Position: Optimization in SciML Should Employ the Function Space Geometry

Johannes Müller [1] **Marius Zeinhofer** [2]

Abstract

Scientific machine learning (SciML) is a relatively new field that aims to solve problems from different fields of natural sciences using machine learning tools. It is well-documented that the optimizers commonly used in other areas of machine learning perform poorly on many SciML problems. We provide an infinite-dimensional view on optimization problems encountered in scientific machine learning and advocate for the paradigm *first optimize, then discretize* for their solution. This amounts to first choosing an appropriate infinite-dimensional algorithm which is then discretized in a second step. To illustrate this point, we show that recently proposed state-of-the-art algorithms for SciML applications can be derived within this framework. As the infinite-dimensional viewpoint is presently underdeveloped in scientific machine learning, we formalize it here and advocate for its use in SciML in the development of efficient optimization algorithms.

1. Introduction

The investigation of optimization problems in function spaces dates back at least to Newton's minimal resistance problem (Newton, 1687), sparking the field today known as the calculus of variations. It took the mathematical community until the beginning of the 20th century for the rigorous concept of function spaces to emerge, initiated by works of Fréchet, Hilbert, Riesz, and Fischer among many others. Today, optimization problems in function spaces are the basis for many applications in engineering and science, including fluid dynamics, solid mechanics, quantum mechanics, and optimal design to name but a few areas of applications. Optimization in function spaces is intimately connected to the solution of *partial differential equations* (PDEs).

Recently, efforts have been made to apply machine learning methods to scientific problems posed in function spaces. Among the goals are the seamless integration of observational data and (partial) physical knowledge in the form of PDEs using physics-informed neural networks (PINNs), the solution of high-dimensional PDEs, such as the many-electron Schrödinger equation, and the design of fast neural surrogate models for many-query applications to replace computationally expensive physics-based models, typically referred to as neural operators. We collect these different approaches under the name Scientific Machine Learning (SciML), see also Subsection 2.1 for more details. Common to all of the methods is that they can be formulated in infinite-dimensional function spaces that are dictated by the underlying PDEs.

Many of the SciML methods lead to notoriously hard optimization problems in practice, different from purely data-driven deep learning applications. For PINNs in particular, it is well documented that first-order methods plateau without reaching high accuracy (Krishnapriyan et al., 2021; Wang et al., 2021; Zeng et al., 2022), which has led many people to believe that

> *there has been a lack of research on optimization tasks for PINNs* (Cuomo et al., 2022).

In particular, for neural network based PDE solvers, the optimization error seems to be dominating the approximation and generalization error, hence rendering these problems very different from many deep learning related tasks. More principled approaches to optimization in SciML have just started to be developed (Müller & Zeinhofer, 2023; Zeng et al., 2022; anonymous, 2024). For variational Monte Carlo methods with neural network ansatz functions the gold standard is K-FAC (Hermann et al., 2022), however, the optimization process is still very resource-consuming (Pfau et al., 2020; Li et al., 2023). In general, there is no clear consensus on best practices for principled choices of optimization algorithms, however, there is a tendency towards second-order optimization algorithms unlike in standard deep learning applications.

[1] Chair of Mathematics of Information Processing, RWTH Aachen University, Aachen, Germany [2] Simula Research Laboratory, Oslo, Norway. Correspondence to: Johannes Müller <mueller@mathc.rwth-aachen.de>, Marius Zeinhofer <mariusz@simula.no>.

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

Function-space inspired methods for SciML Function-space inspired methods provide a principled way to design optimizers that take the specific problem structure into account. Such approaches have been proposed and used in supervised learning and reinforcement learning (Amari, 1998; Kakade, 2001; Martens, 2020) to great success. Here, we discuss their potential in the context of SciML. **We advocate the position that modern optimization algorithms in SciML should employ the function space geometry.** The idea is to choose a suitable infinite-dimensional optimization algorithm based on the properties of the continuous formulation. Only after the choice of an optimization algorithm, the problem is discretized. This corresponds to the well-known *first optimize, then discretize* approach in PDE-constrained optimization, where linear methods rather than networks are used (Hinze et al., 2008). We summarize our main conclusions as follows:

- We argue that most state-of-the-art optimizers in SciML can be obtained from a function-space principle, although they are not commonly motivated this way, see Section 4. This way, we identify an emerging trend in optimization algorithms for SciML.
- Descpite the initial success of function-space inspired methods in SciML, they have not yet been widely adopted in the field. Hence, we advocate for their use and development as we believe this has the potential to help SciML mature into off-the-shelf technology that can be applied at an industrial scale.

The article is structured as follows. In Section 2 we discuss the problems of consideration form the field of Scientific machine learning as well as the importance of the development of more principled and efficient optimizers in this field. In Section 3, we describe the derivation of function-space inspired methods: Formulate the problem at the continuous level. Typically, one views the problem as an infinite-dimensional problem discretized by a neural network ansatz. Choose an optimization algorithm that is well-motivated for the infinite-dimensional problem[1] and discretize the algorithm in the tangent space of the neural network ansatz. In Section 4, we discuss how different state-of-the-art methods in SciML can be derived within this framework and discuss their superiority to first-order and certain second-order methods.

2. Scientific Machine Learning

We describe the problems from the field of scientific machine learning we consider here. Further, we give an overview the current optimization approaches to these problems highlighting that insufficient optimization is widely believed to be the main bottleneck in this field.

2.1. Methods of Scientific Machine Learning

We briefly discuss a sample of influential works in scientific machine learning. The unifying theme is the desire to make neural networks conform to physical laws described in terms of partial differential equations. In the later sections, we refer to these examples and discuss the importance of infinite-dimensional optimization algorithms for them.

Physics-Informed Neural Networks Proposed by (Dissanayake & Phan-Thien, 1994) and popularized by (Raissi et al., 2019), physics-informed neural networks merge observational data with physical prior knowledge in the form of partial differential equations, see also the review articles (Karniadakis et al., 2021; Cuomo et al., 2022). A typical example is the task of finding a function u_θ parametrized as a neural network that matches observational data u_d and satisfies a partial differential equation, say

$$\partial_t u + \mathcal{N}(u) = 0,$$

with given boundary and initial data. Here, $\mathcal{N}$ denotes a partial differential operator. The PINN formulation[2] of the above problem reads

$$\min_{\theta\in\Theta} L(\theta) = \frac{1}{2}\|u_\theta - u_d\|^2_{L^2} + \frac{1}{2}\|\partial_t u_\theta + \mathcal{N}(u_\theta)\|^2_{L^2}. \quad (1)$$

Minimizing L corresponds to solving the PDE and conforming to the observational data u_d. Alternatively, if a variational principle for the PDE at hand is available, this can be used to formulate a loss function. This approach, known as the deep Ritz method, was proposed in (E & Yu, 2018). We remark that physics-informed neural networks and the deep Ritz method can also be regarded as a method to solve PDEs, ignoring the data term in the loss above.

Variational Monte Carlo Methods Recently, neural network based variational Monte Carlo (VMC) methods have shown promising results for the solution of the many-electron Schrödinger equation and presently allow ab initio solutions for system sizes of the order of 100 electrons. We refer to (Hermann et al., 2022) for an overview of this fast-growing field. The idea is to parametrize the wave function ψ by a neural network ψ_θ, where we denote the trainable parameters by θ and to minimize the Raleigh quotient

$$\min_{\theta\in\Theta} \frac{\langle\psi_\theta|\hat{H}|\psi_\theta\rangle}{\langle\psi_\theta|\psi_\theta\rangle}$$

where $\hat{H}$ is the Hamiltonian, which is typically a linear second-order partial differential operator.

[1] The reader may for instance think of Newton's method one-step convergence for the solution of a quadratic problem.

[2] We have omitted terms corresponding to initial and boundary values for brevity of presentation.

Operator Learning Operator learning aims at approximating an operator $\mathcal{G}\colon X \to Y$ between function spaces X and Y by a neural surrogate $\mathcal{G}_\theta \approx \mathcal{G}$, see (Kovachki et al., 2023; Li et al., 2020). A prototypical example is the emulation of the solution map $\mathcal{G}\colon f \mapsto u$ of a given PDE, where $u = \mathcal{G}f$ is the solution of the PDE with data f. Neural operators are typically trained using a regression formulation, although incorporating PDE information is also possible (Li et al., 2021). Combining both, a neural operator loss function has the form

$$L(\theta) = \frac{1}{2}\sum_{i=1}^{N} \|\mathcal{G}_\theta(f_i) - u_i\|_Y^2 + \|\mathcal{D}(\mathcal{G}_\theta(f_i)) - f_i\|_Y^2,$$

where $(f_i, u_i)_{i=1,\dots,N}$ denotes the training data, and the $u_1, \dots, u_N$ is typically generated by a classical solver. The second term in the objective may reduce the need for training data at the expense of a more difficult optimization problem.

2.2. Optimization in Scientific Machine Learning

In their nature, PINNs and related methods are very different from the problems encountered in supervised learning and reinforcement learning. Indeed, the points used in the numerical discretization of the objective function (1) play the role of data points, hence one has access to an unlimited amount of data, which renders the problem as an optimization rather than a statistical one. Further, it is known that the optimization problems are very badly conditioned and consequently, it is commonly believed that optimization is one of the biggest challenges in SciML with training pathologies being well documented (Wang et al., 2021; Krishnapriyan et al., 2021; Cuomo et al., 2022; De Ryck et al., 2023; Liu et al., 2024). To address these difficulties in the optimization, various adaptive weightings of the loss (Wang et al., 2021; 2022b) and sampling strategies (Lu et al., 2021; Nabian et al., 2021; Daw et al., 2022; Zapf et al., 2022; Wang et al., 2022a; Wu et al., 2023; Tang et al., 2023; Jiao et al., 2023) have been suggested, improving naive methods, but failing to produce satisfactory accuracy. Achieving highly accurate PINN solutions has just recently been realized in (Zeng et al., 2022; Müller & Zeinhofer, 2023). As we show in Section 4, all methods that succeed in producing accurate solutions can be interpreted from an infinite-dimensional viewpoint. An illustration of the importance of the infinite-dimensional perspective is provided in Figure 1, which demonstrates that respecting the function space geometry can result in orders of magnitude improvement.

