ANaGRAM: A Natural Gradient Relative to Adapted Model for efficient PINNs learning

Anonymous authors
Paper under double-blind review

Abstract

In the recent years, Physics Informed Neural Networks (PINNs) have received strong interest as a method to solve PDE driven systems, in particular for data assimilation purpose. This method is still in its infancy, with many shortcomings and failures that remain not properly understood. In this paper we propose a natural gradient approach to PINNs which contributes to speed-up and improve the accuracy of the training. Based on an in depth analysis of the differential geometric structures of the problem, we come up with two distinct contributions: (i) a new natural gradient algorithm that scales as $\min(P^2S, S^2P)$, where P is the number of parameters, and S the batch size; (ii) a mathematically principled reformulation of the PINNs problem that allows the extension of natural gradient to it, with proved connections to Green's function theory.

1 Introduction

Following the spectacular success of neural networks for over a decade (LeCun et al., 2015), intensive work has been carried out to apply these methods to numerical analysis (Cuomo et al., 2022).In particular, following the pioneering work of Dissanayake & Phan-Thien (1994) and Lagaris et al. (1998), Raissi et al. (2019a) have introduced Physics Informed Neural Networks (PINNs), a method designed to approximate solutions of partial differential equations (PDEs), using deep neural networks. Theoretically based on the universal approximation theorem of neural networks (Leshno et al., 1993), and put into practice by automatic differentiation (Baydin et al., 2018) for the computation of differential operators, this method has enjoyed a number of successes in fields as diverse as fluid mechanics (Raissi et al., 2019c;b; Sun et al., 2020; Raissi et al., 2020; Jin et al., 2021; de Wolff et al., 2021), bio-engineering (Sahli Costabal et al., 2020; Kissas et al., 2020) or free boundary problems (Wang & Perdikaris, 2021). Nevertheless, many limitations have been pointed out, notably the inability of these methods in their current formulation to obtain high-precision approximations when no additional data is provided (Krishnapriyan et al., 2021; Wang et al., 2021; Karnakov et al., 2022; Zeng et al., 2022). Recent work by Müller & Zeinhofer (2023), however, has substantially altered this state of affairs, proposing an algorithm similar to natural gradient methods in case of linear operator (*cf.* Appendix E), that achieves accuracies several orders of magnitude above previous methods.

Contributions: Müller & Zeinhofer (2024) argue for the need to take function-space geometry into account in order to further understand and perfect scientific machine-learning methods. With this paper, we intend to support and extend their approach by making several contributions:

- (i) We highlight a principled mathematical framework that restates natural gradient in an equivalent, yet simpler way, leading us to propose ANaGRAM, a general-purpose natural gradient algorithm of reduced complexity $\mathcal{O}(\min(P^2S, S^2P))$ compared to $\mathcal{O}(P^3)$, where $P = \#parameters$ and $S = \#batch\ samples$.
- (ii) We reinterpret the PINNs framework from a functional analysis perspective in order to extend ANaGRAM to the PINN's context in a straightforward manner.
- (iii) We establish a direct correspondence between ANaGRAM for PINNs and the Green's function of the operator on the tangent space.

The rest of this article is organized as follows: in Section 2, after introducing neural networks and parametric models in Section 2.1 from a functional analysis perspective, we review two concepts crucial to our work: PINNs framework in Section 2.2, and natural gradient in Section 2.3. In Section 3, we introduce the notions of empirical tangent space and an expression for the corresponding notion of empirical natural gradient leading to ANaGRAM 1. In Section 4, after reinterpreting PINNs as a regression problem from the right functional perspective in Section 4.1, yielding ANaGRAM algorithm 2 for PINNs, we state in Section 4.2 that natural gradient matches the Green's function of the operator on the tangent space and analyse the consequence of this on the interpretation of PINNs training process under ANaGRAM. Finally, in Section 5, we show empirical evidences of the performance of ANaGRAM on a selected benchmark of PDEs.

2 Position of the problem

2.1 Neural Networks and parametric model

Our starting point is the following functional definition of parametric models, of which neural networks are a non-linear special case:

Definition 1 (Parametric model)**.** Given a domain Ω of $\mathbb{R}^n$, $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ and a Hilbert space $\mathcal{H}$ compound of functions $\Omega \to \mathbb{K}^m$, a parametric model is a differentiable functional:

$$u : \left\{ \begin{array}{ccc} \mathbb{R}^P & \to & \mathcal{H} \\ \boldsymbol{\theta} & \mapsto & \big(\boldsymbol{x} \in \Omega \mapsto u(\boldsymbol{x}; \boldsymbol{\theta})\big) \end{array} \right. . \tag{1}$$

To prevent confusion, we will write $u_{|\boldsymbol{\theta}}(\boldsymbol{x})$ instead of $u\big(\boldsymbol{\theta}\big)(\boldsymbol{x})$, for all $\boldsymbol{x} \in \Omega$

Since a parametric model is differentiable by definition, we can define its differential:

Definition 2 (Differential of a parametric model)**.** Let $u : \mathbb{R}^P \to \mathcal{H}$ be a parametric model and $\boldsymbol{\theta} \in \mathbb{R}^P$. Then the differential of the parametric model u in the parameter $\boldsymbol{\theta}$ is:

$$\mathrm{d}u_{|\boldsymbol{\theta}} : \left\{ \begin{array}{ccc} \mathbb{R}^P & \to & \mathcal{H} \\ \boldsymbol{h} & \mapsto & \sum_{p=1}^P \boldsymbol{h}_p \frac{\partial u}{\partial \boldsymbol{\theta}_p} \end{array} \right. , \tag{2}$$

To simplify notations, we will write for all $1 \leqslant p \leqslant P$ and for all $\boldsymbol{\theta} \in \mathbb{R}^P$, $\partial_p u_{|\boldsymbol{\theta}}$, instead of $\frac{\partial u}{\partial \boldsymbol{\theta}_p}$.

Given a parametric model u, we can define the following two objects of interest:

The image set of u : this is the set of functions reached by u, *i.e.* :

$$\mathcal{M} := \operatorname{Im} u := \big\{ u_{|\boldsymbol{\theta}} \,:\, \boldsymbol{\theta} \in \mathbb{R}^P \big\} \tag{3}$$

Although not strictly rigorous[1], $\mathcal{M}$ is often considered in deep-learning as a differential submanifold of $\mathcal{H}$, so we will keep this analogy in mind for pedagogical purposes.

The tangent space of u **at** $\boldsymbol{\theta}$: this is the image set of the differential of u at $\boldsymbol{\theta}$, *i.e.* the linear subspace of $\mathcal{H}$ compound of functions reached by $\mathrm{d}u_{|\boldsymbol{\theta}}$, *i.e.* :

$$T_{\boldsymbol{\theta}}\mathcal{M} := \operatorname{Im} \mathrm{d}u_{|\boldsymbol{\theta}} = \operatorname{Span}\big(\partial_p u_{|\boldsymbol{\theta}} \,:\, 1 \leqslant p \leqslant P\big) \tag{4}$$

Once again, this definition is made with reference to differential geometry.

We give several examples of Parametric models in Appendix B. We now introduce PINNs.

