Sampling Theory, Signal Processing, and Data Analysis (2023) 21:25
https://doi.org/10.1007/s43670-023-00066-6

ORIGINAL ARTICLE

Gauss–Newton method for solving linear inverse problems with neural network coders

Otmar Scherzer[1,2,3] · **Bernd Hofmann**[4] · **Zuhair Nashed**[5]

Received: 29 March 2023 / Accepted: 26 July 2023

Abstract
Neural networks functions are supposed to be able to encode the desired solution of an inverse problem very efficiently. In this paper, we consider the problem of solving linear inverse problems with neural network coders. First we establish some correspondences of this formulation with existing concepts in regularization theory, in particular with state space regularization, operator decomposition and iterative regularization methods. A Gauss–Newton method is suitable for solving encoded linear inverse problems, which is supported by a local convergence result. The convergence studies, however, are not complete, and are based on a conjecture on linear independence of activation functions and its derivatives. Some numerical experiments are presented to support the theoretical findings.

Keywords Gauss–Newton method · Inverse problems · Neural networks

This paper is dedicated to Heinz Engl's 70th birthday.

Communicated by Akram Aldroubi.

✉ Otmar Scherzer
otmar.scherzer@univie.ac.at

Bernd Hofmann
hofmannb@mathematik.tu-chemnitz.de

Zuhair Nashed
zuhair.nashed@ucf.edu

1 Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

2 Johann Radon Institute for Computational and Applied Mathematics (RICAM), Altenbergerstraße 69, 4040 Linz, Austria

3 Christian Doppler Laboratory for Mathematical Modeling and Simulation of Next Generations of Ultrasound Devices (MaMSi), Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

4 Faculty of Mathematics, Chemnitz University of Technology, Reichenhainer Str. 39/41, 09107 Chemnitz, Germany

5 College of Sciences, University of Central Florida, 4393 Andromeda Loop N, Orlando, FL 32816, USA

Published online: 04 August 2023 Birkhäuser

Mathematics Subject Classification 65J22 · 47J07

1 Introduction

We start the discussion by considering a *general nonlinear operator equation*

$$N(\vec{p}\,) = \mathbf{y}, \tag{1.1}$$

where $N : \vec{P} \to \mathbf{Y}$ is a nonlinear operator between Hilbert spaces $\vec{P}$ and $\mathbf{Y}$. Particular emphasis is placed on the case when the numerical solution of Eq. (1.1) is ill-posed or ill-conditioned. In this situation regularization methods need to be implemented (see [15]) to guarantee stable solvability. Essentially two classes of methods exist as the basis for solving inverse problems numerically, which are *variational methods* (see for instance [3, 15, 23, 33, 45, 46]) and *iterative methods* (see for instance [4, 30]).

Modern *coding theory* (see for instance [6, 6, 8, 14, 44]) assumes that *natural images* can be represented efficiently by a combination of *neural network* functions. Therefore the *set of natural images* is given by the range of a nonlinear mapping Ψ, which maps neural network coefficients to images. In coding theory very often the considered operator equation is assumed to be *linear* and *ill-posed*. Let us consider therefore the equation

$$F\mathbf{x} = \mathbf{y}, \tag{1.2}$$

where $F : \mathbf{X} \to \mathbf{Y}$ is a bounded linear operator with non-closed range mapping between the Hilbert spaces $\mathbf{X}$ and $\mathbf{Y}$ such that $\mathbf{x}$ and $\mathbf{y}$ are elements of $\mathbf{X}$ and $\mathbf{Y}$, respectively. Combining this with the assumption that the solution of Eq. (1.2) is a natural image, or in other words that it can be represented as a combination of neural network functions, we get the operator equation

$$N(\vec{p}\,) = F\Psi(\vec{p}\,) = \mathbf{y}, \tag{1.3}$$

where $\Psi : \vec{P} \to \mathbf{X}$ is the before mentioned nonlinear operator that maps neural network parameters to image functions. We call

- $\mathbf{X}$ the *image space* and
- $\mathbf{Y}$ the *data space*, in accordance with a terminology, which we generated in [1, 2]. $\mathbf{X}$ and $\mathbf{Y}$ should be considered spaces of functions or discretizations of such.
- $\vec{P}$ is called the *parameter space*. We make a different notation for $\vec{P}$, because it represents parametrizations, and is often considered a space of vectors below.