Variational Monte Carlo with neural network ansatz functions routinely relies on the natural gradient method as proposed in (Amari, 1998) and uses the K-FAC approximation of the Fisher matrix, see (Martens & Grosse, 2015; Martens, 2020). In Section 4.1, we illustrate the derivation of the natural gradient method for VMC as an L^2-gradient descent algorithm in function space. Note that for large problem sizes, applications become dramatically resource-demanding, and computation times up to 10^4 GPU hours for a single experiment are reported in (Li et al., 2023). This illustrates the importance of a well-suited training algorithm.

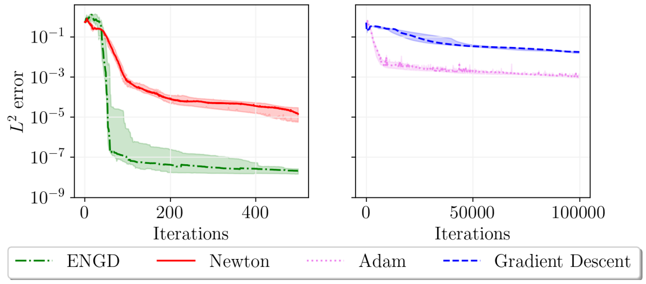

Figure 1. Training curves for a PINN for a 2-dimensional Poisson equation; the first order optimizers (Adam and Gradient Descent) plateau, the second-order optimizers (ENGD and Newton) perform much better, but the function-space inspired optimizer (ENGD) reaches the highest accuracy by several orders of magnitude.

Neural operators for PDE-based applications usually require an offline data generation phase using classical, grid-based solvers to produce training data. Consequently, the neural operator needs to be trained before it is eventually ready for downstream applications. The training is typically carried out with stochastic first-order methods like Adam.

3. Function-Space Inspired Optimization

In this section, we discuss the general principle of the discretization of iterative algorithms in Hilbert spaces, specific examples are provided in Section 4. Assume that we aim to solve a problem in a Hilbert space $\mathcal{H}$, say an optimization or saddle-point problem. To tackle this problem numerically, we consider neural network functions of a given architecture that form a parametric class of functions

$$\mathcal{M} = \{u_\theta : \theta \in \Theta\} \subseteq \mathcal{H},$$

where $\Theta = \mathbb{R}^p$ is the parameter space. It is usually straightforward to design neural networks such that $u_\theta \in \mathcal{H}$ as this typically relies on smoothness properties of the activation function. We will sometimes require the map

$$P\colon \Theta \to \mathcal{M} \subseteq \mathcal{H}, \quad \theta \mapsto u_\theta, \tag{2}$$

which we call the *parametrization* and assume that it is differentiable. We furthermore define the *generalized tangent space* of $\mathcal{M}$ at u_θ as

$$\operatorname{span}\{\partial_{\theta_1} u_\theta, \dots, \partial_{\theta_p} u_\theta\} = \operatorname{ran}(DP(\theta)) \subset \mathcal{H}, \tag{3}$$

where $DP(\theta)$ denotes the derivative of the parametrization. Here, $\partial_{\theta_1} u_\theta, \dots, \partial_{\theta_p} u_\theta$ might be linearly dependent.

The definitions above are not specific for neural networks but are meaningful for general parametric ansatz classes.

The idea of designing optimization algorithms for parametric models that take the function space into account dates back to the seminal works of Amari, who provided a discretization of a function space gradient descent known as natural gradient descent (Amari, 1998) and we refer to the surveys of (Ollivier et al., 2017; Martens, 2020) on function space inspired methods utilizing the Fisher-Rao geometry. Apart from their use in supervised learning, where it is attributed to be more robust to data noise (Amari et al., 2020), natural gradient methods enjoy great popularity in reinforcement learning (Kakade, 2001; Peters et al., 2003; Morimura et al., 2008; Moskovitz et al., 2020). Note that the methods developed in supervised and reinforcement learning can not directly be applied in SciML, where the function spaces often consists of deterministic functions rather then probabilistic models. Here, we describe the general philosophy of function-space inspired methods.

3.1. Discretization of Minimization Methods

Consider an uncontrained optimization problem

$$\min_{u\in\mathcal{H}} E(u), \tag{4}$$

where $E\colon\mathcal{H}\to\mathbb{R}$ is a differentiable function on a Hilbert space $\mathcal{H}$. For this we consider the following scheme

$$u_{k+1} = u_k + \eta_k d_k, \tag{5}$$

with an update direction d_k that is given by

$$d_k = -T_{u_k}^{-1}(DE(u_k)), \tag{6}$$

where $T_{u_k}\colon\mathcal{H}\to\mathcal{H}^*$ is an invertible linear map that is given by the concrete algorithm at hand. For example, we recover gradient descent with $d_k = -\nabla E(u_k)$ and Newton's method with $d_k = -D^2E(u_k)^{-1}(DE(u_k))$. In Section 4, we provide explicit examples and discuss different choices of function space algorithms in more detail.

Corresponding to (4), we define the loss function

$$L\colon\Theta\to\mathbb{R},\quad L(\theta) = E(u_\theta)$$

and aim to design an algorithm in parameter space

$$\theta_{k+1} = \theta_k + \eta_k w_k, \tag{7}$$

such that $u_{\theta_k}\approx u_k$. To understand the dynamics we apply Taylor's theorem to $u_{\theta_{k+1}} = P(\theta_k+\eta_k w_k)$ and obtain

$$u_{\theta_{k+1}} = u_{\theta_k} + \eta_k DP(\theta_k)w_k + O(\eta_k^2\|w_k\|^2).$$

To mimic (5), we would like that $DP(\theta_k)w_k\approx d_k$ for which we choose

$$w_k \in \arg\min_{w\in\mathbb{R}^p}\frac{1}{2}\|DP(\theta_k)w - d_k\|^2_{T_{u_{\theta_k}}}, \tag{8}$$

where $\|w\|^2_{T_{u_{\theta_k}}} = \langle T_{u_{\theta_k}}w, w\rangle$. Computing the normal equations of (8) yields that the update direction is given by

$$w_k = -G(\theta_k)^\dagger\nabla L(\theta_k), \tag{9}$$

where $G(\theta_k)\in\mathbb{R}^{p\times p}$ denotes the Gramian matrix

$$G(\theta_k)_{ij} = \langle T_{u_\theta}\partial_{\theta_i}u_\theta, \partial_{\theta_j}u_\theta\rangle, \tag{10}$$

see Appendix A.1. By $G(\theta_k)^\dagger$ we denote a pseudo-inverse of $G(\theta_k)$, i.e., a matrix satisfying $GG^\dagger G = G$. The discretized algorithm now reads

$$\theta_{k+1} = \theta_k - \eta_k G(\theta_k)^\dagger\nabla L(\theta_k). \tag{11}$$

A typical damping strategy consists of adding an ϵ_k-scaled identity to G. We have derived the discretized algorithm by fitting the update direction (8). This implies that the update direction $DP(\theta_k)w_k$ in function space is the projection of d_k onto $\operatorname{ran}(DP(\theta_k))$, i.e., the tangent space of the model, which is well known for natural gradients in the finite-dimensional setting, see (Amari, 2016; van Oostrum et al., 2022), where we defer the proof to Appendix A.2.

Theorem 1. *Assume we are in the above setting, i.e., consider an algorithm of the form* (5) *that satisfies* (6). *We assume additionally that the $T_{u_{\theta_k}}$ are symmetric and positive definite. Then, for the discretized algorithm* (11) *it holds*

$$u_{\theta_{k+1}} = u_{\theta_k} - \eta_k\Pi_{u_{\theta_k}}[T_{u_{\theta_k}}^{-1}(DE(u_{\theta_k}))] + \epsilon_k, \tag{12}$$

where Π_u denotes the orthogonal projection onto the tangent space with respect to the inner product $\langle T_u\cdot,\cdot\rangle$. The term ϵ_k corresponds to an error vanishing quadratically in the step and step size length

$$\epsilon_k = O(\eta_k^2\|G(\theta_k)^\dagger\nabla L(\theta_k)\|^2).$$

Inspecting the function space dynamics (12) of the discretized algorithm, we see that they agree with the original function space algorithm (5) up to the orthogonal projection onto the tangent space of the model and an error vanishing quadratically with the step size. See also Figure 2, which confirms that the function-space inspired methods (E-NG and GN-NG) lead to function updates that compensate the error of the method much better.

Remark 2 (Connection to Galerkin Discretization). Galerkin schemes in numerical analysis refer to the discretization of linear forms and linear maps using finite dimensional vector spaces (Brenner, 2008). We can interpret the discretization procedure discussed above as a Galerkin scheme in the models tangent space, i.e., using the basis functions $\partial_{\theta_1}u_\theta,\ldots,\partial_{\theta_p}u_\theta$. Discretizing the Fréchet derivative $DE(u_\theta)$ and T_{u_θ} this way yields $\nabla L(\theta)$ and $G(\theta)$. However, unlike in classical methods, the space used in the Galerkin discretization changes at every iteration.

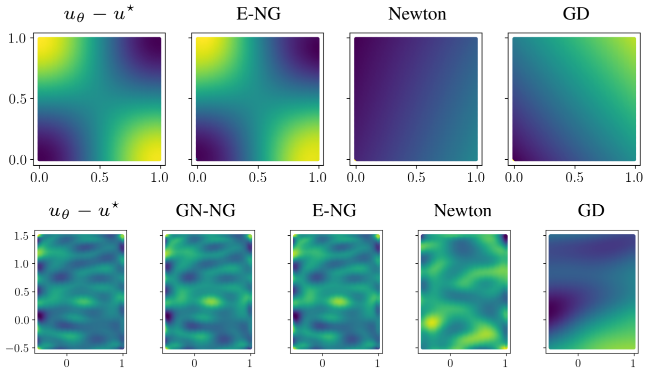

Figure 2. Shown is the heat map over the domain Ω of the function space updates for two different problems, a linear Poisson equation (top) and a steady-state Navier-Stokes system (bottom), for different optimizers, being an energy natural gradient (E-NG), a Gauß-Newton natural gradient (GN-NG), which agrees with E-NG for the Poisson equation, Newton's methods and gradient descent (GD); In addition, the error $u_\theta - u^\star$ is shown, where $u^\star$ is the true solutions; note that the functions-space inspired methods (E-NG and GN-NG) lead to updates that can compensate the error better.