2.2 Physics Informed Neural Networks (PINNs)

As in Definition 1, let us consider a domain Ω of $\mathbb{R}^n$ endowed with a probability measure μ, $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$, $\partial\Omega$ its boundary endowed with a probability measure σ, and $\mathcal{H}$ a Hilbert space compound of functions $\Omega \to \mathbb{K}^m$. Then let us consider two functional operators:

$$D : \left\{ \begin{array}{ccc} \mathcal{H} & \to & \mathrm{L}^2(\Omega \to \mathbb{R}, \mu) \\ u & \mapsto & D[u] \end{array} \right. , \qquad B : \left\{ \begin{array}{ccc} \mathcal{H} & \to & \mathrm{L}^2(\partial\Omega \to \mathbb{R}, \sigma) \\ u & \mapsto & B[u] \end{array} \right. , \tag{5}$$

[1] In particular, because u may not be injective.

that we will assume to be differentiable[2]. We can then consider the PDE:

$$\begin{cases} D(u) = f \in \mathrm{L}^2(\Omega \to \mathbb{R}, \mu) & \text{in } \Omega \\ B(u) = g \in \mathrm{L}^2(\partial\Omega \to \mathbb{R}, \sigma) & \text{on } \partial\Omega \end{cases}. \tag{6}$$

The PINNs framework, as introduced by Raissi et al. (2019a) consists then in approximating a solution to the PDE by making the ansatz $u = u_{|\boldsymbol{\theta}}$, with $u_{|\boldsymbol{\theta}}$ a neural network, sampling points $(x_i^D)_{1 \leqslant i \leqslant S_D}$ in Ω according to μ, $(x_i^B)_{1 \leqslant i \leqslant S_B}$ in $\partial\Omega$ according to σ and then to optimize the loss:

$$\ell(\boldsymbol{\theta}) := \frac{1}{2S_D} \sum_{i=1}^{S_D} \left(D[u_{|\boldsymbol{\theta}}](x_i^D) - f(x_i^D) \right)^2 + \frac{1}{2S_B} \sum_{i=1}^{S_B} \left(B[u_{|\boldsymbol{\theta}}](x_i^B) - g(x_i^B) \right)^2 \tag{7}$$

by classical gradient descent techniques, used in the context of deep learning, such as Adam (Kingma & Ba, 2014), or L-BFGS (Liu & Nocedal, 1989). One of the cornerstones of Raissi et al. (2019a) is also to use automatic differentiation (Baydin et al., 2018) to calculate the operators D and B, thus obtaining quasi-exact calculations, whereas most classic techniques require either approximating operators as for Finite Differences, or carrying out the calculations manually as for Finite Elements.

Although appealing due to its simplicity and relative ease of implementation, this approach suffers from several well-documented empirical pathologies (Krishnapriyan et al., 2021; Wang et al., 2021; Grossmann et al., 2024), which can be understood as an ill conditioned problem (De Ryck et al., 2024; Liu et al., 2024) and for which several *ad hoc* procedures has been proposed (Karnakov et al., 2022; Zeng et al., 2022; McClenny & Braga-Neto, 2022). Following Müller & Zeinhofer (2024), we argue in this work that the key point is rather to theoretically understand the geometry of the problem and adapt PINNs training accordingly.

2.3 Natural Gradient

Natural gradient has been introduced, in the context of Information Geometry by Amari & Douglas (1998). Given a loss: $\ell : \boldsymbol{\theta} \to \mathbb{R}^+$, the gradient descent:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \, \nabla \ell,$$

is replaced by the update:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \, F_{\boldsymbol{\theta}_t}^{\dagger} \nabla \ell, \tag{8}$$

with $F_{\boldsymbol{\theta}_t}$ being the Gram-Matrix associated to a Fisher-Rao information metric (Amari, 2016) or equivalently, the Hessian of some Kullback-Leibler divergence (Kullback & Leibler, 1951), and $\dagger$ the Moore-Penrose pseudo-inverse. This notion has been later further extended to the more abstract setting of Riemannian metrics in the context of neural-networks by Ollivier (2015). In this case, given a Riemannian-(pseudo) metric $\mathcal{G}_{\boldsymbol{\theta}}$, the gradient-descent update is replaced by:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta G_{\boldsymbol{\theta}_t}^{\dagger} \nabla \ell, \tag{9}$$

where $G_{\boldsymbol{\theta}_t \, p,q} := \mathcal{G}_{\boldsymbol{\theta}_t}(\partial_p u_{|\boldsymbol{\theta}_t}, \partial_q u_{|\boldsymbol{\theta}_t})$ is the Gram matrix of partial derivatives relative to $\mathcal{G}_{\boldsymbol{\theta}_t}$. Despite its mathematically principled advantage, natural gradient suffers from its computational cost, which makes it prohibitive, if not untractable for real world applications. Indeed:

- Computation of the Gram matrix $G_{\boldsymbol{\theta}_t}$ is quadratic in the number of parameters.
- Inversion of $G_{\boldsymbol{\theta}_t}$ is cubic in the number of parameters.

Different approaches have been proposed to circumvent this limitations. The most prominent one is K-FAC introduced by Heskes (2000) and further extended by Martens & Grosse (2015); Grosse & Martens (2016), which approximates the Gram matrix by block-diagonal matrices. This approximation can be understood as making the ansatz that the partial derivatives of weights belonging to different layers are orthogonal. A refinement of this method has been proposed by George et al. (2018), in which the eigen-structure of the block-diagonal matrices are carefully taken into account in order to provide a better approximation of the diagonal rescaling induced by the inversion of the Gram matrix. In a completely different vein, Ollivier (2017) has proposed a statistical approach that has been proved to converge to the natural gradient update in the 0 learning rate limit.

[2]It can be shown that, if D and B are defined and differentiable on $\mathcal{C}^\infty(\Omega \to \mathbb{R}^m)$ then such a $\mathcal{H}$ always exists; *cf.* chapter 12 of Berezansky et al. (1996).

To conclude this section, let us give a more geometric interpretation of natural gradient. To this end, let us consider the classical quadratic regression problem :

$$\ell(\boldsymbol{\theta}) := \frac{1}{2S} \sum_{i=1}^{S} \left(u_{|\boldsymbol{\theta}}(x_i) - f(x_i) \right)^2 , \tag{10}$$

with $u_{|\boldsymbol{\theta}}$ a parametric model, for instance a neural-network, (x_i) sampled from some probability measure μ on some domain Ω of $\mathbb{R}^N$. In the limit $S \to \infty$ (population limit), this loss can be reinterpreted as the evaluation at $u_{|\boldsymbol{\theta}}$ of the functional loss:

$$\mathcal{L} : v \in \mathrm{L}^2(\Omega, \mu) \mapsto \frac{1}{2} \left\| v - f \right\|^2_{\mathrm{L}^2(\Omega,\mu)} . \tag{11}$$

Taking the Fréchet derivative, one gets: for all $v, h \in \mathrm{L}^2(\Omega, \mu)$

$$\mathrm{d}\mathcal{L}|_v(h) = \langle v - f \, , \, h \rangle_{\mathrm{L}^2(\Omega,\mu)} ,$$

i.e. the functional gradient of $\mathcal{L}$ is $\nabla \mathcal{L}_{|v} := v - f$. As noted for instance in Verbockhaven et al. (2024), Natural gradient has then to be interpreted from the functional point of view as the projection of $\nabla \mathcal{L}_{|u_{|\boldsymbol{\theta}}}$ onto the tangent space $T_{\boldsymbol{\theta}}\mathcal{M}$ from Equation (4) with respect to the $\mathrm{L}^2(\Omega, \mu)$ metric. However, this functional update must be converted into a parameter space update. Since the parameter space $\mathbb{R}^P$ is somehow identified with $T_{\boldsymbol{\theta}}\mathcal{M}$ via the differential application $\mathrm{d}u_{|\boldsymbol{\theta}}$, it would be sufficient to take the inverse of this application to obtain the parametric update. In general $\mathrm{d}u_{|\boldsymbol{\theta}}$ is not invertible but at least it admits a pseudo-inverse $\mathrm{d}u^{\dagger}_{|\boldsymbol{\theta}}$. Moreover, since $T_{\boldsymbol{\theta}}\mathcal{M} = \operatorname{Im} \mathrm{d}u_{|\boldsymbol{\theta}}$ by definition, $\mathrm{d}u^{\dagger}_{|\boldsymbol{\theta}}$ is defined on all $T_{\boldsymbol{\theta}}\mathcal{M}$. Thus, we have that the natural gradient in the population limits corresponds to the update:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \, \mathrm{d}u^{\dagger}_{|\boldsymbol{\theta}_t} \left(\Pi^{\perp}_{T_{\boldsymbol{\theta}_t}\mathcal{M}} \left(\nabla \mathcal{L}_{|u_{|\boldsymbol{\theta}_t}} \right) \right) . \tag{12}$$

Note that the use of the pseudo-inverse implies that the update in the parameter space happens in the subspace $(\operatorname{Ker} \mathrm{d}u_{|\boldsymbol{\theta}})^{\perp} \subset \mathbb{R}^P$.

