

Convergence and Sketching-Based Efficient Computation of Neural Tangent Kernel Weights in Physics-Based Loss

Max Hirsch 

Department of Mathematics
University of California, Berkeley
Berkeley, CA 94720, USA
mhirsch@berkeley.edu

Federico Pichi 

mathLab, Mathematics Area
SISSA
I-34136 Trieste, Italy
fpichi@sissa.it

November 20, 2025

Abstract

In multi-objective optimization, multiple loss terms are weighted and added together to form a single objective. These weights are chosen to properly balance the competing losses according to some meta-goal. For example, in physics-informed neural networks (PINNs), these weights are often adaptively chosen to improve the network's generalization error. A popular choice of adaptive weights is based on the neural tangent kernel (NTK) of the PINN, which describes the evolution of the network in predictor space during training. The convergence of such an adaptive weighting algorithm is not clear *a priori*. Moreover, these NTK-based weights would be updated frequently during training, further increasing the computational burden of the learning process. In this paper, we prove that under appropriate conditions, gradient descent enhanced with adaptive NTK-based weights is convergent in a suitable sense. We then address the problem of computational efficiency by developing a randomized algorithm inspired by a predictor-corrector approach and matrix sketching, which produces unbiased estimates of the NTK up to an arbitrarily small discretization error. Finally, we provide numerical experiments to support our theoretical findings and to show the efficacy of our randomized algorithm.

Code Availability: <https://github.com/maxhirsch/Efficient-NTK>

1 Introduction

In multi-objective optimization, the goal is to simultaneously minimize multiple competing loss objectives $\{\mathcal{L}_i(\theta)\}_{i=1}^n$ w.r.t. some parameter θ [2, 40]. The objectives $\mathcal{L}_i(\theta)$ may not be minimized at the same $\bar{\theta}$, so it is often impossible to minimize each objective individually at the same time. In general, one can only find a solution in a set of Pareto solutions for which none of the objectives can be decreased without another objective increasing. One approach to finding such a solution to this problem is the so-called *weighted sum method* [19, 16].

In the weighted sum method, one attempts to reach the goal by minimizing the weighted objective

$$\mathcal{L}(\theta) = \sum_{i=1}^n w_i \mathcal{L}_i(\theta)$$

for some weights $w_i \in [0, \infty)$. Choosing a weight w_i that is larger than some w_j will encourage the objective $\mathcal{L}_i(\theta)$ to be minimized more than $\mathcal{L}_j(\theta)$, and modifying the weights results in new minimizers being found. The weighted sum method is known to not be able to find all Pareto solutions in non-convex regions of the Pareto solution set, but the method is still widely used due to its simplicity [16]. In particular, in the scientific machine learning (SciML) context [29, 27, 28], it is a common task when training physics-informed neural networks (PINNs).

A PINN is a method for solving partial differential equations (PDEs) using a neural network [30, 15, 22, 10]. The neural network takes as input the spatial (and possibly temporal) coordinates at which the PDE

solution must be evaluated, and the output of the network is the PDE solution evaluated at that point. To train a PINN, there are typically a few separate objectives or loss terms depending on whether the PDE is time dependent. For steady problems, the loss objectives include the residual of the differential operator for the PDE (usually the strong form of the problem), and the residual of the boundary condition (when imposed via weak constraints). If the problem is time dependent, there is an additional term corresponding to the (weak) imposition of the initial condition. These losses are combined via the weighted sum method, giving a single loss function to train the PINN. However, choosing the weights for the loss function is known to be a difficult problem, so several methods have been developed to learn these weights adaptively during the training procedure [37, 5, 39, 35, 17].

One such method of choosing adaptive weights is based on *neural tangent kernel* (NTK) theory [37]. In that paper, it was shown, and observed for the Poisson equation, that as the width of a physics-informed neural network increases to infinity, the dynamics of the residuals used to define the PINN loss becomes linear. The matrix in the resulting linear ordinary differential equation for the residuals is called the neural tangent kernel. The singular values of the NTK thus determine the rates at which the different residuals, corresponding to the objective terms, decrease to zero. This inspired the authors of [37] to devise a heuristic algorithm for choosing the weights in PINN loss based on the NTK of the PINN. The idea of the algorithm is to weight the losses roughly according to the inverses of the singular values of the NTK to make the rates at which the residuals decrease nearly the same. Furthermore, since in a practical setting the neural network is not infinitely wide, the computed NTK is not constant throughout epochs, requiring the NTK-based loss weights to be updated periodically during training.

This adaptive choice of weights in the loss function raises two main questions. First, does the optimization algorithm used to minimize the aggregate loss function converge when adaptive weights are used? Indeed, multi-objective problems optimized using the weighted sum method result in different minimizers of the objective when different weights are used. If these weights change frequently, it is not clear that an optimization algorithm with adaptive loss weights will ever converge to a single minimizer. Second, can the NTK weight updates be made more efficiently? Computing the NTK is a very costly operation since the matrix multiplication needed to compute a single entry of the NTK matrix of a neural network with O outputs and P parameters has computational complexity $\mathcal{O}(O^2P)$, and P is typically large for modern neural network models, with $P \sim 10^7$ [25]. Thus, recomputing the exact weights can hinder the whole training phase by making it unbearable. However, based on the heuristics of the NTK-based weighting algorithm, it seems to be desirable to keep the weights updated. To do so, it is important to find faster ways of approximating the NTK-based weights.

We address each of these two questions. For the first, we apply a well-known “descent lemma” for non-convex optimization [38, 24, 7] to a setting in which a squared error loss is computed with respect to an adaptive inner product. This allows us to show that the average squared norm of the loss gradients using the adaptive NTK weights converges to zero. Then, to address the second question, we develop a simple approach inspired from predictor-corrector methods to obtain fast matrix-vector products with the NTK.

In particular, in a “predictor” step, we use an initial gradient descent step from θ_t to $\hat{\theta}_{t+1}$ with a special choice of randomized weights, allowing us to approximate the product of the NTK with a Gaussian random vector. Combining this with ideas from matrix sketching, commonly used to approximate the range of a matrix with repeated randomized matrix-vector multiplications [20], we are able to obtain a fast randomized approximation of the NTK and its trace. We then use this approximation in a “corrector” step, which consists of returning to the original gradient descent iterate θ_t and computing the NTK-based weights from our NTK approximation to finally compute θ_{t+1} via gradient descent.

Outline. The remainder of the manuscript is organized as follows: in Section 1.1, we present a literature review of related works. Next in Section 2, we present the use of NTK-based weights in the loss function for training PINNs. In Section 3, we prove convergence results for optimization of the PINN loss function involving adaptive weights. Then in Section 4, we introduce a method for obtaining a fast approximation of the NTK and to allow frequent updates of the adaptive PINN loss weights. Finally, we present some numerical applications in Section 5 that verify our theoretical results and demonstrate the efficacy of our fast NTK approximation to compute the adaptive loss weights.

1.1 Literature Review

In the physics-informed machine learning community, there has been extensive work on the use of adaptive weights for PINN losses. In [37], the authors use weights for the loss based on the neural tangent kernel. The paper [5] develops a weighting algorithm that assigns larger loss weights to the largest objectives. The authors of [39] and [35] use maximum likelihood-based approaches to compute the loss weights. In [36], the authors connect the difficulty of training PINNs to an imbalance in the magnitudes of the gradients of the different constituent loss functions, and develop adaptive loss weights based on the relative sizes of the various loss gradients. The paper [17] also presents a variant of the gradient-based weights combined with the logarithmic mean. Despite the great body of literature in this important topic, up to our knowledge, there has not been any work showing convergence results for these types of loss weighting algorithms.

Recently, following the work [37] in which the authors show convergence of the PINN NTK in the infinite-width limit, the neural tangent kernel has increasingly been used in the context of physics-informed neural networks, mainly for analyzing new algorithms and architectures. The paper [44] extends the analysis of the PINN NTK to more general PDEs, while [21] uses the NTK to analyze a new scheme for adaptive loss weights. In [12], the authors use the NTK to analyze a PINN trained with an additional loss for the gradient of the PDE residual. Finally, the authors of [4] use the NTK to analyze physics-informed Kolmogorov-Arnold Networks.

Concerning the convergence of optimization algorithms for PINNs, so far there have been several different results. In [41], the authors show for linear second-order elliptic and parabolic PDEs that, given a sequence of training sets for PINNs, the sequence of minimizers of the PINN loss converges to the PDE solution. In the setting of linear second-order PDEs, the paper [6] shows convergence of gradient descent for overparameterized two-layer PINNs to global minimizers of the loss, while [14] considers a similar problem focused on stochastic gradient descent and Poisson's equation. In [34], the authors present a more applied study of optimization for PINNs. Finally, [32] considers the effect of PDE order on the PINN optimization.

There are several works on the fast estimation of the neural tangent kernel in different contexts. In [25], the authors analyze the cost of exact computation of the NTK for finite-width neural networks. The authors in [23] approximate the full neural tangent kernel of a network using a smaller matrix independent of the dimension of the network output. Moreover, the authors of [26] propose a Monte Carlo estimator of the NTK, while in [42] the NTK is computed by combining sketching with random features. Finally, there are also works on faster computation of graph neural tangent kernels for graph neural networks [43, 31] using sketching techniques [13].

2 Neural Tangent Kernel

We now describe the setting of physics-informed neural networks and their corresponding neural tangent kernels. In fact, the NTK of a PINN differs from that of a typical neural network because of the presence of a differential operator in the PINN loss function. Thus, we start by presenting the loss function of a general PINN before defining its NTK, along with the adaptive loss weighting algorithm from [37].

2.1 Physics-Informed Neural Network Loss

We consider the problem of finding $u_\theta : \Omega \subseteq \mathbb{R}^d \rightarrow \mathbb{R}^O$ parametrized by the network parameters $\theta \in \mathcal{P} \subseteq \mathbb{R}^p$ which approximately solves the following general partial differential equation defined in the domain Ω w.r.t. the unknown u :

$$\begin{cases} \mathcal{D}u = f & \text{in } \Omega, \\ \mathcal{B}u = g & \text{in } \partial\Omega, \end{cases}$$

where \mathcal{D} is a differential operator, \mathcal{B} defines boundary (and possibly initial) conditions, $f : \Omega \rightarrow \mathbb{R}^O$ is a forcing term, and $g : \partial\Omega \rightarrow \mathbb{R}^O$ are boundary data. For simplicity of presentation, we assume $O = 1$, but everything extends in a straightforward manner to multi-dimensional output, i.e. $O > 1$ (e.g. the numerical example in Section 5.2.3 has $O = 2$).

To be more precise in the formulation of the problem, we consider the collocation points $\mathcal{X}_D = \{x_D^i\}_{i=1}^{n_D}$ and $\mathcal{X}_B = \{x_B^i\}_{i=1}^{n_B}$ in Ω , respectively for the interior and the boundary regions, with $n_D + n_B = n$ and

$\mathcal{X} = \mathcal{X}_D \cup \mathcal{X}_B$, and define the *residual* as $\mathcal{R} : \mathcal{P} \rightarrow \mathbb{R}^n$ by

$$\mathcal{R}(\theta)_i = \begin{cases} \mathcal{D}u_\theta(x_D^i) - f(x_D^i) & \text{if } 1 \leq i \leq n_D, \\ \mathcal{B}u_\theta(x_B^{i-n_D}) - g(x_B^{i-n_D}) & \text{if } n_D + 1 \leq i \leq n. \end{cases}$$

Then, we want to minimize the physics-informed loss function defined as the weighted sum

$$\begin{aligned} \mathcal{L}(\theta) &= \lambda_D \sum_{i=1}^{n_D} |\mathcal{D}u_\theta(x_D^i) - f(x_D^i)|^2 + \lambda_B \sum_{i=1}^{n_B} |\mathcal{B}u_\theta(x_B^i) - g(x_B^i)|^2 \\ &= \lambda_D \sum_{i=1}^{n_D} |\mathcal{R}(\theta)_i|^2 + \lambda_B \sum_{i=n_D+1}^n |\mathcal{R}(\theta)_i|^2, \end{aligned}$$

where we refer to $\lambda_D, \lambda_B > 0$ as our loss weights.

Remark 1. It is common to refer to the network parameters θ as the weights of a neural network, and this should not be confused with the loss weights λ_D, λ_B defined above. The way in which we use “weights” and “parameters” should be clear from the context, but we will never refer to the loss weights as parameters.

More generally, one could minimize the loss function $\mathcal{L} : \mathbb{R}^p \rightarrow [0, \infty)$ given by

$$\mathcal{L}(\theta) = \frac{1}{2} \mathcal{R}(\theta)^\top \Lambda \mathcal{R}(\theta), \quad (1)$$

where $\Lambda \in \mathbb{R}^{n \times n}$ is a fixed symmetric and positive definite matrix.

It follows that \mathcal{L} is the squared norm of \mathcal{R} with respect to some inner product determined by Λ (which is yet to be chosen), thus we will also use the notation $\mathcal{L}(\theta) = \frac{1}{2} \|\mathcal{R}(\theta)\|_\Lambda^2$ to denote the same quantity. We remark that this generalization includes the case in which we give three separate weights for the interior PDE residual, the boundary conditions, and the initial condition for time dependent PDEs. In this case, we will denote the number of points used for different parts of the boundary of Ω as n_{B_1}, n_{B_2}, \dots , and the number of points used for the initial condition and initial time derivative as n_{B_i} and n_{D_i} , respectively. We will also differentiate between norms by using the notation $\|\cdot\|$ to denote the spectral norm for matrices and the Euclidean norm for vectors, as well as $\|\cdot\|_F$ for the Frobenius norm. Analogously, $\langle \cdot, \cdot \rangle$ will denote the Euclidean inner product and $\langle \cdot, \cdot \rangle_F$ the Frobenius inner product.

2.2 Neural Tangent Kernel Weights for Loss

Having defined the PINN loss, we now consider the gradient flow used to identify suitable parameters θ that make u_θ a good approximation to the PDE solution u . We start by assuming that Λ is independent of θ , as is the typical context for training PINNs with non-adaptive weights. The reason for starting with non-adaptive weights is to motivate the adaptive weighting algorithm from [37], which is the object of our analysis. The learning process is thus governed by

$$\frac{d\theta}{dt} = -\nabla \mathcal{L}(\theta(t)) = -\nabla \mathcal{R}(\theta) \Lambda \mathcal{R}(\theta), \quad (2)$$

where $\nabla \mathcal{R}(\theta) = [\nabla \mathcal{R}_1(\theta) \quad \dots \quad \nabla \mathcal{R}_n(\theta)] \in \mathbb{R}^{p \times n}$. In general, the variables with respect to which the gradient is taken will be clear from the context. When it requires specifying, we will denote it as, *e.g.*, ∇_θ .