Remark 3 (Extension to Non-Symmetric & Indefinite T_u). If T_u is symmetric and positive definite, it naturally defines a Riemannian metric. In this case, the discretized algorithm in (11) agrees with a natural gradient method. When T_u is non-symmetric or indefinite, (8) is not the right notion to obtain an update direction, as T_u does not correspond to an orthogonal projection in the strict sense. In this case, we still use (9) although it is not the optimality condition of (8). Assuming, for instance, coercivity or suitable inf-sup conditions of T_u guarantees that w_k is a quasi-best approximation of d_k. For the details, we refer to Appendix A.3.

3.2. Discretization of Saddle Point Problems

Another important problem class is given by saddle point problems. Assume we are given a function

$$\mathcal{L}\colon \mathcal{H} \times \mathcal{V} \to \mathbb{R}$$

and we are looking for a saddle point (u^*, v^*), which means

$$\mathcal{L}(u^*, v) \leq \mathcal{L}(u^*, v^*) \leq \mathcal{L}(u, v^*)$$

is satisfied for all $u \in \mathcal{H}$ and $v \in \mathcal{V}$. Such a point solves a minimax problem of the form

$$\min_{u\in\mathcal{H}} \max_{v\in\mathcal{V}} \mathcal{L}(u, v).$$

Saddle point problems arise, for instance, from Lagrangian approaches to constrained minimization problems. For their solution, we again consider algorithms of the form

$$(u_{k+1}, v_{k+1}) = (u_k, v_k) + (d_k^u, d_k^v)$$

where we assume that $\mathcal{L}$ is Fréchet differentiable and the update direction is given by

$$(d_k^u, d_k^v) = T^{-1}_{u_k,v_k}(D_u\mathcal{L}(u_k, v_k), D_v\mathcal{L}(u_k, v_k)).$$

Here, $T_{u,v}\colon \mathcal{H} \times \mathcal{V} \to \mathcal{H}^* \times \mathcal{V}^*$ is a bounded, linear, and invertible map. Algorithms of the form described above include Newton's method for the solution of a critical point of $\mathcal{L}$, gradient descent-ascent, competitive gradient descent (Schäfer & Anandkumar, 2019) as well as natural hidden gradients (Mladenovic et al., 2021).

We discretize the problem with two neural networks u_θ and v_ψ with parameter spaces Θ and Ψ, respectively. This yields

$$L\colon \Theta \times \Psi \to \mathbb{R}, \quad L(\theta, \psi) = \mathcal{L}(u_\theta, v_\psi)$$

and the parameter update

$$(\theta_{k+1}, \psi_{k+1}) = (\theta_k, \psi_k) + \eta_k(w_k^u, w_k^v),$$

the update directions are computed according to

$$\begin{pmatrix} w_k^u \\ w_k^v \end{pmatrix} = G(\theta_k, \psi_k)^\dagger \begin{pmatrix} \nabla_\theta L(\theta_k, \psi_k), \\ \nabla_\psi L(\theta_k, \psi_k) \end{pmatrix}. \tag{13}$$

Here, the Gramian carries a block structure and is obtained by using

$$(\partial_{\theta_1} u_\theta, 0), \ldots, (\partial_{\theta_{p_\Theta}} u_\theta, 0), (0, \partial_{\psi_1} v_\psi), \ldots, (0, \partial_{\psi_{p_\Psi}} v_\psi)$$

as a generating system for the tangent spaces, for details see Appendix B. For projection results along the lines of Theorem 1 see Appendix A.3

4. Specific Examples of Algorithms

We discuss several infinite-dimensional algorithms, illustrating the abstract framework of the previous section. As examples for applications, we discuss (i) neural network ansatz classes for the solution of the many-electron Schrödinger equation using an L^2 gradient descent, (ii) the solution of a nonlinear variational problem using the deep Ritz formulation and Newton's method in function space, (iii) Lagrange-Newton for the solution of a saddle-point formulation of a Poisson equation, and (iv) Gauss-Newton for the PINN formulation of the Navier-Stokes equations. In all cases, the infinite-dimensional algorithms translate to state-of-the-art methods. Some of these methods were previously proposed, but lacking the infinite dimensional viewpoint.

4.1. Hilbert Space Gradient Descent

We show that gradient descent in a Hilbert space corresponds to natural gradient descent in parameter space, with the Riemannian metric induced by the inner product of the Hilbert space. We exemplify this using the variational formulation of Schrödinger's equation. We find that the Hilbert space gradient descent discretized in tangent space corresponds to the state-of-the-art optimization method used in quantum variational Monte Carlo methods, where the wavefunction is parametrized as a neural network, see for instance (Hermann et al., 2020; Pfau et al., 2020).

Problem Formulation We are interested in solving the problem

$$\min_{\psi} E(\psi) = \frac{\langle \psi | \hat{H} | \psi \rangle}{\langle \psi | \psi \rangle} \tag{14}$$

where $\hat{H}$ is the Hamiltonian, typically a linear second-order partial differential operator, and ψ denotes the wave function, see for instance (Toulouse et al., 2016). Using an $L^2(\Omega)$ gradient descent with initial value u_0 for the minimization of E amounts to

$$\psi_{k+1} = \psi_k - \eta_k \mathcal{I}^{-1}(DE(\psi_k)), \quad k = 0, 1, 2, \dots \tag{15}$$

where $\eta_k > 0$ denotes a step size and $\mathcal{I}\colon L^2(\Omega) \to L^2(\Omega)^*$ is the Riesz isometry of $L^2(\Omega)$, given by $\psi \mapsto \langle \psi | \cdot \rangle$. Note that (15) is formal[3] as we can not in general guarantee that $DE(\psi_k) \in L^2(\Omega)^*$.

Neural Network Discretization We choose a neural network ansatz ψ_θ for the wavefunction and define the loss

$$L(\theta) = \frac{\langle \psi_\theta | \hat{H} | \psi_\theta \rangle}{\langle \psi_\theta | \psi_\theta \rangle},$$

where in practice a reformulation is used and the integrals are computed using Monte Carlo quadrature. We use the shorthand notation $\hat{\psi} = \frac{\psi}{\|\psi\|_{L^2}}$. To discretize (15), note that the Riesz isometry $\mathcal{I}$ corresponds to the map T_{u_θ} and leads to the Fisher matrix

$$G(\theta)_{ij} = \mathcal{I}(\partial_{\theta_i} \hat{\psi}_\theta)(\partial_{\theta_j} \hat{\psi}_\theta) = \int \partial_{\theta_i} \hat{\psi}_\theta \partial_{\theta_j} \hat{\psi}_\theta \, \mathrm{d}x.$$

The update in parameter space thus becomes

$$\theta_{k+1} = \theta_k - \eta_k G(\theta_k)^\dagger \nabla L(\theta_k).$$

Correspondence to Natural Gradient Descent The above discussion shows that the $L^2(\Omega)$ gradient descent leads to the well-known natural gradient descent using the Fisher information matrix. This algorithm is state of the art for quantum variational Monte Carlo methods with neural network discretization (Pfau et al., 2020; Li et al., 2023). Note, that in this context the scalability of the method relies on the K-FAC approximation of G. We discuss scalability issues in more detail in Section 4.5.

4.2. Newton's Method

Next, we showcase Newton's method for the solution of a semilinear elliptic equation in variational form, i.e., we use the Deep Ritz method (E & Yu, 2018) for its solution. The example is taken from (Müller & Zeinhofer, 2023), where it is shown that this approach yields highly accurate solutions, for both PINN type objectives of linear PDEs and convex minimization problems like the example discussed here.

Problem Formulation Let $\Omega \subset \mathbb{R}^d$ with $d = 1, 2, 3$ and consider the minimization problem[4]

$$\min_{u \in H^1(\Omega)} E(u) = \int_\Omega \frac{|\nabla u|^2}{2} + \frac{u^4}{4} - fu \, \mathrm{d}x. \tag{16}$$

Newton's method for the minimization of (16) for a given initial value $u_0 \in H^1(\Omega)$ is

$$u_{k+1} = u_k - D^2 E(u_k)^{-1}(DE(u_k)), \quad k = 0, 1, 2, \dots$$

where the Hessian is given by

$$D^2 E(u)(v, w) = (\nabla v, \nabla w)_\Omega + 3(u^2 v, w)_\Omega.$$

Neural Network Discretization We choose a neural network ansatz u_θ with parameter space Θ and define the loss function

$$L(\theta) = E(u_\theta) = \int_\Omega \frac{|\nabla u_\theta|^2}{2} + \frac{u_\theta^4}{4} - f u_\theta \, \mathrm{d}x.$$

For the discretization we note that $T_{u_\theta} = D^2 E(u_\theta)$, which yields

$$\begin{aligned} G(\theta)_{ij} &= D^2 E(u_\theta)(\partial_{\theta_i} u_\theta, \partial_{\theta_i} u_\theta) \\ &= (\nabla \partial_{\theta_i} u_\theta, \nabla \partial_{\theta_j} u_\theta)_\Omega + 3(u_\theta^2 \, \partial_{\theta_i} u_\theta, \partial_{\theta_j} u_\theta)_\Omega. \end{aligned}$$

The algorithm in parameter space becomes

$$\theta_{k+1} = \theta_k - \eta_k G(\theta_k)^\dagger \nabla L(\theta_k), \quad k = 0, 1, 2, \dots \tag{17}$$

for some initial parameters θ_0 and a stepsize $\eta_k > 0$.