3 Empirical Natural Gradient and ANaGRAM

In practice, one cannot reach the population limit and thus Equation (12) is only an asymptotic update. Nevertheless, we can derive a more accurate update, when we can rely only on a finite set of points $(x_i)_{i=1}^S$ that is usually called a batch. Following Jacot et al. (2018), we now that quadratic classical gradient descent update with respect to a batch in the vanishing learning rate limit $\eta \to 0$, rewrites in the functional space as:

$$\frac{\mathrm{d}u_{|\boldsymbol{\theta}_t}}{\mathrm{d}t}(x) = -\sum_{i=1}^{S} NTK_{\boldsymbol{\theta}_t}(x, x_i)(u_{|\boldsymbol{\theta}_t}(x_i) - y_i), \quad NTK_{\boldsymbol{\theta}}(x, y) := \sum_{p=1}^{P} \left(\partial_p u_{|\boldsymbol{\theta}}(x) \right) \left(\partial_p u_{|\boldsymbol{\theta}}(y) \right)^t . \tag{13}$$

Furthermore, Rudner et al. (2019); Bai et al. (2022) show that under natural gradient descent, the **Neural Tangent Kernel** $NTK_{\boldsymbol{\theta}_t}$ should be replaced in Equation (13) by the **Natural NTK**:

$$NNTK_{\boldsymbol{\theta}}(x, y) := \sum_{1 \leqslant p,q \leqslant P} \left(\partial_p u_{|\boldsymbol{\theta}}(x) \right) G^{\dagger}_{\boldsymbol{\theta} pq} (\partial_p u_{|\boldsymbol{\theta}}(y))^t, \quad G_{\boldsymbol{\theta}_t p,q} := \left\langle \partial_p u_{|\boldsymbol{\theta}_t} \, , \, \partial_q u_{|\boldsymbol{\theta}_t} \right\rangle_{\mathcal{H}} . \tag{14}$$

As a consequence, one may see that the update under natural gradient descent with respect to a batch $(x_i)_{i=1}^S$ happens in a subspace of the tangent space, namely the **empirical Tangent Space**:

$$\widehat{T}^{NNTK}_{\boldsymbol{\theta},(x_i)}\mathcal{M} := \operatorname{Span}(NNTK_{\boldsymbol{\theta}}(\cdot, x_i) \, : \, (x_i)_{1 \leqslant i \leqslant S}) \subset T_{\boldsymbol{\theta}}\mathcal{M}. \tag{15}$$

Subsequently, Equation (12) can then be adapted to define the **empirical Natural Gradient update**:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \, \mathrm{d}u^{\dagger}_{|\boldsymbol{\theta}_t} \left(\Pi^{\perp}_{\widehat{T}^{NNTK}_{\boldsymbol{\theta},(x_i)}\mathcal{M}} \left(\nabla \mathcal{L}_{|u_{|\boldsymbol{\theta}_t}} \right) \right) . \tag{16}$$

Note that this update can be understood from the functional perspective as the standard Nyström method (Sun et al., 2015), bridging the gap between our work and the many methods developed in this field. Nevertheless, the $NNTK_{\boldsymbol{\theta}}$ kernel cannot be computed explicitly in our case, since it requires *a priori* inverting the Gram matrix, which adds further challenge. With this in mind, we present a first result, encapsulated in the following theorem, which is one of our main contributions:

Theorem 1 (ANaGRAM). *Let us be for all* $1 \leqslant i \leqslant S$ *and for all* $1 \leqslant p \leqslant P$*:*
$\widehat{\phi}_{\boldsymbol{\theta} i,p} := \partial_p u_{|\boldsymbol{\theta}}(x_i)\,; \quad \widehat{\nabla\mathcal{L}}_{|u_{|\boldsymbol{\theta}}\,i} := \nabla\mathcal{L}_{|u_{|\boldsymbol{\theta}}}(x_i) = u_{|\boldsymbol{\theta}}(x_i) - f(x_i) \quad$ *and* $\quad \widehat{V}_{\boldsymbol{\theta}}\widehat{\Delta}_{\boldsymbol{\theta}}\widehat{U}_{\boldsymbol{\theta}}^t = SVD(\widehat{\phi}_{\boldsymbol{\theta}})$.
Then:
$$du_{|\boldsymbol{\theta}}^{\dagger}\left(\Pi^{\perp}_{\widehat{T}_{\boldsymbol{\theta},(x_i)}\mathcal{M}}\nabla\mathcal{L}_{|u_{|\boldsymbol{\theta}}}\right) = \left(\widehat{\phi}^{\dagger}_{\boldsymbol{\theta}} + I^{\perp}_{\boldsymbol{\theta}}\right)\left(\widehat{\nabla\mathcal{L}}_{|u_{|\boldsymbol{\theta}}} - \widehat{\nabla\mathcal{L}}^{\perp}_{|u_{|\boldsymbol{\theta}}}\right), \tag{17}$$
with:
- $I^{\perp}_{\boldsymbol{\theta}} = \widehat{V}_{\boldsymbol{\theta}}\left(\boldsymbol{I}_P - \Pi_r\right)\widehat{V}^t_{\boldsymbol{\theta}}G^{\dagger}_{\boldsymbol{\theta}}\widehat{V}_{\boldsymbol{\theta}}\,\Pi_r\,\left(\Pi_r\widehat{V}^t_{\boldsymbol{\theta}}G^{\dagger}_{\boldsymbol{\theta}}\widehat{V}_{\boldsymbol{\theta}}\,\Pi_r\right)^{\dagger}\widehat{\Delta}^{\dagger}_{\boldsymbol{\theta}}\widehat{U}^t_{\boldsymbol{\theta}}$; $\Pi_r := \sum_{p=1}^{r}\boldsymbol{e}^{(p)}\boldsymbol{e}^{(p)\,t}$.
- $\widehat{\nabla\mathcal{L}}^{\perp}_{|u_{|\boldsymbol{\theta}}} = \left(\nabla\mathcal{L}(x_i) - \langle NNTK_{\boldsymbol{\theta}}(x_i,\cdot)\,,\,\nabla\mathcal{L}\rangle_{\mathcal{H}}\right)_{1\leqslant i\leqslant S} = \left(\left(\Pi^{\perp}_{T^{\perp}_{\boldsymbol{\theta}}\mathcal{M}}\nabla\mathcal{L}\right)(x_i)\right)_{1\leqslant i\leqslant S}$.