On the other hand, when computing the time derivative of the residual term, we have

$$\frac{d\mathcal{R}(\theta(t))}{dt} = \nabla \mathcal{R}(\theta)^\top \frac{d\theta}{dt} = -\nabla \mathcal{R}(\theta)^\top \nabla \mathcal{R}(\theta) \Lambda \mathcal{R}(\theta) = -K(\theta) \Lambda \mathcal{R}(\theta),$$

where $K(\theta) = \nabla \mathcal{R}(\theta)^\top \nabla \mathcal{R}(\theta)$ is defined as the *neural tangent kernel* of the PINN. To better understand the contribution of the interior and boundary components, we can partition $K(\theta)$ into four blocks $K_{DD} \in \mathbb{R}^{n_D \times n_D}$, $K_{BB} \in \mathbb{R}^{n_B \times n_B}$, $K_{DB} \in \mathbb{R}^{n_D \times n_B}$, and $K_{BD} \in \mathbb{R}^{n_B \times n_D}$ given the original arrangement of the residual $\mathcal{R}(\theta)$ so that

$$K(\theta) = \begin{pmatrix} K_{DD}(\theta) & K_{DB}(\theta) \\ K_{BD}(\theta) & K_{BB}(\theta) \end{pmatrix}.$$

Because $K(\theta) = \nabla\mathcal{R}(\theta)^\top \nabla\mathcal{R}(\theta)$ is symmetric, we have that K_{DD} and K_{BB} are symmetric, and $K_{DB} = K_{BD}^\top$. If we consider the discrete approximation of (2), *i.e.*, the gradient descent algorithm

$$\theta_{t+1} = \theta_t - \eta \nabla\mathcal{R}(\theta_t) \Lambda \mathcal{R}(\theta_t), \quad \theta_0 = \theta(0), \quad (3)$$

where $\eta > 0$ is the learning rate, then the loss weighting scheme proposed in [37] suggests modifying Λ at each iteration to obtain parameter-dependent weights $\Lambda = \Lambda(\theta_t)$ which balance the two components of the loss given by the PDE residuals ($\mathcal{R}(\theta)_i$ for $1 \leq i \leq n_D$) and the boundary residuals ($\mathcal{R}(\theta)_i$ for $n_D + 1 \leq i \leq n$). This results in the following iteration scheme with parameter-dependent weights

$$\theta_{t+1} = \theta_t - \eta \nabla\mathcal{R}(\theta_t) \Lambda(\theta_t) \mathcal{R}(\theta_t), \quad \theta_0 = \theta(0). \quad (4)$$

One particular choice of $\Lambda(\theta)$ proposed in [37] is to consider a term-wise rescaling

$$\Lambda(\theta_t) = \begin{pmatrix} \lambda_D(\theta_t) I_{n_D} & 0 \\ 0 & \lambda_B(\theta_t) I_{n_B} \end{pmatrix}, \quad \text{where } \lambda_D(\theta_t) = \frac{\text{Tr}(K(\theta_t))}{\text{Tr}(K_{DD}(\theta_t))}, \quad \lambda_B(\theta_t) = \frac{\text{Tr}(K(\theta_t))}{\text{Tr}(K_{BB}(\theta_t))}, \quad (5)$$

and I_m denotes the $m \times m$ identity matrix. In practice, such a choice suggests a combination of the weight-independent scheme (3) and the weight-dependent one (4), mainly for efficiency reasons, since adaptive weights (5) are slow to compute. From a practical point of view, a way to overcome this issue is to perform many gradient descent steps with constant Λ before being updated with the current weights θ_t . The procedure is summarized in Algorithm 1.

Algorithm 1 Gradient Descent with Adaptive Loss Weights

Require: Update frequency n , number of training steps T

- 1: Initialize $\theta_0 \in \mathcal{P}$
 - 2: **for** $t = 0, 1, \dots, T$ **do**
 - 3: **if** t is divisible by n **then**
 - 4: Compute $K(\theta_t)$ and define $\Lambda = \Lambda(\theta_t)$ according to (5)
 - 5: **end if**
 - 6: $\theta_{t+1} = \theta_t - \eta \nabla\mathcal{R}(\theta_t) \Lambda \mathcal{R}(\theta_t)$
 - 7: **end for**
-

The heuristic motivation for the particular choice of Λ in (5) is that when we consider u_θ to be in a class of feedforward neural networks of fixed depth, then in a suitable infinite-width limit of this network for particular PDEs, $K(\theta_t) = K(\theta_0)$ remains constant throughout training (see Theorems 4.3 and 4.4 in [37]), giving the continuous time dynamics

$$\frac{d\mathcal{R}(\theta(t))}{dt} = -K \Lambda \mathcal{R}(\theta(t)). \quad (6)$$

Thus, the weighting parameters in Equation (5) can be thought of as a rough approximation to a scaled inverse of K (e.g., if K were approximately diagonal), trying to balance the rates at which the PDE residuals and the boundary residuals decrease. Indeed, if $\Lambda = \text{Tr}(K)K^{-1}$ holds exactly, then Equation (6) yields the solutions $\mathcal{R}(\theta(t))_i = e^{-\text{Tr}(K)t} \mathcal{R}(\theta(0))_i$ for all i , which means that the rate $e^{-\text{Tr}(K)t}$ is the same for each residual $\mathcal{R}(\theta)_i$.

However, practically speaking, we cannot work with infinite-width neural networks, so the exact limit will not hold, and $K(\theta_t)$ will change with t . This means that $\Lambda(\theta_t)$ needs to be updated, possibly frequently, and without affecting the speed of training too much.

In the following, we will not focus on the concerns of efficiently exploiting Equations (3) and (4), and instead assume that the loss weights can be updated at each iteration. Toward this goal, we present convergence results for the heuristically better weighting scheme (4) in which the weights $\Lambda(\theta_t)$ evolve over time rather than keeping them constant in certain intervals. As a corollary to our results, we do offer a convergence result with fewer assumptions which gives a rule for the frequency with which updates to the NTK-based weights should be made. We will then present a solution to the efficiency issue in Section 4.

3 Convergence Analysis

We first consider the convergence of Algorithm 1, keeping the weights $\Lambda(\theta_t)$ general, and not specifically defined by (5). The main difficulty in the analysis lies in the fact that Λ is not fixed and may depend on the neural tangent kernel $\nabla\mathcal{R}(\theta_t)^\top \nabla\mathcal{R}(\theta_t)$.

As a corollary of our analysis, we are also able to obtain results for the more specific choice of the weights given by (5). To investigate the convergence properties, we will require a number of assumptions.

Assumption 1 (Convergence Assumptions). *In the following we will assume:*

1. *The parameter domain $\mathcal{P} \subseteq \mathbb{R}^p$ is open and convex.*
2. *For all $t \in \mathbb{N} \cup \{0\}$, $\Lambda(\theta_t)$ is symmetric positive definite with eigenvalues bounded as $0 < \ell_{\min} \leq \lambda_{\min}(\Lambda(\theta_t)) \leq \lambda_{\max}(\Lambda(\theta_t)) \leq \ell_{\max} < \infty$.*
3. *For all $t \in \mathbb{N} \cup \{0\}$, $K(\theta_t)$ is symmetric positive definite with eigenvalues bounded as $0 < k_{\min} \leq \lambda_{\min}(K(\theta_t)) \leq \lambda_{\max}(K(\theta_t)) \leq k_{\max} < \infty$.*
4. *Let $G(\theta; z) = \frac{1}{2}\|\mathcal{R}(\theta)\|_{\Lambda(z)}^2$, then $\nabla_\theta G(\theta; z)$ is uniformly L -Lipschitz for any z , i.e., there exists $L > 0$ such that for any $\theta, \theta' \in \mathcal{P}$ and for all $z \in \mathcal{P}$,*

$$\|\nabla_\theta G(\theta; z) - \nabla_\theta G(\theta'; z)\| \leq L\|\theta - \theta'\|.$$

5. *Let $F(\theta) = \frac{1}{2}\|\mathcal{R}(\theta)\|^2$, then $\nabla F(\theta) = \nabla\mathcal{R}(\theta)\mathcal{R}(\theta)$ is L -Lipschitz.*

Assumptions 1.1, 1.4, and 1.5 are technical assumptions required to apply the standard descent lemma below [38, 24, 7]. Moreover, Assumption 1.1 is not very restrictive, since, for example, we could take $\mathcal{P} = \mathbb{R}^p$ for our convergence results. We also remark that if the weights $\Lambda(\theta_t)$ are taken as in [37], then Assumption 1.2 is automatically satisfied by Assumption 1.3. Furthermore, in the regime in which the NTK converges in the infinite-width limit, the upper bound in Assumption 1.3 will be satisfied. The lower bound depend on the sample of training data. If, for instance, the gradients of the residuals for the training points have a strong linear correlation, then $\nabla\mathcal{R}(\theta)^\top \nabla\mathcal{R}(\theta)$ could be approximately low rank. Moreover, the lower bound cannot be satisfied if $n > p$ since the rank of $K(\theta) = \nabla\mathcal{R}(\theta)^\top \nabla\mathcal{R}(\theta)$ is bounded by rank $\nabla\mathcal{R}(\theta) \leq \min\{p, n\}$, making $K(\theta)$ rank-deficient. In other words, there must be more parameters in the neural network than the amount of data used to train the model.

Remark 2. *For certain linear PDEs, if one considers the infinite-width NTK limit, it may be possible to derive even stronger convergence results. We instead focus on a more general and realistic setting without assuming that our differential operator is linear and the neural networks have finite width.*

Lemma 1 (Descent Lemma). *If $F \in C^1(\mathcal{P})$ with $\mathcal{P} \subseteq \mathbb{R}^p$ convex and ∇F is L -Lipschitz, then for all $x, y \in \mathcal{P}$,*

$$F(x) - F(y) \leq \langle \nabla F(y), x - y \rangle + \frac{L}{2}\|x - y\|^2.$$

Proof. See Appendix A. □

Using this result, we can prove that the time average of the residual norms $\|\mathcal{R}(\theta_t)\|^2$ converges. This will be used later to prove that the loss gradients also converge in some sense.

Theorem 1 (Convergence of Residual Averages). *Let $\{\theta_t\}_t$ be computed via the scheme (4). Then, under Assumptions 1.1, 1.2, 1.3, and 1.5, it holds*

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\mathcal{R}(\theta_t)\|^2 \leq \frac{\|\mathcal{R}(\theta_0)\|^2 - \|\mathcal{R}(\theta_T)\|^2}{T\eta}, \quad \text{for all } \eta \leq \frac{2k_{\min}\ell_{\min}}{Lk_{\max}\ell_{\max}^2}, \quad \text{and } T \geq 1.$$

In particular, the following limit holds

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \|\mathcal{R}(\theta_t)\|^2 = 0.$$

Proof. By defining $F(\theta) = \frac{1}{2}\|\mathcal{R}(\theta)\|^2$, we note that $\nabla F(\theta) = \nabla\mathcal{R}(\theta)\mathcal{R}(\theta)$ is L -Lipschitz by Assumption 1.5. Then by Lemma 1 combined with (4), we have

$$\begin{aligned} F(\theta_{t+1}) &\leq F(\theta_t) + \langle \nabla F(\theta_t), \theta_{t+1} - \theta_t \rangle + \frac{L}{2}\|\theta_{t+1} - \theta_t\|^2 \\ &= F(\theta_t) - \eta\mathcal{R}(\theta_t)^\top\nabla\mathcal{R}(\theta_t)^\top\nabla\mathcal{R}(\theta_t)\Lambda(\theta_t)\mathcal{R}(\theta_t) + \frac{L\eta^2}{2}\|\nabla\mathcal{R}(\theta_t)\Lambda(\theta_t)\mathcal{R}(\theta_t)\|^2. \end{aligned} \quad (7)$$

Now, note that for symmetric positive definite matrices A and B , the following bound holds

$$\frac{1}{\lambda_{\min}(AB)} = \lambda_{\max}((AB)^{-1}) = \|(AB)^{-1}\| \leq \|A^{-1}\|\|B^{-1}\| = \lambda_{\max}(A^{-1})\lambda_{\max}(B^{-1}) = \frac{1}{\lambda_{\min}(A)\lambda_{\min}(B)}.$$

Taking $A = K(\theta_t)$ and $B = \Lambda(\theta_t)$, we have that $\lambda_{\min}(K(\theta_t)\Lambda(\theta_t)) \geq \lambda_{\min}(K(\theta_t))\lambda_{\min}(\Lambda(\theta_t))$, since $K(\theta_t) = \nabla\mathcal{R}(\theta_t)^\top\nabla\mathcal{R}(\theta_t)$ and $\Lambda(\theta_t)$ are symmetric positive definite matrices by Assumptions 1.2 and 1.3. Combining this with the fact that $\text{tr}(CD) \geq \lambda_{\min}(C)\text{tr}(D)$ for symmetric positive definite matrices C and D and with the cyclic property of the trace, it follows that

$$\begin{aligned} \mathcal{R}(\theta_t)^\top\nabla\mathcal{R}(\theta_t)^\top\nabla\mathcal{R}(\theta_t)\Lambda(\theta_t)\mathcal{R}(\theta_t) &= \text{tr}(\mathcal{R}(\theta_t)^\top\nabla\mathcal{R}(\theta_t)^\top\nabla\mathcal{R}(\theta_t)\Lambda(\theta_t)\mathcal{R}(\theta_t)) \\ &= \|\mathcal{R}(\theta_t)\|^2 \text{tr} \left(\underbrace{\nabla\mathcal{R}(\theta_t)^\top\nabla\mathcal{R}(\theta_t)\Lambda(\theta_t)}_C \underbrace{\frac{\mathcal{R}(\theta_t)}{\|\mathcal{R}(\theta_t)\|} \frac{\mathcal{R}(\theta_t)^\top}{\|\mathcal{R}(\theta_t)\|}}_D \right) \\ &\geq \|\mathcal{R}(\theta_t)\|^2 \lambda_{\min}(\nabla\mathcal{R}(\theta_t)^\top\nabla\mathcal{R}(\theta_t)\Lambda(\theta_t)) \text{tr} \left(\frac{\mathcal{R}(\theta_t)}{\|\mathcal{R}(\theta_t)\|} \frac{\mathcal{R}(\theta_t)^\top}{\|\mathcal{R}(\theta_t)\|} \right) \\ &\geq \lambda_{\min}(K(\theta_t))\lambda_{\min}(\Lambda(\theta_t))\|\mathcal{R}(\theta_t)\|^2, \end{aligned}$$

where we used the fact that by the cyclic property of the trace, it holds $\text{tr} \left(\frac{\mathcal{R}(\theta_t)}{\|\mathcal{R}(\theta_t)\|} \frac{\mathcal{R}(\theta_t)^\top}{\|\mathcal{R}(\theta_t)\|} \right) = 1$. Additionally, we have the inequality

$$\|\nabla\mathcal{R}(\theta_t)\Lambda(\theta_t)\mathcal{R}(\theta_t)\|^2 = \mathcal{R}(\theta_t)^\top\Lambda(\theta_t)\nabla\mathcal{R}(\theta_t)^\top\nabla\mathcal{R}(\theta_t)\Lambda(\theta_t)\mathcal{R}(\theta_t) \leq \lambda_{\max}(K(\theta_t))\lambda_{\max}(\Lambda(\theta_t))^2\|\mathcal{R}(\theta_t)\|^2.$$

Using these bounds in Equation (7) together with Assumptions 1.2 and 1.3 yields

$$\begin{aligned} F(\theta_{t+1}) &\leq F(\theta_t) - \eta\lambda_{\min}(K(\theta_t))\lambda_{\min}(\Lambda(\theta_t))\|\mathcal{R}(\theta_t)\|^2 + \frac{L\eta^2}{2}\lambda_{\max}(K(\theta_t))\lambda_{\max}(\Lambda(\theta_t))^2\|\mathcal{R}(\theta_t)\|^2 \\ &= F(\theta_t) - \left(\eta\lambda_{\min}(K(\theta_t))\lambda_{\min}(\Lambda(\theta_t)) - \frac{L\eta^2}{2}\lambda_{\max}(K(\theta_t))\lambda_{\max}(\Lambda(\theta_t))^2 \right) \|\mathcal{R}(\theta_t)\|^2 \\ &\leq F(\theta_t) - \left(\eta k_{\min}\ell_{\min} - \frac{L\eta^2}{2}k_{\max}\ell_{\max}^2 \right) \|\mathcal{R}(\theta_t)\|^2. \end{aligned}$$

Dividing by $T\eta$ and summing over $t = 0, \dots, T-1$ gives

$$\left(k_{\min}\ell_{\min} - \frac{L\eta}{2}k_{\max}\ell_{\max}^2 \right) \frac{1}{T} \sum_{t=0}^{T-1} \|\mathcal{R}(\theta_t)\|^2 \leq \frac{F(\theta_0) - F(\theta_T)}{T\eta}.$$

Since $\eta \leq \frac{2k_{\min}\ell_{\min}}{Lk_{\max}\ell_{\max}^2}$, the result follows by definition of F . \square

Remark 3. Note that the admissible learning rate $\eta \leq \frac{2k_{\min}\ell_{\min}}{Lk_{\max}\ell_{\max}^2}$ depends on the inverses of the worst possible condition numbers of K and Λ . This means that, practically speaking, while the average squared residuals will converge to zero by Theorem 1, the required learning rate may need to be very small.