Correspondence to Generalized Gauss-Newton To see the connection of the above method to a generalized Gauss-Newton approach *in parameter space*, we compute the Hessian of L. In fact, $D^2 L(\theta)_{ij}$ equals

$$D^2 E(u_\theta)(\partial_{\theta_i} u_\theta, \partial_{\theta_j} u_\theta) + DE(u_\theta)(\partial_{\theta_i} \partial_{\theta_j} u_\theta).$$

Generalized Gauss-Newton methods use the first term in the above expansion in an algorithm of the form (17), which shows that the discretized Newton method and generalized Gauss-Newton coincide, which was recently proposed for the deep Ritz method by (Hao et al., 2023).

[3] For the discretized algorithm this is not a problem.

[4] Dimensions larger than 3 can be considered as well, but require a different functional setting as in four or more dimensions members of $H^1(\Omega)$ are not integrable in fourth power.

Remark 4. The method described above was proposed as a natural gradient method in (Müller & Zeinhofer, 2023) under the name *energy natural gradient descent*. It was shown that it is highly efficient and capable of producing accurate solutions for PINN-type formulations of linear PDEs and deep Ritz formulations of semilinear elliptic PDEs.

Remark 5. Newton in function space for linear PDEs using a PINN formulation yields the standard Gauss-Newton algorithm for l^2 regression in parameter space. This is a consequence of the discussion in Section 4.4. We refer to the Remark 11 for details.

4.3. Lagrange-Newton

The Lagrange Newton algorithm is a solution method for equality-constrained problems (Hinze et al., 2008). It applies Newton's method to find a critical point of the Lagrangian formulation of the constrained problem. We exemplify its application for the solution of a Poisson equation which yields the recently proposed competitive physics-informed neural networks (CPINNs) formulation (Zeng et al., 2022). Moreover, we demonstrate that for linear PDEs the discretization of the Lagrange-Newton algorithm leads to Competitive Gradient Descent (CGD), see (Schäfer & Anandkumar, 2019). The work (Zeng et al., 2022) demonstrates in great detail that the application of CGD to the saddle point formulation of linear PDEs yields state-of-the-art accuracy for PINN type optimization problems.

Problem Formulation Given $\Omega \subset \mathbb{R}^d$, $f \in L^2(\Omega)$, $g \in H^{3/2}(\partial\Omega)$ we consider the constant energy $J(u) = 0$ and the following problem

$$\min_{u \in H^2(\Omega)} J(u) \quad \text{s.t.} \begin{cases} \Delta u + f & = 0 \quad \text{in } \Omega, \\ u - g & = 0 \quad \text{on } \partial\Omega. \end{cases} \tag{18}$$

Note that the only feasible point of the above minimization problem is the solution u^* of Poisson's equation $-\Delta u^* = f$ with boundary data g. The corresponding Lagrangian functional $\mathcal{L}\colon H^2(\Omega) \times L^2(\Omega) \times L^2(\partial\Omega) \to \mathbb{R}$ is given by

$$\mathcal{L}(u, \lambda, \mu) = (\lambda, f + \Delta u)_\Omega + (\mu, u - g)_{\partial\Omega}. \tag{19}$$

The unique saddle point or Nash equilibrium of $\mathcal{L}$ is $(u^*, 0, 0)$ and satisfies

$$\max_{\lambda,\mu} \min_{u} \Big[(\lambda, f + \Delta u)_\Omega + (\mu, u - g)_{\partial\Omega} \Big],$$

see Appendix C.1 for details. The minimax formulation above is the CPINN formulation proposed in (Zeng et al., 2022) applied to Poisson's equation.

Employing the Lagrange-Newton algorithm, we aim to solve

$$D\mathcal{L}(u, \lambda, \mu) = 0$$

via Newton's method. The unique zero of $D\mathcal{L}$ is $(u^*, 0, 0)$ which corresponds to the sought-after saddle point. As $D\mathcal{L}$ is linear, Newton's method converges in one step for any initial value (u_0, λ_0, μ_0), hence it holds that

$$(u^*, 0, 0) = (u_0, \lambda_0, \mu_0) + (d_{u_0}, d_{\lambda_0}, d_{\Delta\mu_0})$$

where the update direction $(d_{u_0}, d_{\lambda_0}, d_{\mu_0})$ is given by

$$-D^2\mathcal{L}(u_0, \lambda_0, \mu_0)^{-1}[D\mathcal{L}(u_0, \lambda_0, \mu_0)].$$

Further, for test functions $(\delta_u, \delta_\lambda, \delta_\mu)$ and $(\bar\delta_u, \bar\delta_\lambda, \bar\delta_\mu)$ the Hessian

$$D^2\mathcal{L}(u_0, \lambda_0, \mu_0)((\delta_u, \delta_\lambda, \delta_\mu), (\bar\delta_u, \bar\delta_\lambda, \bar\delta_\mu))$$

is given via the formula

$$\begin{pmatrix} \delta_u \\ \delta_\lambda \\ \delta_\mu \end{pmatrix}^T \begin{pmatrix} 0 & (\cdot, \Delta \cdot)_\Omega & (\cdot, \cdot)_{\partial\Omega} \\ (\cdot, \Delta \cdot)_\Omega & 0 & 0 \\ (\cdot, \cdot)_{\partial\Omega} & 0 & 0 \end{pmatrix} \begin{pmatrix} \bar\delta_u \\ \bar\delta_\lambda \\ \bar\delta_\mu \end{pmatrix}. \tag{20}$$

Neural Network Discretization We choose neural networks u_θ, λ_ψ, μ_ξ and set as a competitive loss

$$L(\theta, \psi, \xi) = (\lambda_\psi, \Delta u_\theta + f)_\Omega + (\mu_\xi, u_\theta - g)_{\partial\Omega}.$$

For given initial parameters θ_0, ψ_0, ξ_0 we discretize the Lagrange-Newton algorithm above in tangent space via inserting $\partial_{\theta_i} u_\theta, \partial_{\psi_i} \lambda_\psi, \partial_{\xi_i} \mu_\xi$ into the Hessian $D^2\mathcal{L}$ and the derivative $D\mathcal{L}$, see Appendix B for details. This yields a block matrix $G = G(\theta, \psi, \xi)$ of the form

$$G(\theta, \psi, \xi) = \begin{pmatrix} 0 & A & B \\ A^T & 0 & 0 \\ B^T & 0 & 0 \end{pmatrix},$$

where

$$A_{ij} = (\partial_{\psi_j} \lambda_\psi, \Delta \partial_{\theta_i} u_\theta)_\Omega, \quad B_{ij} = (\partial_{\xi_j} \mu_\xi, \partial_{\theta_i} u_\theta).$$

The algorithm becomes

$$(\theta, \psi, \xi)_{k+1} = (\theta, \psi, \xi)_k - \eta_k G_k^\dagger \nabla L(\theta_k, \psi_k, \xi_k), \tag{21}$$

where η_k is a suitably chosen stepsize where damping yields

$$(\theta, \psi, \xi)_{k+1} = (\theta, \psi, \xi)_k - \eta_k (G_k + \epsilon_k \operatorname{Id})^\dagger \nabla L(\theta_k, \psi_k, \xi_k).$$

Correspondence to Competitive GD Adapting the notation, we translate equation (4) of (Schäfer & Anandkumar, 2019) to our setting, which describes an iteration of CGD. For a fixed small $\eta > 0$, the update direction $(d_\theta, d_\psi, d_\xi)$ is given by

$$-\begin{pmatrix} \operatorname{Id} & \eta^{-1} D^2_{\theta,\psi} L & \eta^{-1} D^2_{\theta,\xi} L \\ \eta^{-1} D^2_{\psi\theta} L & \operatorname{Id} & \eta^{-1} D^2_{\psi,\xi} L \\ \eta^{-1} D^2_{\xi,\theta} L & \eta^{-1} D^2_{\xi,\psi} L & \operatorname{Id} \end{pmatrix}^{-1} \begin{pmatrix} \nabla_\theta L \\ \nabla_\psi L \\ \nabla_\xi L \end{pmatrix}$$

and this matrix coincides precisely with $\eta^{-1}G + \mathrm{Id}$, for which we provide the elementary computations in the appendix C.1. In compact notation, this corresponds to

$$\begin{aligned}(d_\theta, d_\psi, d_\xi) &= (\eta^{-1}G + \mathrm{Id})^{-1}DL \\ &= \eta[G + \eta\,\mathrm{Id}]^{-1}DL.\end{aligned}$$

This means it is precisely a discretized Lagrange-Newton method with damping.

Remark 6. The connection between Competitive Gradient Descent on parameter space and Lagrange-Newton in function space can be extended to general *linear* PDEs. However, it does not extend to the nonlinear case, which leads to a non-vanishing first diagonal entry in the matrix (20).

4.4. Gauss-Newton

We discuss a function space version of the Gauss-Newton algorithm for the solution of nonlinear least-squares problems. We exemplify the algorithm using a PINN-type formulation for the Navier-Stokes equations. We show that Gauss-Newton in function space leads to Gauss-Newton in parameter space. This approach was proposed in (anonymous, 2024) for the solution of the Navier-Stokes equations with neural network ansatz. There it is shown that it yields state of the art accuracy for neural network approximations of solutions of the Navier-Stokes equations.

