \geq \frac{3}{4}\gamma,$$

which proves item 1. To see the second claim, observe for any $w, w' \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$ the triangle inequality implies

$$\|(H_L(w') + \gamma I) - (H_L(w) + \gamma I)\| \leq \frac{\varepsilon\mu}{2} \leq \frac{2}{3}\varepsilon\left(\frac{3}{4}\gamma\right).$$

As $\lambda_{\text{min}}\left(H_L(w) + \gamma I\right) \geq \frac{3}{4}\gamma$, it follows from Lemma G.5 that

$$\left(1 - \frac{2}{3}\varepsilon\right)\left[H_L(w) + \gamma I\right] \preceq H_L(w') + \gamma I \preceq \left(1 + \frac{2}{3}\varepsilon\right)\left[H_L(w) + \gamma I\right],$$

which establishes item 2. □

The next result characterizes the behavior of the tangent kernel and Gauss-Newton matrix in $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$.

Lemma G.7 (Tangent kernel and Gauss-Newton matrix in $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$)**.** *Let $\gamma \geq \mu$. Then for any $w, w' \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$, the following statements hold:*

1. *(Tangent kernels stay close)*
$$\left(1 - \frac{\varepsilon}{2}\right) K_{\mathcal{F}}(w_\star) \preceq K_{\mathcal{F}}(w) \preceq \left(1 + \frac{\varepsilon}{2}\right) K_{\mathcal{F}}(w_\star)$$
2. *(Gauss-Newton matrices stay close)*
$$\left(1 - \frac{\varepsilon}{2}\right) [G(w) + \gamma I] \preceq G(w_\star) + \gamma I \preceq \left(1 + \frac{\varepsilon}{2}\right) [G(w) + \gamma I]$$
3. *(Damped Hessian is close to damped Gauss-Newton matrix)*
$$(1 - \varepsilon) [G(w) + \gamma I] \preceq H_L(w) + \gamma I \preceq (1 + \varepsilon) [G(w) + \gamma I].$$
4. *(Jacobian has full row-rank) The Jacobian satisfies* $\text{rank}(J_{\mathcal{F}}(w)) = n$.

Proof. 1. Observe that

$$\begin{aligned}
\|K_{\mathcal{F}}(w) - K_{\mathcal{F}}(w_\star)\| &= \|J_{\mathcal{F}}(w) J_{\mathcal{F}}(w)^T - J_{\mathcal{F}}(w_\star) J_{\mathcal{F}}(w_\star)^T\| \\
&= \left\| [J_{\mathcal{F}}(w) - J_{\mathcal{F}}(w_\star)] J_{\mathcal{F}}(w)^T + J_{\mathcal{F}}(w_\star) [J_{\mathcal{F}}(w) - J_{\mathcal{F}}(w_\star)]^T \right\| \\
&\leq 2\mathcal{L}_{\mathcal{F}} \mathcal{L}_{J_{\mathcal{F}}} \|w - w_\star\| \leq \frac{2\mathcal{L}_{\mathcal{F}} \mathcal{L}_{J_{\mathcal{F}}}}{\sqrt{\gamma}} \|w - w_\star\|_{H_L(w_\star) + \gamma I} \leq \frac{\varepsilon \mu^{3/2}}{\sqrt{\gamma}} \leq \frac{\varepsilon}{2} \mu,
\end{aligned}$$

where in the first inequality we applied the fundamental theorem of calculus to reach

$$\|J_{\mathcal{F}}(w) - J_{\mathcal{F}}(w_\star)\| \leq \mathcal{L}_{J_{\mathcal{F}}} \|w - w_\star\|.$$

Hence the claim follows from Lemma G.5.

2. By an analogous argument to item 1, we find

$$\|(G(w) + \gamma I) - (G(w_\star) + \gamma I)\| \leq \frac{\varepsilon}{2} \mu,$$

so the result again follows from Lemma G.5.

3. First observe $H_L(w_\star) + \gamma I = G(w_\star) + \gamma I$. Hence the proof of Lemma G.6 implies,

$$\left(1 - \frac{\varepsilon}{4}\right) [G(w_\star) + \gamma I] \preceq H_L(w) + \gamma I \preceq \left(1 + \frac{\varepsilon}{4}\right) [G(w_\star) + \gamma I].$$

Hence the claim now follows from combining the last display with item 2.