Using the previous result, we can show the convergence of the gradients of G as the inner product changes.

Theorem 2 (Convergence of Gradient Averages). *Let $\{\theta_t\}_t$ be computed via the scheme (4). Then under Assumptions 1.1–1.5, it holds that*

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla_{\theta} G(\theta_t; \theta_t)\|^2 \leq \frac{G(\theta_0; \theta_0) - G(\theta_T; \theta_T)}{T\eta(1 - L\eta/2)} + \frac{2\ell_{\max}}{T\eta^2} \|\mathcal{R}(\theta_0)\|^2,$$

for any $\eta < \min \left\{ \frac{2k_{\min}\ell_{\min}}{Lk_{\max}\ell_{\max}^2}, \frac{2}{L} \right\}$ and $T \geq 1$. In particular,

$$\min_{t=0,1,\dots,T-1} \|\nabla_{\theta} G(\theta_t; \theta_t)\|^2 \leq \frac{1}{T} \sum_{t=0}^{T-1} \|\nabla_{\theta} G(\theta_t; \theta_t)\|^2 \rightarrow 0 \quad \text{as } T \rightarrow \infty.$$

Proof. Having defined $G(\theta; z) = \frac{1}{2} \|\mathcal{R}(\theta)\|_{\Lambda(z)}^2$, we remark that Equation (4) can be rewritten as

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} G(\theta_t; \theta_t). \quad (8)$$

By Assumption 1.4 and Lemma 1, for all $z \in \mathcal{P}$, we have

$$G(\theta_{t+1}; z) - G(\theta_t; z) \leq \langle \nabla_{\theta} G(\theta_t; z), \theta_{t+1} - \theta_t \rangle + \frac{L}{2} \|\theta_{t+1} - \theta_t\|^2,$$

that for the specific choice of $z = \theta_t$, and using Equation (8) gives

$$G(\theta_{t+1}; \theta_t) - G(\theta_t; \theta_t) \leq -\eta \|\nabla_{\theta} G(\theta_t; \theta_t)\|^2 + \frac{L\eta^2}{2} \|\nabla_{\theta} G(\theta_t; \theta_t)\|^2. \quad (9)$$

On the other side, we note that

$$\begin{aligned} G(\theta_{t+1}; \theta_{t+1}) - G(\theta_{t+1}; \theta_t) &= \frac{1}{2} \|\mathcal{R}(\theta_{t+1})\|_{\Lambda(\theta_{t+1})}^2 - \frac{1}{2} \|\mathcal{R}(\theta_{t+1})\|_{\Lambda(\theta_t)}^2 \\ &= \frac{1}{2} \mathcal{R}(\theta_{t+1})^\top (\Lambda(\theta_{t+1}) - \Lambda(\theta_t)) \mathcal{R}(\theta_{t+1}) \\ &\leq \frac{1}{2} \lambda_{\max}(\Lambda(\theta_{t+1}) - \Lambda(\theta_t)) \|\mathcal{R}(\theta_{t+1})\|^2, \end{aligned}$$

so that, combining the former inequality with Equation (9), we have

$$G(\theta_{t+1}; \theta_{t+1}) - G(\theta_t; \theta_t) \leq \frac{1}{2} \lambda_{\max}(\Lambda(\theta_{t+1}) - \Lambda(\theta_t)) \|\mathcal{R}(\theta_{t+1})\|^2 - \eta \|\nabla_{\theta} G(\theta_t; \theta_t)\|^2 + \frac{L\eta^2}{2} \|\nabla_{\theta} G(\theta_t; \theta_t)\|^2.$$

Similarly as before, dividing by $T\eta$ and summing over $t = 0, 1, \dots, T-1$ gives

$$\left(1 - \frac{L\eta}{2}\right) \frac{1}{T} \sum_{t=0}^{T-1} \|\nabla_{\theta} G(\theta_t; \theta_t)\|^2 \leq \frac{G(\theta_0; \theta_0) - G(\theta_T; \theta_T)}{T\eta} + \frac{1}{2T\eta} \sum_{t=0}^{T-1} \lambda_{\max}(\Lambda(\theta_{t+1}) - \Lambda(\theta_t)) \|\mathcal{R}(\theta_{t+1})\|^2. \quad (10)$$

Now by Assumption 1.2 and Theorem 1, we have

$$\begin{aligned} &\frac{1}{2T\eta} \sum_{t=0}^{T-1} \lambda_{\max}(\Lambda(\theta_{t+1}) - \Lambda(\theta_t)) \|\mathcal{R}(\theta_{t+1})\|^2 \\ &\leq \frac{\ell_{\max}}{T\eta} \sum_{t=0}^{T-1} \|\mathcal{R}(\theta_{t+1})\|^2 \leq \frac{2\ell_{\max}}{\eta(T+1)} \sum_{t=0}^T \|\mathcal{R}(\theta_t)\|^2 \leq \frac{2\ell_{\max}}{T\eta^2} \|\mathcal{R}(\theta_0)\|^2, \end{aligned}$$

from which it follows that

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla_{\theta} G(\theta_t; \theta_t)\|^2 \leq \frac{G(\theta_0; \theta_0) - G(\theta_T; \theta_T)}{T\eta(1 - L\eta/2)} + \frac{2\ell_{\max}}{T\eta^2} \|\mathcal{R}(\theta_0)\|^2,$$

as desired. \square

Using this result, we can also show the convergence of the gradients of F if the weights Λ are chosen according to Equation (5), as done in [37].

Corollary 1 (Convergence for NTK-Based Weights). *Let $\{\theta_t\}_t$ be computed via the scheme (4), and assume that $\Lambda(\theta_t)$ is computed according to (5). Then under Assumption 1, it holds that*

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla F(\theta_t)\|^2 \leq \frac{k_{\max}}{k_{\min}} \left(\frac{G(\theta_0; \theta_0) - G(\theta_T; \theta_T)}{T\eta(1 - L\eta/2)} + \frac{2\ell_{\max}}{T\eta^2} \|\mathcal{R}(\theta_0)\|^2 \right),$$

for any $\eta < \min \left\{ \frac{2k_{\min}\ell_{\min}}{Lk_{\max}\ell_{\max}^2}, \frac{2}{L} \right\}$ and $T \geq 1$. In particular,

$$\min_{t=0,1,\dots,T-1} \|\nabla F(\theta_t)\|^2 \leq \frac{1}{T} \sum_{t=0}^{T-1} \|\nabla F(\theta_t)\|^2 \rightarrow 0 \quad \text{as } T \rightarrow \infty.$$

Proof. For all t , we have

$$\|\nabla_\theta G(\theta_t; \theta_t)\|^2 = \|\nabla \mathcal{R}(\theta_t) \Lambda(\theta_t) \mathcal{R}(\theta_t)\|^2 \geq \lambda_{\min}(K(\theta_t)) \|\Lambda(\theta_t) \mathcal{R}(\theta_t)\|^2 \geq k_{\min} \|\mathcal{R}(\theta_t)\|^2,$$

and

$$\|\nabla F(\theta_t)\|^2 = \|\nabla \mathcal{R}(\theta_t) \mathcal{R}(\theta_t)\|^2 \leq \|\nabla \mathcal{R}(\theta_t)\|^2 \|\mathcal{R}(\theta_t)\|^2 \leq k_{\max} \|\mathcal{R}(\theta_t)\|^2$$

by Assumption 1.3. Combining these gives $\|\nabla_\theta G(\theta_t; \theta_t)\|^2 \geq \frac{k_{\min}}{k_{\max}} \|\nabla F(\theta_t)\|^2$ so that by Theorem 2, we obtain

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla F(\theta_t)\|^2 \leq \frac{k_{\max}}{k_{\min}} \left(\frac{G(\theta_0; \theta_0) - G(\theta_T; \theta_T)}{T\eta(1 - L\eta/2)} + \frac{2\ell_{\max}}{T\eta^2} \|\mathcal{R}(\theta_0)\|^2 \right),$$

as desired. \square

We now present a corollary that provides results analogous to the ones in Theorem 2 but under fewer assumptions. In particular, we do not require assumptions on Λ and K , as long as the inner product Λ is changed only periodically according to some prescribed rule. Intuitively, if Λ does not change between iterations, then the extra term in (10) depending on the largest eigenvalue of the difference between Λ at successive time steps is zero, in which case we would have the desired convergence. Thus, if the changes of Λ are sufficiently sparse in time, then we can still obtain the convergence of Theorem 2 without invoking Theorem 1 in the proof.

Corollary 2 (Controlled Gradients via Spaced Updates). *Let $\{\theta_t\}_t$ be computed via the scheme (4), and let $h : [0, \infty) \rightarrow [0, \infty)$ be any non-decreasing function of T such that $h = o(T)$ as $T \rightarrow \infty$. Letting $\tilde{\Lambda}(\theta_{t+1})$ be any predicted update to $\Lambda(\theta_t)$ and*

$$S(t) = \begin{cases} \sum_{r=0}^{t-1} \lambda_{\max}(\Lambda(\theta_{r+1}) - \Lambda(\theta_r)) \|\mathcal{R}(\theta_{r+1})\|^2 & \text{if } t > 0, \\ 0 & \text{otherwise,} \end{cases}$$

consider the rule

$$\Lambda(\theta_{t+1}) = \begin{cases} \tilde{\Lambda}(\theta_{t+1}) & \text{if } S(t) + \lambda_{\max}(\tilde{\Lambda}(\theta_{t+1}) - \Lambda(\theta_t)) \|\mathcal{R}(\theta_{t+1})\|^2 \leq h(t), \\ \Lambda(\theta_t) & \text{otherwise.} \end{cases} \quad (11)$$

Then, under Assumptions 1.1, 1.4 and assuming $\eta < \frac{2}{L}$, we have

$$\min_{t=0,1,\dots,T-1} \|\nabla_\theta G(\theta_t; \theta_t)\|^2 \leq \frac{1}{T} \sum_{t=0}^{T-1} \|\nabla_\theta G(\theta_t; \theta_t)\|^2 \rightarrow 0 \quad \text{as } T \rightarrow \infty.$$

Proof. Starting from (10), applying the update rule (11), and dividing by $1 - L\eta/2$, we have

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla_\theta G(\theta_t; \theta_t)\|^2 \leq \frac{G(\theta_0; \theta_0) - G(\theta_T; \theta_T)}{\eta T(1 - L\eta/2)} + \frac{h(T)}{2\eta(1 - L\eta/2)T}.$$

Since $h = o(T)$ by assumption, the result follows. \square

Remark 4. The update rule (11) is based on a technical condition chosen specifically to control the extra term $\frac{1}{2T\eta} \sum_{t=0}^{T-1} \lambda_{\max}(\Lambda(\theta_{t+1}) - \Lambda(\theta_t)) \|\mathcal{R}(\theta_{t+1})\|^2$ in (10). If an additional term $\lambda_{\max}(\widetilde{\Lambda}(\theta_{t+1}) - \Lambda(\theta_t)) \|\mathcal{R}(\theta_{t+1})\|^2$ in the sum causes a growth rate $h(t) = o(t)$ to be exceeded, then the loss weights Λ are not updated.

4 Fast Estimation of the Neural Tangent Kernel

To address the computational bottleneck of computing NTK-based loss weights in (5), we now present a practical algorithm to obtain fast approximations of the Neural Tangent Kernel $K(\theta_t)$ based on matrix sketching techniques [20]. We first describe the algorithm and then provide intuition for why it works, showing a result on the estimator's unbiasedness (up to a discretization parameter). In particular, our objective is to find an approximation of $K(\theta_t)$ in expectation by means of $\widehat{K}(\theta_t)$.

To define a single sample approximation of $K(\theta_t)$, let us suppose that we have already computed θ_t , and set $\Lambda_{\text{pred}}(\theta_t) = \text{diag}(-g_t) \text{diag}(\mathcal{R}(\theta_t))^{\dagger}$, where $g_t \sim N(0, I_n)$, the notation $\text{diag}(v)$ denotes the diagonal matrix with diagonal entries given by the entries of the vector v , and A^{\dagger} denotes the Moore-Penrose pseudoinverse of the matrix A . We remark that, if none of the residual entries $\mathcal{R}(\theta_t)_i$ is equal to 0, then $\text{diag}(\mathcal{R}(\theta_t))^{\dagger} = \text{diag}(\mathcal{R}(\theta_t))^{-1}$. The approximation we present here follows without assumptions on the residuals, while in Theorem 3 we will later assume the residuals are nonzero to ensure that our approximation is indeed close to $K(\theta_t)g_t$. Thus, for now, let us define the following iteration scheme

$$\widehat{\theta}_{t+1} = \theta_t - \Delta t \nabla \mathcal{R}(\theta_t) \Lambda_{\text{pred}}(\theta_t) \mathcal{R}(\theta_t), \quad \text{for any } \Delta t > 0.$$

This step can be thought of as the “predictor” step of a predictor-corrector method to numerically solve an ordinary differential equation, except that $\widehat{\theta}_{t+1}$ is not related to θ_{t+1} which will be computed later. Thus, the sole purpose of $\widehat{\theta}_{t+1}$ is to obtain approximations to the NTK as

$$K(\theta_t)g_t \approx \frac{\mathcal{R}(\widehat{\theta}_{t+1}) - \mathcal{R}(\theta_t)}{\Delta t}.$$

We will show in the proof of Theorem 3 that, through a Taylor expansion, this is indeed an approximation of $K(\theta_t)g_t$. Then, given this approximate matrix-vector product, we may form a single sample approximation of the NTK as

$$\widehat{K}(\theta_t) = \frac{\widetilde{K}(\theta_t) + \widetilde{K}(\theta_t)^{\top}}{2}, \quad \text{where } \widetilde{K}(\theta_t) = \begin{pmatrix} \frac{\mathcal{R}(\widehat{\theta}_{t+1}) - \mathcal{R}(\theta_t)}{\Delta t}(g_t)_1 & \dots & \frac{\mathcal{R}(\widehat{\theta}_{t+1}) - \mathcal{R}(\theta_t)}{\Delta t}(g_t)_n \end{pmatrix} \in \mathbb{R}^{n \times n},$$

and we can estimate the trace of the NTK as

$$\text{Tr}(\widehat{K}(\theta_t)) = g_t^{\top} \left(\frac{\mathcal{R}(\widehat{\theta}_t) - \mathcal{R}(\theta_t)}{\Delta t} \right).$$

Note that if our approximation of $K(\theta_t)g_t$ is correct, then $(g_t)_i K(\theta_t)g_t$ in expectation equals the i -th column of $K(\theta_t)$, which shows that $\widehat{K}(\theta_t)$ would indeed be an approximation of $K(\theta_t)$. Multiple samples constructed in this way may be averaged together to obtain a more accurate estimate. The steps for the single sample approximation are summarized in Algorithm 2. Note that the only step with significant computational cost is the computation of $\widehat{\theta}_{t+1}$, which is simply the cost of a single extra forward and backward pass through the neural network with the data batch without significant extra memory costs. Now we can use the result of this “predictor” step to define $\widehat{\Lambda}(\theta_t)$ analogously to (5):

$$\widehat{\Lambda}(\theta_t) = \begin{pmatrix} \widehat{\lambda}_D(\theta_t)I_{n_D} & 0 \\ 0 & \widehat{\lambda}_B(\theta_t)I_{n_B} \end{pmatrix}, \quad \text{where } \widehat{\lambda}_D(\theta_t) = \frac{\text{Tr}(\widehat{K}(\theta_t))}{\text{Tr}(\widehat{K}_{DD}(\theta_t))}, \quad \widehat{\lambda}_B(\theta_t) = \frac{\text{Tr}(\widehat{K}(\theta_t))}{\text{Tr}(\widehat{K}_{BB}(\theta_t))}, \quad (12)$$

Having defined $\widehat{\Lambda}(\theta_t)$, we can perform the “corrector” step given by the typical gradient descent step (4).

