NONLINEAR DISCRETIZATIONS AND NEWTON'S METHOD: CHARACTERIZING STATIONARY POINTS

Conor Rowan
Smead Aerospace Engineering Sciences
University of Colorado Boulder
Boulder, CO 80309
conor.rowan@colorado.edu

ABSTRACT

Second-order methods are emerging as promising alternatives to standard first-order optimizers such as gradient descent and ADAM for training neural networks. Though the advantages of including curvature information in computing optimization steps have been celebrated in the scientific machine learning literature, the only second-order methods which have been studied are quasi-Newton, meaning that the Hessian matrix of the objective is approximated. Though one would expect only to gain from using the true Hessian in place of its approximation, we show that neural network training fails spectacularly when relying on exact curvature information. This failure mode provides insight both into the geometry of nonlinear discretizations as well as the distribution of stationary points in the loss landscape, leading us to question the conventional wisdom that the loss landscape is replete with local minima.

Keywords Non-convex optimization · Machine learning · Physics-informed machine learning · Second-order optimization · Nonlinear discretizations

1 Introduction

First-order optimization methods have historically dominated approaches to train neural networks. These methods find a minimum of the objective function by iteratively updating the neural network parameters in the direction which most rapidly decreases this objective. In the case of ADAM—a widely-used first-order algorithm which includes "momentum"—the update to the parameters is a weighted combination of the current and past steepest descent directions [9]. While ADAM has been successful on an impressive array of problems in scientific machine learning (SciML)

[17, 8, 6, 12, 11], recent years have seen growing interest in using second-order optimization methods for physics-informed machine learning problems. Second-order optimization strategies use a local quadratic approximation of the objective function in order to choose the step direction and step size. An advantage of a second-order method is that, because it contains stationary points, the local quadratic approximation of the objective naturally suggests a step size, unlike the unbounded linear approximation of steepest descent methods [20]. Another advantage is that the step direction avoids oscillations and slow convergence in ill-conditioned regions of the loss landscape, a well-known shortcoming of first-order methods [15]. Because of these advantages, second-order optimization methods are already standard in many areas of science and engineering. Accordingly, many authors have begun to investigate these strategies for optimization problems arising from machine learning. In this work, our attention will be limited to regression problems inspired by problems from the SciML literature. In particular, we have in mind physics-informed neural networks (PINNs), for which the target function is defined in space and/or time and the regression problem involves spatial and/or temporal derivatives of the neural network regressor. The basic PINNs approach was first introduced in [23, 17], and has since been explored extensively in many interesting scientific and engineering application areas [13, 7, 2, 4].

Before proceeding, we distinguish between exact Newton and quasi-Newton second-order optimization. An exact Newton method will use the Hessian matrix—in other words, the matrix of second-derivatives of the objective function—in order to build a local quadratic approximation of the objective. The Hessian matrix is either computed analytically and supplied to the optimizer or computed with finite differencing, though the cost of this is typically prohibitive. In contrast, a quasi-Newton method constructs a running approximation of the Hessian with observations of the loss and its gradient obtained over the optimization history. At each optimization step, the previous estimate of the Hessian is updated based on new observations of the loss and its gradient. A common update strategy is provided by the method of Broyden-Fletcher-Goldfarb-Shanno (BFGS), which finds a minimum Frobenius norm update to the previous Hessian subject to the constraint that the update is consistent with a Taylor approximation (secant condition) [1]. A popular modification of this approach is the limited-memory BFGS (L-BFGS) algorithm, which avoids storing a dense approximation of the inverse Hessian and is thus more efficient for high-dimensional problems.

To the best of the author's knowledge, all second-order methods explored in the SciML literature have been quasi-Newton. In [26], the authors use PINNs to optimize an airfoil geometry while simultaneously learning the flow distribution using L-BFGS. Because of the ability of second-order quasi-Newton methods to converge rapidly even in ill-conditioned loss landscapes, a novel hybrid of L-BFGS and ADAM is proposed in [18], demonstrating improved performance for physics-informed objective functions. One study compares optimizers in the BFGS family on challenging PINNs problems, demonstrating that self-scaled variants of BFGS and Broyden optimizers lead to marked performance improvements [10]. Another work shows that quasi-Newton methods naturally resolve conflicts between gradients of different terms in the loss function, which often lead to poor convergence for first-order methods [29]. The authors in [27] report that minor modifications to the loss function and standard BFGS algorithm allow PINNs solutions

to outperform incumbents such as finite difference methods. Finally, in [3], a novel quasi-Newton preconditioning strategy is developed and shown to outperform state-of-the-art methods such as L-BFGS.

Though quasi-Newton methods for training neural networks have seen a surge of interest in recent years, exact Newton methods have remained unexplored. While analytically forming, storing, and inverting the Hessian is prohibitively expensive for many large-scale problems of interest, we show in this work that exact Newton methods exhibit surprising behaviors which provide insight into the geometry of neural network discretizations and the nature of stationary points in the loss landscape. In particular, our contributions are as follows:

1. We discuss regression on manifolds, showing that stationary points can be interpreted geometrically;
2. We conceptualize neural networks as a defining a particular approximation manifold, namely one in which a basis and coefficients are fit simultaneously;
3. We identify a stationary point of the regression objective which is specific to neural network discretizations and represents a trivial zero solution to the regression problem;
4. We show through numerical experimentation that exact Newton methods reliably find these trivial solutions, even for simple one-dimensional problems;
5. We discuss the ways in which quasi-Newton methods differ from exact Newton methods and how these differences lead to better performance.

The rest of this work is organized as follows. In Section 2, we discuss nonlinear discretizations and regression problems defined on manifolds. We show that stationary points of the regression objective need not be minima, and that the condition for stationarity can be interpreted geometrically. In Section 3, we turn to regression with neural networks, showing that multilayer perceptron neural networks can be viewed as simultaneously fitting and scaling basis functions. Such structure allows for stationarity of the regression objective to be satisfied by a trivial solution, but one which imposes particular structure on the learned basis functions. We show with numerical examples that neural networks trained with exact Newton methods reliably converge to this trivial solution, which we observe to be a saddle point rather than a local minimum. In Section 4, we explore two boundary value problems with PINNs, showing that trivial solutions are also possible in the case of physics-informed training. Though the conditions under which such a solution is possible seem as or more complex than the original regression problem, we again show that exact Newton methods favor these trivial saddle solutions. Finally, in Section 5 we conclude with light philosophical reflections on high-dimensional loss landscapes and a discussion of how quasi-Newton methods avoid saddle points.

2 Nonlinear discretizations

Consider a discrete regression problem in which a target vector $\mathbf{v}$ is to be approximated by a parameterized vector $\mathbf{N}(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ are the parameters to be determined. The regression problem is solved with a quadratic error objective and its stationarity condition:

$$\mathcal{L}(\boldsymbol{\theta}) = \|\mathbf{N}(\boldsymbol{\theta}) - \mathbf{v}\|^2, \quad \frac{\partial \mathcal{L}}{\partial \theta_k} = (\mathbf{N}(\boldsymbol{\theta}) - \mathbf{v}) \cdot \frac{\partial \mathbf{N}}{\partial \theta_k} = 0. \tag{1}$$

Eq. (1) shows that stationarity of the quadratic objective enforces that the the error vector $\mathbf{e}(\boldsymbol{\theta}) = \mathbf{N}(\boldsymbol{\theta}) - \mathbf{v}$ is orthogonal to the tangent space of the approximation given by span$\{\frac{\partial \mathbf{N}}{\partial \theta_1}, \frac{\partial \mathbf{N}}{\partial \theta_2}, \dots\}$. When the approximation is linear, meaning that the parameters scale fixed basis vectors $\{\mathbf{h}_j\}_{j=1}^{|\boldsymbol{\theta}|}$, the condition for stationarity reads

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = (\sum_{j=1}^{|\boldsymbol{\theta}|} \theta_j \mathbf{h}_j - \mathbf{v}) \cdot \mathbf{h}_k = \mathbf{e}(\boldsymbol{\theta}) \cdot \mathbf{h}_k = 0. \tag{2}$$

This is known as Galerkin optimality, which states that the error vector has zero projection in the approximation space defined by the basis. If the basis functions are linearly independent, the stiffness matrix $K_{jk} = \mathbf{h}_j \cdot \mathbf{h}_k$ is both full-rank and positive definite. These conditions ensure that the optimization problem 1) has a unique solution and 2) that the solution is a minimum. In other words, when the basis is fixed, the stationary point is as good of an approximation as the basis allows. No such guarantees exist for nonlinear discretizations, which is any parameterization $\mathbf{N}(\boldsymbol{\theta})$ other than that of a fixed basis with variable coefficients. We view these nonlinear discretizations as defining approximation spaces which are manifolds, or surfaces embedded in a higher-dimensional space than that of their parameterization. To see this, consider the following regression problem:

$$\mathbf{N}(\theta) = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \quad \theta \in [0, 2\pi).$$

Galerkin optimality requires that the error vector is orthogonal to the tangent of the approximation, which reads

$$\frac{\partial \mathcal{L}}{\partial \theta} = \begin{bmatrix} \cos(\theta) - 2 \\ \sin(\theta) - 2 \end{bmatrix} \cdot \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} = 2(\sin(\theta) - \cos(\theta)) = 0.$$

This equation can be satisfied for $\theta = \pi/4, 5\pi/4$. Evidently, unlike the linear discretization, the uniqueness of the solution is no longer guaranteed. Furthermore, computing the second derivative of the loss as $\partial^2 \mathcal{L}/\partial \theta^2 = 2(\cos(\theta) + \sin(\theta))$, we observe that

$$\left.\frac{\partial^2 \mathcal{L}}{\partial \theta^2}\right|_{\pi/4} = 2\sqrt{2} > 0, \quad \left.\frac{\partial^2 \mathcal{L}}{\partial \theta^2}\right|_{5\pi/4} = -2\sqrt{2} < 0,$$

which indicates that $\pi/4$ corresponds to a minimum and to $5\pi/4$ a maximum. Thus, not only does the nonlinear discretization have multiple solutions, the error vector can be orthogonal to the tangent(s) of the approximation when the objective is maximized. Finding a stationary point is no longer a trustworthy guide to good performance on regression problems when the discretization is nonlinear. The satisfaction of the orthogonality condition at both the minimum and

maximum error is shown in Figure 1. We remark that the unit circle approximation space is a manifold in the sense that it is embedded in $\mathbb{R}^2$ but is parameterized by a single coordinate θ. In contrast, linear discretizations always define hyperplane approximation spaces, where the dimension of hyperplane is equivalent to the number of parameters.