A proof of this theorem, as well as a more comprehensive introduction to empirical natural gradient, encompassing a *détour* through RKHS theory, can be found in Appendix C. Note that $I^{\perp}_{\boldsymbol{\theta}}$ is an interlacing term, intuitively accounting for $\mathrm{Im}\left(G_{\boldsymbol{\theta}}\widehat{\phi}_{\boldsymbol{\theta}}\right)\bigcap\left(\mathrm{Im}\,\widehat{\phi}_{\boldsymbol{\theta}}\right)^{\perp}$, while $\widehat{\nabla\mathcal{L}}^{\perp}_{|u_{|\boldsymbol{\theta}}}$ is an over-evaluation term, intuitively accounting for the evaluation of the orthogonal to tangent space part of the functional gradient. In the following, we show that in some cases those latter terms vanish.

Proposition 1. *There exist* P *points* $(\hat{x}_i)_{1\leqslant i\leqslant P}$ *such that* $\widehat{T}^{NNTK}_{\boldsymbol{\theta},(x_i)}\mathcal{M} = T_{\boldsymbol{\theta}}\mathcal{M}$*. In particular* $I^{\perp}_{\boldsymbol{\theta}} = 0$*.*

Proposition 2. *If* u *rewrites as* $u_{|\boldsymbol{\theta}} = L_{|\boldsymbol{\theta}} \circ C_{|\boldsymbol{\theta}}$*, with* L *linear in* $\boldsymbol{\theta}$*, and* $f = 0$*, then* $\widehat{\nabla\mathcal{L}}^{\perp}_{|u_{|\boldsymbol{\theta}}} = 0$*.*

As a first approximation, we can neglect those two terms, yielding the following vanilla algorithm:

Algorithm 1: vanilla ANaGRAM

Input:
- $u : \mathbb{R}^P \to \mathrm{L}^2(\Omega,\mu)$ `// neural network architecture`
- $\boldsymbol{\theta}_0 \in \mathbb{R}^P$ `// initialization of the neural network`
- $f \in \mathrm{L}^2(\Omega,\mu)$ `// target function of the quadratic regression`
- $(x_i) \in \Omega^S$ `// a batch in` Ω
- $\epsilon > 0$ `// cutoff level to compute the pseudo inverse`

repeat
- $\widehat{\phi}_{\boldsymbol{\theta}_t} \leftarrow \left(\partial_p u_{|\boldsymbol{\theta}_t}(x_i)\right)_{1\leqslant p\leqslant P,\,1\leqslant i\leqslant S}$ `// Computed` *via* `auto-differentiation`
- $W_{\boldsymbol{\theta}_t}, \Delta_{\boldsymbol{\theta}_t}, V^t_{\boldsymbol{\theta}_t} \leftarrow SVD(\widehat{\phi}_{\boldsymbol{\theta}_t})$
- $\Delta_{\boldsymbol{\theta}_t} \leftarrow \left(\Delta_{\boldsymbol{\theta}_t p}\text{ if }\Delta_{\boldsymbol{\theta}_t p} > \epsilon\text{ else }0\right)_{1\leqslant p\leqslant P}$
- $\widehat{\nabla\mathcal{L}} \leftarrow \left(u_{|\boldsymbol{\theta}_t}(x_i) - f(x_i)\right)_{1\leqslant i\leqslant S}$
- $d_{\boldsymbol{\theta}_t} \leftarrow W_{\boldsymbol{\theta}_t}\Delta^{\dagger}_{\boldsymbol{\theta}_t}V^t_{\boldsymbol{\theta}_t}\,\widehat{\nabla\mathcal{L}}$
- $\eta_t \leftarrow \underset{\eta\in\mathbb{R}^+}{\arg\min}\sum_{1\leqslant i\leqslant S}\left(f(x_i) - u_{|\boldsymbol{\theta}_t - \eta d_{\boldsymbol{\theta}_t}}(x_i)\right)^2$ `// Using e.g. line search`
- $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta_t\, d_{\boldsymbol{\theta}_t}$

until *stop criterion met*

Note that algorithm 1 is equivalent to Gauss-Newton algorithm applied to the empirical loss in Equation (10) also considered recently in **?** with a different setting. Nevertheless, our work aims at a more general approach, giving rise to different algorithms depending on the approximations of $I^{\perp}_{\boldsymbol{\theta}}$ and $\widehat{\nabla\mathcal{L}}^{\perp}_{u_{|\boldsymbol{\theta}}}$.One of the pleasant byproducts of the ANaGRAM framework is also that it leads to a straightforward criterion to choose points in the batch, namely:

$$(x_i^*) := \underset{(x_i)\in\Omega^S}{\arg\min}\,\|\Pi^{\perp}_{\mathrm{Span}(NNTK_\theta(x_i,\cdot):1\leqslant i\leqslant S)}\left(\nabla\mathcal{L}\right) - \nabla\mathcal{L}\|_{\mathcal{H}}, \tag{18}$$

which is amenable to various approximations, subject to further investigations. Taking the best advantage of this criterion should eventually allow us to use natural gradient in a stochastic setting while staying close to the convergence rate of the full batch natural gradient as characterized in Xu et al. (2024). We will now show how ANaGRAM can be applied to the PINNs framework.

4 ANaGRAM for PINNs

Generalizing ANaGRAM to PINNs only requires to change the problem perspective.

4.1 PINNs AS A LEAST-SQUARE REGRESSION PROBLEM

The only difference between the losses of Equation (7) and Equation (10) is the use of the differential operator D and the boundary operator B in Equation (7). More precisely, PINNs and classical quadratic regression problems are essentially similar, except that in the case of PINNs we use the compound model $(D, B) \circ u$ instead of u directly, where, using the definitions of Equation (5):

$$(D,B)\circ u : \left\{ \begin{array}{ccccc} \mathbb{R}^P & \to & \mathcal{H} & \to & \mathrm{L}^{\mathbf{2}}\left(\mathbf{\Omega}, \partial\mathbf{\Omega}\right) := \mathrm{L}^2(\Omega \to \mathbb{R}, \mu) \times \mathrm{L}^2(\partial\Omega \to \mathbb{R}, \sigma) \\ \boldsymbol{\theta} & \mapsto & u_{|\boldsymbol{\theta}} & \mapsto & (D[u_{|\boldsymbol{\theta}}], B[u_{|\boldsymbol{\theta}}]) \end{array} \right. . \quad (19)$$

The derivation of vanilla ANaGRAM in PINNs context is then straightforward:

Algorithm 2: vanilla ANaGRAM for PINNs

Input:
- $u : \mathbb{R}^P \to \mathcal{H}$ `// neural network architecture`
- $\boldsymbol{\theta}_0 \in \mathbb{R}^P$ `// initialization of the neural network`
- $D : \mathcal{H} \to \mathrm{L}^2(\Omega \to \mathbb{R}, \mu)$ `// differential operator`
- $B : \mathcal{H} \to \mathrm{L}^2(\partial\Omega \to \mathbb{R}, \sigma)$ `// boundary operator`
- $f \in \mathrm{L}^2(\Omega \to \mathbb{R}, \mu)$ `// source term`
- $g \in \mathrm{L}^2(\partial\Omega \to \mathbb{R}, \sigma)$ `// boundary value`
- $(x_i^D) \in \Omega^{S_D}$ `// a batch in` Ω
- $(x_i^B) \in \Omega^{S_B}$ `// a batch in` $\partial\Omega$
- $\epsilon > 0$ `// cutoff level to compute the pseudo inverse`

repeat

$\widehat{\phi}_{\boldsymbol{\theta}_t} \leftarrow \left(\left(\partial_p D[u_{|\boldsymbol{\theta}_t}](x_i^D)\right)_{i=1}^{S_D}, \quad \left(\partial_p B[u_{|\boldsymbol{\theta}_t}](x_i^B)\right)_{i=1}^{S_B} \right)_{p=1}^{P}$ `// via autodiff`