Algorithm 2 Single sample approximation of $K(\theta_t)$

Require: Assumption 2, θ_t is given.

Ensure: $\widehat{K}(\theta_t) \approx K(\theta_t)$ in expectation

$$\begin{aligned} & \text{Sample } g_t \sim N(0, I_n) \\ & \Lambda_{\text{pred}}(\theta_t) \leftarrow \text{diag}(-g_t) \text{diag}(\mathcal{R}(\theta_t))^{-1} \\ & \widehat{\theta}_{t+1} \leftarrow \theta_t - \Delta t \nabla \mathcal{R}(\theta_t) \Lambda_{\text{pred}}(\theta_t) \mathcal{R}(\theta_t) \\ & \tilde{K}(\theta_t) \leftarrow \begin{pmatrix} \frac{\mathcal{R}(\widehat{\theta}_{t+1}) - \mathcal{R}(\theta_t)}{\Delta t}(g_t)_1 & \dots & \frac{\mathcal{R}(\widehat{\theta}_{t+1}) - \mathcal{R}(\theta_t)}{\Delta t}(g_t)_n \end{pmatrix} \\ & \widehat{K}(\theta_t) \leftarrow \frac{\tilde{K}(\theta_t) + \tilde{K}(\theta_t)^\top}{2} \end{aligned}$$

4.1 Error Analysis of Fast NTK Approximation

We begin the analysis of the proposed Algorithm 2 by introducing the assumptions that we will require for this purpose.

Assumption 2 (Fast NTK Approximation Assumptions). *Assume the following hold:*

1. For all $t = 0, 1, \dots, T$, the residual $\mathcal{R}(\theta_t)_i \neq 0$ for $i = 1, 2, \dots, n$.
2. The parameter domain $\mathcal{P} \subseteq \mathbb{R}^p$ is open, convex, and bounded.
3. The residual \mathcal{R} is smooth and has bounded derivatives in \mathcal{P} .

Note that Assumption 2.3 will be guaranteed by using a smooth activation function for our PINN (which will also be required in general for allowing higher order differential operators in the loss) and by Assumption 2.2. The Assumption 2.1 is reasonable since we expect that the residuals are never exactly minimized in a typical PINN optimization. The second Assumption 2.2 then is the most restrictive but is reasonable if the gradient descent iterates are converging. Now under this assumption, we will show that Algorithm 2 provides an accurate approximation of the NTK.

Theorem 3 (Unbiased Estimates of NTK up to Discretization). *Under Assumption 2, there exists a constant $C > 0$ such that for any $\Delta t > 0$ it holds*

$$\mathbb{E} \left[\left\| \frac{\mathcal{R}(\widehat{\theta}_{t+1}) - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t)g_t \right\| \right] \leq Cn\Delta t, \quad \text{and} \quad \mathbb{E} \left[\left\| \frac{\mathcal{R}(\widehat{\theta}_{t+1}) - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t)g_t \right\|^2 \right] \leq C^2 n(n+2)\Delta t^2.$$

In particular, the following are satisfied

$$\left\| \mathbb{E} [\widehat{K}(\theta_t)] - K(\theta_t) \right\|_F \leq Cn(n+1)\Delta t, \quad \text{and} \quad \left| \mathbb{E} [\text{Tr}(\widehat{K}(\theta_t))] - \text{Tr}(K(\theta_t)) \right| \leq Cn(n+1)\Delta t.$$

Proof. Thanks to the fact that derivatives of \mathcal{R} are smooth in θ and bounded by Assumption 2.3, by definition of $\widehat{\Lambda}(\theta_t)$ we can use $\widehat{\theta}_{t+1} = \theta_t - \Delta t \nabla \mathcal{R}(\theta_t) \Lambda_{\text{pred}}(\theta_t) \mathcal{R}(\theta_t) = \theta_t + \Delta t \nabla \mathcal{R}(\theta_t) g_t$, and then perform a Taylor expansion to obtain

$$\begin{aligned} \left\| \frac{\mathcal{R}(\widehat{\theta}_{t+1}) - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t)g_t \right\| &= \left\| \frac{\mathcal{R}(\theta_t + \Delta t \nabla \mathcal{R}(\theta_t) g_t) - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t)g_t \right\| \\ &\leq \left\| \frac{\mathcal{R}(\theta_t) + \Delta t \nabla \mathcal{R}(\theta_t)^\top \nabla \mathcal{R}(\theta_t) g_t - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t)g_t \right\| + \frac{C_1 \Delta t^2 \|\nabla \mathcal{R}(\theta_t) g_t\|^2}{\Delta t} \\ &= C_1 \Delta t \|\nabla \mathcal{R}(\theta_t) g_t\|^2 \leq C_2 \Delta t \|g_t\|^2. \end{aligned} \tag{13}$$

Taking the expectation of this yields the first part of the theorem since $\mathbb{E}[\|g_t\|^2] = n$ and $\mathbb{E}[\|g_t\|^4] = n(n+2)$, for the first and second inequality, respectively.

Now defining

$$\mathcal{K} = \frac{1}{2} \left[((g_t)_1 K(\theta_t) g_t \quad \dots \quad (g_t)_n K(\theta_t) g_t) + ((g_t)_1 K(\theta_t) g_t \quad \dots \quad (g_t)_n K(\theta_t) g_t)^\top \right],$$

and using the fact that $\mathbb{E}[\mathcal{K}] = K(\theta_t)$ since $\mathbb{E}[(g_t)_i K(\theta_t) g_t] = K(\theta_t) e_i$ with e_i the i -th standard basis vector of \mathbb{R}^n , we then have

$$\begin{aligned} \left\| \mathbb{E} [\widehat{K}(\theta_t)] - K(\theta_t) \right\|_F &= \left\| \mathbb{E} [\widehat{K}(\theta_t)] - \mathbb{E}[\mathcal{K}] \right\|_F \\ &\leq \mathbb{E} \left[\left\| \widehat{K}(\theta_t) - \mathcal{K} \right\|_F \right] \\ &\leq \sum_{i=1}^n \mathbb{E} \left[|(g_t)_i| \cdot \left\| \frac{\mathcal{R}(\widehat{\theta}_t) - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t) g_t \right\| \right] \\ &\leq \sum_{i=1}^n \sqrt{\mathbb{E}[(g_t)_i^2]} \sqrt{\mathbb{E} \left[\left\| \frac{\mathcal{R}(\widehat{\theta}_t) - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t) g_t \right\|^2 \right]} \\ &\leq n \sqrt{C^2 n(n+2) \Delta t^2} \leq Cn(n+1)\Delta t, \end{aligned}$$

where in the second to last line, we used the Cauchy-Schwarz inequality, while in the last line we used $n(n+2) < (n+1)^2$, the fact that $\mathbb{E}[(g_t)_i^2] = 1$, and our previously derived bound on the mean squared error of the difference quotient.

Lastly, following a similar sequence of inequalities, we have

$$\begin{aligned} \left| \mathbb{E} [\text{Tr}(\widehat{K}(\theta_t))] - \text{Tr}(K(\theta_t)) \right| &= \left| \mathbb{E} [\text{Tr}(\widehat{K}(\theta_t))] - \mathbb{E} [\text{Tr}(\mathcal{K})] \right| \\ &\leq \mathbb{E} \left[\left| \text{Tr}(\widehat{K}(\theta_t)) - \mathcal{K} \right| \right] \\ &\leq \sum_{i=1}^n \mathbb{E} \left[\left| \widehat{K}(\theta_t)_{ii} - \mathcal{K}_{ii} \right| \right] \\ &= \sum_{i=1}^n \mathbb{E} \left[\left| \frac{\mathcal{R}(\widehat{\theta}_t)_i - \mathcal{R}(\theta_t)_i}{\Delta t} (g_t)_i - (K(\theta_t) g_t)_i (g_t)_i \right| \right] \\ &\leq \sum_{i=1}^n \sqrt{\mathbb{E}[(g_t)_i^2]} \sqrt{\mathbb{E} \left[\left| \frac{\mathcal{R}(\widehat{\theta}_t)_i - \mathcal{R}(\theta_t)_i}{\Delta t} - (K(\theta_t) g_t)_i \right|^2 \right]} \\ &\leq \sum_{i=1}^n \sqrt{\mathbb{E} \left[\left\| \frac{\mathcal{R}(\widehat{\theta}_t) - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t) g_t \right\|^2 \right]} \\ &\leq Cn(n+1)\Delta t, \end{aligned}$$

as desired. \square

Remark 5. In place of Gaussian random vectors, there are other choices of vectors that could be used in Algorithm 2. For example, one could choose Rademacher random vectors with i.i.d. entries taking the values ± 1 , each with probability $1/2$. The main trick of the algorithm is that we can obtain a matrix-vector product of K with any vector v up to a discretization error of $O(\Delta t)$ by Taylor's theorem.

Remark 6. By definition of the construction of $\widehat{K}(\theta_t)$, some of the entries of the estimator may be negative. It follows that $\widehat{K}(\theta_t)$ is dominated by an estimator obtained by taking the maximum of the entries of $\widehat{K}(\theta_t)$ and 0. In other words, one could define the estimator $\check{K}(\theta_t)$ given by

$$\check{K}(\theta_t)_{ij} = \max(\widehat{K}(\theta_t)_{ij}, 0), \quad \text{for } i, j = 1, \dots, n,$$

which has a better mean absolute (and mean squared) error:

$$\mathbb{E} [\|\check{K}(\theta_t) - K(\theta_t)\|_F] \leq \mathbb{E} [\|\hat{K}(\theta_t) - K(\theta_t)\|_F].$$

We now present a result which provides a Monte Carlo error rate, up to a discretization error, for an approximation of the NTK obtained by averaging samples from Algorithm 2.

Theorem 4 (Error Rates of NTK Approximation). *Assume that Assumption 2 holds, and let $\{\hat{K}^{(j)}(\theta_t)\}_{j=1}^N$ be the N results of Algorithm 2 with N i.i.d. copies $g_t^{(j)} \sim N(0, I_n)$. Then for any $\Delta t > 0$ it holds*

$$\mathbb{E} \left[\left| \frac{1}{N} \sum_{j=1}^N \text{Tr}(\hat{K}^{(j)}(\theta_t)) - \text{Tr}(K(\theta_t)) \right|^2 \right] \leq Cn(n+2)(n+4)\Delta t^2 + \frac{4}{N} \|K(\theta_t)\|_F^2,$$

and

$$\mathbb{E} \left[\left\| \frac{1}{N} \sum_{j=1}^N \hat{K}^{(j)}(\theta_t) - K(\theta_t) \right\|_F^2 \right] \leq Cn^2(n+2)(n+4)\Delta t^2 + \frac{2n(n+2)}{N} \|K(\theta_t)\|_F^2,$$

where $C > 0$ is a constant.

Proof. First, by the triangle inequality and Young's inequality we have

$$\begin{aligned} & \mathbb{E} \left[\left| \frac{1}{N} \sum_{j=1}^N \text{Tr}(\hat{K}^{(j)}(\theta_t)) - \text{Tr}(K(\theta_t)) \right|^2 \right] \\ & \leq \mathbb{E} \left[2 \underbrace{\left| \frac{1}{N} \sum_{j=1}^N \text{Tr}(\hat{K}^{(j)}(\theta_t)) - \frac{1}{N} \sum_{j=1}^N (g_t^{(j)})^\top K(\theta_t) g_t^{(j)} \right|^2}_{(a)} + 2 \underbrace{\left| \frac{1}{N} \sum_{j=1}^N (g_t^{(j)})^\top K(\theta_t) g_t^{(j)} - \text{Tr}(K(\theta_t)) \right|^2}_{(b)} \right]. \end{aligned}$$

For (a), first note that for all $j = 1, 2, \dots, N$, by Taylor expansion as in the proof of Theorem 3 we have,

$$\text{Tr}(\hat{K}^{(j)}(\theta_t)) = (g_t^{(j)})^\top \frac{\mathcal{R}(\hat{\theta}_t^{(j)}) - \mathcal{R}(\theta_t)}{\Delta t} = (g_t^{(j)})^\top (K(\theta_t)g_t^{(j)} + \mathcal{E}_t^{(j)}),$$

where $\|\mathcal{E}_t^{(j)}\| \leq C\Delta t \|g_t^{(j)}\|^2$. It follows that

$$\begin{aligned} (a) &= 2\mathbb{E} \left[\left| \frac{1}{N} \sum_{i=1}^N \left(\text{Tr}(\hat{K}^{(i)}(\theta_t)) - (g_t^{(i)})^\top K(\theta_t) g_t^{(i)} \right) \right|^2 \right] = 2\mathbb{E} \left[\left| \frac{1}{N} \sum_{j=1}^N (g_t^{(j)})^\top \mathcal{E}_t^{(j)} \right|^2 \right] \\ &\leq 2\mathbb{E} \left[\left(\frac{1}{N} \sum_{j=1}^N C\Delta t \|g_t^{(j)}\|^3 \right)^2 \right] \leq \frac{2}{N^2} \mathbb{E} \left[N \sum_{j=1}^N C^2 \Delta t^2 \|g_t^{(j)}\|^6 \right] = 2C^2 \Delta t^2 \mathbb{E}[\|g_t^{(1)}\|^6], \end{aligned}$$

where the last inequality comes from the fact that $(a_1 + \dots + a_N)^2 \leq N(a_1^2 + \dots + a_N^2)$. Using $\mathbb{E}[\|g_t^{(1)}\|^6] = n(n+2)(n+4)$, we obtain $(a) \leq 2C^2 n(n+2)(n+4)\Delta t^2$.

Now (b) is the standard error for the Hutchinson estimator exploited in randomized numerical linear algebra to estimate the trace of a matrix using random matrix-vector products. By Theorem 2.1 in [8], we have that

$$(b) = 2\mathbb{E} \left[\left| \frac{1}{N} \sum_{j=1}^N (g_t^{(j)})^\top K(\theta_t) g_t^{(j)} - \text{Tr}(K(\theta_t)) \right|^2 \right] = \frac{4}{N} \|K(\theta_t)\|_F^2.$$

Combining the estimates for (a) and (b) yields the result.