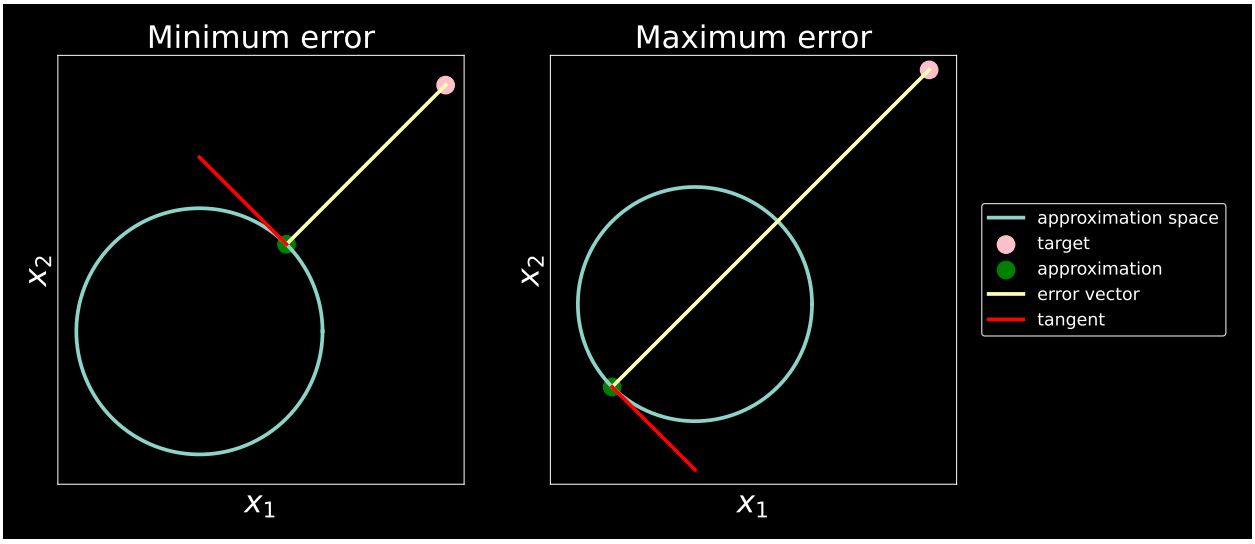

Figure 1: The error vector is orthogonal to the tangent of the approximation space both when the error is minimized and when it is maximized.

When the discretization has only one parameter, the solution to the regression problem is necessarily either a minimum or a maximum. With two or more parameters, it is possible to find stationary points of the loss that are saddle points. A saddle point is a minimum with respect to some parameters and a maximum with respect to others. In order to visualize the geometry of saddle points with nonlinear discretizations, we now define our approximation space as the surface of an ellipsoidal torus. The torus is parameterized by

$$\mathbf{N}(\boldsymbol{\theta}) = \begin{bmatrix} (R + r\cos(\theta_2)\cos(\theta_1) \\ (R + r\cos(\theta_2))e\sin(\theta_1) \\ r\sin(\theta_2) \end{bmatrix}, \quad \theta_1, \theta_2 \in [0, 2\pi),$$

where $\boldsymbol{\theta} = [\theta_1, \theta_2]$ are the parameters, R is the radius of the axis of the torus, r is the thickness of the torus, and e is the eccentricity defining the ratio of the major to the minor axis in the elliptical footprint of the torus. See Figure 2 to visualize the geometry of the approximation space. We look for points on the ellipse to approximate the origin, i.e. $\mathbf{v} = [0, 0, 0]^T$. The regression problem is

$$\mathcal{L}(\boldsymbol{\theta}) = \|\mathbf{N}(\boldsymbol{\theta})\|^2, \quad \frac{\partial \mathcal{L}}{\partial \theta_k} = \mathbf{N}(\boldsymbol{\theta}) \cdot \frac{\partial \mathbf{N}}{\partial \theta_k} = 0. \tag{3}$$

This stationarity condition states that the position vector is orthogonal to the two tangent vectors of the surface of the torus. To solve this nonlinear system of equations, we use Newton's method:

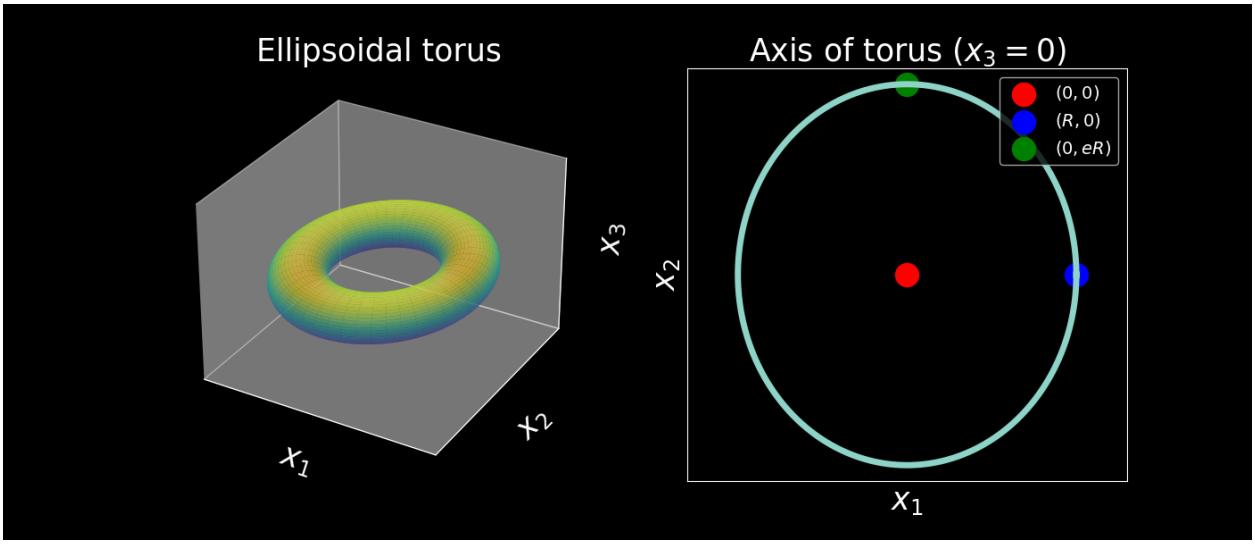

Figure 2: The nonlinear discretization is constructed with two parameters that traverse the surface of an ellipsoidal torus. The approximation space is visualized in 3D (left) and in cross-section (right).

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \left(\frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}}\right)^{-1}\Bigg|_{\boldsymbol{\theta}_k} \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}\right)\Bigg|_{\boldsymbol{\theta}_k}, \tag{4}$$

where we define the Hessian matrix as $\mathbf{J} := \partial^2 \mathcal{L} / \partial \boldsymbol{\theta} \partial \boldsymbol{\theta}$ [20]. The Newton iterations continue until $\|\partial \mathcal{L} / \partial \boldsymbol{\theta}\| \approx 0$, meaning that the nonlinear system of equations for stationarity has been approximately solved. All parameter gradients are computed with automatic differentiation in PyTorch. The initial guess of each parameter is independent and uniformly distributed in $[0, 2\pi)$. The geometric properties of the torus are given by $R = 1$, $r = 0.35$, and $e = 1.2$. See Figure 3 for the results of the Newton solution for three different initializations of the parameters. In each of the three runs, we obtain a different point on the torus as satisfying stationarity. As before, it is clear geometrically that the error vector is orthogonal to the tangent space of the approximation, yet only one of these solutions minimizes the error. To understand the nature of the stationary points, we turn to the Hessian matrix $\mathbf{J}$, whose eigenvalues dictate whether the solution is a minimum, maximum, or saddle. Because the Hessians at the three solutions are diagonal, this allows us to read off the eigenvalues directly. The three Hessian matrices are:

$$\mathbf{J}_1 = \begin{bmatrix} 3.7 & 0 \\ 0 & 0.7 \end{bmatrix}, \quad \mathbf{J}_2 = \begin{bmatrix} 1.6 & 0 \\ 0 & -0.7 \end{bmatrix}, \quad \mathbf{J}_3 = \begin{bmatrix} -1.6 & 0 \\ 0 & -1.1 \end{bmatrix}.$$

As seen by the sign of the diagonal entries, Solution 1 is a minimum, Solution 2 is a saddle point, and Solution 3 is a maximum. The eigenvalues report how the distance from the origin changes as we traverse the surface of the torus in each of the eigenvector directions, which in this case align with tangent directions. A positive eigenvalue corresponding to the tangent direction means that the loss is locally convex in the direction of the tangent. A negative eigenvalue indicates concavity. In the case of Solution 2, it is interesting to consider how the elliptical shape of the x_3

cross-sections of the torus ensures that moving along the torus with the tangent in the x_3 direction decreases the distance from the origin whereas moving with the x_2 tangent increases the distance. Such mixed curvature is the definitin of a saddle point.