Problem Formulation For $\Omega \subset \mathbb{R}^d$ and forcing f, the steady-state Navier-Stokes equations in velocity-pressure formulation are given by

$$\begin{aligned}-\Delta u + (u\cdot\nabla)u + \nabla p = f \quad &\text{in } \Omega \\ \operatorname{div} u = 0 \quad &\text{in } \Omega,\end{aligned} \tag{22}$$

with suitable boundary conditions. We reformulate (22) in least-squares, i.e., PINN form and assume – for brevity of presentation mainly – that the ansatz space for the velocity consists of divergence-free functions that satisfy the desired boundary conditions.[5] This yields

$$E(u,p) = \frac{1}{2}\|R(u,p)\|^2_{L^2(\Omega)^d}, \tag{23}$$

for a suitably defined nonlinear residual R. Note that the problem is neither quadratic nor convex in (u,p). Here, we choose Gauß-Newton as a function space algorithm as we are facing a least squares problem. The Gauss-Newton algorithm in function space linearizes R at the current iterate and explicitly solves the resulting quadratic problem, see also (Dennis Jr & Schnabel, 1996). Following this strategy, we obtain

$$(u_{k+1}, p_{k+1}) = (u_k, p_k) - T_k^{-1}[DE(u_k,p_k)],$$

where T_k is given by[6]

$$T_k = DR(u_k,p_k)^* DR(u_k,p_k).$$

Neural Network Discretization We choose neural networks u_θ and p_ψ as an ansatz for the velocity and the pressure and denote the PINN loss by

$$L(\theta,\psi) = \frac{1}{2}\| - \Delta u_\theta + (u_\theta\cdot\nabla)u_\theta + \nabla p_\psi - f\|^2_{L^2(\Omega)^d}.$$

Next, we discretize $DR(u_\theta,p_\psi)^*DR(u_\theta,p_\psi)$ in the tangent space of the neural network ansatz following the abstract framework. This yields a block-matrix

$$G = G(\theta,\psi) = \begin{pmatrix} A & B \\ B^T & C\end{pmatrix}.$$

Setting $\xi_i = \Delta\partial_{\theta_i}u_\theta + (u_\theta\cdot\nabla)\partial_{\theta_i}u_\theta + (\partial_{\theta_i}u_\theta\cdot\nabla)u_\theta$ as an abbreviation, we have

$$A_{ij} = (\xi_i,\xi_j)_{L^2(\Omega)^d}, \quad B_{ij} = (\xi_i, \nabla\partial_{\psi_j}p_\psi)_{L^2(\Omega)^d} \tag{24}$$

and

$$C_{ij} = (\nabla\partial_{\psi_i}p_\psi, \nabla\partial_{\psi_j}p_\psi)_{L^2(\Omega)^d}. \tag{25}$$

Using the above computations, the algorithm in parameter space becomes

$$\begin{pmatrix}\theta_{k+1}\\ \psi_{k+1}\end{pmatrix} = \begin{pmatrix}\theta_k \\ \psi_k\end{pmatrix} - \eta_k G(\theta_k,\psi_k)^\dagger \nabla L(\theta_k,\psi_k).$$

Correspondence to Gauss-Newton in Parameter Space We show that Gauss-Newton in parameter space and function space coincide, given a suitable integral discretization. For quadrature points $(x_i)_{i=1,\dots,N}$ in Ω we define the discrete residual $r\colon (\theta,\psi)\to\mathbb{R}^N$ to be

$$\frac{|\Omega|}{\sqrt{N}}\begin{pmatrix}(-\Delta u_\theta + (u_\theta\cdot\nabla)u_\theta + \nabla p_\psi - f)(x_1)\\ \vdots \\ (-\Delta u_\theta + (u_\theta\cdot\nabla)u_\theta + \nabla p_\psi - f)(x_N)\end{pmatrix}.$$

The discretized PINN formulation of (22) reads

$$\min L(\theta,\psi) = \frac{1}{2}\|r(\theta,\psi)\|^2_{l^2}. \tag{26}$$

It is straight-forward to see that applying Gauss-Newton to (26) yields the matrix $G = G(\theta,\psi)$ – if in the discretization of the integrals in the matrices (24) and (25) the same quadrature points as in the definition of r are being used. We provide the details in Appendix C.2.

Remark 7. The correspondence between Gauss-Newton in function space and its counterpart in parameter space holds for general nonlinear least-squares problems, not only for the Navier-Stokes equations, see Appendix C.2.

[5] Modifying neural network ansatz spaces to fulfill these requirements is not uncommon, see (Sukumar & Srivastava, 2022; Richter-Powell et al., 2022)

[6] More precisely this means that $\langle T_k(\delta_u,\delta_p),(\bar\delta_u,\bar\delta_p)\rangle$ equals $(DR(u_k,p_k)(\delta_u,\delta_p), DR(u_k,p_k)(\bar\delta_u,\bar\delta_p))_{L^2(\Omega)^d}$.

4.5. Scalability

Function-space inspired algorithms like those discussed in the previous section yield require the solution of a large system of linear equations of the size of the parameters d_Θ, which is intractable in high parameter dimension d_Θ. Since this is a critical aspect we briefly review matrix-free second-order optimization as discussed in (Schraudolph, 2002), and the K-FAC approach (Martens & Grosse, 2015) that allows to efficiently compute approximate inverses of $G(\theta_k)$.

Matrix-Free Second-Order Optimization In certain situations, the matrix $G(\theta)$ allows for the computation of matrix-vector products $v \mapsto G(\theta)v$ without the need for matrix assembly and storage at a comparable computational cost to the gradient $\nabla L(\theta)$. With matrix-vector products available one can resort to iterative linear solvers that only require matrix-vector products, such as the CG or GMRES methods (Trefethen & Bau, 2022). For example, matrix-vector products are available for PINN-type loss functions of linear PDEs that are optimized using Newton's method in function space. In this setting, the resulting optimization in parameter space is the Gauss-Newton method, see also Remark 5 and Remark 11. More precisely, it holds $G(\theta) = J^T(\theta) \cdot J(\theta)$, where $J(\theta)$ is the Jacobian of a suitably scaled residual r. Then, Jacobian-vector products and vector-Jacobian products can be efficiently computed using automatic differentiation, relying on a combination of forward and backward modes. The matrix-vector product $G(\theta) \cdot v$ for a given vector $v \in \mathbb{R}^p$ can be computed as

$$w = J(\theta)v, \quad G(\theta)v = (w^T \cdot J(\theta))^T.$$

Further details on matrix-free optimization methods can be found in (Schraudolph, 2002) and a successful application is demonstrated in (Zeng et al., 2022).

K-FAC Kronecker-Factorized Approximate Curvature (K-FAC) is used to approximate the Fisher information matrix or Gauß-Newton matrix (Martens & Grosse, 2015; Eschenhagen et al., 2024). The approach approximates the matrix $G(\theta)$ as the Kronecker product of much smaller matrices $G(\theta) \approx A(\theta) \otimes B(\theta)$, which by the properties of the Kronecker product yields $G(\theta)^{-1} \approx A(\theta)^{-1} \otimes B(\theta)^{-1}$. The Kronecker structure stems from the linear layers in the neural network ansatz making the concrete K-FAC approximation dependent on the structure of the ansatz set. K-FAC is the state-of-the-art method for optimization in neural network based variational Monte Carlo methods (Pfau et al., 2020; Li et al., 2023; Scherbela et al., 2023).

5. Conclusion and Outlook

We consider a variety of problems in scientific machine learning for which optimization is arguably the biggest challenge and no principled way or commonly accepted best-practices for optimization exists. We provide a principled way to transfer infinite dimensional optimization algorithms to nonlinear neural network ansatz classes which follows the paradigm of *first optimize, then discretize*. Here, it is the idea to first choose an algorithm in function space that is well aligned with the problem and then to discretize is using neural networks. We show that this approach offers a unified view on many state-of-the-art optimization routines currently employed in SciML, see Table 1. This leads us to the following conclusions:

Function Space	Parameter Space	Name
Gradient Descent	NGD	NGD
Newton	GGN	ENGD
Lagrange-Newton	CGD	CPINNs
Gauss-Newton	GN	GNNGD

Table 1. Translation of optimization algorithms; here NGD stands for natural gradient descent, GN for Gauss-Newton, GGN for generalized Gauss-Newton, ENGD for energy natural gradient descent (Müller & Zeinhofer, 2023), GNNGD for Gauss-Newton natural gradient descent (anonymous, 2024), and CPINNs for competitive PINNs (Schäfer & Anandkumar, 2019).

- Function-space inspired optimization is currently underdeveloped in the field of scientific machine learning.
- The Function-space perspective yields principled way to design problem-specific optimization algorithms in SciML. This has the potential to greatly improve the performance on many current SciML tasks.

Based on our discussion, we propose the following program for the development of efficient function space-algorithms in scientific machine learning:

- Design function-space inspired methods for more SciML problems, in particular this requires understanding what an appropriate function-space algorithm is for a given problem at hand.
- Provide fast implementations of function-space inspired algorithms. Note that the methods developed in the other contexts can usually not be applied as the function-space geometries in SciML often incorporate PDE specific terms.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

JM acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the project number 442047500 through the Collaborative Research Center *Sparsity and Singular Structures* (SFB 1481).

References

Amari, S. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Amari, S. *Information geometry and its applications*, volume 194. Springer, Japan, 2016.

Amari, S.-i., Ba, J., Grosse, R., Li, X., Nitanda, A., Suzuki, T., Wu, D., and Xu, J. When does preconditioning help or hurt generalization? *arXiv preprint arXiv:2006.10732*, 2020.

anonymous. Gauss-newton natural gradient. *under review*, 2024.

Boffi, D., Brezzi, F., Fortin, M., et al. *Mixed finite element methods and applications*, volume 44. Springer, 2013.

Brenner, S. C. *The mathematical theory of finite element methods*. Springer, 2008.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Daw, A., Bu, J., Wang, S., Perdikaris, P., and Karpatne, A. Rethinking the importance of sampling in physics-informed neural networks. *arXiv preprint arXiv:2207.02338*, 2022.

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

Dennis Jr, J. E. and Schnabel, R. B. *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

Dissanayake, M. and Phan-Thien, N. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

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

Eschenhagen, R., Immer, A., Turner, R., Schneider, F., and Hennig, P. Kronecker-factored approximate curvature for modern neural network architectures. *Advances in Neural Information Processing Systems*, 36, 2024.

Grisvard, P. *Elliptic problems in nonsmooth domains*. SIAM, 2011.

Hao, W., Hong, Q., Jin, X., and Wang, Y. Gauss newton method for solving variational problems of pdes with neural network discretizaitons. *arXiv preprint arXiv:2306.08727*, 2023.

Hermann, J., Schätzle, Z., and Noé, F. Deep-neural-network solution of the electronic schrödinger equation. *Nature Chemistry*, 12(10):891–897, 2020.