For the second result, we have

$$\begin{aligned} & \mathbb{E} \left[\left\| \frac{1}{N} \sum_{j=1}^N \widehat{K}^{(j)}(\theta_t) - K(\theta_t) \right\|_F^2 \right] \\ & \leq 2\mathbb{E} \left[\left\| \frac{1}{N} \sum_{j=1}^N \widehat{K}^{(j)}(\theta_t) - \frac{1}{N} \sum_{j=1}^N \mathcal{K}^{(j)}(\theta_t) \right\|_F^2 + \left\| \frac{1}{N} \sum_{j=1}^N \mathcal{K}^{(j)}(\theta_t) - K(\theta_t) \right\|_F^2 \right] =: (c) + (d). \end{aligned}$$

Considering (c), we have

$$\begin{aligned} (c) &= 2\mathbb{E} \left[\left\| \frac{1}{N} \sum_{j=1}^N \widehat{K}^{(j)}(\theta_t) - \frac{1}{N} \sum_{j=1}^N \mathcal{K}^{(j)}(\theta_t) \right\|_F^2 \right] \\ &\leq \frac{2}{N^2} \mathbb{E} \left[\left\| \sum_{j=1}^N \left(\left((g_t^{(j)})_1 \frac{\mathcal{R}(\widehat{\theta}_t^{(j)}) - \mathcal{R}(\theta_t)}{\Delta t} \dots (g_t^{(j)})_n \frac{\mathcal{R}(\widehat{\theta}_t^{(j)}) - \mathcal{R}(\theta_t)}{\Delta t} \right) - \left((g_t^{(j)})_1 K(\theta_t) g_t^{(j)} \dots (g_t^{(j)})_n K(\theta_t) g_t^{(j)} \right) \right) \right\|_F^2 \right] \\ &= \frac{2}{N^2} \mathbb{E} \left[\sum_{i=1}^N \left\| \sum_{j=1}^N (g_t^{(j)})_i \left(\frac{\mathcal{R}(\widehat{\theta}_t^{(j)}) - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t) g_t^{(j)} \right) \right\|_F^2 \right] \\ &= \frac{2n}{N^2} \mathbb{E} \left[\left\| \sum_{j=1}^N (g_t^{(j)})_1 \left(\frac{\mathcal{R}(\widehat{\theta}_t^{(j)}) - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t) g_t^{(j)} \right) \right\|_F^2 \right] \\ &\leq \frac{2n}{N} \mathbb{E} \left[\sum_{j=1}^N \left\| (g_t^{(j)})_1 \left(\frac{\mathcal{R}(\widehat{\theta}_t^{(j)}) - \mathcal{R}(\theta_t)}{\Delta t} - K(\theta_t) g_t^{(j)} \right) \right\|_F^2 \right] \\ &\leq Cn\Delta t^2 \mathbb{E}[\|g_t^{(1)}\|^6] = Cn^2(n+2)(n+4)\Delta t^2, \end{aligned}$$

where in the last line, we applied Equation 13.

Now considering (d), repeatedly using the fact that $\mathcal{K}^{(j)}(\theta_t)$ are i.i.d. with mean $K(\theta_t)$, we have

$$\begin{aligned} (d) &= 2\mathbb{E} \left[\left\| \frac{1}{N} \sum_{j=1}^N \mathcal{K}^{(j)}(\theta_t) - K(\theta_t) \right\|_F^2 \right] \\ &= 2\mathbb{E} \left[\left\| \frac{1}{N} \sum_{j=1}^N \mathcal{K}^{(j)}(\theta_t) \right\|_F^2 - 2 \left\langle \frac{1}{N} \sum_{j=1}^N \mathcal{K}^{(j)}(\theta_t), K(\theta_t) \right\rangle_F + \|K(\theta_t)\|_F^2 \right] \\ &= -2\|K(\theta_t)\|_F^2 + 2\mathbb{E} \left[\left\| \frac{1}{N} \sum_{j=1}^N \mathcal{K}^{(j)}(\theta_t) \right\|_F^2 \right] \\ &= -2\|K(\theta_t)\|_F^2 + \frac{2}{N^2} \sum_{j=1}^N \mathbb{E} \left[\left\| \mathcal{K}^{(j)}(\theta_t) \right\|_F^2 \right] + \frac{2}{N^2} \sum_{i \neq j} \left\langle \mathbb{E} [\mathcal{K}^{(i)}(\theta_t)], \mathbb{E} [\mathcal{K}^{(j)}(\theta_t)] \right\rangle_F \\ &= -2\|K(\theta_t)\|_F^2 + \frac{2}{N^2} \sum_{j=1}^N \mathbb{E} [\|K(\theta_t) + \mathcal{E}_t^{(j)}\|_F^2] + \frac{2(N-1)}{N} \|K(\theta_t)\|_F^2, \end{aligned}$$

where we redefine $\mathcal{E}_t^{(j)} = \mathcal{K}^{(j)}(\theta_t) - K(\theta_t)$. Now note that

$$\begin{aligned} \frac{2}{N^2} \sum_{j=1}^N \mathbb{E} \left[\|K(\theta_t) + \mathcal{E}_t^{(j)}\|_F^2 \right] &= \frac{2}{N} \left(\|K(\theta_t)\|_F^2 + \left\langle K(\theta_t), \mathbb{E} \left[\mathcal{E}_t^{(1)} \right] \right\rangle_F + \mathbb{E} \left[\|\mathcal{E}_t^{(1)}\|_F^2 \right] \right) \\ &= \frac{2}{N} \left(\|K(\theta_t)\|_F^2 + \mathbb{E} \left[\|\mathcal{E}_t^{(1)}\|_F^2 \right] \right), \end{aligned}$$

where we used the fact that $\mathbb{E}[\mathcal{E}_t^{(1)}] = 0$. Now using that $\mathbb{E}[\|g_t\|^4] = n(n+2)$, we have

$$\begin{aligned} \mathbb{E} \left[\|\mathcal{E}_t^{(1)}\|_F^2 \right] &= \mathbb{E}[\|\mathcal{K}^{(1)}(\theta_t)\|_F^2] - \|K(\theta_t)\|_F^2 \\ &\leq \mathbb{E}[\|((g_t)_1 K(\theta_t) g_t \dots (g_t)_n K(\theta_t) g_t)\|_F^2] - \|K(\theta_t)\|_F^2 \\ &= \mathbb{E} \left[\sum_{i=1}^n |(g_t)_i|^2 \|K(\theta_t) g_t\|_F^2 \right] - \|K(\theta_t)\|_F^2 \\ &\leq \mathbb{E}[\|g_t\|^4] \cdot \|K(\theta_t)\|_F^2 - \|K(\theta_t)\|_F^2 \\ &\leq n(n+2) \|K(\theta_t)\|_F^2. \end{aligned}$$

It follows that

$$(d) = \frac{2}{N} \mathbb{E} \left[\|\mathcal{E}_t^{(1)}\|_F^2 \right] \leq \frac{2n(n+2)}{N} \|K(\theta_t)\|_F^2.$$

Combining this with the bound for (c) gives the desired result. \square

We now report another result on approximating the trace of the neural tangent kernel based on Theorem 4.1 in [33] to compare it with our own strategy.

Theorem 5 (Alternative Approximation of the NTK Trace). *Let $g_t \sim N(0, \varepsilon^2 I_n)$ and let θ_t be given. Under Assumption 2, there is $C > 0$ such that*

$$-C(n+1)^2 \varepsilon \leq \mathbb{E} \left[\left\| \frac{\mathcal{R}(\theta_t + g_t) - \mathcal{R}(\theta_t)}{\varepsilon} \right\|_F^2 \right] - \text{Tr}(K(\theta_t)) \leq C(n+1)^2 (\varepsilon + \varepsilon^2).$$

Proof. See Appendix B. \square

From Theorem 5 we can see that

$$\frac{\mathcal{R}(\theta_t + g_t) - \mathcal{R}(\theta_t)}{\varepsilon}, \quad \text{with } g_t \sim N(0, \varepsilon^2 I_n) \tag{14}$$

also provides an estimate of the trace of $K(\theta_t)$, but since it does not allow arbitrary matrix-vector products with the NTK one cannot recover a full approximation of $K(\theta_t)$. Indeed, it may be seen from the proof of Theorem 5 given in Appendix B that one can use the standard basis vectors of \mathbb{R}^n to obtain approximations to the diagonal entries of $K(\theta_t)$, but not other entries of the NTK. Thus, the main difference between our estimator $\hat{K}(\theta_t)$ and (14) is that our approach gives additional information about off-diagonal entries of $K(\theta_t)$. In terms of computational cost, the proposed estimator requires a single extra backpropagation to compute $\hat{\theta}_t$ compared to this alternative estimator.

4.2 Practical Algorithm with Moving Average

We conclude this section by presenting a practical algorithm for computing an adaptive approximation of the NTK (or its trace). We begin by noting that an accurate computation of the NTK would take many samples due to the Monte Carlo error rate noted in Theorem 4. However, doing this every iteration would result in a high computational cost, which is exactly what we wish to avoid. Instead, we propose using a moving average over training steps of our single sample NTK estimates to obtain adaptive NTK estimates.

This allows for an adaptive estimate of the NTK which costs just two extra forward passes and one extra backward pass in the neural network optimization per iteration, effectively only multiplying the training time by a factor of 2.5. Indeed, computing the weights $\widehat{\theta}_{t+1}$ is the same cost of a typical gradient descent step of a single forward and backward pass, and computing $\mathcal{R}(\widehat{\theta}_{t+1})$ for the finite difference approximation is an extra forward pass ($\mathcal{R}(\theta_t)$ need not be recomputed). These are the main costs since they depend on the number of network parameters which is thought to be large. Now, we can first obtain a good approximation of the NTK at the initialization:

$$\widehat{K}_0(\theta_0) = \frac{1}{N} \sum_{j=1}^N \widehat{K}^{(j)}(\theta_0),$$

where we denote our moving average NTK approximation at time t and parameter θ_t by $\widehat{K}_t(\theta_t)$ to differentiate it from the N i.i.d. samples $\widehat{K}^{(j)}(\theta_t)$ taken according to Algorithm 2. Then, for a fixed parameter $\alpha \in (0, 1]$ and $t > 0$, we compute

$$\widehat{K}_t(\theta_t) = (1 - \alpha)\widehat{K}_{t-1}(\theta_{t-1}) + \alpha\widehat{K}^{(1)}(\theta_t),$$

that is, for each new training step, we compute a single new sample from Algorithm 2 and average it with the NTK approximation from the previous time step. If the NTK were constant between iterations, the moving average would effectively become a sort of Monte Carlo estimator of the NTK, approximating the NTK better as $t \rightarrow \infty$. Thus, if the NTK changes smoothly between iterations, one would expect that the moving average will adapt to well-approximate the NTK over time. The full training procedure is summarized in Algorithm 3, and an analogous procedure may be done for the trace of the NTK.

Lastly, note that as the learning rate $\eta \rightarrow 0$, the gradient descent updates change less and less between iterations due to the fact that the loss is smooth, so the moving average should become closer to the actual NTK.

Algorithm 3 Moving average approximation of $K(\theta_t)$

Require: Assumption 2, $\alpha \in (0, 1]$ and N are given.

Ensure: $\widehat{K}_t(\theta_t) \approx K(\theta_t)$ in expectation for all t .

```

Initialize  $\theta_0$ 
 $\widehat{K}_0(\theta_0) \leftarrow \frac{1}{N} \sum_{j=1}^N \widehat{K}^{(j)}(\theta_0),$   $\triangleright \{\widehat{K}^{(j)}(\theta_0)\}_j$  are sampled i.i.d. according to Algorithm 2
 $\widehat{\Lambda}(\theta_0) = \begin{bmatrix} \frac{\text{Tr}(\widehat{K}(\theta_0))}{\text{Tr}(\widehat{K}_{DD}(\theta_0))} I_{n_D}, 0; 0, \frac{\text{Tr}(\widehat{K}(\theta_0))}{\text{Tr}(\widehat{K}_{BB}(\theta_0))} I_{n_B} \end{bmatrix}$   $\triangleright$  As in Eq. (5) with  $\widehat{K}_0(\theta_0)$  in place of  $K(\theta_0)$ 
 $\theta_1 \leftarrow \theta_0 - \eta \nabla \mathcal{R}(\theta_0) \widehat{\Lambda}(\theta_0) \mathcal{R}(\theta_0)$ 
for  $t = 1, 2, \dots, T$  do
     $\widehat{K}_t(\theta_t) \leftarrow (1 - \alpha)\widehat{K}_{t-1}(\theta_{t-1}) + \alpha\widehat{K}^{(1)}(\theta_t),$   $\triangleright \widehat{K}^{(1)}(\theta_t)$  is sampled according to Algorithm 2
     $\widehat{\Lambda}(\theta_t) = \begin{bmatrix} \frac{\text{Tr}(\widehat{K}(\theta_t))}{\text{Tr}(\widehat{K}_{DD}(\theta_t))} I_{n_D}, 0; 0, \frac{\text{Tr}(\widehat{K}(\theta_t))}{\text{Tr}(\widehat{K}_{BB}(\theta_t))} I_{n_B} \end{bmatrix}$   $\triangleright$  As in Eq. (5) with  $\widehat{K}_t(\theta_t)$  in place of  $K(\theta_t)$ 
     $\theta_{t+1} \leftarrow \theta_t - \eta \nabla \mathcal{R}(\theta_t) \widehat{\Lambda}(\theta_t) \mathcal{R}(\theta_t)$ 
end for

```

5 Numerical Results

We now present numerical results to verify our theoretical findings and demonstrate the efficacy of the proposed method for fast NTK estimation. First, in Section 5.1 we present a one-dimensional example exhibiting the optimization convergence rates given in Theorems 1 and 2. Then, we present the results for fast NTK estimation in Section 5.2, including: (i) a quadratically parameterized predictor (Section 5.2.1), comparing the results to the exact NTK and verifying the Monte Carlo error rates in Theorem 4, (ii) a PINN problem to solve a wave equation (Section 5.2.2), and (iii) a multidimensional nonlinear PDE modeling liquid crystals (Section 5.2.3) [9, 18]. Numerical examples involving PINNs have been developed within the framework of the open source software PINA [3].

5.1 Convergence Test

In this experiment, we consider the second-order and one-dimensional equation

$$\begin{cases} u_{xx}(x) = -16\pi^2 \sin(4\pi x), & x \in [0, 1], \\ u(0) = u(1) = 0, \end{cases}$$

and we seek a PINN u_θ , parameterized by θ , to approximate the solution to the equation, which is exactly given by $u(x) = \sin(4\pi x)$.

For this task, we use a neural network with one hidden layer and 100 neurons, hyperbolic tangent as the activation function, initialized with Xavier strategy, and with normalized input to have zero mean and unit standard deviation. To verify the theoretical results from Theorems 1 and 2, we will use the exact NTK in the weights given by (5), following Algorithm 1 with $n = 1$. Because computing the exact NTK is costly, we use a minimal number of collocation points to verify these results, taking $n_{B_1} = n_{B_2} = 1$ collocation points at the boundary points $x = 0$ and $x = 1$, as well as $n_D = 2$ collocation points in $(0, 1)$. With this number of points, we see that $n = n_D + n_{B_1} + n_{B_2} \leq p$, which we recall was required for the lower bound in Assumption 1.3 to hold. The loss from the training is shown in Figure 1.

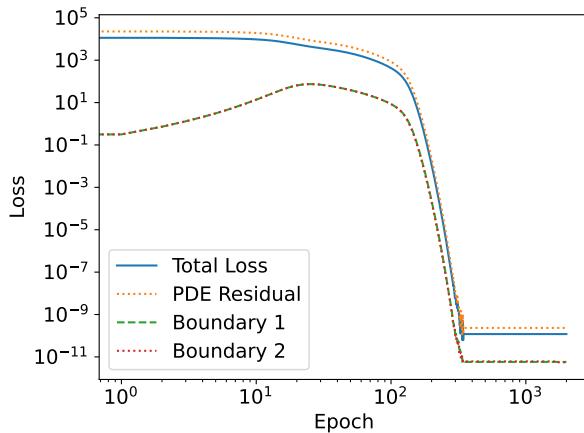


Figure 1: Convergence behavior for the total training loss, composed by equal contribution from the mean PDE residual, and the mean boundary terms.

The plot in Figure 1 shows that the optimization task quickly minimized the loss up to machine precision. Because we take so few collocation points, we cannot expect the resulting PINN u_θ to approximate well the exact solution u , but this setting is already sufficient to see the optimization behavior described by our theory. Indeed, we start by verifying Assumptions 1.2 and 1.3 with the results in Figure 2. In particular, the loss weights λ_D and λ_B remain bounded over the epochs as required by Assumption 1.2, and they remain stable at the end of training. Additionally, we observe that the NTK eigenvalues always increase during training, while they are no longer changing towards the end of optimization. The plot also shows a seemingly continuous evolution of the eigenvalues themselves. This is expected since the NTK is a continuous function of the neural network parameters which should also change continuously. We depicted the sorted eigenvalues of the NTK, so the colors of the eigenvalue trajectories had to be inferred by minimizing measures of distance between values and derivatives of the trajectories which we parameterized to produce what looked qualitatively correct. For example, the smallest eigenvalue at the initial time appears to evolve smoothly to become the largest eigenvalue at the final time, as shown in the blue curve.