$W_{\boldsymbol{\theta}_t}, \Delta_{\boldsymbol{\theta}_t}, V_{\boldsymbol{\theta}_t}^t \leftarrow SVD(\widehat{\phi}_{\boldsymbol{\theta}_t})$

$\Delta_{\boldsymbol{\theta}_t} \leftarrow (\Delta_{\boldsymbol{\theta}_t r} \text{ if } \Delta_{\boldsymbol{\theta}_t r} > \epsilon \text{ else } 0)_{1 \leqslant r \leqslant P}$

$\widehat{\nabla \mathcal{L}} \leftarrow \begin{pmatrix} \left(D[u_{|\boldsymbol{\theta}_t}](x_i^D) - f(x_i^D)\right)_{1 \leqslant i \leqslant S_D} \\ \left(B[u_{|\boldsymbol{\theta}_t}](x_i^B) - g(x_i^B)\right)_{1 \leqslant i \leqslant S_B} \end{pmatrix}$

$d_{\boldsymbol{\theta}_t} \leftarrow W_{\boldsymbol{\theta}_t} \Delta_{\boldsymbol{\theta}_t}^{\dagger} V_{\boldsymbol{\theta}_t}^t \widehat{\nabla \mathcal{L}}$

$\eta_t \leftarrow \underset{\eta \in \mathbb{R}^+}{\arg\min} \frac{1}{2S_D} \sum_{1 \leqslant i \leqslant S_D} \left(f(x_i^D) - D[u_{|\boldsymbol{\theta}_t - \eta d_{\boldsymbol{\theta}_t}}](x_i^D) \right)^2 + \frac{1}{2S_B} \sum_{1 \leqslant i \leqslant S_B} \left(g(x_i^B) - B[u_{|\boldsymbol{\theta}_t - \eta d_{\boldsymbol{\theta}_t}}](x_i^B) \right)^2$ `// Using e.g. line search`

$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta_t \, d_{\boldsymbol{\theta}_t}$

until *stop criterion met*

More precisely, this comes from the adaptation of definitions of Section 2.3 as follows:

The image set of the model $\Gamma := \mathrm{Im}\left((D,B)\circ u\right) = \left\{ \left(D[u_{|\boldsymbol{\theta}}], B[u_{|\boldsymbol{\theta}}]\right) : \boldsymbol{\theta} \in \mathbb{R}^P \right\} \subset \mathrm{L}^2\left(\Omega, \partial\Omega\right)$

The model differential $\mathrm{d}\left((D,B)\circ u\right)_{|\boldsymbol{\theta}} : \left\{ \begin{array}{ccc} \mathbb{R}^P & \to & \mathrm{L}^2\left(\Omega, \partial\Omega\right) \\ h & \mapsto & \sum_{p=1}^P h_p \partial_p \left((D,B)\circ u\right)_{|\boldsymbol{\theta}} \end{array} \right.$.

The tangent space $T_{\boldsymbol{\theta}}\Gamma := \mathrm{Im}\,\mathrm{d}\left((D,B)\circ u\right)_{|\boldsymbol{\theta}} = \left\{ \sum_{p=1}^P h_p \left(\partial_p D[u_{|\boldsymbol{\theta}}], \partial_p B[u_{|\boldsymbol{\theta}}]\right) \, : \, h \in \mathbb{R}^P \right\}$

The functional loss $\mathcal{L} : v \in \mathrm{L}^2(\Omega, \partial\Omega) \mapsto \frac{1}{2} \left\| v - (f,g) \right\|_{\mathrm{L}^2(\Omega, \partial\Omega)}^2$

The functional gradient $\nabla \mathcal{L}_{\boldsymbol{\theta}} := \nabla \mathcal{L}_{|((D,B)\circ u)_{|\boldsymbol{\theta}}} = \left(\left((D,B)\circ u\right)_{|\boldsymbol{\theta}} - (f,g) \right) \in \mathrm{L}^2\left(\Omega, \partial\Omega\right)$.

PINN's natural gradient $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \, \mathrm{d}\left((D,B)\circ u\right)_{|\boldsymbol{\theta}_t}^{\dagger} \left(\Pi_{T_{\boldsymbol{\theta}_t}\Gamma}^{\perp} \left(\nabla \mathcal{L}_{\boldsymbol{\theta}_t} \right) \right)$

Appendix C.4 details the slightly more technical definitions of $NNTK$ and empirical Tangent Space. We now present the link between PINN's natural gradient and the operator's Green's function.

4.2 PINNs Natural Gradient is a Green's Function

Knowing the Green's function of a linear operator is one of the most optimal ways of solving the associated PDE, since it then suffices to estimate an integral to approximate a solution (Duffy, 2015). However, this requires prior knowledge of the Green's function, which is not always possible. Here, we show that using the natural gradient for PINNs implicitly uses the operator's Green's function. In Appendix D, we briefly recall the main definitions required to state and prove the following theorem:

Theorem 2. *Let $D : \mathcal{H} \to L^2(\Omega \to \mathbb{R}, \mu)$ be a linear differential operator and $u : \mathbb{R}^P \to \mathcal{H}$ a parametric model. Then for all $\boldsymbol{\theta} \in \mathbb{R}^P$, the generalized Green's function of D on $T_{\boldsymbol{\theta}}\mathcal{M} = \operatorname{Im} du_{|\boldsymbol{\theta}}$ is given by: for all $x, y \in \Omega$*

$$g_{T_{\boldsymbol{\theta}}\mathcal{M}}(x, y) := \sum_{1 \leqslant p,q \leqslant P} \partial_p u_{|\boldsymbol{\theta}}(x)\, G^{\dagger}_{p,q} \partial_q D[u_{|\boldsymbol{\theta}}](y), \tag{20}$$

with: for all $1 \leqslant p, q \leqslant P$

$$G_{pq} := \left\langle \partial_p D[u_{|\boldsymbol{\theta}}]\,,\, \partial_q D[u_{|\boldsymbol{\theta}}] \right\rangle_{L^2(\Omega \to \mathbb{R}, \mu)}. \tag{21}$$

In particular, the natural gradient of PINNs defined at the tend of Section 4.1 can be rewritten:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\, du^{\dagger}_{|\boldsymbol{\theta}_t} \left(x \in \Omega \mapsto \int_{\Omega} g_{T_{\boldsymbol{\theta}_t}\mathcal{M}}(x, y) \nabla \mathcal{L}_{|\boldsymbol{\theta}_t}(y) \mu(dy) \right), \tag{22}$$

A few comments should be made about Equation (22). First, if $\eta = 1$, then the natural gradient can be understood as the least-square's solution of $D[u] = f$ at order 1, *i.e.* in the affine space $u_{|\boldsymbol{\theta}_t} + T_{\boldsymbol{\theta}_t}\mathcal{M}$. However, it does not hold *a priori* that:

- $D[u_{|\boldsymbol{\theta}_t} + T_{\boldsymbol{\theta}_t}\mathcal{M}]$ correctly approximates $f \in \mathrm{L}^2(\Omega \to \mathbb{R}, \mu)$.
- $u_{|\boldsymbol{\theta}_t} + T_{\boldsymbol{\theta}_t}\mathcal{M}$ correctly approximates the image space $\mathcal{M} = \left\{u_{|\boldsymbol{\theta}} \,:\, \boldsymbol{\theta} \in \mathbb{R}^P\right\}$.