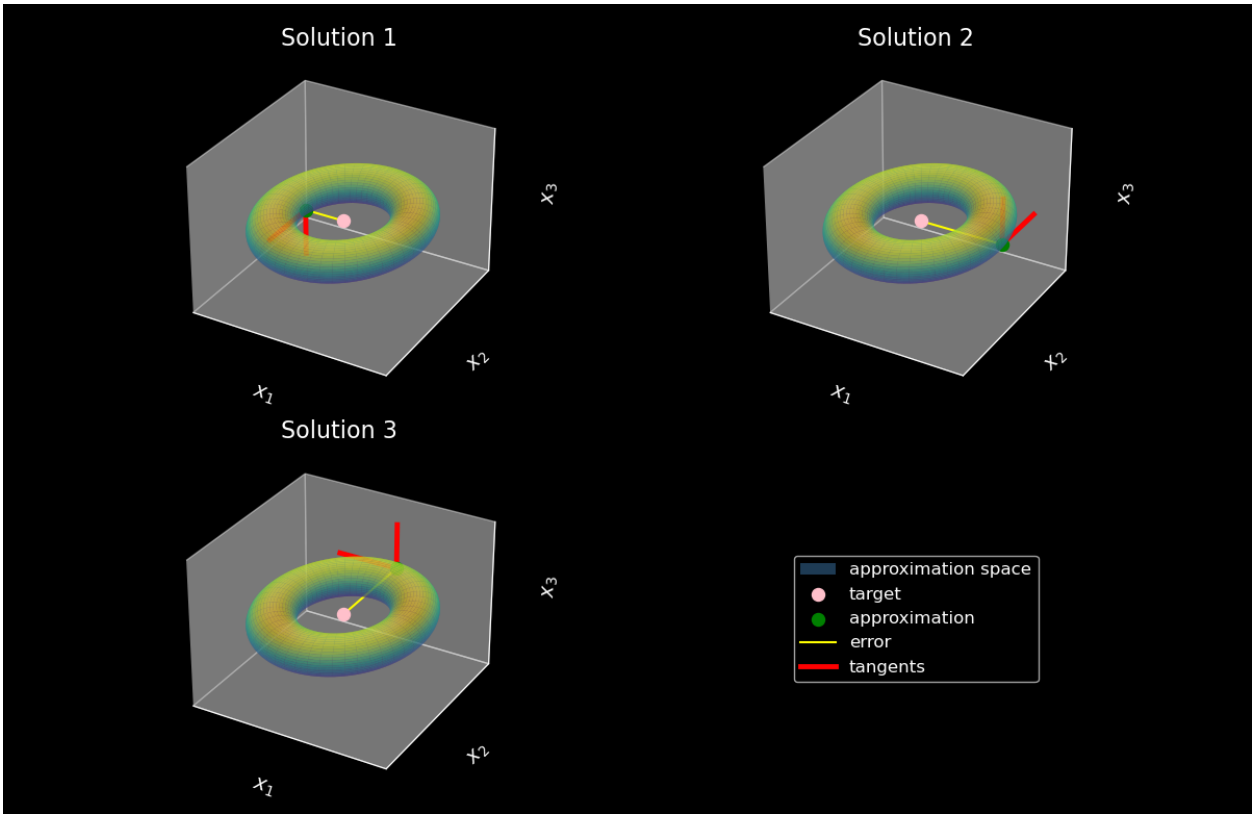

Figure 3: There are multiple stationary points of the objective function for the regression problem. Using the Hessian matrices, we classify Solution 1 as a minimum, Solution 2 as a saddle point, ans Solution 3 as a maximum.

In total, there are 8 stationary points of the regression given in Eq. (2). With the geometric interpretation of stationarity in mind, it is straightforward to see that there 8 positions on the torus where the position vector is orthogonal to the tangent space. All stationary points lie in the $x_3 = 0$ plane. See Figure 4 for a visualization and characterization of each of the stationary points, as well as a depiction of the dynamics of Newton optimization in the loss landscape. There are 2 minima, 2 maxima, and 4 saddle points. The convergence trajectories indicate that optimization based on standard Newton's method has no preference for minima over other stationary points. This is a well-known consequence of Newton's method looking for a zero of the gradient, rather than a minimum of the objective.

3 Regression with neural networks

The most widely used nonlinear discretization is that of a neural network. We now leverage the intuition from the circular and toroidal approximation spaces to interpret a standard neural network regression problem solved using

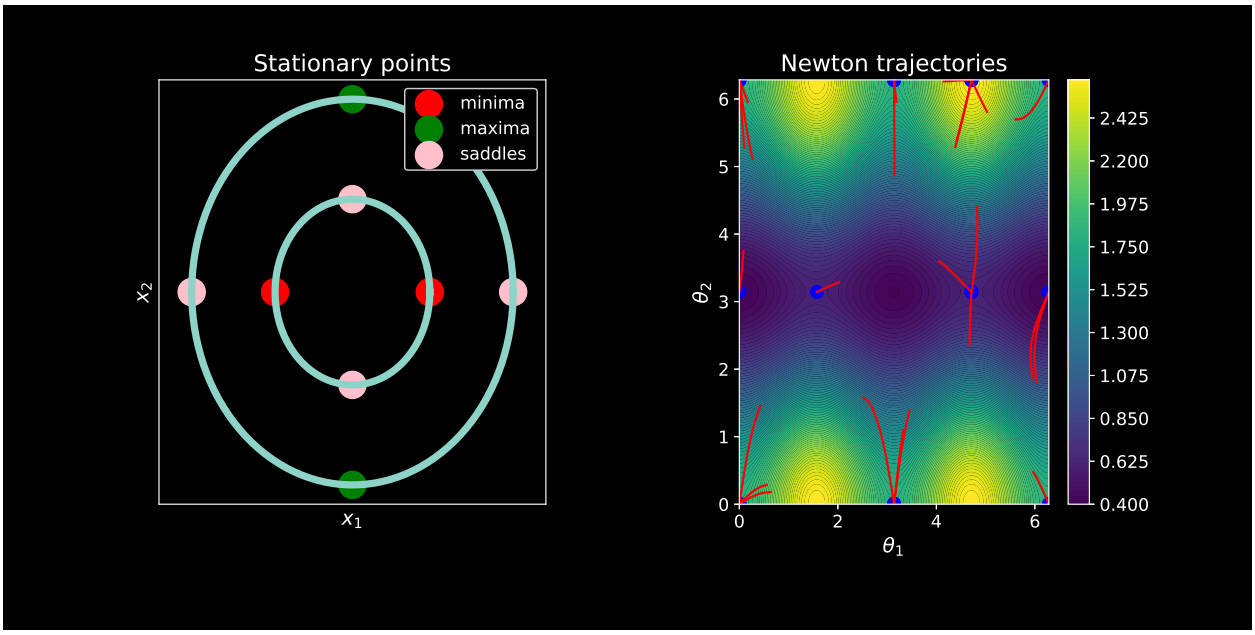

Figure 4: All stationary points are in the $x_3 = 0$ plane and lie along one of the coordinate axes (left). We show 25 convergence histories of Newton's method for random initializations of the parameters (right). Converged solutions are indicated by blue dots. Note that by periodicity, the saddle point at each of the four corners is actually the same solution. This is also the case for the minimum found along the left and right edge of the domain and the saddle at the center of the top and bottom edges.

Newton methods. We take the architecture to be a multilayer perceptron (MLP) neural network. In an MLP, the input-output relation for the i-th hidden layer is

$$\mathbf{y}_i = \sigma\Big(\mathbf{W}_i \mathbf{y}_{i-1} + \mathbf{B}_i\Big),$$

where $\sigma(\cdot)$ is a nonlinear "activation function" applied element-wise. As shown, the output $\mathbf{y}_i$ then becomes the next layer's input. The parameters of the neural network are the collection of the weight matrices $\mathbf{W}_i$ and bias vectors $\mathbf{B}_i$ for each layer. Thus, we can write the neural network parameters as $\boldsymbol{\theta} = [\mathbf{W}_1, \mathbf{B}_1, \mathbf{W}_2, \mathbf{B}_2, \dots]$. Typically, a linear mapping with no bias is used to go from the last hidden layer to the output.

Returning to the regression problem, we call the scalar target function $v(x)$ and the neural network discretization $\mathcal{N}(x; \boldsymbol{\theta})$ where $x \in [0, 1]$ and $\boldsymbol{\theta}$ is the collection of trainable parameters. The objective function is

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} \int_0^1 \Big(\mathcal{N}(x; \boldsymbol{\theta}) - v(x)\Big)^2 dx. \tag{5}$$

Using Newton's method, we find a zero of the following nonlinear system of equations corresponding to stationarity of the objective:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \int_0^1 \Big(\mathcal{N}(x;\boldsymbol{\theta}) - v(x)\Big)\frac{\partial \mathcal{N}}{\partial \boldsymbol{\theta}}dx \tag{6}$$

The connection to the finite dimensional stationary condition is made more clear when Eq. (6) is numerically integrated. Approximating the integral on a uniform grid, this reads

$$\frac{\partial \mathcal{L}}{\partial \theta_j} = \Delta x \sum_{i=1}^{N} \Big(\mathcal{N}(x_i;\boldsymbol{\theta}) - v(x_i)\Big)\frac{\partial \mathcal{N}(x_i,\boldsymbol{\theta})}{\partial \theta_j} = \Delta x \mathbf{e}(\boldsymbol{\theta}) \cdot \frac{\partial \mathbf{N}}{\partial \theta_j} = 0, \quad j = 1, 2, \dots$$

where $\{x_i\}_{i=1}^N$ are the integration points. Even in the continuous setting, stationarity of a quadratic loss dictates that the error is orthogonal to the tangent space of the approximation. With this in mind, we show that structure of MLP neural networks gives certain stationary points interpretable structure. First, we write the neural network discretization in a more revealing form as

$$\mathcal{N}(x,\boldsymbol{\theta}) = \sum_{k=1}^{|\boldsymbol{\theta}^{\mathrm{O}}|} \theta_k^{\mathrm{O}} h_k(x;\boldsymbol{\theta}^{\mathrm{I}}), \tag{7}$$

where the total parameter set $\boldsymbol{\theta} = [\boldsymbol{\theta}^{\mathrm{I}}, \boldsymbol{\theta}^{\mathrm{O}}]$ is decomposed into an "inner" and "outer" part. The inner parameters define the last layer of the network, which we interpret as basis functions $h_k(x;\boldsymbol{\theta}^{\mathrm{I}})$, and the outer parameters act as coefficients scaling these basis functions. Note that this assumes there is no bias in the output layer of the network. The most intuitive solution to Eq. (6) is when the error vector is zero, given that the zero vector is to any tangent vector. But, as the examples of the circle and torus suggest, nonlinear discretizations harbor other stationary points of the regression objective. To see an example of this, make the ansatz that $\mathcal{N}(x;\boldsymbol{\theta}) = 0$ is a solution to Eq. (6). In this case, the stationarity condition becomes