To verify the predicted behavior obtained from Theorems 1 and 2, we depict in Figure 3 the time averaged loss and time averaged squared norm of the gradient if the residuals, respectively. We can see that soon after the initialization the rates given by the theorems are empirically recovered during training. However, it is interesting to note that this rate is only observed after the losses appear to begin to decrease significantly (see Figure 1). Indeed, the loss components appear to be more effectively minimized around epoch 100, when at the same time the NTK eigenvalues stabilize, eventually reaching machine precision at about epoch 200.

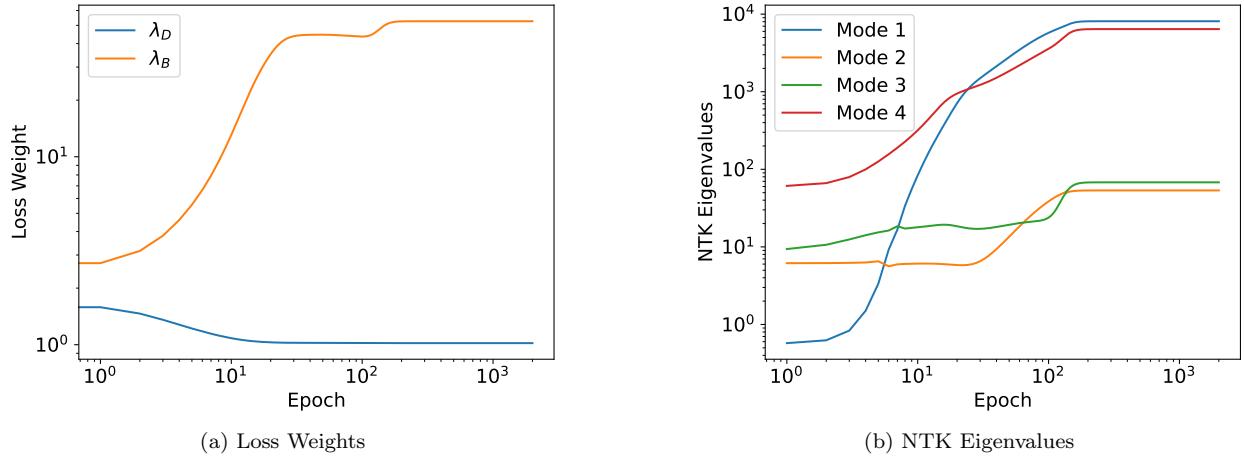


Figure 2: Loss weights and NTK eigenvalues which verify Assumptions 1.2 and 1.3.

These results correspond to the expected decay in Figure 3, where around epoch 100 we begin to observe the $O(T^{-1})$ rate for the time averaged quantities obtained from our theoretical results. In summary, it seems that the NTK eigenvalues were increasing without bound until epoch 100, at which point the eigenvalues and gradient directions could stabilize. Then, the eigenvalues remained bounded, and our theory could be applied. However, the loss also exhibited a decay faster than $O(T^{-1})$, so this may indicate that the loss landscape was approximately convex, allowing faster convergence.

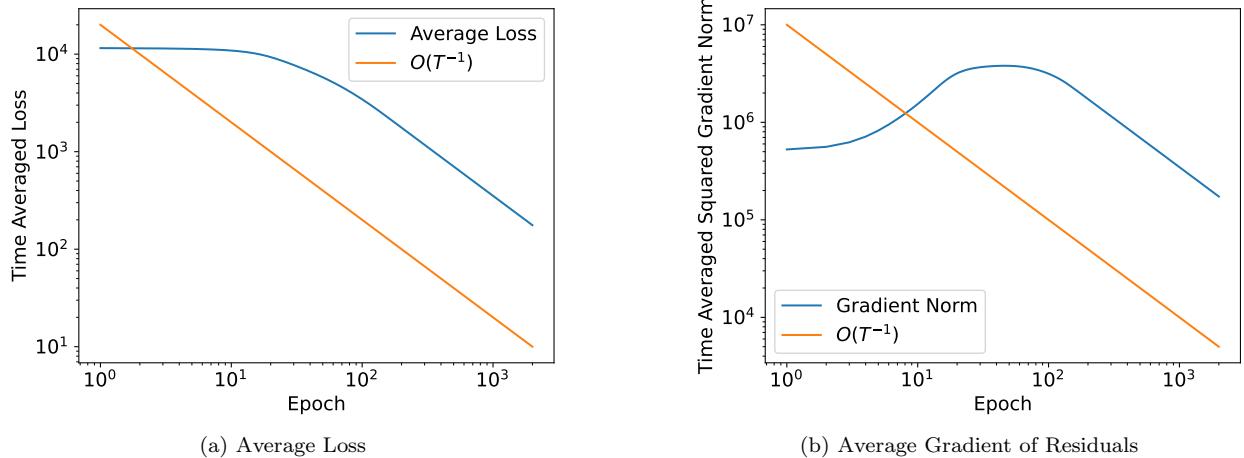


Figure 3: Convergence rates of the time averaged loss and the time averaged norm of gradient of the residuals.

5.2 Fast NTK Estimation

Here, we aim to demonstrate the behavior of the proposed algorithm for estimating the NTK and its trace, as well as our approximation of the NTK-based PINN weights.

5.2.1 Quadratically parameterized predictor

In this example, we verify the theoretical error rate that we obtained for the NTK approximation with respect to the number of samples taken. This example is a simple regression problem, i.e. not in the physics-informed

setting, with a quadratically parameterized predictor. The setup is as follows: we take 50 equispaced values in $[-1, 1]$ denoted by $\{x_i\}_{i=1}^{50}$ and sample the data as

$$y_i = \pi x_i^2 + ex_i + \sqrt{2} + \xi_i, \quad \text{where } \xi_i \stackrel{\text{iid}}{\sim} N(0, 1/\sqrt{2}),$$

where the “true” output is thus given by

$$y = f(x) \doteq \pi x^2 + ex + \sqrt{2}.$$

We aim to learn an approximation \hat{f} of f from the pairs $\{(x_i, y_i)\}_{i=1}^{50}$. By defining the feature map as $u(x) = (1, x, x^2)$, we can write $f(x) = (\theta^* \odot \theta^*) \cdot u(x)$, where $\theta^* = (\pi^{1/2}, e^{1/2}, 2^{1/4})$ and \odot denotes element-wise multiplication. Our goal is to learn $\hat{\theta} \in \mathbb{R}^3$ such that $\hat{f}(x) = (\hat{\theta} \odot \hat{\theta}) \cdot u(x) \approx f(x)$.

Let us define the residual $\mathcal{R}(\theta) \in \mathbb{R}^{50}$ so that

$$\mathcal{R}(\theta)_i = (\theta \odot \theta) \cdot u(x_i) - y_i, \quad \text{for } i = 1, 2, \dots, 50,$$

so that we can then compute $\nabla \mathcal{R}(\theta) \in \mathbb{R}^{3 \times 50}$ exactly as

$$\nabla \mathcal{R}(\theta)_{ji} = 2\theta_j u(x_i)_j.$$

We note that we parameterized our predictor in this precise way so that $\nabla \mathcal{R}(\theta)$ would depend on θ . Indeed, an even simpler linearly parameterized predictor would result in a neural tangent kernel that is independent of θ . The data, exact function f , and predictor \hat{f} found via Algorithm 3 can be seen in Figure 4. As expected, obtaining a reasonable estimator through gradient descent in this example is easy.

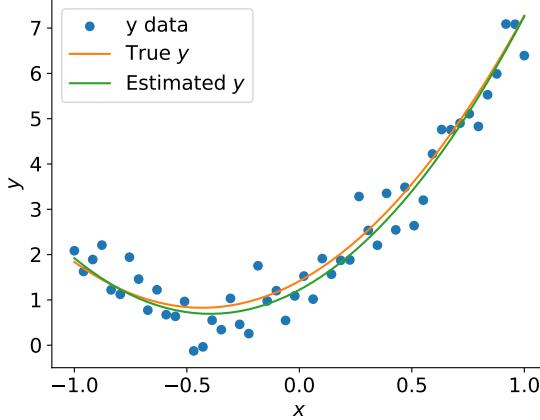


Figure 4: Quadratically parameterized predictor, sampled data, and true output.

The main goal of this section is to verify the theoretical results obtained in Theorems 3 and 4. To verify Theorem 3, we note that, by the law of large numbers, a Monte Carlo approximation of the expected value of our estimator at time 0 will be approximately the expected value of the estimator at that time:

$$\mathbb{E} [\hat{K}(\theta_0)] \approx \frac{1}{N} \sum_{j=1}^N \hat{K}^{(j)}(\theta_0).$$

In Figure 5, we show a sequence of such estimators, where the final estimate with $N = 2 \times 10^4$ samples exhibits an approximate NTK that pretty well matches the exact one at time 0. We note, however, that to obtain an accurate estimate of the NTK, we require a large number of samples. Despite the fact that, exploiting only one sample, the NTK estimate does not look like the exact one, we observe that the estimate with 2000 samples already provides a good qualitative approximation, even though it still contains some artifacts. To quantitatively explore this phenomenon, we now consider our theoretical results for the sequence of estimates.

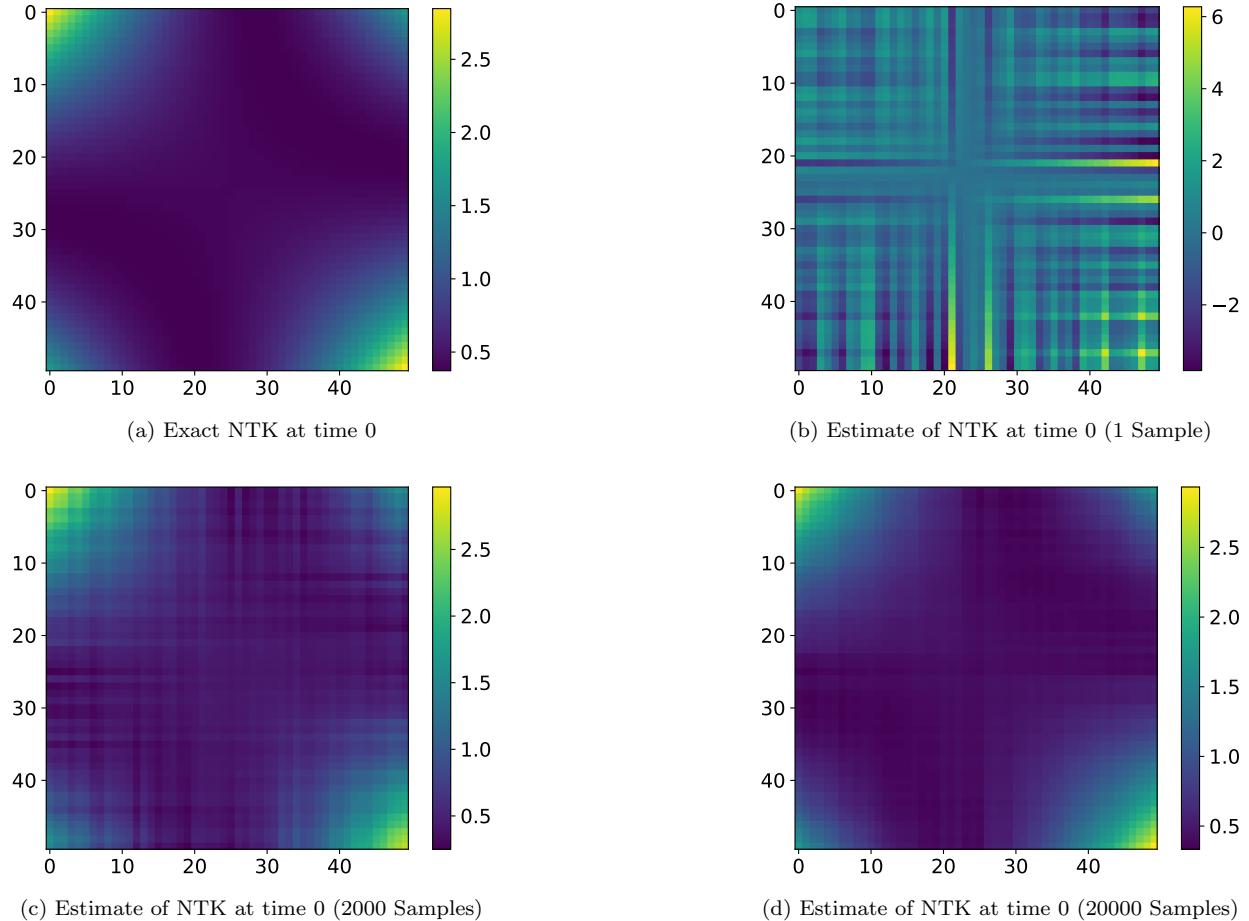


Figure 5: Approximation of NTK at a fixed time for differing numbers of samples.

To verify the Monte Carlo error rate in Theorem 4, we compute empirical averages of errors for $M = 100$ independent Monte Carlo estimates of the NTK with varying N :

$$\mathbb{E} \left[\left\| \frac{1}{N} \sum_{j=1}^N \widehat{K}^{(j,N)}(\theta_0) - K(\theta_0) \right\|_F^2 \right] \approx \frac{1}{M} \sum_{i=1}^M \left\| \frac{1}{N} \sum_{j=1}^N \widehat{K}^{(j,N)}(\theta_0) - K(\theta_0) \right\|_F^2,$$

where the superscript (j, N) denotes the fact that each $\widehat{K}^{(j,N)}$ is an i.i.d. sample from Algorithm 2 for each j and N . An analogous approximation is done for the Monte Carlo trace estimator of the NTK. The results are shown in Figure 6, and, as expected, we observe a rate of the order $O(N^{-1})$ for the error, according to Theorem 4. We note that while the error may seem large, this is because the error is computed for the entire matrix with 2500 entries, resulting in a small average error per entry.

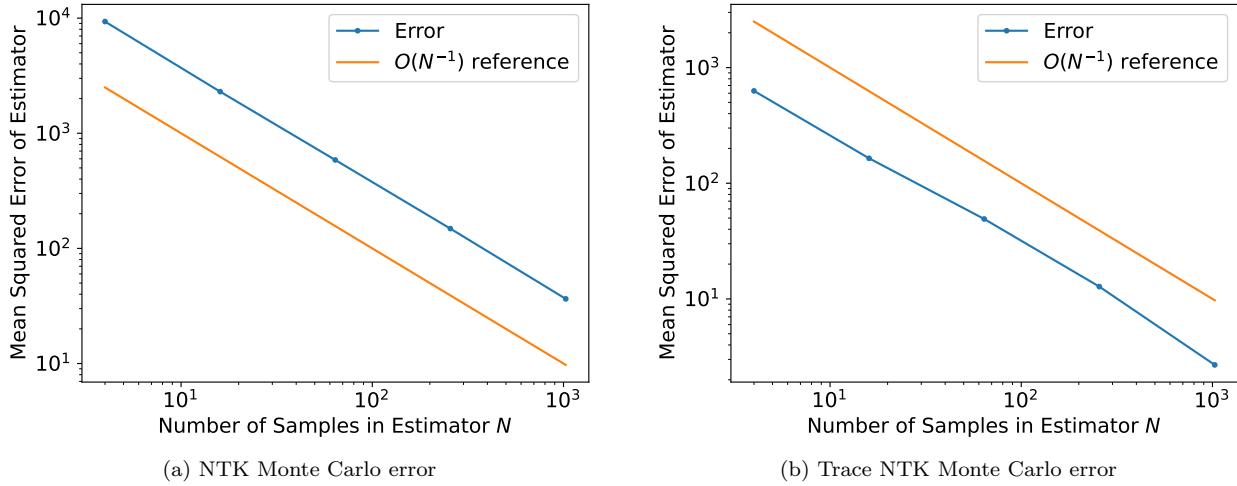


Figure 6: Monte Carlo error rates of efficient NTK and NTK trace estimators.