Multiplying by a learning rate $\eta \ll 1$ is then essential. In this way, natural gradient can be understood as moving in the direction of the solution of $D[u] = f$ in the affine space $u_{|\boldsymbol{\theta}_t} + T_{\boldsymbol{\theta}_t}\mathcal{M}$, and thus getting closer to the solution, while expecting that the change induced by this update will improve the approximation space $u_{|\boldsymbol{\theta}_{t+1}} + T_{\boldsymbol{\theta}_{t+1}}\mathcal{M}$[3]. On the other hand, when we approach the end of the optimization, *i.e.* when the space $D[u_{|\boldsymbol{\theta}_t} + T_{\boldsymbol{\theta}_t}\mathcal{M}]$ approximates f "well enough", while $\mathrm{d}u_{|\boldsymbol{\theta}_t}$ approximates "well enough" $\mathcal{M}$, then it is in our best interest to solve the equation completely, *i.e.* to take learning rates η close to 1. This is why the use of line search in ANaGRAM (*cf.* line 6 in Algorithm 2) is essential. We should then conclude that the quality of the solution found by the parametric model u **depends only** on:

- How well $\Gamma = \{D[u_{|\boldsymbol{\theta}}] \,:\, \boldsymbol{\theta} \in \mathbb{R}^P\}$ can approximate the source $f \in \mathrm{L}^2(\Omega \to \mathbb{R}, \mu)$.
- The curvature of Γ. More precisely, if its non-linear structure induces convergence to a $D[u_{|\boldsymbol{\theta}}]$ such that $f - D[u_{|\boldsymbol{\theta}}]$ is non-negligible, while being orthogonal to the tangent space $D[T_{\boldsymbol{\theta}_t}\mathcal{M}]$.

If we assume now that D is also nonlinear, then all the above analysis also holds for the linear operators $\mathrm{d}D_{|u_{|\boldsymbol{\theta}_t}}$, the difference being that the operator changes at each step. This means that in the case of non-linear operators, we have to deal with both the non-linearity of D and u, but that does not change the overall dynamic.

Finally, assuming that both D and u are linear (this is for instance the case when we assume u to be a linear combination of basis functions, like in Finite Elements, or Fourier Series). Then "learning" $u_{|\boldsymbol{\theta}}$ with natural gradient (and learning rate 1) corresponds to solve the equation in the least-squares sense with a generalized Green's function.

5 Experiments

We test ANaGRAM on four problems: 2 D Laplace equation ; 1+1 D heat equation ; 5 D Laplace equation ; and 1+1 D Allen-Cahn equation. The first three problems comes from Müller & Zeinhofer (2023), while the last one is proposed in Lu et al. (2021).

[3] To our best knowledge, rigorous proof of this phenomenon has yet to be provided. We can therefore only rely on empirical evidence, which we will detail in Section 5.

For training, we use multilayer perceptrons with varying layer sizes and $\tanh$ activations, along with fixed batches of points: a batch of size S_D to discretize Ω and a batch of size S_B to discretize $\partial\Omega$. The layer size specifications, cutoff factor ϵ, values of S_D and S_B, and discretization procedures are specified separately for each problem. Currently, the cutoff factor is chosen manually and warrants further investigation.

For these various problems, we display as a function of gradient descent steps, the medians over 10 different initializations, of L^2 error E_{L^2} and test loss E_{test}, with shaded area indicating the range between the first and third quartiles. E_{L^2} is defined as: given test points $(x_i)_{i=1}^{S}$, $E_{\mathrm{L}^2}(\boldsymbol{\theta}) := \sqrt{\frac{1}{S_{\mathrm{L}^2}} \sum_{i=1}^{S_{\mathrm{L}^2}} \left|u_{|\boldsymbol{\theta}}(x_i) - u^*(x_i)\right|^2}$, where u^* is a known solution to the PDE and S is taken 10 times bigger than the Ω batch size S_D, while E_{test} is the empirical PINNs loss ℓ of Equation (7), computed with a distinct set of points, of size 5 times bigger than the Ω batch size S_D. We compare ANaGRAM to Energy Natural Gradient descent (E-NGD) (Müller & Zeinhofer, 2023), vanilla gradient descent (GD) with line-search, Adam (Kingma & Ba, 2014) with exponentially decaying learning-rate after 10^{15} steps as in Müller & Zeinhofer (2023) as well as L-BFGS (Liu & Nocedal, 1989). The corresponding CPU times are also provided in tables for reference. The code is made avaible at `https://anonymous.4open.science/r/ANaGRAM-3815/` and further implementation and computation details are provided in Appendix A.1.

2 D Laplace equation : We consider the two dimensional Laplace equation and its solution:

$$\begin{cases} \Delta u = -2\pi^2 \sin(\pi x_1)\sin(\pi x_2) & \text{in } \Omega = [0,1]^2 \\ u = 0 & \text{on } \partial\Omega \end{cases}; \quad u^*(x_1, x_2) = \sin(\pi x_1)\sin(\pi x_2). \tag{23}$$

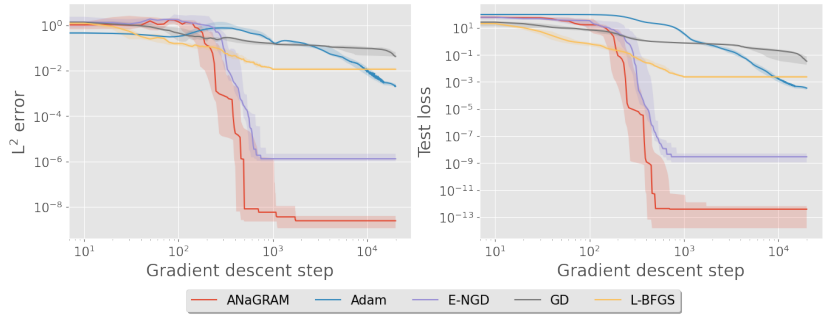

Figure 1: Median absolute L^2 errors and Test losses for the 2 D Laplace equation.

We choose $S_D = 900$ equi-distantly spaced points in the interior of Ω and $S_B = 120$ equally spaced points on the boundary $\partial\Omega$ (30 on each side). ANaGRAM, E-NGD and L-BFGS are applied for 2000 iterations each, while GD and Adam are trained for 20×10^3 iterations. The network consists of a single hidden layer with a width of 32, resulting in a total of $P = 129$ parameters. The cutoff factor is set to $\epsilon = 1 \times 10^{-6}$.

CPU time (s)	Per step	Full
ANaGRAM	7.16e-02	**1.25e+02**
Adam	**1.23e-02**	2.44e+02
E-NGD	1.94e-01	1.88e+02
GD	2.07e-02	4.13e+02
L-BFGS	1.95e-01	1.95e+02

1+1 D Heat equation : We consider the $(1+1)$ dimensional Heat equation and its solution:

$$\begin{cases} \partial_t u - \frac{1}{4}\partial_{xx} u = 0 & \text{in } \Omega = [0,1]^2 \\ u = 0 & \text{on } \partial\Omega_{\text{border}} = [0,1] \times \{0,1\}; \\ u(0,x) = \sin(\pi x) & \text{on } \partial\Omega_0 = \{0\} \times [0,1] \end{cases} \quad u^*(t,x) = \exp\left(-\frac{\pi^2 t}{4}\right)\sin(\pi x). \tag{24}$$

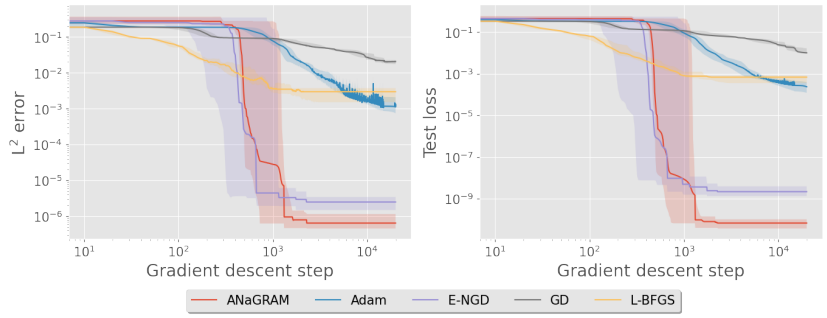

Figure 2: Median absolute L^2 errors and Test losses for the Heat equation.