4. This last claim follows immediately from item 1, as for any $w \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$,

$$\sigma_n (J_{\mathcal{F}}(w)) = \sqrt{\lambda_{\min}(K_{\mathcal{F}}(w))} \geq \sqrt{\left(1 - \frac{\varepsilon}{2}\right) \mu} > 0.$$

Here the last inequality uses $\lambda_{\min}(K_{\mathcal{F}}(w_\star)) \geq \mu$, which follows as $w_\star \in B(w_0, 2R)$.

□

The next lemma is essential to proving convergence. It shows in $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$ that $L(w)$ is uniformly smooth with respect to the damped Hessian, with nice smoothness constant $(1 + \varepsilon)$. Moreover, it establishes that the loss is uniformly PŁ*with respect to the damped Hessian in $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$.

Lemma G.8 (Preconditioned smoothness and PŁ*)**.** *Suppose $\gamma \geq \mu$. Then for any $w, w', w'' \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$, the following statements hold:*

1. $L(w'') \leq L(w') + \langle \nabla L(w'), w'' - w' \rangle + \frac{1+\varepsilon}{2} \|w'' - w'\|^2_{H_L(w)+\gamma I}$.
2. $\frac{\|\nabla L(w)\|^2_{(H_L(w)+\gamma I)^{-1}}}{2} \geq \frac{1}{1+\varepsilon} \frac{1}{(1+\gamma/\mu)} L(w)$.

Proof. 1. By Taylor's theorem

$$L(w'') = L(w') + \langle \nabla L(w'), w'' - w' \rangle + \int_0^1 (1-t) \|w'' - w'\|^2_{H_L(w'+t(w''-w'))} dt$$

Note $w' + t(w'' - w') \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$ as $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$ is convex. Thus we have,

$$\begin{aligned} L(w'') &\leq L(w') + \langle \nabla L(w'), w'' - w' \rangle + \int_0^1 (1-t) \|w'' - w'\|^2_{H_L(w'+t(w''-w'))+\gamma I} dt \\ &\leq L(w') + \langle \nabla L(w'), w'' - w' \rangle + \int_0^1 (1-t)(1+\varepsilon) \|w'' - w'\|^2_{H_L(w)+\gamma I} dt \\ &= L(w') + \langle \nabla L(w'), w'' - w' \rangle + \frac{(1+\varepsilon)}{2} \|w'' - w'\|^2_{H_L(w)+\gamma I}. \end{aligned}$$

2. Observe that

$$\frac{\|\nabla L(w)\|^2_{(H_L(w)+\gamma I)^{-1}}}{2} = \frac{1}{2} (\mathcal{F}(w) - y)^T \left[J_{\mathcal{F}}(w) \left(H_L(w) + \gamma I \right)^{-1} J_{\mathcal{F}}(w)^T \right] (\mathcal{F}(w) - y).$$

Now,

$$\begin{aligned} J_{\mathcal{F}}(w) \left(H_L(w) + \gamma I \right)^{-1} J_{\mathcal{F}}(w)^T &\succeq \frac{1}{(1+\varepsilon)} J_{\mathcal{F}}(w) \left(G(w) + \gamma I \right)^{-1} J_{\mathcal{F}}(w)^T \\ &= \frac{1}{(1+\varepsilon)} J_{\mathcal{F}}(w) \left(J_{\mathcal{F}}(w)^T J_{\mathcal{F}}(w) + \gamma I \right)^{-1} J_{\mathcal{F}}(w)^T \end{aligned}$$

Lemma G.7 guarantees $J_{\mathcal{F}}(w)$ has full row-rank, so the SVD yields

$$J_{\mathcal{F}}(w) \left(J_{\mathcal{F}}(w)^T J_{\mathcal{F}}(w) + \gamma I \right)^{-1} J_{\mathcal{F}}(w)^T = U \Sigma^2 (\Sigma^2 + \gamma I)^{-1} U^T \succeq \frac{\mu}{\mu + \gamma} I.$$