Finally, in Figure 7 we provide a comparison between the exact and estimated NTKs at the initial and final time $T = 10^5$ after applying Algorithm 3 with $\alpha = 10^{-4}$. We see that at the initial time, the upper right and lower left corners of the exact NTK have larger values than at the final time. The NTK approximation reflects this, showing that the moving average approximation of the NTK is able to adapt to a changing NTK over time. We will further demonstrate this adaptivity via the loss weights in the next experiments.

5.2.2 Wave Equation with PINNs

Here, we aim to show the comparison between the efficient computation of the weights with respect to the exact NTK ones (5). We consider the following problem: find a neural network $u_\theta(t, x)$ with parameters θ that approximately solves the wave equation

$$\begin{cases} u_{tt}(x, t) - 4u_{xx}(x, t) = 0, & (x, t) \in (0, 1) \times (0, 1), \\ u(x, 0) = \sin(\pi x) + \frac{1}{2} \sin(4\pi x), & x \in [0, 1], \\ u(0, t) = u(1, t) = 0, & t \in [0, 1], \\ u_t(x, 0) = 0, & x \in [0, 1]. \end{cases} \quad (15)$$

To obtain an accurate approximation of the physical phenomenon, within this benchmark, we further resample training points at each training iteration, using $n_D = 300$ collocation points for the PDE residual, $n_{D_i} = 300$ points in space for the initial time derivative, $n_{B_i} = 100$ points in space for the initial condition, and $n_{B_1} = n_{B_2} = 100$ points in space for each of the boundaries. Resampling is a common technique that was also used for the original presentation of the NTK-PINN loss weighting algorithm [37], considering a

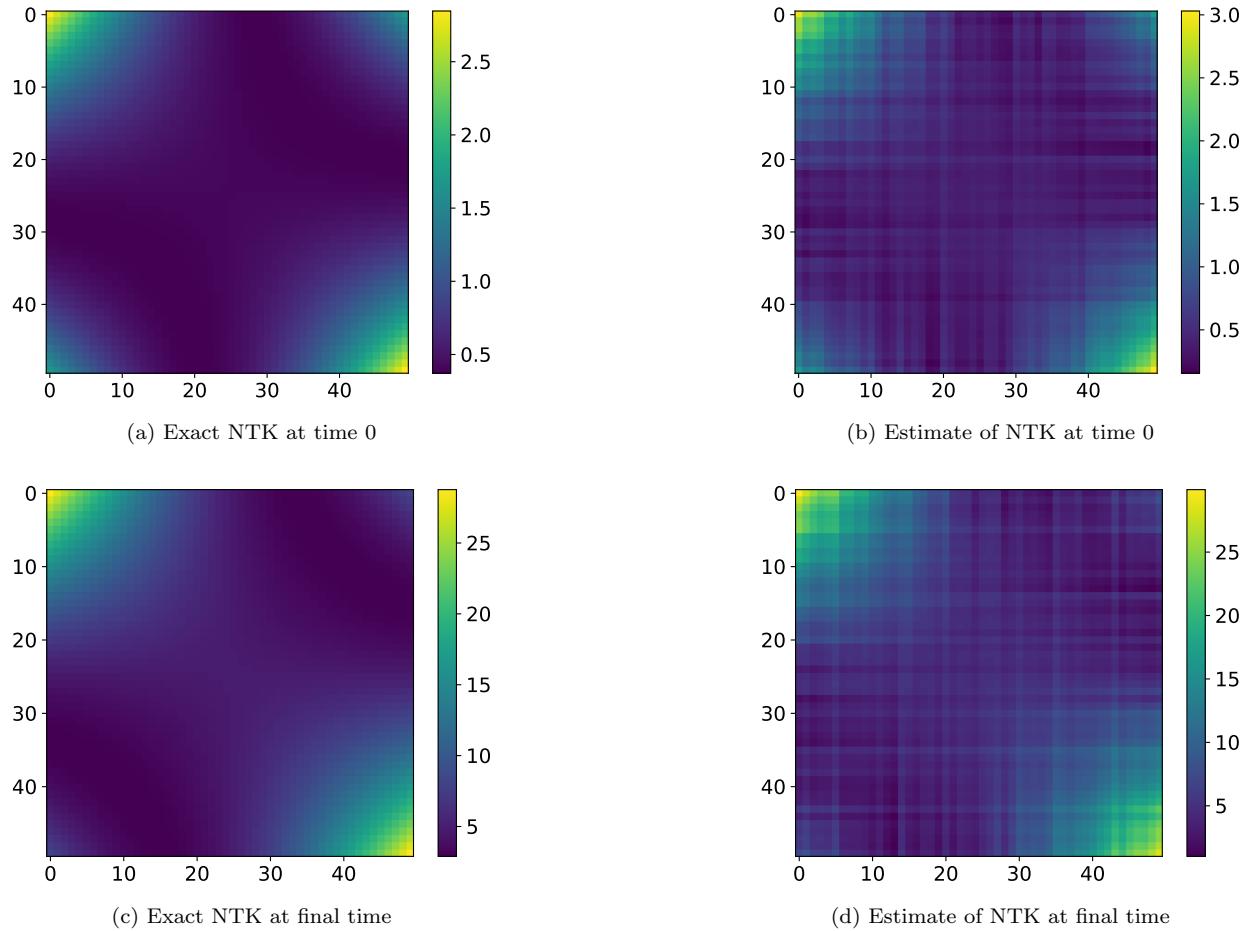


Figure 7: Approximation of NTK with evolution in time.

fixed training batch size sampled from the infinitely many points in the space-time domain. This leads to a smaller generalization error compared to using a fixed training set of the same size. So instead of fixing the points at which we compute the residuals to form the vector \mathcal{R} , we now sample new points at each training step t , resulting in a newly defined residual denoted as $\mathcal{R}_t(\theta)$. Thus, we now effectively approximate $\mathbb{E}[\nabla \mathcal{R}_t(\theta_t)^\top \nabla \mathcal{R}_t(\theta_t)]$, instead of $\nabla \mathcal{R}_t(\theta_t)^\top \nabla \mathcal{R}_t(\theta_t)$, and we use it in place of $K(\theta_t)$ in (5). By applying Algorithm 3 with $\alpha = 10^{-3}$, we obtain the resulting loss weights depicted in Figure 8. In general, we observed that the performance of the estimator is fairly robust to the choice of α , and smaller α lead to smoother loss weight trajectories. In particular, $\alpha = 10^{-4}$ is small enough to see the loss weight trajectories without being obscured by noise in the random estimates of the NTK.

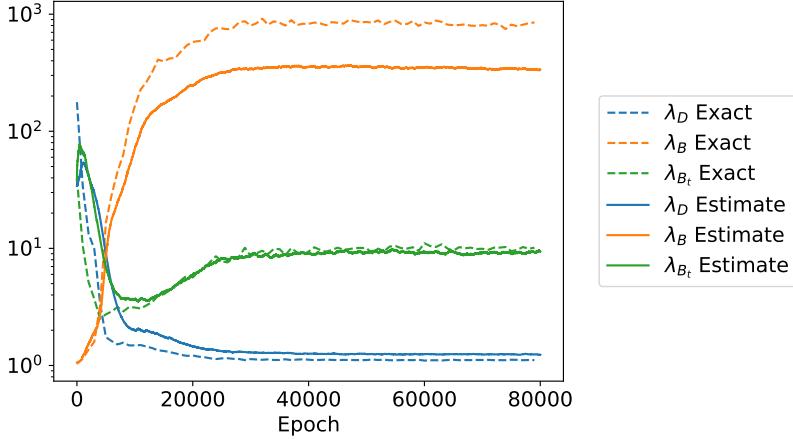


Figure 8: Exact NTK-based loss weights (shown with dashed lines) given by (5), and the loss weights based on the approximated NTK (shown with solid lines) given by Algorithm 3 combined with resampled data.

We see that the exact and approximated loss weights exhibit a very similar behavior, though the largest loss weight is under-approximated, while the smallest is over-approximated. Note that because the loss weights involve the quotient of random variables, even though our trace estimator is unbiased up to a negligible discretization error, the results obtained by replacing K in (5) with its randomized approximations are not unbiased. Despite this discrepancy, the resulting PINN approximation to the wave equation, as shown in Figure 9, has a low error compared to the exact solution given by

$$u(x, t) = \sin(\pi x) \cos(2\pi t) + \frac{1}{2} \sin(4\pi x) \cos(8\pi t).$$

To obtain these results, we used a feedforward neural network with three hidden layers and 500 neurons each, Xavier initialization, and the hyperbolic tangent as the activation function.

5.2.3 Liquid Crystal Q-Tensor PINN

In this final example, we show the efficacy of our weighting algorithm for a nonlinear PDE which describes the evolution of liquid crystal molecule orientations. The model in two spatial dimensions is given by

$$\begin{cases} Q_t = L\Delta Q - aQ - c\text{Tr}(Q^2)Q, & (\mathbf{x}, t) \in \Omega \times [0, T], \\ Q(\mathbf{x}, 0) = Q_0(\mathbf{x}), & \mathbf{x} \in \Omega, \\ Q(\mathbf{x}, t) = 0, & (t, \mathbf{x}) \in \partial\Omega \times [0, T], \end{cases}$$

where $\Omega \subseteq \mathbb{R}^2$ and $Q : \bar{\Omega} \times [0, T] \rightarrow \mathbb{R}^{2 \times 2}$ is a trace-free and symmetric matrix-valued function referred to as the Q-tensor [9, 18]. Additionally, $a \in \mathbb{R}$ and $L, c > 0$ are parameters that characterize the behavior of the solution, and because Q is trace-free and symmetric, there are only two independent variables in this equation. The initial condition is given by

$$Q_0(x, y) = dd^\top - \frac{\|d\|^2}{2}I, \quad d = \frac{[(2-x)(2+x)(2-y)(2+y) \quad \sin(\pi x)\sin(\pi y)]^\top}{\sqrt{1 + [(2-x)(2+x)(2-y)(2+y)]^2 + \sin(\pi x)^2\sin(\pi y)^2}}.$$

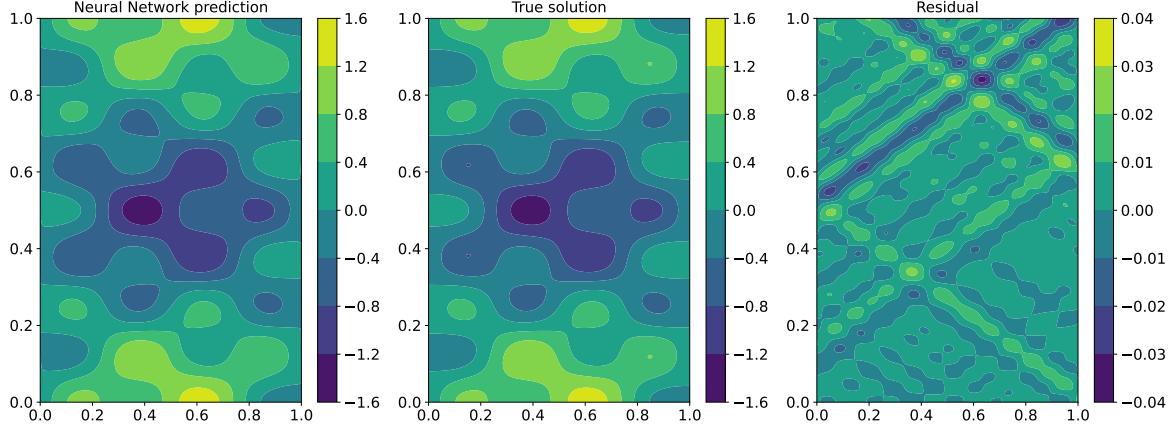


Figure 9: Wave equation solution obtained using the loss weighting Algorithm 3 combined with resampled data.

The eigenvectors of the function Q encode the average direction of the liquid crystal molecules at a point in space, and depending on the direction of the molecules, light may or may not pass through the liquid crystal. The light passing through the liquid crystal can be seen by a contour plot of $|\lambda_1(\mathbf{x}, t)(v_1(\mathbf{x}, t) \cdot (1, 0))|$, where $v_1(\mathbf{x}, t)$ is the eigenvector corresponding to the positive eigenvalue of $Q(\mathbf{x}, t)$.

As in the previous example, we train a PINN model Q_θ with our adaptive weighting algorithm to approximate the solution to the Q-tensor PDE, using the same feedforward neural network with three hidden layers and 500 neurons each. We sample $n_D = n_{B_i} = 300$ data points each for the interior residual and for the initial condition, while for each portion of the boundaries we use $n_{B_k} = 10$ data samples for $k = 1, 2, 3, 4$, still resampling the points at each iteration of training.

In Figure 10, we plot the resulting PINN (denoted NTK-PINN) compared to a reference Finite Element solution computed with the invariant quadratization method [18], as well as compared to a PINN trained without the adaptive weighting algorithm (denoted PINN). The contours show the aforementioned visualization of light passing through the liquid crystal, while the blue vectors represent the eigenvectors $v_1(\mathbf{x}, t)$. We see that all three solutions are qualitatively very similar, providing an accurate reconstruction of the contour plots and of the eigenvectors reaching the horizontal equilibrium state at time $t = 2$. We note that the artifacts around the boundary for the NTK-PINN and PINN solutions are due to the fact that $Q = 0$ on the boundary, so computing the eigenvectors will be sensitive to any error, while for the Finite Element solution, Q is constrained to equal 0 exactly on the boundary. Thus, even though the use of the adaptive loss weights makes little difference in the quality of solution for this example, we emphasize that the Algorithm 3 can offer enough improvement in computational efficiency to make such an adaptive NTK-based loss weighting feasible.