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \int_0^1 v(x)\frac{\partial \mathcal{N}}{\partial \boldsymbol{\theta}}dx = \begin{bmatrix} \int_0^1 v(x)\frac{\partial \mathcal{N}}{\partial \boldsymbol{\theta}^{\mathrm{O}}}dx \\ \int_0^1 v(x)\frac{\partial \mathcal{N}}{\partial \boldsymbol{\theta}^{\mathrm{I}}}dx \end{bmatrix} = \mathbf{0}, \tag{8}$$

where we have decomposed the parameter gradient defining the tangent vectors into inner and outer components. Referring to Eq. (7), it is clear that the outer parameter tangents are simply the basis functions at the current setting of the inner parameters:

$$\frac{\partial \mathcal{N}}{\partial \theta_j^{\mathrm{O}}} = h_j(x;\boldsymbol{\theta}^{\mathrm{I}}).$$

Thus, with the trivial regression solution $\mathcal{N} = 0$, the orthogonality of the error to the outer parameter gradients can be satisfied by fitting basis functions that are orthogonal to the target function $v(x)$. But, Eq. (8) shows that in order to satisfy stationarity generally, we also require orthogonality with respect to the inner parameter gradients. The inner parameter gradients can be written as

$$\frac{\partial \mathcal{N}}{\partial \theta_\ell^{\mathrm{I}}} = \sum_{k=1}^{|\boldsymbol{\theta}^{\mathrm{O}}|} \theta_k^{\mathrm{O}} \frac{\partial h_k(x)}{\partial \theta_\ell^{\mathrm{I}}}.$$

Given that all functions $v(x)$ are trivially orthogonal to the zero function, we simply require that $\boldsymbol{\theta}^{\mathrm{O}} = \mathbf{0}$ in order to satisfy orthogonality with respect to the inner parameter tangents. Thus, stationarity of the loss can be obtained by fitting zero coefficients on a basis that is orthogonal to the target function. At this point, we do not claim that neural networks trained with Newton's method actually find this solution. We only show that it is possible in principle to obtain trivial solutions to regression problems when using MLP neural networks trained with Newton's method. Note that such pathological behavior is made possible by the structure of the output layer of MLP neural networks, which gives a linear combination of a customizable basis. There is no analogue of setting the coefficients $\boldsymbol{\theta}^{\mathrm{O}} = \mathbf{0}$ in order to zero a subset of the tangents in the torus discretization. The tangent vectors of the torus are always nonzero, ensuring that orthogonality of the error cannot be trivially satisfied.

We now explore what solutions exact Newton optimization finds to a neural network-based regression problem. To the best of the author's knowledge, all second-order optimization studies in the literature have been with quasi-Newton methods. To this end, we define the orthogonality of the basis with the target function over the course of training as

$$O_j(t) = \int_0^1 \hat{v}(x)\hat{h}_j(x; \boldsymbol{\theta}^{\mathrm{I}}(t))dx,$$

where t is a pseudo-time variable indicating the optimization epochs, $\hat{v}(x)$ is the forcing function normalized to have a squared integral of unity, and $\hat{h}_j$ is the j-th basis function normalized in the same way. The objective is that of the regression problem given in Eq. (5), and the inner and outer gradient magnitudes are computed as $\|\partial\mathcal{L}/\partial\boldsymbol{\theta}^{\mathrm{I}}\|^2/|\boldsymbol{\theta}^{\mathrm{I}}|$ and $\|\partial\mathcal{L}/\partial\boldsymbol{\theta}^{\mathrm{O}}\|^2/|\boldsymbol{\theta}^{\mathrm{O}}|$ respectively. The standard Newton optimization is modified per

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta\left(\frac{\partial^2\mathcal{L}}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}} + \epsilon\mathbf{I}\right)^{-1}\Bigg|_{\boldsymbol{\theta}_k}\left(\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}}\right)\Bigg|_{\boldsymbol{\theta}_k}, \tag{9}$$

where $0 < \eta < 1$ relaxes the Newton step and $\epsilon > 0$ introduces convexity into the quadratic approximation of the loss to avoid excessively large steps when the loss has little to no curvature. This is known as the Levenberg-Marquardt algorithm [14]. Convergence of the Newton updating will halt once $\|\partial\mathcal{L}/\partial\boldsymbol{\theta}\| < \mathcal{T}$ where $\mathcal{T}$ is a problem-specific convergence criterion.

In the regression problem, we use a standard MLP architecture with two hidden layers, hyperbolic tangent activation functions, and 10 neurons per hidden layer with an output layer containing no bias. This corresponds to $|\boldsymbol{\theta}| = 140$ trainable parameters. Going forward, all neural network parameters will be initialized with the built-in Xavier initialization in PyTorch. We use an integration grid of 100 equally spaced points. The target function is $v(x) = 2\sin(4\pi x)$. In this example, we take $\eta = \epsilon = 5 \times 10^{-2}$ and $\mathcal{T} = 1 \times 10^{-5}$. See Figure 5 for the results of the Newton

optimization. Exactly as discussed above, we obtain a trivial solution by finding a basis which is orthogonal to the target function and setting the coefficients to zero. Because the constant function is orthogonal to $\sin(4\pi x)$, all but one of the basis functions is set is constant. The one non-constant basis resembles $\sin(\pi x) + c$ where c is a constant shift, which is also orthogonal to the target. Figure 6 shows the distribution of eigenvalues of the Hessian at the converged solution. Most eigenvalues are approximately zero, and the remainder are an even split between positive and negative, indicating that the solution is saddle point in this high-dimensional loss landscape.

We re-run the problem with different initializations to investigate the robustness of this trivial solution. In one numerical experiment, we obtained the trivial solution 9 out of 10 runs, though the non-constant basis functions is not always replicated. Note that we verify that the network is capable of accurately representing the target function by solving the regression problem with ADAM optimization, for which convergence to saddle points is not an issue.

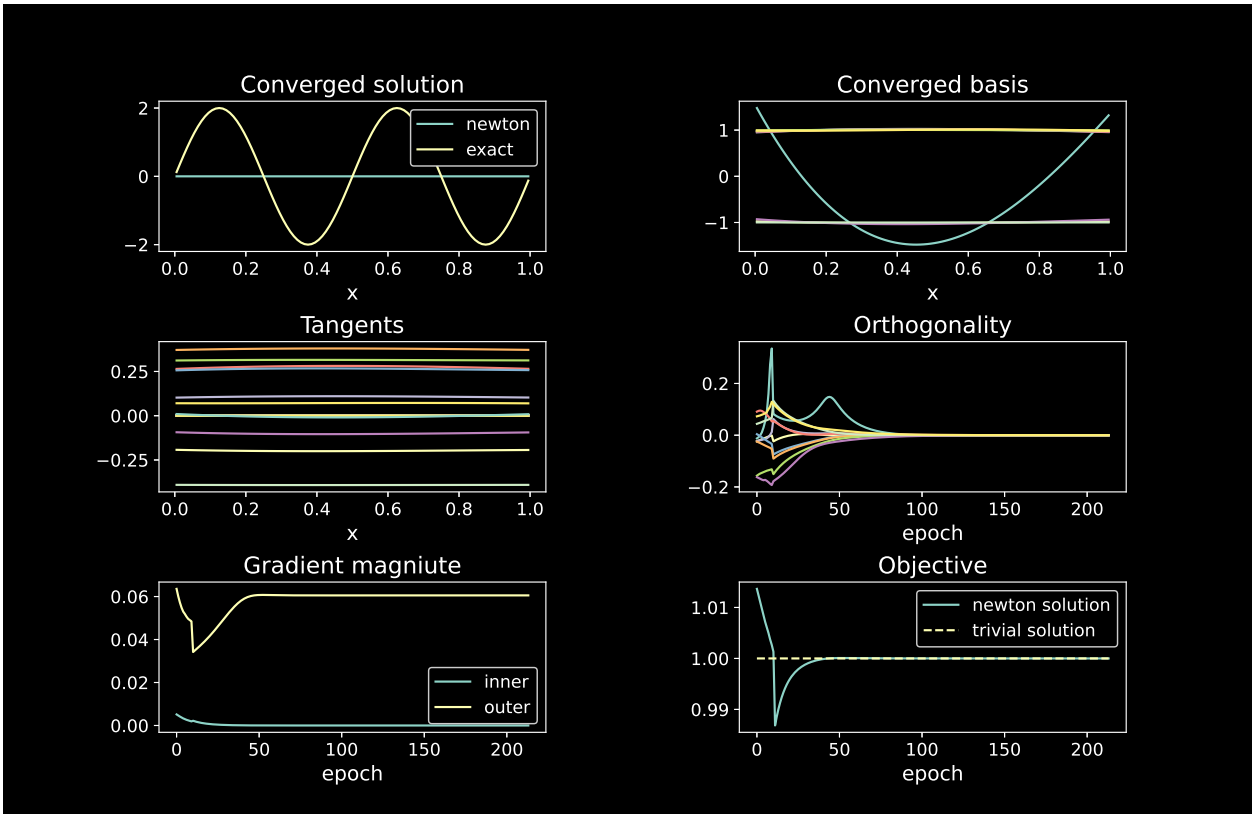

Figure 5: Exact Newton optimization obtains the trivial solution we identified. Note that the magnitude of each basis function is normalized to unity for visualization and in computing the evolution of orthogonality.