Hence

$$\frac{\|\nabla L(w)\|^2_{(H_L(w)+\gamma I)^{-1}}}{2} \geq \frac{\mu}{(1+\varepsilon)(\mu+\gamma)} \frac{1}{2} \|\mathcal{F}(w) - y\|^2 = \frac{\mu}{(1+\varepsilon)(\mu+\gamma)} L(w).$$

□

Lemma G.9 (Local preconditioned-descent)**.** *Run Phase II of Algorithm 1 with $\eta_{\text{DN}} = (1+\varepsilon)^{-1}$ and $\gamma = \mu$. Suppose that $\tilde{w}_k, \tilde{w}_{k+1} \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$, then*

$$L(\tilde{w}_{k+1}) \leq \left(1 - \frac{1}{2(1+\varepsilon)^2} \right) L(\tilde{w}_k).$$

Proof. As $\tilde{w}_k, \tilde{w}_{k+1} \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$, item 1 of Lemma G.8 yields

$$L(\tilde{w}_{k+1}) \leq L(\tilde{w}_k) - \frac{\|\nabla L(\tilde{w}_k)\|^2_{(H_L(\tilde{w}_k)+\mu I)^{-1}}}{2(1+\varepsilon)}.$$

Combining the last display with the preconditioned PŁ*condition, we conclude

$$L(\tilde{w}_{k+1}) \leq \left(1 - \frac{1}{2(1+\varepsilon)^2} \right) L(\tilde{w}_k).$$

□

The following lemma describes how far an iterate moves after one-step of Phase II of Algorithm 1.

Lemma G.10 (1-step evolution)**.** *Run Phase II of Algorithm 1 with* $\eta_{\text{DN}} = (1+\varepsilon)^{-1}$ *and* $\gamma \geq \mu$*. Suppose* $\tilde{w}_k \in \mathcal{N}_{\frac{\varepsilon}{3}}(w_\star)$*, then* $\tilde{w}_{k+1} \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$*.*

Proof. Let $P = H_L(\tilde{w}_k) + \gamma I$. We begin by observing that

$$\|\tilde{w}_{k+1} - w_\star\|_{H_L(w_\star)+\mu I} \leq \sqrt{1+\varepsilon}\|\tilde{w}_{k+1} - w_\star\|_P.$$

Now,

$$\begin{aligned}
\|\tilde{w}_{k+1} - w_\star\|_P &= \frac{1}{1+\varepsilon}\|\nabla L(\tilde{w}_k) - \nabla L(w_\star) - (1+\varepsilon)P(w_\star - \tilde{w}_k)\|_{P^{-1}} \\
&= \frac{1}{1+\varepsilon}\left\|\int_0^1 \left[\nabla^2 L(w_\star + t(w_k - w_\star)) - (1+\varepsilon)P\right] dt(w_\star - \tilde{w}_k)\right\|_{P^{-1}} \\
&= \frac{1}{1+\varepsilon}\left\|\int_0^1 \left[P^{-1/2}\nabla^2 L(w_\star + t(w_k - w_\star))P^{-1/2} - (1+\varepsilon)I\right] dt P^{1/2}(w_\star - \tilde{w}_k)\right\| \\
&\leq \frac{1}{1+\varepsilon}\int_0^1 \left\|P^{-1/2}\nabla^2 L(w_\star + t(w_k - w_\star))P^{-1/2} - (1+\varepsilon)I\right\| dt\|\tilde{w}_k - w_\star\|_P
\end{aligned}$$

We now analyze the matrix $P^{-1/2}\nabla^2 L(w_\star + t(w_k - w_\star))P^{-1/2}$. Observe that

$$\begin{aligned}
&P^{-1/2}\nabla^2 L(w_\star + t(w_k - w_\star))P^{-1/2} = P^{-1/2}(\nabla^2 L(w_\star + t(w_k - w_\star)) + \gamma I - \gamma I)P^{-1/2} \\
&= P^{-1/2}(\nabla^2 L(w_\star + t(w_k - w_\star)) + \gamma I)P^{-1/2} - \gamma P^{-1} \succeq (1-\varepsilon)I - \gamma P^{-1} \succeq -\varepsilon I.
\end{aligned}$$