6 Conclusion

In this paper, we presented a convergence analysis for a neural tangent kernel-based loss weighting scheme in multi-objective optimization with a physics-informed loss. We showed that, under certain assumptions, the average squared norms of the residuals and of the loss gradients converge to zero. To address concerns with computational complexity, we further proposed a randomized algorithm which allowed us to obtain sketches of the NTK. This way, we demonstrated that, using a moving average, computing adaptive NTK approximations only costs one additional evaluation and backpropagation of our physics-informed neural network per iteration. We provided error estimates to justify the proposed randomized algorithm, and we verified our theoretical results numerically. We performed further numerical tests demonstrating the efficacy of our methodology for the wave equation and a nonlinear model of liquid crystal dynamics. In the future, we would also like to study the convergence of PINN training in the infinite width limit, particularly for nonlinear PDEs. Another interesting open question is related to the discovery of the optimal weights used to balance

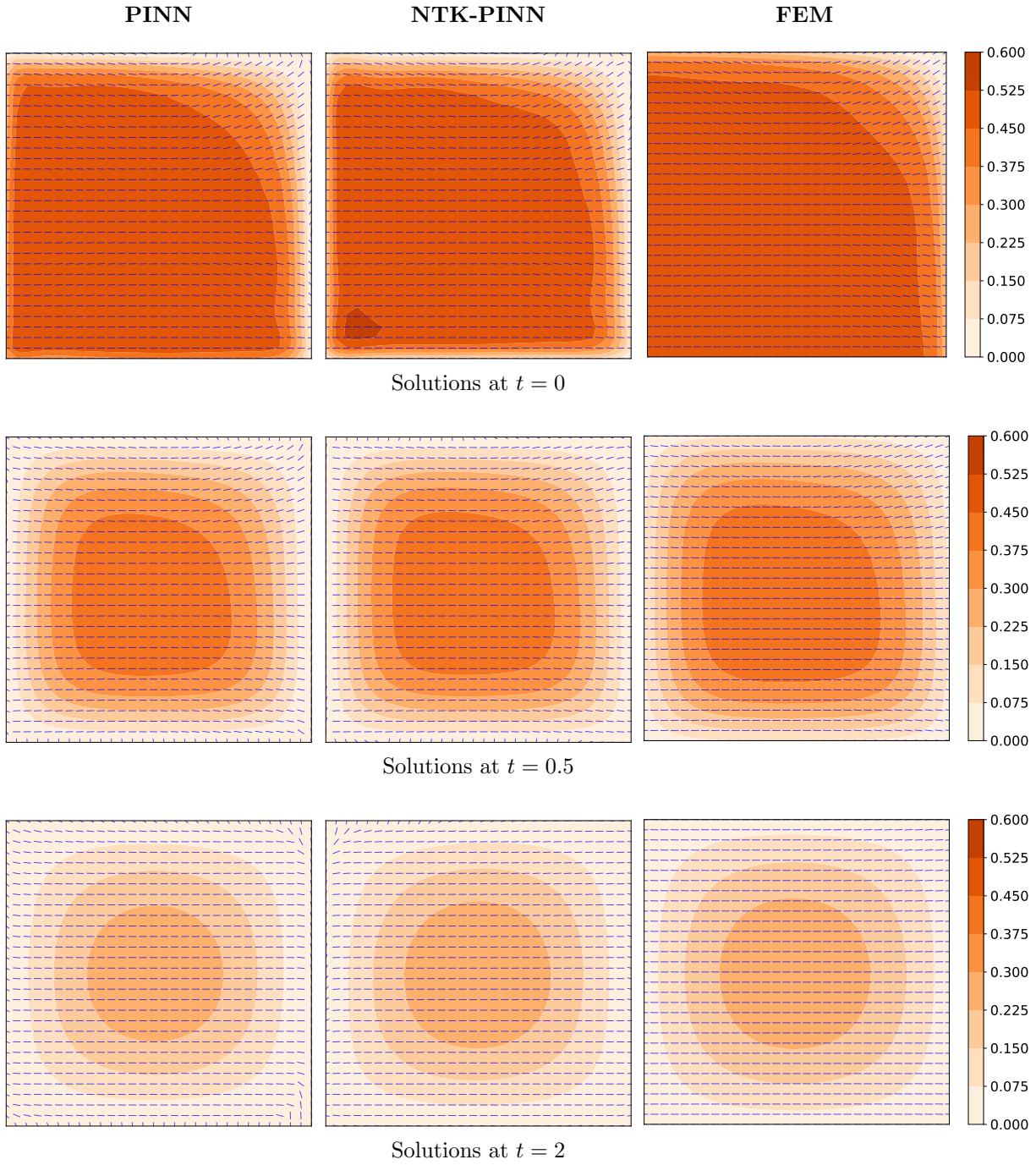


Figure 10: Finite Element solution (FEM) compared to a PINN solution trained with the approximate NTK weight algorithm (NTK-PINN) and a PINN solution trained without adaptive weighting (PINN).

the loss functions. Lastly, an important application of our work will be to physics-informed machine learning for reduced order modeling. In this context, the neural networks used are vector-valued, and computing the loss weights based on the NTK is particularly expensive, so we envision our NTK approximation algorithm having great benefits in that context [1, 11].

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 2146752. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. **FP** acknowledges the support provided by the European Union - NextGenerationEU, in the framework of the iNEST - Interconnected Nord-Est Innovation Ecosystem (iNEST ECS00000043 - CUP G93C22000610007) consortium and its CC5 Young Researchers initiative. **FP** also acknowledges INdAM-GNSC and the project “Sviluppo e analisi di modelli di ordine ridotto basati su tecniche di deep learning” (CUP E53C24001950001) for its support.

A Proof of Descent Lemma 1

Proof. Let $g(t) = F(y + t(x - y))$, which is well-defined since \mathcal{P} is convex. Then $g'(t) = \langle \nabla F(y + t(x - y)), x - y \rangle$. By the fundamental theorem of calculus, we have

$$F(x) - F(y) = g(1) - g(0) = \int_0^1 g'(t) dt = \int_0^1 \langle \nabla F(y + t(x - y)), x - y \rangle dt.$$

Subtract $\langle \nabla F(y), x - y \rangle$ from both sides of this equation and take absolute values to obtain

$$\begin{aligned} |F(x) - F(y) - \langle \nabla F(y), x - y \rangle| &\leq \int_0^1 |\langle \nabla F(y + t(x - y)) - \nabla F(y), x - y \rangle| dt \\ &\leq \int_0^1 \|\nabla F(y + t(x - y)) - \nabla F(y)\| \cdot \|x - y\| dt \\ &\leq \int_0^1 Lt \|x - y\|^2 dt \\ &= \frac{L}{2} \|x - y\|^2, \end{aligned}$$

which completes the proof. \square

B Proof of Theorem 5

Proof. Performing a Taylor expansion, we have

$$\begin{aligned} \mathbb{E} [\|\mathcal{R}(\theta_t + g_t) - \mathcal{R}(\theta_t)\|_F^2] &= \mathbb{E} [\|\mathcal{R}(\theta_t) + \nabla \mathcal{R}(\theta_t)^\top g_t + \mathcal{E}(\theta_t, g_t) - \mathcal{R}(\theta_t)\|_F^2] \\ &= \mathbb{E} [\|\nabla \mathcal{R}(\theta_t)^\top g_t + \mathcal{E}(\theta_t, g_t)\|_F^2], \end{aligned}$$

where we define $\mathcal{E}(\theta_t, g_t)$ to be the Taylor remainder, which by Assumption 2.3 satisfies

$$\|\mathcal{E}(\theta_t, g_t)\|_F \leq C\varepsilon^2 \|g_t/\varepsilon\|^2.$$

Combining this with the fact that $\mathbb{E}[\|g_t/\varepsilon\|^3] \leq \mathbb{E}[\|g_t/\varepsilon\|^4] = n(n+2)$, since g_t/ε is a standard Gaussian vector, gives

$$\mathbb{E} [\|g_t\| \|\mathcal{E}(\theta_t, g_t)\|_F] = \varepsilon \mathbb{E} [\|g_t/\varepsilon\| \|\mathcal{E}(\theta_t, g_t)\|_F] \leq C\varepsilon^3 n(n+2), \quad \text{and} \quad \mathbb{E} [\|\mathcal{E}(\theta_t, g_t)\|_F^2] \leq C^2 \varepsilon^4 n(n+2).$$

Again, by Assumption 2.3, it follows that

$$\begin{aligned}\mathbb{E} [\|\mathcal{R}(\theta_t + g_t) - \mathcal{R}(\theta_t)\|_F^2] &= \mathbb{E} [\|\nabla \mathcal{R}(\theta_t)^\top g_t\|_F^2 + 2\langle \nabla \mathcal{R}(\theta_t)^\top g_t, \mathcal{E}(\theta_t, g_t) \rangle + \|\mathcal{E}(\theta_t, g_t)\|_F^2] \\ &\leq \mathbb{E} [\|\nabla \mathcal{R}(\theta_t)^\top g_t\|_F^2 + 2C\|g_t\| \|\mathcal{E}(\theta_t, g_t)\|_F + \|\mathcal{E}(\theta_t, g_t)\|_F^2] \\ &\leq \mathbb{E} [\|\nabla \mathcal{R}(\theta_t)^\top g_t\|_F^2] + 2C\varepsilon^3 n(n+2) + C^2\varepsilon^4 n(n+2),\end{aligned}$$

and similarly,

$$\mathbb{E} [\|\mathcal{R}(\theta_t + g_t) - \mathcal{R}(\theta_t)\|_F^2] \geq \mathbb{E} [\|\nabla \mathcal{R}(\theta_t)^\top g_t\|_F^2] - 2C\varepsilon^3 n(n+2).$$

Now, using the cyclic property of the trace, we have that

$$\mathbb{E} [\|\nabla \mathcal{R}(\theta_t)^\top g_t\|_F^2] = \mathbb{E} [\text{Tr}(g_t^\top \nabla \mathcal{R}(\theta_t) \nabla \mathcal{R}(\theta_t)^\top g_t)] = \text{Tr}(\nabla \mathcal{R}(\theta_t)^\top \mathbb{E}[g_t g_t^\top] \nabla \mathcal{R}(\theta_t)) = \varepsilon^2 \text{Tr}(K(\theta_t)).$$

Thus, we obtain

$$-2C\varepsilon n(n+2) \leq \mathbb{E} \left[\left\| \frac{\mathcal{R}(\theta_t + g_t) - \mathcal{R}(\theta_t)}{\varepsilon} \right\|_F^2 \right] - \text{Tr}(K(\theta_t)) \leq 2C\varepsilon n(n+2) + C^2\varepsilon^2 n(n+2),$$

and the result follows from $n(n+2) < (n+1)^2$ and by bounding from above with a larger suitable constant. \square

References

- [1] W. Chen, Q. Wang, J. S. Hesthaven, and C. Zhang. Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Journal of Computational Physics*, 446:110666, 2021.
- [2] Y. Collette and P. Siarry. *Multiobjective Optimization*. Decision Engineering. Springer Berlin, Heidelberg, 1 edition, 2004. Original French edition published by Groupe Eyrolles, 2002. Part of the Springer Book Archive. Copyright © Springer-Verlag Berlin Heidelberg 2004.
- [3] D. Coscia, A. Ivagnes, N. Demo, and G. Rozza. Physics-Informed Neural networks for Advanced modeling. *Journal of Open Source Software*, 8(87):5352, 2023.
- [4] S. A. Faroughi and F. Mostajeran. Neural Tangent Kernel Analysis to Probe Convergence in Physics-informed Neural Solvers: PIKANs vs. PINNs, 2025.
- [5] B. Gao, R. Yao, and Y. Li. Physics-informed neural networks with adaptive loss weighting algorithm for solving partial differential equations. *Computers & Mathematics with Applications*, 181:216–227, 2025.
- [6] Y. Gao, Y. Gu, and M. Ng. Gradient Descent Finds the Global Optima of Two-Layer Physics-Informed Neural Networks. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10676–10707. PMLR, 2023.
- [7] S. Ghadimi and G. Lan. Stochastic First- and Zeroth-Order Methods for Nonconvex Stochastic Programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.
- [8] A. Girard. A fast ‘Monte-Carlo cross-validation’ procedure for large least squares problems with noisy data. *Numerische Mathematik*, 56(1):1–23, 1989.
- [9] V. M. Gudibanda, F. Weber, and Y. Yue. Convergence Analysis of a Fully Discrete Energy-Stable Numerical Scheme for the Q-Tensor Flow of Liquid Crystals. *SIAM Journal on Numerical Analysis*, 60(4):2150–2181, 2022.
- [10] Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, and J. Zhu. Physics-Informed Machine Learning: A Survey on Problems, Methods and Applications, 2023.

- [11] M. Hirsch, F. Pichi, and J. S. Hesthaven. Neural Empirical Interpolation Method for Nonlinear Model Reduction. *SIAM Journal on Scientific Computing*, pages C1264–C1293, 2025.
- [12] N. Jha and E. Mallik. GPINN with Neural Tangent Kernel Technique for Nonlinear Two Point Boundary Value Problems. *Neural Processing Letters*, 56(3):192, 2024.
- [13] S. Jiang, Y. Man, Z. Song, Z. Yu, and D. Zhuo. Fast Graph Neural Tangent Kernel via Kronecker Sketching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):7033–7041, 2022.
- [14] B. Jin and L. Wu. Convergence of Stochastic Gradient Methods for Wide Two-Layer Physics-Informed Neural Networks, 2025.
- [15] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.
- [16] I. Y. Kim and O. L. de Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29(2):149–158, 2005.
- [17] Y. Liu, L. Cai, Y. Chen, and B. Wang. Physics-informed neural networks based on adaptive weighted loss functions for Hamilton-Jacobi equations. *Mathematical Biosciences and Engineering*, 19(12):12866–12896, 2022.
- [18] M. Hirsch, F. Weber, and Y. Yue. The zero inertia limit for the Q-tensor model of liquid crystals: analysis and numerics. *ESAIM: M2AN*, 59(5):2515–2556, 2025.
- [19] R. T. Marler and J. S. Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization*, 41(6):853–862, 2010.
- [20] P.-G. Martinsson and J. A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020.
- [21] L. D. McCleny and U. M. Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722, 2023.
- [22] C. Meng, S. Griesemer, D. Cao, S. Seo, and Y. Liu. When physics meets machine learning: a survey of physics-informed machine learning. *Machine Learning for Computational Science and Engineering*, 1(1):20, 2025.
- [23] M. A. Mohamadi, W. Bae, and D. J. Sutherland. A fast, well-founded approximation to the empirical neural tangent kernel. In *Proceedings of the 40th International Conference on Machine Learning*, ICML’23. JMLR.org, 2023.
- [24] Y. Nesterov. *Introductory Lectures on Convex Optimization*, volume 87 of *Applied Optimization*. Springer US, Boston, MA, 2004.
- [25] R. Novak, J. Sohl-Dickstein, and S. S. Schoenholz. Fast Finite Width Neural Tangent Kernel. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17018–17044. PMLR, 2022.
- [26] R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, J. Hron, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein. Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes. In *ICLR (Poster)*. OpenReview.net, 2019.
- [27] S. J. Prince. *Understanding Deep Learning*. The MIT Press, 2023.
- [28] A. Quarteroni, P. Gervasio, and F. Regazzoni. Combining Physics-Based and Data-Driven Models: Advancing the Frontiers of Research with Scientific Machine Learning. *Mathematical Models and Methods in Applied Sciences*, 35(04):905–1071, 2025.

- [29] C. Rackauckas. Parallel Computing and Scientific Machine Learning (SciML): Methods and Applications, 2022.
- [30] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- [31] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko. A Gentle Introduction to Graph Neural Networks. *Distill*, 2021. <https://distill.pub/2021/gnn-intro>.
- [32] C. Song, Y. Park, and M. Kang. How does PDE order affect the convergence of PINNs? In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS ’24, Red Hook, NY, USA, 2025. Curran Associates Inc.
- [33] Z. Tan, T. Liu, and W. Huang. Understanding Neural Tangent Kernel Dynamics Through Its Trace Evolution, 2025.
- [34] J. F. Urbán, P. Stefanou, and J. A. Pons. Unveiling the optimization process of physics informed neural networks: How accurate and competitive can PINNs be? *Journal of Computational Physics*, 523:113656, 2025.
- [35] J. Wang, X. Xiao, X. Feng, and H. Xu. An improved physics-informed neural network with adaptive weighting and mixed differentiation for solving the incompressible Navier–Stokes equations. *Nonlinear Dynamics*, 112(18):16113–16134, 2024.
- [36] S. Wang, Y. Teng, and P. Perdikaris. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.
- [37] S. Wang, X. Yu, and P. Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.
- [38] R. Ward, X. Wu, and L. Bottou. AdaGrad stepsizes: sharp convergence over nonconvex landscapes. *J. Mach. Learn. Res.*, 21(1), 2020.
- [39] Z. Xiang, W. Peng, X. Liu, and W. Yao. Self-adaptive loss balanced physics-informed neural networks. *Neurocomputing*, 496:11–34, 2022.
- [40] X.-S. Yang. Chapter 14 - Multi-Objective Optimization. In X.-S. Yang, editor, *Nature-Inspired Optimization Algorithms*, pages 197–211. Elsevier, Oxford, 2014.
- [41] S. Yeonjong, D. Jérôme, and E. Karniadakis, George. On the Convergence of Physics Informed Neural Networks for Linear Second-Order Elliptic and Parabolic Type PDEs. *Communications in Computational Physics*, 28(5):2042–2074, 2020.
- [42] A. Zandieh, I. Han, H. Avron, N. Shoham, C. Kim, and J. Shin. Scaling Neural Tangent Kernels via Sketching and Random Features. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1062–1073. Curran Associates, Inc., 2021.
- [43] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.
- [44] Z. Zhou and Z. Yan. Is the neural tangent kernel of PINNs deep learning general partial differential equations always convergent? *Physica D: Nonlinear Phenomena*, 457:133987, 2024.