We choose $S_D = 900$ equi-distantly spaced points in the interior of Ω and $S_B = 90$ equally spaced points on the boundary $\partial\Omega$ (30 on $\partial\Omega_0$ and 30 on each side of $\partial\Omega_{\text{border}}$). ANaGRAM, E-NGD and L-BFGS are applied for 2000 iterations each, while GD and Adam are trained for 20×10^3 iterations. The network consists of a single hidden layer with a width of 64, resulting in a total of $P = 257$ parameters. The cutoff factor is set to $\epsilon = 1 \times 10^{-5}$.

CPU time (s)	Per step	Full
ANaGRAM	1.29e-01	**3.78e+02**
Adam	**2.12e-02**	4.15e+02
E-NGD	1.78e-01	4.04e+02
GD	3.87e-02	7.68e+02
L-BFGS	1.30e-01	3.91e+02

5 D Laplace equation : We consider the five dimensional Laplace equation and its solution:

$$\begin{cases} \Delta u = \pi^2 \sum_{k=1}^{5} \sin(\pi x_k) & \text{in } \Omega = [0,1]^5 \\ u = \sum_{k=1}^{5} \sin(\pi x_k) & \text{on } \partial\Omega \end{cases}; \qquad u^*(x) = \sum_{k=1}^{5} \sin(\pi x_k), \tag{25}$$

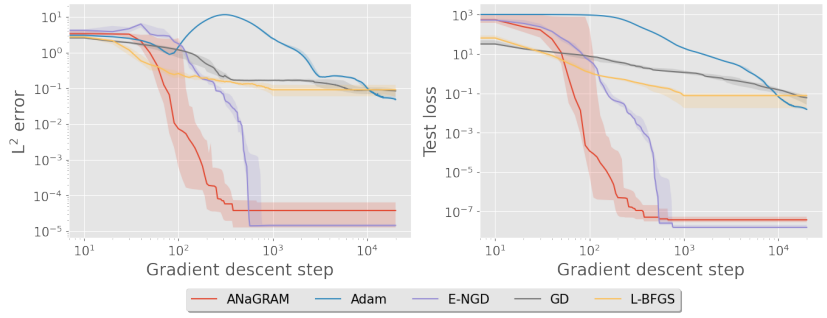

Figure 3: Median absolute L^2 errors and Test losses for the 5 D Laplace equation.

We choose $S_D = 4000$ uniformly drawn points in the interior of Ω and $S_B = 500$ uniformly drawn points on the boundary $\partial\Omega$. ANaGRAM, E-NGD and L-BFGS are applied for 1000 iterations each, while GD and Adam are trained for 20×10^3 iterations. The network consists of a single hidden layer with a width of 64, resulting in a total of $P = 449$ parameters. The cutoff factor is set to $\epsilon = 5.10^{-7} \times \Delta_{\theta \max}$, where $\Delta_{\theta \max}$ is the maximal eigenvalue of $\widehat{\phi}_\theta$ (*cf.* line 1 of algorithm 2).

CPU time (s)	Per step	Full
ANaGRAM	7.18e-01	4.88e+02
Adam	**6.65e-02**	1.29e+03
E-NGD	6.52e+00	4.96e+03
GD	2.69e-01	5.38e+03
L-BFGS	2.96e-01	**2.96e+02**

1+1 D Allen-Cahn equation We consider the $(1+1)$ dimensional Allen-Cahn equation:

$$\begin{cases} \partial_t u - 10^{-3}\, \partial_{xx} u - 5(u - u^3) = 0 & \text{in } \Omega = [0,1] \times [-1,1] \\ u = -1 & \text{on } \partial\Omega_{\text{border}} = [0,1] \times \{-1,1\} \\ u(0,x) = x^2 \cos(\pi x) & \text{on } \partial\Omega_0 = \{0\} \times [-1,1] \end{cases} \tag{26}$$

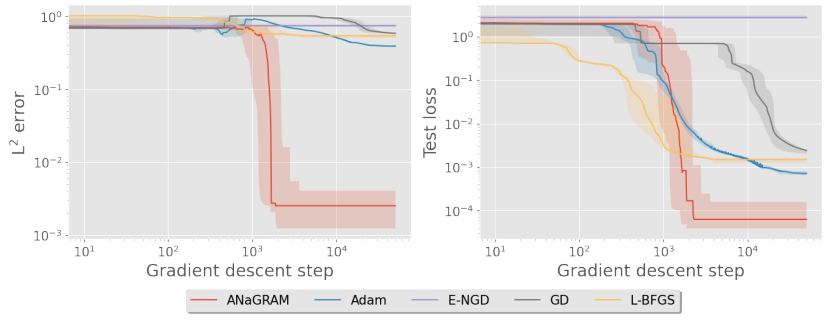

Figure 4: Median absolute L^2 errors and Test losses for the Allen-Cahn equation.

We choose $S_D = 900$ equi-distantly spaced points in the interior of Ω and $S_B = 90$ equally spaced points on the boundary $\partial\Omega$ (30 on $\partial\Omega_0$ and 30 on each side of $\partial\Omega_{\text{border}}$). ANaGRAM and L-BFGS are applied for 4000 iterations each, E-NGD for 1000 iterations, while classical gradient descent (GD) and Adam are trained for 50×10^3 iterations. The network consists of three hidden layers with a width of 20, resulting in a total of $P = 921$ parameters. The cutoff factor is set to $\epsilon = 5.10^{-7} \times \Delta_{\theta\max}$, where $\Delta_{\theta\max}$ is the maximal eigenvalue of $\widehat{\phi}_\theta$ (*cf.* line 1 of algorithm 2).

CPU time (s)	Per step	Full
ANaGRAM	6.01e-01	2.16e+03
Adam	**2.82e-02**	**1.18e+03**
E-NGD	1.30e+00	6.52e+03
GD	8.59e-02	4.28e+03
L-BFGS	4.07e-01	1.60e+03

Results summary : We demonstrated that our approach can achieve comparable accuracy to Müller & Zeinhofer (2023) on linear problems, consistent with the equivalence established in Appendix E, while maintaining a per-step computational cost at most, reasonably higher than that of Adam. Excluding Adam and GD, which consistently get stuck at high error levels, the bottom line is that ANaGRAM consistently outperforms both E-NGD and L-BFGS—often by a significant margin—on at least one or even both criteria: precision and computation time. The cases where the computation times of E-NGD and ANaGRAM are similar occur when small-sized architectures are sufficient for the problem.

6 Conclusion and Perspectives

We introduce empirical Natural Gradient, a new kind of natural gradient that scales linearly with respect to the number of parameters and extend it to PINNs framework through a mathematically principled reformulation. We show that this update implicitly corresponds to the use of the Green's function of the operator. We give empirical evidences that this optimization in its simplest form (vanilla ANaGRAM) already achieves highly accurate solutions, comparable to Müller & Zeinhofer (2023) for linear PDEs at a fraction of the computational cost, and with significant improvements for non-linear equations, for which equivalence of the two algorithms does not hold anymore.