Moreover,

$$P^{-1/2}\nabla^2 L(w_\star + t(w_k - w_\star))P^{-1/2} \preceq P^{-1/2}(\nabla^2 L(w_\star + t(w_k - w_\star)) + \gamma I)P^{-1/2} \preceq (1+\varepsilon)I.$$

Hence,

$$0 \preceq (1+\varepsilon)I - P^{-1/2}\nabla^2 L(w_\star + t(w_k - w_\star))P^{-1/2} \preceq (1+2\varepsilon)I,$$

and so

$$\|\tilde{w}^{k+1} - w_\star\|_P \leq \frac{1+2\varepsilon}{1+\varepsilon}\|\tilde{w}_k - w_\star\|_P.$$

Thus,

$$\|\tilde{w}^{k+1} - w_\star\|_{H_L(w_\star)+\mu I} \leq \frac{1+2\varepsilon}{\sqrt{1+\varepsilon}}\|\tilde{w}_k - w_\star\|_P \leq (1+2\varepsilon)\|\tilde{w}_k - w_\star\|_{H_L(w_\star)+\mu I} \leq \varepsilon_{\text{loc}}.$$

□

The following lemma is key to establishing fast local convergence; it shows that the iterates produced by damped Newton's method remain in $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$, the region of local convergence.

Lemma G.11 (Staying in $\mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$)**.** *Suppose that* $w_{\text{loc}} \in \mathcal{N}_\rho(w_\star)$*, where* $\rho = \frac{\varepsilon_{\text{loc}}}{19\sqrt{\beta_L/\mu}}$*. Run Phase II of Algorithm 1 with* $\gamma = \mu$ *and* $\eta = (1+\varepsilon)^{-1}$*, then* $\tilde{w}_{k+1} \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$ *for all* $k \geq 1$*.*

Proof. In the argument that follows $\kappa_P = 2(1+\varepsilon)^2$. The proof is via induction. Observe that if $w_{\text{loc}} \in \mathcal{N}_\varrho(w_\star)$ then by Lemma G.10, $\tilde{w}_1 \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$. Now assume $\tilde{w}_j \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$ for $j = 2, \ldots, k$. We shall show $\tilde{w}_{k+1} \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$. To this end, observe that

$$\|\tilde{w}_{k+1} - w_\star\|_{H_L(w_\star)+\mu I} \leq \|w_{\text{loc}} - w_\star\|_{H_L(w_\star)+\mu I} + \frac{1}{1+\varepsilon}\sum_{j=1}^{k}\|\nabla L(w_j)\|_{(H_L(w_\star)+\mu I)^{-1}}$$

Now,

$$\|\nabla L(w_j)\|_{(H_L(w_\star)+\mu I)^{-1}} \leq \frac{1}{\sqrt{\mu}}\|\nabla L(w_j)\|_2 \leq \sqrt{\frac{2\beta_L}{\mu}L(w_j)}$$
$$\leq \sqrt{\frac{2\beta_L}{\mu}}\left(1-\frac{1}{\kappa_P}\right)^{j/2}\sqrt{L(w_{\text{loc}})},$$

Here the second inequality follows from $\|\nabla L(w)\| \leq \sqrt{2\beta_L L(w)}$, and the last inequality follows from Lemma G.9, which is applicable as $\tilde{w}_0, \ldots, \tilde{w}_k \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$. Thus,

$$\|\tilde{w}_{k+1} - w_\star\|_{H_L(w_\star)+\mu I} \leq \rho + \sqrt{\frac{2\beta_L}{\mu}}\sum_{j=1}^{k}\left(1-\frac{1}{\kappa_P}\right)^{j/2}\sqrt{L(\tilde{w}_0)}$$
$$\leq \rho + \sqrt{\frac{(1+\varepsilon)\beta_L}{2\mu}}\|w_{\text{loc}} - w_\star\|_{H_L(w_\star)+\mu I}\sum_{j=1}^{k}\left(1-\frac{1}{\kappa_P}\right)^{j/2}$$
$$\leq \left(1+\sqrt{\frac{\beta_L}{\mu}}\sum_{j=0}^{\infty}\left(1-\frac{1}{\kappa_P}\right)^{j/2}\right)\rho$$
$$= \left(1+\frac{\sqrt{\beta_L/\mu}}{1-\sqrt{1-\frac{1}{\kappa_P}}}\right)\rho \leq \varepsilon_{\text{loc}}.$$