It is both interesting and surprising that the neural network reliably converges to a trivial solution by learning an orthogonal basis rather than minimizing the error with the target function. Figure 5 shows that among all choices of functions orthogonal to $\sin(4\pi x)$, the network favors the two lowest frequency options. We posit that this is a case of the well-known spectral bias of neural networks, which states that standard neural network architectures converge most rapidly on low frequency functions [16]. In order to push our findings a step further, we manually inject

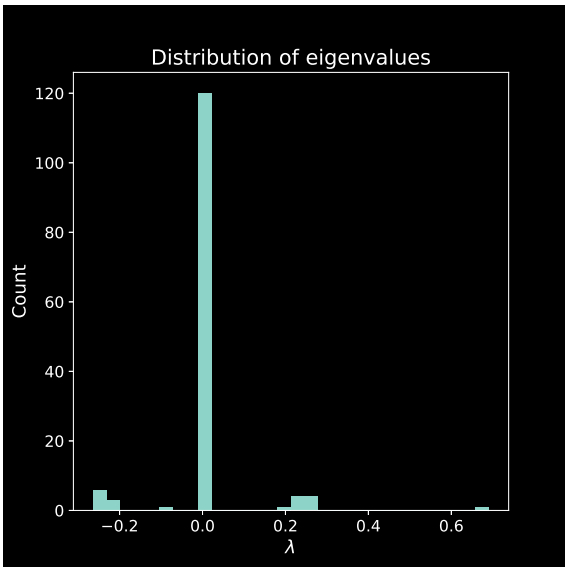

Figure 6: Most eigenvalues of the Hessian matrix at the converged solution cluster around zero, indicating that the loss landscape has little to no curvature in the corresponding eigenvector directions. The remaining eigenvalues are an even mix of positive and negative, indicating a saddle point solution. We note that a similar clustering of eigenvalues around zero is observed over the course of training as well, which explains why the Levenberg-Marquardt modification in Eq. (9) is required to stabilize training.

high-frequency behavior into the network by using sinusoidal activation functions, which is known as the sinusoidal representation network (SIREN) [24]. We use the same two hidden layer network but replace hyperbolic tangent activation functions with $\sin(\omega_0(\cdot))$ where ω_0 is a hyperparameter controlling the frequency content of the neural network approximation. We use the same target function and the same value of $\eta = 5 \times 10^{-2}$ but increase the convexity parameter to $\epsilon = 1 \times 10^{-1}$ and the convergence criterion to $\mathcal{T} = 1 \times 10^{-3}$. The frequency hyperparameter is set at $\omega_0 = 4$. These adjustments account for the more complex loss landscape arising from the oscillatory activation functions. See Figure 7 for the results of the Newton optimization. Once again, we obtain a trivial solution to the regression problem. However, the basis functions are now 1) high-frequency and 2) non-redundant. As before, the eigenvalues of the Hessian indicate that this solution is a saddle point. *Learning a high-frequency orthogonal basis as opposed to fitting the target function represents a spectacular failure of exact Newton optimization.* Fitting an orthogonal basis of the sort shown in Figure 7 seems to be a more complex problem than driving the error to zero by matching the target function. In fact, [19] shows that orthogonality of a basis is a prohibitively complex optimization objective for neural networks, though this was without high-frequency behavior injected into the network. In spite of the complexity of explicitly learning an orthogonal basis, one numerical experiment indicated that the trivial solution was obtained 4 out of 5 runs of the Newton optimization.

Another strategy for introducing high-frequency behavior into MLP neural networks is that of Fourier feature embedding [30]. Whereas the standard MLP network takes in the spatial coordinate x, the Fourier feature network's input layer is

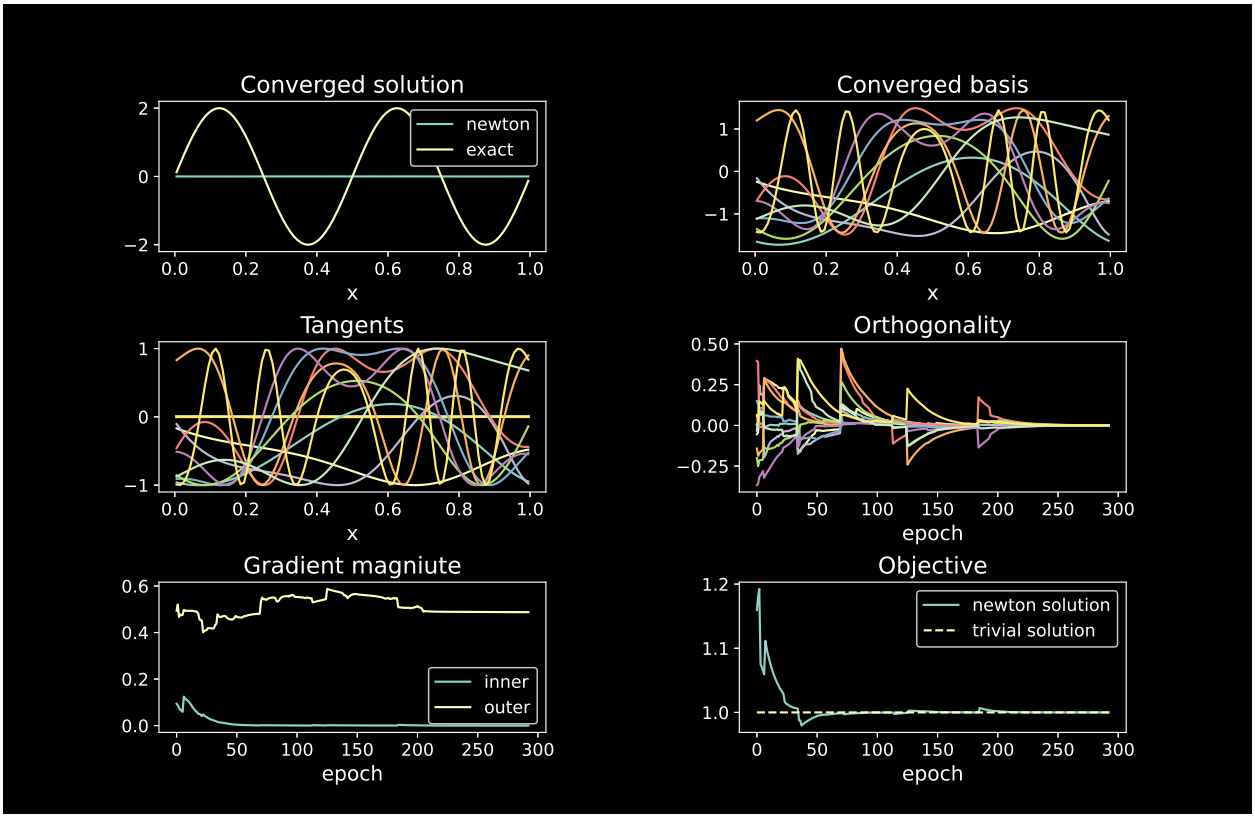

Figure 7: The SIREN network introduces manually introduces high-frequency behavior into the neural network, which leads to higher frequency basis functions but does not avoid the trivial saddle point solution.

$$\boldsymbol{\gamma}(x) = [\sin(2\pi \mathbf{B}x), \cos(2\pi \mathbf{B}x)]^T,$$

where $\mathbf{B} \in \mathbb{R}^f$ is a vector whose components are normally distributed with variance σ^2 and $2f$ is the number of Fourier features. The variance of the components of the random embedding matrix $\mathbf{B}$ determine the frequency content of the discretization. Taking $\sigma^2 = 1.5$, and $f = 10$, we use a two hidden layer MLP with hyperbolic tangent activation functions which takes in the Fourier features at the input layer. The width of the hidden layers is again 10, and we set $\eta = 5 \times 10^{-2}$, $\epsilon = 1 \times 10^{-1}$ and $\mathcal{T} = 1 \times 10^{-3}$. See Figure 8 for the results of the Newton optimization. In spite of the high-frequency basis, we again converge to the trivial solution. Like the SIREN network, the basis functions are high-frequency and non-redundant. From our experience, the trivial solution is obtained with the Fourier features approximately half of all runs, indicating again that the Newton optimization routinely fails to solve the desired regression problem. The majority of the remaining runs fail to converge at all.

4 Physics-informed machine learning

We explore whether the same trivial solution is obtained for physics-informed training, in addition to the regression problems of the previous section. As a first test, we solve the second-order elliptic boundary value problem given by

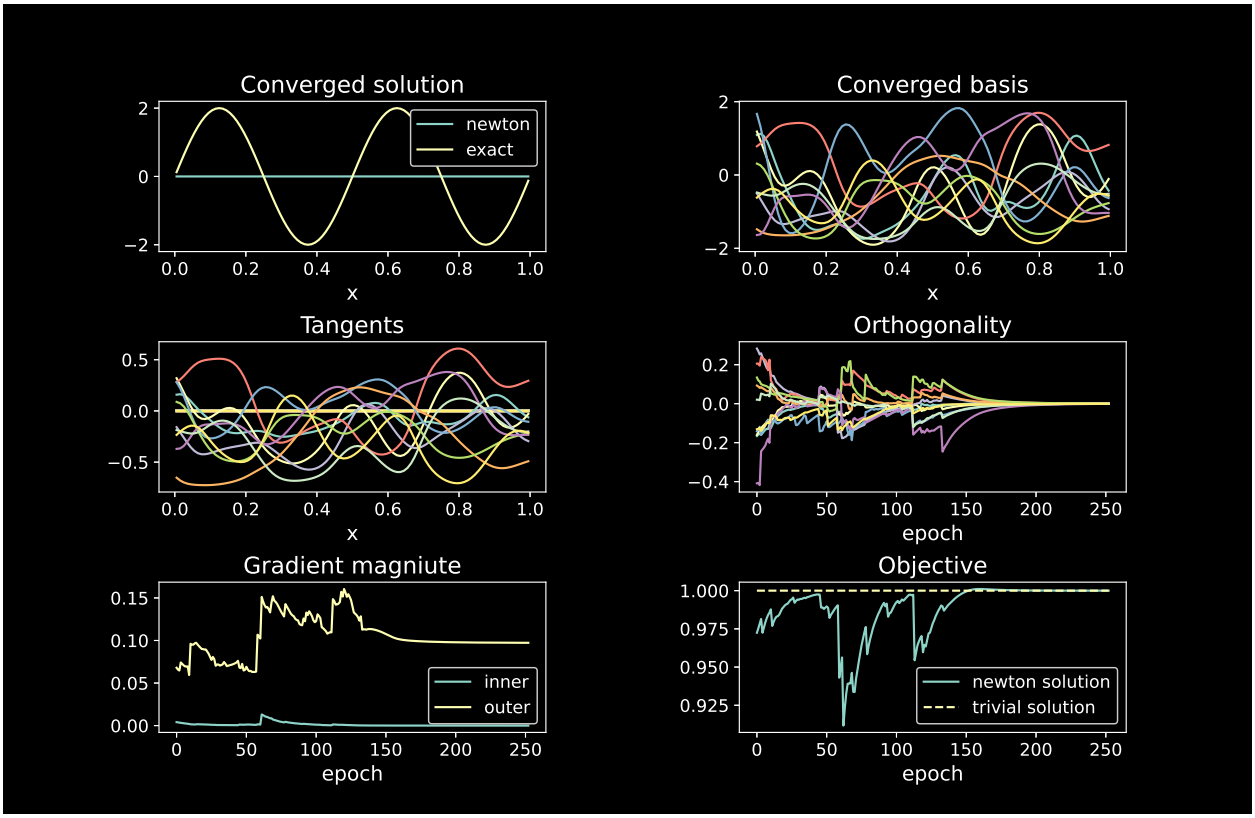

Figure 8: The MLP network with $2f = 20$ embedded Fourier features also finds a trivial solution to the regression problem with Newton optimization, in spite of the high-frequency basis.