Hermann, J., Spencer, J., Choo, K., Mezzacapo, A., Foulkes, W., Pfau, D., Carleo, G., and Noé, F. Ab-initio quantum chemistry with neural-network wavefunctions. *arXiv preprint arXiv:2208.12590*, 2022.

Hinze, M., Pinnau, R., Ulbrich, M., and Ulbrich, S. *Optimization with PDE constraints*, volume 23. Springer Science & Business Media, 2008.

Jiao, Y., Li, D., Lu, X., Yang, J. Z., and Yuan, C. Gas: A gaussian mixture distribution-based adaptive sampling method for pinns. *arXiv preprint arXiv:2303.15849*, 2023.

Kakade, S. M. A natural policy gradient. *Advances in Neural Information Processing Systems*, 14, 2001.

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

Kovachki, N. B., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A. M., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.*, 24(89):1–97, 2023.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

Li, R., Ye, H., Jiang, D., Wen, X., Wang, C., Li, Z., Li, X., He, D., Chen, J., Ren, W., et al. Forward laplacian: A new computational framework for neural network-based variational monte carlo. *arXiv preprint arXiv:2307.08214*, 2023.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

Liu, S., Su, C., Yao, J., Hao, Z., Su, H., Wu, Y., and Zhu, J. Preconditioning for physics-informed neural networks. *arXiv preprint arXiv:2402.00531*, 2024.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

Martens, J. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

Martens, J. and Grosse, R. Optimizing neural networks with Kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

Mladenovic, A., Sakos, I., Gidel, G., and Piliouras, G. Generalized natural gradient flows in hidden convex-concave games and GANs. In *International Conference on Learning Representations*, 2021.

Morimura, T., Uchibe, E., Yoshimoto, J., and Doya, K. A new natural policy gradient by stationary distribution metric. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 82–97. Springer, 2008.

Moskovitz, T., Arbel, M., Huszar, F., and Gretton, A. Efficient wasserstein natural gradients for reinforcement learning. *arXiv preprint arXiv:2010.05380*, 2020.

Müller, J. and Zeinhofer, M. Achieving high accuracy with PINNs via energy natural gradient descent. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25471–25485. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/muller23b.html.

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

Newton, I. Philosophiæ naturalis principia mathematica (mathematical principles of natural philosophy). 1687.

Ollivier, Y., Arnold, L., Auger, A., and Hansen, N. Information-geometric optimization algorithms: A unifying picture via invariance principles. *Journal of Machine Learning Research*, 18(18):1–65, 2017.

Peters, J., Vijayakumar, S., and Schaal, S. Reinforcement learning for humanoid robotics. In *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pp. 1–20, 2003.

Pfau, D., Spencer, J. S., Matthews, A. G., and Foulkes, W. M. C. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Physical Review Research*, 2(3):033429, 2020.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Richter-Powell, J., Lipman, Y., and Chen, R. T. Neural conservation laws: A divergence-free perspective. *Advances in Neural Information Processing Systems*, 35: 38075–38088, 2022.

Schäfer, F. and Anandkumar, A. Competitive gradient descent. *Advances in Neural Information Processing Systems*, 32, 2019.

Scherbela, M., Gerard, L., and Grohs, P. Variational monte carlo on a budget–fine-tuning pre-trained neural wavefunctions. *arXiv preprint arXiv:2307.09337*, 2023.

Schraudolph, N. N. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

Sukumar, N. and Srivastava, A. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022.

Tang, K., Wan, X., and Yang, C. Das-pinns: A deep adaptive sampling method for solving high-dimensional partial differential equations. *Journal of Computational Physics*, 476:111868, 2023.

Toulouse, J., Assaraf, R., and Umrigar, C. J. Introduction to the variational and diffusion monte carlo methods. In *Advances in Quantum Chemistry*, volume 73, pp. 285–314. Elsevier, 2016.

Trefethen, L. N. and Bau, D. *Numerical linear algebra*, volume 181. Siam, 2022.

van Oostrum, J., Müller, J., and Ay, N. Invariance properties of the natural gradient in overparametrised systems. *Information Geometry*, pp. 1–17, 2022.

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

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

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

Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

Xu, J. and Zikatanov, L. Some observations on babuska and brezzi theories. *Numerische Mathematik*, 94(1):195–202, 2003.

Zapf, B., Haubner, J., Kuchta, M., Ringstad, G., Eide, P. K., and Mardal, K.-A. Investigating molecular transport in the human brain from mri with physics-informed neural networks. *Scientific Reports*, 12(1):1–12, 2022.

Zeidler, E. *Applied functional analysis: main principles and their applications*, volume 109. Springer Science & Business Media, 2012.

Zeng, Q., Bryngelson, S. H., and Schaefer, F. T. Competitive physics informed networks. In *ICLR 2022 Workshop on Gamification and Multiagent Solutions*, 2022. URL https://openreview.net/forum?id=rMz_scJ6lc.

A. Details Abstract Framework

For now, we will work under the assumption that the map T_u is symmetric and positive semi-definite. We denote the inner product induced by T by

$$[w_1, w_2]_{T_u} \coloneqq \langle T_u w_1, w_2 \rangle$$

and the generalized norm by $\|w\|_{T_u}^2 = [w, w]_{T_u}$.

A.1. Derivation of Normal Equations of (8)

Lemma 8. *We consider a differentiable parametrization $P\colon \Theta \to \mathcal{H}$ and $d_k = T_{u_{\theta_k}}^{-1}(DE(u_{\theta_k})) \in \mathcal{H}$, where we assume $T_{u_{\theta_k}} : \mathcal{H} \to \mathcal{H}^*$ to be symmetric, linear and bounded. Then it holds that*

$$w_k \in \arg\min \frac{1}{2} \|DP(\theta_k)w - d_k\|_{T_{u_{\theta_k}}}^2, \tag{27}$$

if and only if

$$G(\theta_k)w_k = -\nabla L(\theta_k),$$

where

$$G(\theta)_{ij} = \langle T_{u_{\theta_k}} \partial_{\theta_i} u_{\theta_k}, \partial_{\theta_j} u_\theta \rangle.$$

Proof. The right-hand side in (27) is up to the constant term $\frac{1}{2}\|d_k\|_{T_{u_{\theta_k}}}^2$ given by

$$\ell(w) \coloneqq \frac{1}{2} \|DP(\theta_k)w\|_{T_{u_{\theta_k}}}^2 - [DP(\theta_k)w, d_k]_{T_{u_{\theta_k}}}. \tag{28}$$

Using $DP(\theta)w = \sum_i w_i \partial_{\theta_i} u_\theta$ and the definition of $[\cdot,\cdot]_{T_u}$, the first term of (28) takes the form

$$\frac{1}{2} \sum_{i,j} w_i w_j \langle T_{u_{\theta_k}} \partial_{\theta_i} u_{\theta_k}, \partial_{\theta_j} u_{\theta_k} \rangle = \frac{1}{2} w^\top G(\theta_k) w,$$

where $G(\theta)_{ij} = \langle T_{u_{\theta_k}} \partial_{\theta_i} u_{\theta_k}, \partial_{\theta_j} u_\theta \rangle$. The second term in (28) amounts to

$$-\sum_i \langle w_i \partial_{\theta_i} u_{\theta_k}, T_{u_{\theta_k}} d_k \rangle = \sum_i \langle \partial_{\theta_i} u_{\theta_k}, DE(u_{\theta_k}) \rangle = w^\top \nabla L(\theta_k),$$

where we used $d_k = -T_{u_{\theta_k}}^{-1}(DE(u_{\theta_k}))$ and the chain rule $\partial_{\theta_i} L(\theta) = DE(u_\theta) \partial_{\theta_i} u_\theta$. Overall, this yields

$$\ell(w) = \frac{1}{2} w^\top G(\theta_k) w + w^\top \nabla L(\theta_k),$$

where the optimizers of this function are characterized by

$$0 = \nabla \ell(w) = G(\theta_k)w + \nabla L(\theta_k).$$

□

A.2. Proof of Theorem 1

Theorem 1. *Assume we are in the above setting, i.e., consider an algorithm of the form (5) that satisfies (6). We assume additionally that the $T_{u_{\theta_k}}$ are symmetric and positive definite. Then, for the discretized algorithm (11) it holds*

$$u_{\theta_{k+1}} = u_{\theta_k} - \eta_k \Pi_{u_{\theta_k}} [T_{u_{\theta_k}}^{-1}(DE(u_{\theta_k}))] + \epsilon_k, \tag{12}$$

where Π_u denotes the orthogonal projection onto the tangent space with respect to the inner product $\langle T_u \cdot, \cdot \rangle$. The term ϵ_k corresponds to an error vanishing quadratically in the step and step size length

$$\epsilon_k = O(\eta_k^2 \|G(\theta_k)^\dagger \nabla L(\theta_k)\|^2).$$

Proof. This is a direct consequence of (27) as the best approximation is given by the orthogonal projection. □

A.3. Extension to Non-Symmetric and Indefinite T_u

Recall that we consider an iterative algorithm in a Hilbert space $\mathcal{H}$ of the form

$$u_{k+1} = u_k + \eta_k d_k, \quad \text{with } d_k = T_{u_k}^{-1}(DE(u_k)).$$

Here, $T_{u_k} : \mathcal{H} \to \mathcal{H}^*$ is a continuous, linear, and bijective map, $E : \mathcal{H} \to \mathbb{R}$ is a function whose extrema or saddle points we aim to find. The update direction d_k satsifies the equation

$$\langle T_{u_k} d_k, w \rangle = \langle DE(u_k), w \rangle \quad \text{for all } w \in \mathcal{H}.$$

Now we again introduce a neural network discretization of this algorithm and assume that we are at step k with parameters θ_k and the corresponding function u_{θ_k}. The discretization in tangent space and the solution of equation (9) is nothing but replacing the space of test functions by the tangent space of the neural network ansatz at θ_k, i.e., with $w_k = G(\theta_k)^\dagger \nabla L(\theta_k)$ and $d_{\theta_k} = DP(\theta_k) w_k$ we have

$$\langle T_{u_{\theta_k}} d_{\theta_k}, w \rangle = \langle DE(u_{\theta_k}), w \rangle \quad \text{for all } w \in T_{u_{\theta_k}} \mathcal{M}.$$

Here $T_{u_{\theta_k}} \mathcal{M}$ is the tangent space of the neural network ansatz at u_{θ_k}, i.e., $\operatorname{span}\{\partial_{\theta_1} u_{\theta_k}, \dots \partial_{\theta_p} u_{\theta_k}\}$. For general invertible, but possibly non-symmetric and indefinite $T_{u_{\theta_k}}$, we are interested in quasi-best approximation results of the form

$$\|d_k - d_{\theta_k}\|_{\mathcal{H}} \leq C \cdot \inf_{d \in T_{u_{\theta_k}} \mathcal{M}} \|d_k - d\|_{\mathcal{H}}. \tag{29}$$

Such an estimate satisfies the best approximation property of an orthogonal projection up to a constant C[7] and thus guarantees that the function space update directions d_k are closely matched. Results like (29) are well-known in the finite element literature and hold if certain inf-sup conditions are satisfied both on the continuous and the discrete level. We refer to (Xu & Zikatanov, 2003; Boffi et al., 2013) for an introduction. Note that the verification depends on the properties on the infinite-dimensional level *and* the concrete structure of the discrete ansatz spaces which is not the case for symmetric and positive definite operators.