Here, in the second inequality we have used $L(\tilde{w}_0) \leq 2(1+\varepsilon)\|w_{\text{loc}} - w_\star\|^2_{H_L(w_\star)+\mu I}$, which is an immediate consequence of Lemma G.8. Hence, $\tilde{w}_{k+1} \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$, and the desired claim follows by induction. □

Theorem G.12 (Fast-local convergence of Damped Newton)**.** *Let w_{loc} be as in Corollary G.4. Consider the iteration*

$$\tilde{w}_{k+1} = \tilde{w}_k - \frac{1}{1+\varepsilon}(H_L(\tilde{w}_k)+\mu I)^{-1}\nabla L(\tilde{w}_k), \quad \textit{where } \tilde{w}_0 = w_{\text{loc}}.$$

Then, after k iterations, the loss satisfies

$$L(\tilde{w}_k) \leq \left(1-\frac{1}{2(1+\varepsilon)^2}\right)^k L(w_{\text{loc}}).$$

Thus after $k = \mathcal{O}\left(\log\left(\frac{1}{\epsilon}\right)\right)$ iterations

$$L(\tilde{w}_k) \leq \epsilon.$$

Proof. Lemma G.11 ensure that $\tilde{w}^k \in \mathcal{N}_{\varepsilon_{\text{loc}}}(w_\star)$ for all k. Thus, we can apply item 1 of Lemma G.8 and the definition of $\tilde{w}^{k+1}$, to reach

$$L(\tilde{w}_{k+1}) \leq L(\tilde{w}_k) - \frac{1}{2(1+\varepsilon)}\|\nabla L(\tilde{w}_k)\|^2_{P^{-1}}.$$

Now, using item 2 of Lemma G.8 and recursing yields

$$L(\tilde{w}_{k+1}) \leq \left(1-\frac{1}{2(1+\varepsilon)^2}\right)L(\tilde{w}_k) \leq \left(1-\frac{1}{2(1+\varepsilon)^2}\right)^{k+1}L(w_{\text{loc}}).$$

The remaining portion of the theorem now follows via a routine calculation. □

G.4. Formal Convergence of Algorithm 1

Here, we state and prove the formal convergence result for Algorithm 1.

Theorem G.13. *Suppose that Assumption 8.1 and Assumption G.1 hold, and that the loss is μ-PŁ* in $B(w_0, 2R)$, where $R = \frac{2\sqrt{2\beta_L L(w_0)}}{\mu}$. Let ε_{loc} and ρ be as in Corollary G.4, and set $\varepsilon = 1/6$ in the definition of ε_{loc}. Run Algorithm 1 with parameters: $\eta_{\text{GD}} = 1/\beta_L, K_{\text{GD}} = \frac{\beta_L}{\mu} \log\left(\frac{4\max\{2\beta_L, 1\}L(w_0)}{\mu\rho^2}\right), \eta_{\text{DN}} = 5/6, \gamma = \mu$ and $K_{\text{DN}} \geq 1$. Then Phase II of Algorithm 1 satisfies*

$$L(\tilde{w}_k) \leq \left(\frac{2}{3}\right)^k L(w_{K_{\text{GD}}}).$$

Hence after $K_{\text{DN}} \geq 3\log\left(\frac{L(w_{K_{\text{GD}}})}{\epsilon}\right)$ iterations, Phase II of Algorithm 1 outputs a point satisfying

$$L(\tilde{w}_{K_{\text{DN}}}) \leq \epsilon.$$

Proof. By assumption the conditions of Corollary G.4 are met, therefore $w_{K_{\text{GD}}}$ satisfies $\|w_{K_{\text{GD}}} - w_\star\|_{H_L(w_\star)+\mu I} \leq \rho$, for some $w_\star \in \mathcal{W}_\star$. Hence, we may invoke Theorem G.12 to conclude the desired result. □