$$\frac{\partial^2 u}{\partial x^2} + v(x) = 0, \quad u(0) = u(1) = 0.$$

Per the standard PINNs approach [17], we take the regression objective and modify it to minimize the strong form loss of the governing differential equation:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\int_0^1 \left(\frac{\partial^2 \mathcal{N}(x;\boldsymbol{\theta})}{\partial x^2} + v(x)\right)^2 dx, \quad \mathcal{N}(0;\boldsymbol{\theta}) = \mathcal{N}(1;\boldsymbol{\theta}) = 0. \tag{10}$$

We choose to build the boundary conditions into the neural network discretization of the solution with the distance function method. A standard distance function discretization for homogeneous Dirichlet boundaries is given by $\mathcal{N}(x;\boldsymbol{\theta}) = \sin(\pi x)\tilde{\mathcal{N}}(x;\boldsymbol{\theta})$, where $\tilde{\mathcal{N}}$ is a neural network that need not satisfy the Dirichlet boundary conditions [28, 25, 21]. We modify this distance function approach in order to preserve the interpretation of the inner and outer parameters of the network as building and scaling basis functions. As such, we use the distance function to enforce the homogeneous Dirichlet boundaries at the level of the basis functions themselves. The neural network discretization then reads

$$\mathcal{N}(x;\boldsymbol{\theta}) = \sum_{k=1}^{|\boldsymbol{\theta}^{\mathrm{O}}|} \theta_k^{\mathrm{O}} \sin(\pi x)\tilde{h}_k(x;\boldsymbol{\theta}^{\mathrm{I}}),$$

where $\tilde{h}_k$ represents the k-th neuron in the final layer of a standard MLP network. The basis functions to are then $h_k(x;\boldsymbol{\theta}^{\mathrm{I}}) = \sin(\pi x)\tilde{h}_k(x;\boldsymbol{\theta}^{\mathrm{I}})$. This ensures that the solution satisfies the homogeneous Dirichlet boundaries automatically without complicating our breakdown of the neural network into inner and outer parameters. Taking a gradient of Eq. (10), stationarity of the physics objective is given by

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = \int_0^1 \left(\frac{\partial^2 \mathcal{N}(x;\boldsymbol{\theta})}{\partial x^2} + v(x) \right) \frac{\partial^3 \mathcal{N}}{\partial \theta_k \partial x^2} dx = 0.$$

In order to obtain a trivial solution in the case of PINNs, it is necessary that $\mathcal{N} = 0$ and that the forcing function $v(x)$ is orthogonal to second spatial derivatives of each basis function. The relevant orthogonality condition tracked during Newton iteration becomes

$$O_j(t) = \int_0^1 \hat{v}(x) \frac{\partial^2 \hat{h}_j(x;\boldsymbol{\theta}^{\mathrm{I}})}{\partial x^2} dx,$$

where hats again indicate normalized quantities. We take $v(x) = 100\sin(4\pi x)$ and discretize the solution with a two hidden layer SIREN network of width 10 and $\omega_0 = 4$. Owing to the two order of magnitude increase in the scale of the objective, we take the convergence criterion to be $\mathcal{T} = 1$ and perform relaxed Newton optimization with $\eta = 5 \times 10^{-2}$ and $\epsilon = 1 \times 10^{-1}$. Spatial gradients as well as parameter gradients are now computed with automatic differentiation using PyTorch. See Figure 9 for the results. Once again, Newton's method converges to a trivial saddle solution. The interpretation of the trivial solution in the case of the PINNs solution is a simple analogue of the standard regression problem: the network learns basis functions whose image under the differential operator are orthogonal to the target function. This trivially enforces the stationarity of the physics loss. Like the regression examples, convergence to this solution is robust, occurring 5 out of 5 runs in one numerical experiment.

Thus far, all tests have been in one spatial dimension. As a final exploration of saddle point solutions with neural network discretizations, we solve a diffusion-reaction boundary value problem in two dimensions with homogeneous Dirichlet boundary conditions. The governing equation is given by

$$\begin{aligned} \nabla^2 u(\mathbf{x}) + u(\mathbf{x}) + v(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega, \end{aligned}$$

where the domain is $\Omega = [0,1]^2$. Using the same modification of the distance function method, the boundary conditions are built into the basis functions:

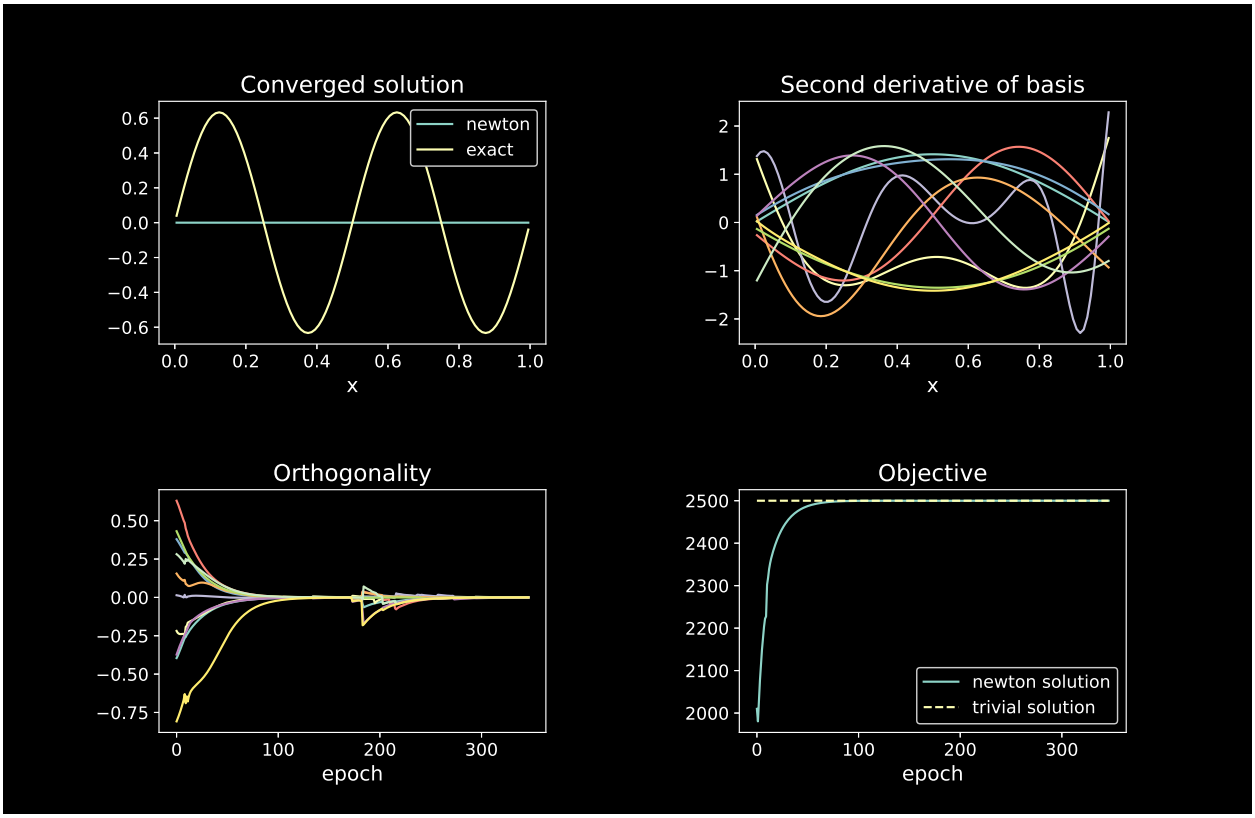

Figure 9: Training the network with a physics-based objective does not prevent convergence to trivial solutions. Understanding the nature of these saddle point solutions in the physics-informed regime only requires changing the orthogonality condition to involve the image of the basis under the differential operator.

$$\mathcal{N}(\mathbf{x};\boldsymbol{\theta}) = \sum_{k=1}^{|\boldsymbol{\theta}^{\mathrm{O}}|} \theta_k^{\mathrm{O}} \sin(\pi x_1)\sin(\pi x_2)\tilde{h}_k(\mathbf{x};\boldsymbol{\theta}^{\mathrm{I}}).$$

With the boundary conditions satisfied automatically by the discretization, the strong form loss is

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}\int_{\Omega} \left(\nabla^2\mathcal{N}(\mathbf{x};\boldsymbol{\theta}) + \mathcal{N}(\mathbf{x};\boldsymbol{\theta}) + v(\mathbf{x})\right)^2 d\Omega.$$

Analogous to the one dimensional PINNs problems, the condition for stationarity of the objective states that the residual is orthogonal to the differential operator applied to the discretization. In the case of the trivial solution, this reads

$$\frac{\partial\mathcal{L}}{\partial\theta_k} = \int_{\Omega} v(\mathbf{x})\left(\frac{\partial^3\mathcal{N}}{\partial x_i\partial x_i\partial\theta_k} + \frac{\partial\mathcal{N}}{\partial\theta_k}\right)d\Omega = 0. \tag{11}$$

Eq. (11) is satisfied when the target function is orthogonal to the image of the basis under the differential operator. The orthogonality condition we track over the course of optimization is

$$O_j(t) = \int_0^1 \hat{v}(\mathbf{x})\Big(\nabla^2 \hat{h}_j(\mathbf{x};\boldsymbol{\theta}^{\mathrm{I}}) + \hat{h}_j(\mathbf{x};\boldsymbol{\theta}^{\mathrm{I}})\Big) d\Omega.$$

We demonstrate that Newton methods again find this trivial solution. The forcing function is given by $v(\mathbf{x}) = 100\sin(4\pi x_1)\sin(4\pi x_2)$, and we use a SIREN network with with two hidden layers and a width of 10 hidden units per layer. The frequency hyperparameter is set at $\omega_0 = 5$. The step size and convexity parameters are $\epsilon = 5 \times 10^{-2}$ and $\epsilon = 1 \times 10^{-1}$. The convergence threshold is set at $\mathcal{T} = 1$. We also compare against ADAM optimization with a learning rate of 1×10^{-2} with the same architecture to verify that the network is sufficiently expressive to accurately represent the true solution. We note that the exact solution to the problem can be written down by inspection as

$$u(\mathbf{x}) = \frac{100}{32\pi^2}\sin(4\pi x_1)\sin(4\pi x_2).$$

See Figure 10 for the comparison of the Newton optimization to ADAM. As seen from the eigenvalues of the Hessian, we obtain a trivial saddle solution. This is in contrast to the ADAM optimizer, which drives the strong form loss down to near zero. Figure 11 gives a visual comparison of the ADAM and Newton solutions to the exact solution, as well as showing 6 of the 10 basis functions learned from Newton optimization. As before, it is remarkable that it is easier for the Newton to learn a basis which satisfies a complex orthogonality condition involving spatial gradients of the basis in place of simply solving the differential equation.