B. Discretization of Saddle Point Problems

We recall our notation for saddle point problems. Given a map

$$\mathcal{L} : \mathcal{H} \times \mathcal{V} \to \mathbb{R}, \quad (u, v) \mapsto \mathcal{L}(u, v)$$

we are looking for a saddle point (u^*, v^*). For their solution we employ an iterative algorithm of the form

$$(u_{k+1}, v_{k+1}) = (u_k, v_k) + (d_{u_k}, d_{v_k}).$$

We assume that $\mathcal{L}$ is Fréchet differentiable and the update direction is given by

$$(d_{u_k}, d_{v_k}) = T_{u_k, v_k}^{-1}(D\mathcal{L}(u, v)).$$

Here,

$$T_{u_k, v_k} : \mathcal{H} \times \mathcal{V} \to \mathcal{H}^* \times \mathcal{V}^*$$

is a bounded, linear, and invertible map. Recall our notation for the competitive loss L

$$L : \Theta \times \Psi \to \mathbb{R}, \quad L(\theta, \psi) = \mathcal{L}(u_\theta, v_\psi),$$

where u_θ and v_ψ are two neural networks with parameter spaces Θ and Ψ, respectively. We then employ

$$\{(\partial_{\theta_i} u_\theta, 0)\}_{i=1,\dots,p_\Theta} \quad \text{and} \quad \{(0, \partial_{\psi_i} v_\psi)\}_{i=1,\dots,p_\Psi}$$

[7] This constant should ideally not depend on critical parameters.

for the discretization in the neural network's tangent space. To discretize the linear map $T = T_{u_\theta, v_\psi}$, note that we can write it in block structure

$$T = \begin{pmatrix} T^1 & T^2 \\ T^3 & T^4 \end{pmatrix}$$

with $T_1 : \mathcal{H} \to \mathcal{H}^*$, $T_2 : \mathcal{V} \to \mathcal{H}^*$, $T_3 : \mathcal{H} \to \mathcal{V}^*$, and $T_4 : \mathcal{V} \to \mathcal{V}^*$. The corresponding matrix $G = G(\theta, \psi)$ inherits this block structure

$$G = \begin{pmatrix} G^1 & G^2 \\ G^3 & G^4 \end{pmatrix}$$

and it holds

$$G^1_{ij} = \langle T_1 \partial_{\theta_i} u_\theta, \partial_{\theta_j} u_\theta \rangle_{\mathcal{H}}, \quad G^2_{ij} = \langle T_2 \partial_{\psi_j} v_\psi, \partial_{\theta_i} u_\theta \rangle_{\mathcal{H}}$$

and

$$G^3_{ij} = \langle T_3 \partial_{\theta_j} u_\theta, \partial_{\psi_i} v_\psi \rangle_{\mathcal{V}}, \quad G^2_{ij} = \langle T_4 \partial_{\psi_i} v_\psi, \partial_{\psi_j} v_\psi \rangle_{\mathcal{V}}.$$

For the derivative of $\mathcal{L}$ note that the chain rule implies

$$(D_u \mathcal{L}(u_\theta, v_\psi)(\partial_{\theta_i} u_\theta), D_v \mathcal{L}(u_\theta, v_\psi)(\partial_{\psi_j} v_\psi)) = (\nabla_\theta L(\theta, \psi)_i, \nabla_\psi L(\theta, \psi)_j),$$

i.e., corresponds to the gradient of the function L. The algorithm in parameter space with the additional introduction of a step size $\eta_k > 0$ reads

$$(\theta_{k+1}, \psi_{k+1}) = (\theta_k, \psi_k) - \eta_k G_k^\dagger \nabla L(\theta_k, \psi_k),$$

A typical damping strategy consists of adding an ϵ-scaled identity to G and reads as

$$(\theta_{k+1}, \psi_{k+1}) = (\theta_k, \psi_k) - \eta_k (G_k + \epsilon_k \operatorname{Id})^\dagger \nabla L(\theta_k, \psi_k).$$

Remark 9 (Interpretation of Update Direction). A similar projection result to the one provided for minimization problems can obtained for discretized saddle point problems *under appropriate assumtions on* T, see Appendix A.3.

C. Extended Examples

This Appendix provides detailed proofs for omitted details in Section 4.

C.1. Details for Section 4.3 on Lagrange-Newton Methods

Recall that we aim to solve

$$\min_{u \in H^2(\Omega)} J(u) = 0 \quad \text{s.t.} \begin{cases} \Delta u + f & = 0 \quad \text{in } \Omega, \\ u - g & = 0 \quad \text{on } \partial\Omega. \end{cases} \tag{30}$$

Given $\Omega \subset \mathbb{R}^d$ open and bounded with a $C^{1,1}$ boundary, $f \in L^2(\Omega)$, and $g \in H^{3/2}(\partial\Omega)$. Note that elliptic regularity theory (Grisvard, 2011) guarantees the existence of a solution $u^* \in H^2(\Omega)$ to this problem.

Equivalence of Saddle Points and Critical Points of the Lagrangian We now considering the Lagrangian of the constrained minimization problem which is given by

$$\mathcal{L}\colon H^2(\Omega) \times L^2(\Omega) \times L^2(\partial\Omega) \to \mathbb{R}, \quad \mathcal{L}(u, \lambda, \mu) = (\lambda, f + \Delta u)_\Omega + (\mu, u - g)_{\partial\Omega}. \tag{31}$$

As $\mathcal{L}$ is a sum of continuous bilinear forms, it is Fréchet differentiable with derivative

$$D\mathcal{L}(u, \lambda, \mu)((\delta_u, \delta_\lambda, \delta_\mu)) = ((\lambda, \Delta \delta_u)_\Omega + (\mu, \delta_u)_{\partial\Omega}, (\delta_\lambda, f + \Delta u)_\Omega, (\delta_\mu, u - g)_{\partial\Omega}). \tag{32}$$

Recall that a saddle point – or in this case equivalently a Nash equilibrium – of $\mathcal{L}$ is a triplet (u^*, λ^*, μ^*) that satisfies

$$\mathcal{L}(u^*, \lambda, \mu) \le \mathcal{L}(u^*, \lambda^*, \mu^*) \le \mathcal{L}(u, \lambda^*, \mu^*), \quad \forall u, \lambda, \mu \in H^2(\Omega) \times L^2(\Omega) \times L^2(\partial\Omega). \tag{33}$$

It is well-known, see for instance (Zeidler, 2012) Theorem 2.F, that a saddle point (u^*, λ^*, μ^*) satisfies the minimax problem

$$\mathcal{L}(u^*, \lambda^*, \mu^*) = \max_{\lambda,\mu} \min_{u} \mathcal{L}(u, \lambda, \mu) = \min_{u} \max_{\lambda,\mu} \mathcal{L}(u, \lambda, \mu) = \min_{u} \max_{\lambda,\mu} \Big[(\lambda, f + \Delta u)_\Omega + (\mu, u - g)_{\partial\Omega}\Big].$$

The minimax formulation above is the CPINN formulation proposed in (Zeng et al., 2022) applied to Poisson's equation.

Lemma 10. *The unique saddle point of $\mathcal{L}$ is $(u^*, 0, 0)$, where $u^* \in H^2(\Omega)$ denotes the solution to (30). Furthermore, the triplet $(u^*, 0, 0)$ is the unique zero of $D\mathcal{L}$.*

Proof. It is easy to see that the triplet $(u^*, 0, 0)$ satisfies the defining inequalities (33) of a saddle point as all expressions evaluate to zero. Given another triplet $(u^{**}, \lambda^{**}, \mu^{**})$ that satisfies 33 we realize that the condition

$$\mathcal{L}(u^{**}, \lambda, \mu) \leq \mathcal{L}(u^{**}, \lambda^{**}, \mu^{**}), \quad \text{for all } \lambda \in L^2(\Omega), \mu \in L^2(\partial\Omega)$$

can only hold with a finite value for $\mathcal{L}(u^{**}, \lambda^{**}, \mu^{**})$ if $u^{**} = u^*$. Moreover

$$\mathcal{L}(u^*, \lambda^{**}, \mu^{**}) \leq \mathcal{L}(u, \lambda^{**}, \mu^{**}), \quad \text{for all } u \in H^2(\Omega)$$

can only hold with a finite value for $\mathcal{L}(u^*, \lambda^{**}, \mu^{**})$ if $\lambda^{**} = \mu^{**} = 0$. Hence $(u^{**}, \lambda^{**}, \mu^{**}) = (u^*, 0, 0)$.