Still, the present formulation of the algorithm has two limitations: one concerns the chosing procedure of the batch points, which is so far limited to simple heuristics; the second is the hyperparameter tunning, more specifically the cutoff factor, which is so far chosen by hand, while it may probably be automatically chosen based on the spectrum of the $\widehat{\phi}_{\boldsymbol{\theta}}$.

Important perspectives include exploring approximations schemes for terms $I^{\perp}_{\boldsymbol{\theta}}$ (*e.g.* using Nyström's methods, *cf.* Sun et al. (2015)) and $\widehat{\nabla\mathcal{L}}^{\perp}_{|u_{|\boldsymbol{\theta}}}$ (*e.g.* using Cohen & Migliorati (2017)), introduced in Theorem 1, the design of an optimal collocation points procedure, coupled with SVD cut-off factor adaptation strategy for ANaGRAM, as well as incorporation of common optimization techniques, such as momentum. From a theoretical point of view, it seems particularly important to us to include data assimilation in this theoretical setting, and understand its regularizing effect, while establishing connections to classical solvers such as FEMs.

REFERENCES

Shun-ichi Amari. *Information Geometry and Its Applications*, volume 194. Springer, 2016.

Shun-Ichi Amari and Scott C. Douglas. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 2, pp. 1213–1216. IEEE, 1998.

Qinxun Bai, Steven Rosenberg, and Wei Xu. A Geometric Understanding of Natural Gradient, February 2022.

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: A survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

Yurij M. Berezansky, Zinovij G. Sheftel, and Georgij F. Us. *Functional Analysis. Vol. II*, volume 86 of *Operator Theory Advances and Applications*. Birkhäuser, 1996.

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

Albert Cohen and Giovanni Migliorati. Optimal weighted least-squares methods. *The SMAI Journal of computational mathematics*, 3:181–203, 2017. ISSN 2426-8399. doi: 10.5802/smai-jcm.24.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, 92(3):88, July 2022. ISSN 1573-7691. doi: 10.1007/s10915-022-01939-z.

Tim De Ryck, Florent Bonnet, Siddhartha Mishra, and Emmanuel de Bézenac. An operator preconditioning perspective on training in physics-informed machine learning, May 2024.

Taco de Wolff, Hugo Carrillo, Luis Martí, and Nayat Sanchez-Pi. Assessing physics informed neural networks in ocean modelling and climate change applications. In *AI: Modeling Oceans and Climate Change Workshop at ICLR 2021*, 2021.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3): 195–201, March 1994. ISSN 1069-8299, 1099-0887. doi: 10.1002/cnm.1640100303.

Dean G. Duffy. *Green's Functions with Applications*. Chapman and Hall/CRC, 2015.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast Approximate Natural Gradient Descent in a Kronecker Factored Eigenbasis. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

Roger Grosse and James Martens. A Kronecker-factored approximate Fisher matrix for convolution layers. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 573–582. PMLR, June 2016.

Tamara G Grossmann, Urszula Julia Komorowska, Jonas Latz, and Carola-Bibiane Schönlieb. Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics*, 89(1):143–174, January 2024. ISSN 0272-4960. doi: 10.1093/imamat/hxae011.

Tom Heskes. On "Natural" Learning and Pruning in Multilayered Perceptrons. *Neural Computation*, 12(4):881–901, April 2000. ISSN 0899-7667. doi: 10.1162/089976600300015637.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, February 2021. ISSN 0021-9991. doi: 10.1016/j.jcp.2020.109951.

Petr Karnakov, Sergey Litvinov, and Petros Koumoutsakos. Solving inverse problems in physics by optimizing a discrete loss is much more faster and accurate without neural networks. 2022.

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

Georgios Kissas, Yibo Yang, Eileen Hwuang, Walter R. Witschey, John A. Detre, and Paris Perdikaris. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358:112623, January 2020. ISSN 0045-7825. doi: 10.1016/j.cma.2019.112623.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26548–26560. Curran Associates, Inc., 2021.

S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. ISSN 0003-4851.

Isaac E. Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539.

Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, August 1989. ISSN 1436-4646. doi: 10.1007/BF01589116.

Songming Liu, Chang Su, Jiachen Yao, Zhongkai Hao, Hang Su, Youjia Wu, and Jun Zhu. Preconditioning for Physics-Informed Neural Networks, February 2024.

Lu Lu, Xuhui Meng, Zhiping Mao, and George E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, January 2021. ISSN 0036-1445, 1095-7200. doi: 10.1137/19M1274067.

James Martens and Roger Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 2408–2417. PMLR, June 2015.

Levi McClenny and Ulisses Braga-Neto. Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism, April 2022.

Johannes Müller and Marius Zeinhofer. Achieving high accuracy with PINNs via energy natural gradient descent. In *International Conference on Machine Learning*, pp. 25471–25485. PMLR, 2023.

Johannes Müller and Marius Zeinhofer. Position: Optimization in SciML Should Employ the Function Space Geometry. In *Forty-First International Conference on Machine Learning*, February 2024.

Yann Ollivier. Riemannian metrics for neural networks I: Feedforward networks. *Information and Inference: A Journal of the IMA*, 4(2):108–153, 2015.

Yann Ollivier. True Asymptotic Natural Gradient Optimization, December 2017.

Vern I. Paulsen and Mrinal Raghupathi. *An Introduction to the Theory of Reproducing Kernel Hilbert Spaces*, volume 152. Cambridge university press, 2016.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019a. ISSN 00219991. doi: 10.1016/j.jcp.2018.10.045.

Maziar Raissi, Hessam Babaee, and Peyman Givi. Deep learning of turbulent scalar mixing. *Physical Review Fluids*, 4(12):124501, December 2019b. doi: 10.1103/PhysRevFluids.4.124501.

Maziar Raissi, Zhicheng Wang, Michael S. Triantafyllou, and George Em Karniadakis. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119–137, February 2019c. ISSN 0022-1120, 1469-7645. doi: 10.1017/jfm.2018.872.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, February 2020. doi: 10.1126/science.aaw4741.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Tim GJ Rudner, Florian Wenzel, Yee Whye Teh, and Yarin Gal. The natural neural tangent kernel: Neural network training dynamics under natural gradient descent. In *4th Workshop on Bayesian Deep Learning (NeurIPS 2019)*, 2019.

Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E. Hurtado, and Ellen Kuhl. Physics-Informed Neural Networks for Cardiac Activation Mapping. *Frontiers in Physics*, 8, February 2020. ISSN 2296-424X. doi: 10.3389/fphy.2020.00042.

Bernhard Schölkopf, Alexander J. Smola, and Francis Bach. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, 2002.

Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, April 2020. ISSN 0045-7825. doi: 10.1016/j.cma.2019.112732.

Shiliang Sun, Jing Zhao, and Jiang Zhu. A review of Nyström methods for large-scale machine learning. *Information Fusion*, 26:36–48, 2015.

Manon Verbockhaven, Sylvain Chevallier, and Guillaume Charpiat. Growing tiny networks: Spotting expressivity bottlenecks and fixing them optimally. *arXiv preprint arXiv:2405.19816*, 2024.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Sifan Wang and Paris Perdikaris. Deep learning of free boundary and Stefan problems. *Journal of Computational Physics*, 428:109914, 2021.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

Xianliang Xu, Ting Du, Wang Kong, Ye Li, and Zhongyi Huang. Convergence Analysis of Natural Gradient Descent for Over-parameterized Physics-Informed Neural Networks, August 2024.

Qi Zeng, Spencer H. Bryngelson, and Florian Tobias Schaefer. Competitive Physics Informed Networks. In *ICLR 2022 Workshop on Gamification and Multiagent Solutions*, April 2022.