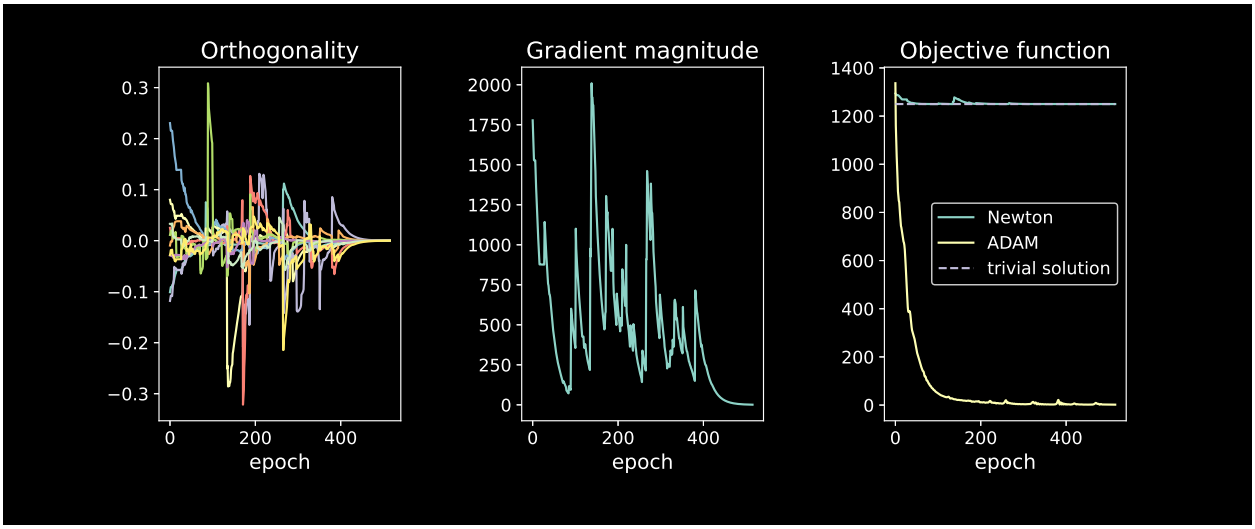

Figure 10: The converged solution is such that the learned basis satisfies the differential orthogonality condition. In contrast, ADAM has no trouble finding a solution to the governing equation, indicating that the network is sufficiently expressive.

It is surprising the lengths the neural network goes to in finding this trivial saddle solution. We remark that we never observe the neural network finding a *maximum* error solution, in spite of this remaining a theoretical possibility with exact Newton methods. Perhaps it is the case the saddle points are more prevalent in the loss landscape than minima or maxima? While the relative frequency of the three different kinds of stationary points is difficult to asses in high

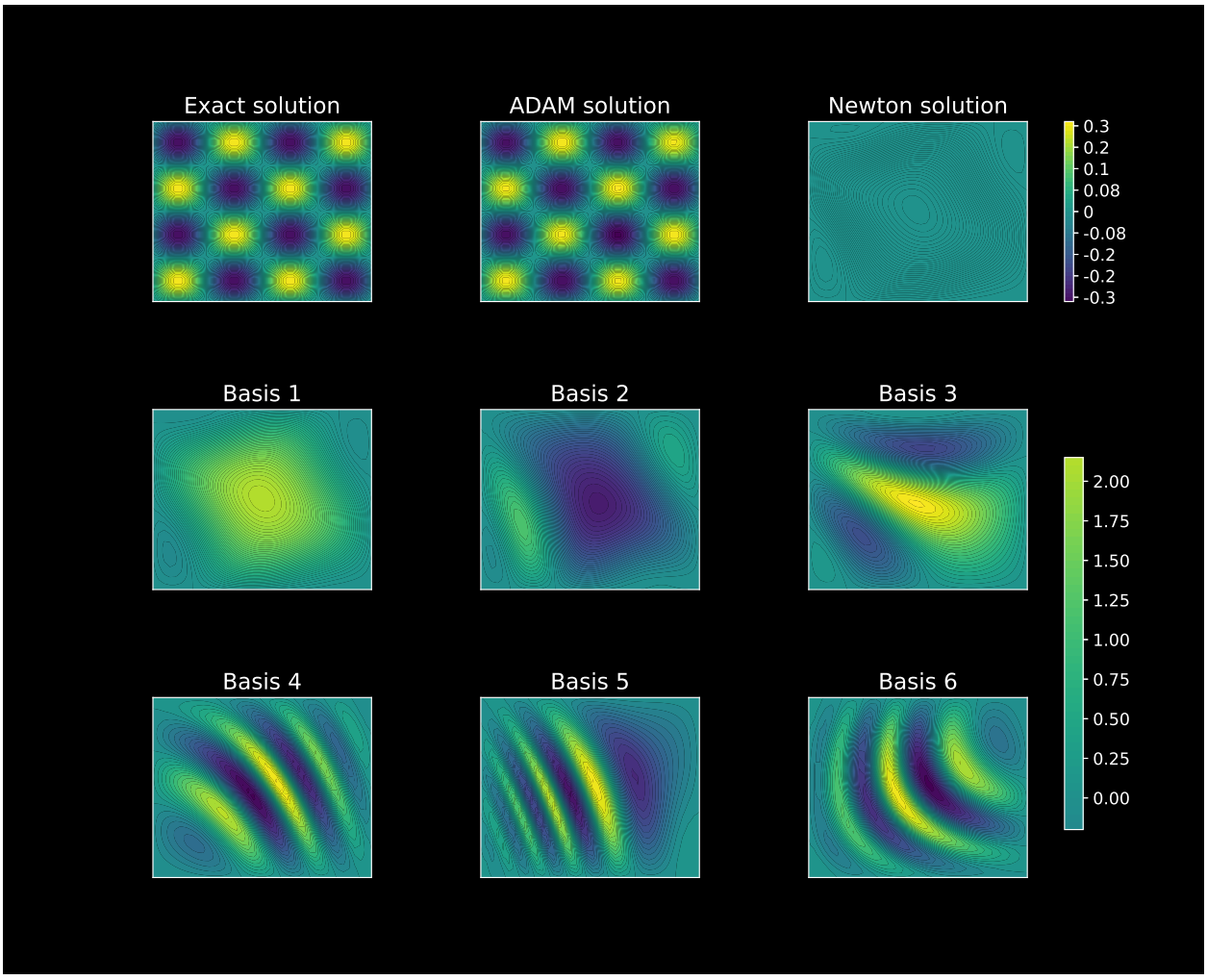

Figure 11: ADAM obtains the exact solution to the PINNs problem whereas Newton finds a trivial solution. The basis functions from the SIREN network are not themselves trivial, in spite of the solution they ultimately construct.

dimensions, a very simple-minded argument suggests that saddle points are more common than other extrema. We know that the eigenvalues of the Hessian matrix characterize the stationary point, and we know that the Hessian matrix is symmetric. Let us assume for the sake of argument that the independent components of the Hessian are distributed as independent standard random normal variables. We assume without loss of generality that the variance of the random normal variable is unity, as a constant scale factor will not affect the sign of the eigenvalues of a matrix. The number of parameters in the one-dimensional discretizations was $|\boldsymbol{\theta}| = 140$, so we take the Hessian matrix to be

$$\mathbf{J} = \frac{1}{2}(\mathbf{M} + \mathbf{M}^T), \quad \mathbf{M} \in \mathbb{R}^{140\times 140}, \quad M_{ij} \overset{\text{i.i.d.}}{\sim} N(0,1). \tag{12}$$

We randomly generate 10^5 such Hessian matrices and compute the eigenvalues. Out of these trials, not one random Hessian matrix has purely negative or purely positive eigenvalues. Given that the squared error regression objective is strictly positive, there is guaranteed to be a minimum somewhere, meaning there is some Hessian matrix with entirely

non-negative eigenvalues. The fact of the existence of this Hessian matrix indicates that the structure of these matrices is not purely random. However, it is not clear what other structure can be imposed on Hessian matrices apart from the one guaranteed minimum. As such, the independent and random components is a rough model. Certainly, this model suggests that saddle points drastically outnumber minima and maxima in a high-dimensional loss landscape. This sheds some light on the reliability of the convergence to saddle solutions. The predominance of saddle points in high dimensions is discussed in [5], where more sophisticated arguments in favor of this conclusion are provided.