Investigating the derivative of the Lagrangian (32), we see that $(u^*, 0, 0)$ is a critical point. Furthermore, for any other critical point $(u^{**}, \lambda^{**}, \mu^{**})$ the last two equations of (32) imply that $u^{**} = u^*$. The first equation reads

$$(\lambda, \Delta\delta_u)_\Omega + (\mu, \delta_u)_{\partial\Omega} = 0 \quad \text{for all } \delta_u \in H^2(\Omega).$$

We assume first that $\mu \in H^{3/2}(\partial\Omega)$ and consider the auxiliary problem of finding $\delta_u^* \in H^2(\Omega)$ that satisfies

$$\begin{aligned} \Delta\delta_u^* &= \lambda \quad \text{in } \Omega, \\ \delta_u^* &= \mu \quad \text{on } \partial\Omega. \end{aligned}$$

Testing with this δ_u^* yields

$$\|\lambda\|^2_{L^2(\Omega)} + \|\mu\|^2_{L^2(\partial\Omega)} = 0$$

and thus the desired fact $\lambda = \mu = 0$. If μ is merely $L^2(\partial\Omega)$ we use the density of $H^{3/2}(\partial\Omega) \subset L^2(\partial\Omega)$ and replace μ in the auxiliary equation by $\mu_\epsilon \in H^{3/2}(\partial\Omega)$ such that $(\mu, \mu_\epsilon)_{L^2(\partial\Omega)} > 0$. This then implies again $\lambda = 0$ and then also $\mu = 0$. □

Correspondence of Lagrange-Newton and Competitive Gradient Descent We provide the missing computations to verify the correspondence claimed in the main text. Recall that

$$D^2\mathcal{L}(u_0, \lambda_0, \mu_0)((\delta_u, \delta_\lambda, \delta_\mu), (\bar{\delta}_u, \bar{\delta}_\lambda, \bar{\delta}_\mu))$$

is given by

$$\begin{pmatrix} \delta_u & \delta_\lambda & \delta_\mu \end{pmatrix} \begin{pmatrix} 0 & (\cdot, \Delta\cdot)_\Omega & (\cdot,\cdot)_{\partial\Omega} \\ (\cdot, \Delta\cdot)_\Omega & 0 & 0 \\ (\cdot,\cdot)_{\partial\Omega} & 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{\delta}_u \\ \bar{\delta}_\lambda \\ \bar{\delta}_\mu \end{pmatrix}.$$

Discretizing this matrix in the tangent space of the neural network ansatz, i.e., using the functions $\partial_{\theta_i} u_\theta, \partial_{\psi_i} \lambda_\psi, \partial_{\xi_i} \mu_\xi$ yields a block matrix of the form

$$G(\theta, \psi, \xi) = \begin{pmatrix} 0 & A & B \\ A^T & 0 & 0 \\ B^T & 0 & 0 \end{pmatrix}, \tag{34}$$

where

$$A_{ij} = (\partial_{\psi_j} \lambda_\psi, \Delta\partial_{\theta_i} u_\theta)_\Omega, \quad B = (\partial_{\xi_j} \mu_\xi, \partial_{\theta_i} u_\theta).$$

The matrix employed in competitive gradient descent for this problem is

$$\begin{pmatrix} \mathrm{Id} & \eta D^2_{\theta,\psi} L & \eta D^2_{\theta,\xi} L \\ \eta D^2_{\psi\theta} L & \mathrm{Id} & \eta D^2_{\psi,\xi} L \\ \eta D^2_{\xi,\theta} L & \eta D^2_{\xi,\psi} L & \mathrm{Id} \end{pmatrix}.$$

For the reader's convenience we recall the loss function

$$L(\theta,\psi,\xi) = (\lambda_\psi, \Delta u_\theta + f)_\Omega + (\mu_\xi, u_\theta - g)_{\partial\Omega}.$$

We compute the partial derivatives of L

$$\begin{aligned} \partial_{\theta_i} L(\theta,\psi,\xi) &= (\lambda_\psi, \Delta \partial_{\theta_i} u_\theta)_\Omega + (\mu_\xi, \partial_{\theta_i} u_\theta)_{\partial\Omega}, \\ \partial_{\psi_j}\partial_{\theta_i} L(\theta,\psi,\xi) &= (\partial_{\psi_j}\lambda_\psi, \Delta\partial_{\theta_i} u_\theta)_\Omega, \\ \partial_{\xi_j}\partial_{\theta_i} L(\theta,\psi,\xi) &= (\partial_{\xi_j}\mu_\xi, \partial_{\theta_i} u_\theta)_{\partial\Omega}. \end{aligned}$$

This shows that the first row and column of the CGD matrix and the Lagrange-Newton matrix agree. We proceed to compute

$$\begin{aligned} \partial_{\xi_i} L(\theta,\psi,\xi) &= (\partial_{\xi_i}\mu_\xi, u_\theta - g)_{\partial\Omega}, \\ \partial_{\psi_j}\partial_{\xi_i} L(\theta,\psi,\xi) &= 0, \end{aligned}$$

which guarantees the correspondence of the remaining blocks. Note that the derivatives

$$\partial^2_{\theta_i} L(\theta,\psi,\xi), \quad \partial^2_{\psi_i} L(\theta,\psi,\xi), \quad \partial^2_{\theta_i} L(\theta,\psi,\xi), \quad \partial^2_{\xi_i} L(\theta,\psi,\xi)$$

do not vanish unless u_θ, λ_ψ and μ_ξ are linear in θ, ψ and ξ, respectively. This shows once more that applying Newton's method to the discrete problem does not reproduce the infinite-dimensional algorithm.

C.2. Details for Section 4.4 on Gauss-Newton Methods

In this Section we derive the Gauss-Newton method in function spaces for the solution of nonlinear least-squares problems and discuss the connection to the classical Gauss-Newton method in Euclidean space.

Derivation of the Gauss-Newton Method Assume we are given a Fréchet differentiable function $R : \mathcal{H} \to L^2(\Omega)$ defined on a Hilbert space $\mathcal{H}$ and we aim to minimize the energy

$$E : \mathcal{H} \to \mathbb{R}, \quad E(u) = \frac{1}{2}\|R(u)\|^2_{L^2(\Omega)}.$$

Here we discuss the approach for an abstract space $\mathcal{H}$. To obtain the formulas for Navier-Stokes as discussed in Section 4.4 set

$$\mathcal{H} = H^2(\Omega)^d \times H^1(\Omega).$$

Given an iterate $u_k \in \mathcal{H}$, the Gauss-Newton algorithm produces an update $u_{k+1} = u_k + d_k$ by adding the direction d_k which is given via

$$d_k = \underset{v\in\mathcal{H}}{\operatorname{argmin}} \frac{1}{2}\|R(u_k) + DR(u_k)v\|^2_{L^2(\Omega)} \approx \frac{1}{2}\|R(u_k+v)\|^2_{L^2(\Omega)}$$

The optimality condition of the quadratic problem that d_k needs to satisfy is given by

$$DR(u_k)^* R(u_k) + DR(u_k)^* DR(u_k) d_k = 0.$$

This is an equality in the Hilbert space $\mathcal{H}$ and $DR(u_k)^* : L^2(\Omega) \to \mathcal{H}$ denotes the Hilbert space adjoint of the map $DR(u_k)$. Applying the Riesz isomorphism $\mathcal{I} : \mathcal{H} \to \mathcal{H}^*$ to the equation above yields

$$DE(u_k) + \mathcal{I}DR(u_k)^* DR(u_k) d_k = 0$$

as an equality in $\mathcal{H}^*$ and it shows that for any $u \in \mathcal{H}$ the map T_u for Gauss-Newton's method is given by

$$T_u : \mathcal{H} \to \mathcal{H}^*, \quad T_u = \mathcal{I} \circ DR(u)^* \circ DR(u).$$

Thus, for a neural network discretization u_θ the Gramian resulting from T_{u_θ} is given by

$$G(\theta)_{ij} = (DR(u_\theta)[\partial_{\theta_i} u_\theta], DR(u_\theta)[\partial_{\theta_j} u_\theta])_{L^2(\Omega)}.$$

Approximating the integrals in the inner product above by Monte Carlo sampling[8] $x_1, \dots, x_N \in \Omega$ yields

$$G(\theta)_{ij} \approx \frac{|\Omega|}{N} \sum_{k=1}^{N_\Omega} DR(u_\theta)[\partial_{\theta_i} u_\theta](x_k) DR(u_\theta)[\partial_{\theta_j} u_\theta](x_k). \tag{35}$$

Correspondence to Gauss-Newton in Parameter Space Here we prove the correspondence between Gauss-Newton in function space and parameter space. Consider the residual function $r : \Theta \to \mathbb{R}^N$

$$r(\theta) = \sqrt{\frac{|\Omega|}{N}} \begin{pmatrix} R(u_\theta)(x_1) \\ \vdots \\ R(u_\theta)(x_N) \end{pmatrix}.$$

Then we can define a discrete loss function

$$L(\theta) = \frac{1}{2} \|r(\theta)\|_{l^2}^2 \approx \frac{1}{2} \|R(u_\theta)\|_{L^2(\Omega)}^2 = E(u_\theta).$$

Applying the Euclidean version of the Gauss-Newton method to r requires the Jacobian J of r, which is given by

$$J(\theta) = \sqrt{\frac{|\Omega|}{N}} \begin{pmatrix} DR(u_\theta)[\partial_{\theta_1} u_\theta](x_1) & \dots & DR(u_\theta)[\partial_{\theta_p} u_\theta](x_1) \\ & \vdots & \\ DR(u_\theta)[\partial_{\theta_1} u_\theta](x_N) & \dots & DR(u_\theta)[\partial_{\theta_p} u_\theta](x_N) \end{pmatrix}.$$

It is clear that $J^T(\theta)J(\theta)$ is the same as the approximation of the Gramian in equation (35), given that the same quadrature points are used.

Remark 11 (Correspondence of Newton and Gauss-Newton for linear PDEs). In the case of a linear PDE, i.e., when R is an affine linear map the previous computations show that Newton's method in function space, as described in Section 4.2, also leads to Gauss-Newton's method in parameter space. In this case, $DR(u_k)$ is independent of u_k and agrees with the linear part of R.

[8] Any other quadrature is equally possible.