5 Conclusion: avoiding saddle points

The examples given above beg the question: if Newton methods so reliably converge to trivial saddle point solutions, how is it that they are used in practice to train neural networks? First, we remark that true Newton methods involving the exact Hessian are a rarity in machine learning. Though the avoidance of forming and inverting the exact Hessian is usually attributed to computational cost, our results suggest that even if the true inverse Hessian could be obtained for free, it would not be useful. Remember that Newton methods solve for a zero of the gradient of the loss, rather explicitly minimizing the loss. Thus, given the prevalence of saddle points in loss landscapes involving neural networks, the true Hessian points the way to nearby stationary points, rather than in descent directions. While this is well-known, we believe the following point is under-appreciated in the machine learning literature: *second-order quasi-Newton optimizers succeed in practice not in spite of their failures to approximate the true Hessian, but because of them.* BFGS and L-BFGS approximations of the Hessian enforce the so-called “curvature condition" in order to maintain a positive definite approximation of the Hessian, even when steps in the loss landscape suggest otherwise [1]. The saddle-free Newton method modifies the true Hessian matrix in order to repel from saddle points and maxima [5]. See [22] for a review of variants of saddle free Newton methods. Figure 12 depicts a return to the torus regression problem with BFGS and saddle-free Newton methods. Now, all initializations converge to one of the two minima, in spite of the prevalence of saddles and maxima in the loss surface.

Of course, by explicitly minimizing the objective function, first-order optimization methods such as gradient descent and ADAM never ascend in the loss landscape to saddle solutions. They are, however, susceptible to capture by local minima and descending to saddle solutions, though such saddles are unstable fixed points of the gradient flow dynamics. We note that the standard intuition backing modifications of gradient descent such as ADAM is that the introduction of momentum allows escape from saddles and minima. If this is the case, it is not clear why one would expect a quasi-Newton optimizer, even one that avoids ascent directions, to perform better than ADAM. Under this interpretation, such a second-order optimizer—while equipped with curvature information in order to choose optimal step directions and sizes—would find the first local minimum and converge. Yet, a number of works cite quasi-Newton methods as not only being competitive with ADAM, but offering improvements in accuracy [18, 10]. If entrapment in local minima is a primary failure mode of an optimizer without momentum, it is not clear how to interpret the success of

quasi-Newton methods. We believe there remains much intuition to acquire about the nature of the loss landscape with neural networks and the dynamics of different optimizers.

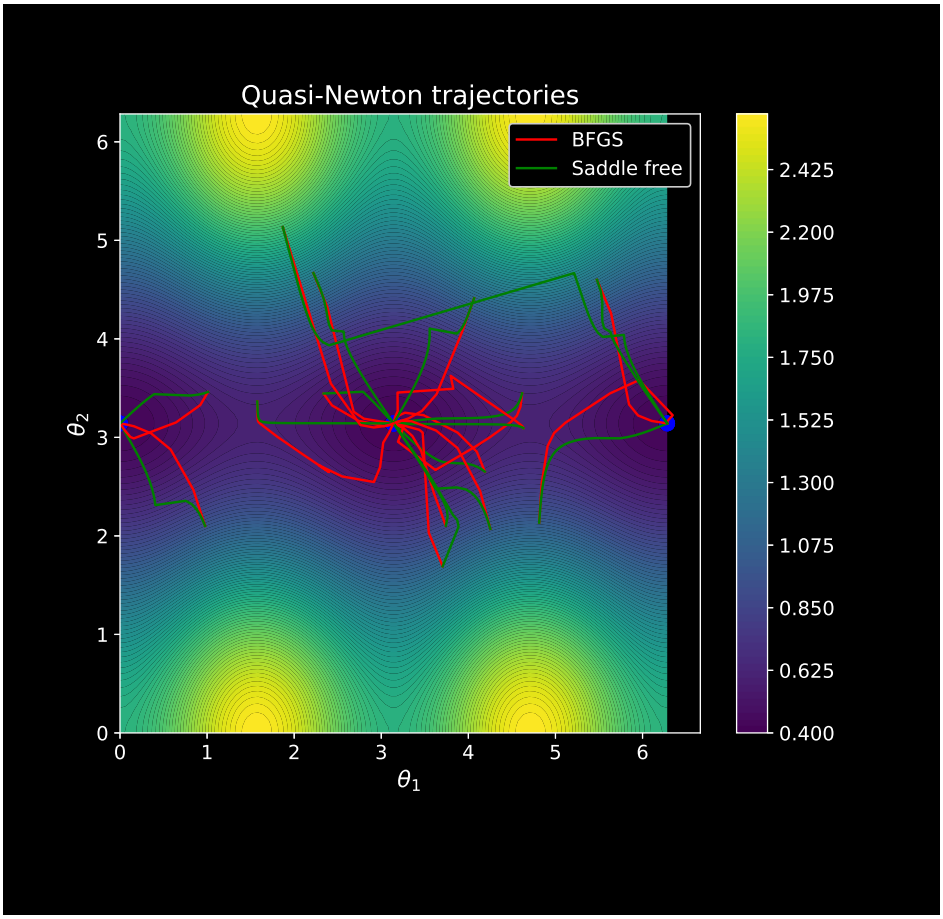

Figure 12: Comparing 15 optimization trajectories for the torus regression problem using the PyTorch BFGS quasi-Newton optimization and saddle free Newton, which modifies the exact Hessian to avoid saddle points and maxima. By neglecting negative curvature, both these methods converge to one of the two minima every time.

Our goals in this work have been to 1) to provide geometric insight into the non-convex loss landscape supplied by nonlinear discretizations, 2) characterize trivial solutions specifically for MLP neural networks, and 3) show that exact Newton methods reliably find these solutions. The robust failures of Newton optimization in no way calls into question the efficacy of second-order quasi-Newton optimization methods demonstrated in works such as [27, 3, 26]. However, our results do provide deeper insight into the nature of this efficacy. The second-order optimizer for machine learning does not improve the speed and accuracy of the solution by incorporating curvature of the loss. It does so by incorporating only that curvature information which pertains to minimizing the loss. This is the curvature which is relevant to finding descent directions and taking adaptively calibrated steps in these directions. If negative curvature information is incorporated into the Hessian, the neural network is prone to converge to a saddle solution, which our examples suggest are ubiquitous. BFGS approximations of the Hessian disregard negative curvature, whereas saddle free Newton methods explicitly step opposite the directions of negative curvature. Though the numerical examples

contained in this work are simple, we believe they are sufficient to provide worthwhile insight into the nature of loss landscapes defined by neural networks, while simultaneously surfacing fine points about the behavior of second-order optimization methods.

References

[1] Quasi-Newton Methods. In Jorge Nocedal and Stephen J. Wright, editors, *Numerical Optimization*, pages 192–221. Springer, New York, NY, 1999.

[2] Diab W. Abueidda, Seid Koric, Rashid Abu Al-Rub, Corey M. Parrott, Kai A. James, and Nahil A. Sobh. A deep learning energy method for hyperelasticity and viscoelasticity. *European Journal of Mechanics - A/Solids*, 95:104639, September 2022. arXiv:2201.08690 [cs].

[3] Shahbaz Ahmad and Muhammad Israr. A preconditioned quasi-newton optimizer for efficient training of PINNs. *Machine Learning for Computational Science and Engineering*, 1(2):34, September 2025.

[4] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-Informed Neural Networks for Heat Transfer Problems. *Journal of Heat Transfer*, 143(060801), April 2021.

[5] Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, June 2014. arXiv:1406.2572 [cs].

[6] Weinan E and Bing Yu. The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, September 2017. arXiv:1710.00211 [cs].

[7] 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.

[8] Reza Khodayi-Mehr and Michael M. Zavlanos. VarNet: Variational Neural Networks for the Solution of Partial Differential Equations, December 2019. arXiv:1912.07443 [cs].

[9] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. arXiv:1412.6980 [cs].

[10] Elham Kiyani, Khemraj Shukla, Jorge F. Urbán, Jérôme Darbon, and George Em Karniadakis. Optimizing the Optimizer for Physics-Informed Neural Networks and Kolmogorov-Arnold Networks, August 2025. arXiv:2501.16371 [cs].

[11] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations, May 2021. arXiv:2010.08895 [cs].

[12] Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. arXiv:1910.03193 [cs].

[13] M. Manav, R. Molinaro, S. Mishra, and L. De Lorenzis. Phase-field modeling of fracture with physics-informed deep learning. *Computer Methods in Applied Mechanics and Engineering*, 429:117104, September 2024.

[14] Donald W. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, June 1963. Publisher: Society for Industrial and Applied Mathematics.

[15] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[16] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the Spectral Bias of Neural Networks, May 2019. arXiv:1806.08734 [stat].

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

[18] Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in Training PINNs: A Loss Landscape Perspective, June 2024. arXiv:2402.01868 [cs].

[19] Conor Rowan, John Evans, Kurt Maute, and Alireza Doostan. Solving engineering eigenvalue problems with neural networks using the Rayleigh quotient, June 2025. arXiv:2506.04375 [math].

[20] Andrzej Ruszczynski. *Nonlinear Optimization*. Princeton University Press, 2006.

[21] Hailong Sheng and Chao Yang. PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries. *Journal of Computational Physics*, 428:110085, March 2021.

[22] Cooper Simpson. *Regularized Saddle-Free Newton: Saddle Avoidance and Ecient Implementation*. PhD thesis, University of Colorado Boulder, 2022.

[23] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, December 2018. arXiv:1708.07469 [q-fin].

[24] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions, June 2020. arXiv:2006.09661 [cs].

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

[26] Yubiao Sun, Ushnish Sengupta, and Matthew Juniper. Physics-informed deep learning for simultaneous surrogate modeling and PDE-constrained optimization of an airfoil geometry. *Computer Methods in Applied Mechanics and Engineering*, 411:116042, June 2023.

[27] Jorge F. Urbán, Petros Stefanou, and José A. Pons. Unveiling the optimization process of physics informed neural networks: How accurate and competitive can PINNs be? *Journal of Computational Physics*, 523:113656, February 2025.

[28] Jiaji Wang, Y.L. Mo, Bassam Izzuddin, and Chul-Woo Kim. Exact Dirichlet boundary Physics-informed Neural Network EPINN for solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 414:116184, September 2023.

[29] Sifan Wang, Ananyae Kumar Bhartari, Bowen Li, and Paris Perdikaris. Gradient Alignment in Physics-informed Neural Networks: A Second-Order Optimization Perspective, September 2025. arXiv:2502.00604 [cs].

[30] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, October 2021. arXiv:2012.10047 [cs].