The advantage of this ansatz is that the solution of Eq. (1.2) is sparsely coded. However, the price to pay is that the reformulated Eq. (1.3) is nonlinear. Operator equations of the form of Eq. (1.3) are not new: They have been studied in abstract settings for instance in the context of

- *state space regularization* [9] and

- in the context of the *degree of ill-posedness* [16, 21, 22, 25, 26] as well as of the *degree of nonlinearity* [25].
- Another related approach is *finite dimensional approximation* of regularization in Banach spaces (see for instance [43]). Particularly, finite dimensional approximations of regularization methods with neural network functions (in the context of frames and *deep synthesis regularization*) have been studied in [39].

In this paper we aim to link the general regularization of the degree of ill-posedness and nonlinearity with coding theory. We investigate generalized Gauss–Newton methods for solving Eq. (1.3); Such methods replace the inverse of standard Newton method by outer inverses or by approximations of such (see [38]).

The outline of the paper is as follows: In Sect. 2 we review two decomposition cases as stated in [22] first, one of them is Eq. (1.3). The study of decomposition cases follows the work on classifying inverse problems and regularization (see [34]). For operators associated to Eq. (1.3), Newton methods seem to be better suited than gradient descent methods in terms of convergence analysis (see Sect. 3). However, this does not necessarily apply to practice (see Sect. 4). Section 3.4 is devoted to solving Eq. (1.3), where Ψ is a shallow neural network synthesis operator. In Sect. 4 some simple numerical experiments are presented, which, in particular, show the advantage of the Gauss–Newton in terms of a simple analysis in comparison with gradient methods. On the other hand one practical problem appears, which is related to stability (see Remark 4.1).

2 Decomposition cases

We start with a definition for nonlinear operator equations possessing forward operators that are compositions of a linear and a nonlinear operator. Precisely, we distinguish between a first decomposition case (i), where the linear operator is the inner operator and the nonlinear is the outer one, and a second decomposition case (ii), where the nonlinear operator is the inner operator and the linear is the outer operator.

Definition 2.1 (*Decomposition cases*) Let $\vec{P}$, $\mathbf{X}$, $\mathbf{Y}$ be Hilbert-spaces.

(i) An operator N is said to satisfy the *first decomposition case* in an open, nonempty neighborhood $\mathcal{B}(\vec{p}^{\,\dagger};\rho) \subseteq \vec{P}$ of some point $\vec{p}^{\,\dagger}$ if there exists a linear operator $F : \vec{P} \to \mathbf{X}$ and a nonlinear operator $\Psi : \mathbf{X} \to \mathbf{Y}$ such that

$$N(\vec{p}\,) = \Psi(F\vec{p}\,) \quad \text{for } \vec{p} \in \mathcal{B}(\vec{p}^{\,\dagger};\rho).$$

(ii) N is said to satisfy the *second decomposition case* in a neighborhood $\mathcal{B}(\vec{p}^{\,\dagger};\rho) \subseteq \vec{P}$ of some point $\vec{p}^{\,\dagger}$ if there exists a linear operator $F : \mathbf{X} \to \mathbf{Y}$ and a nonlinear operator $\Psi : \vec{P} \to \mathbf{X}$ such that

$$N(\vec{p}\,) = F\Psi(\vec{p}\,) \quad \text{for } \vec{p} \in \mathcal{B}(\vec{p}^{\,\dagger};\rho). \tag{2.1}$$

Typically it is assumed that the nonlinear operator Ψ is well-posed.

Remark 2.2 (First decomposition case) In [22], this decomposition case has been studied under structural conditions, relating the second derivative of N with the first derivative. Under such assumptions convergence rates conditions (see [22, Lemma 4.1]) could be proven. The first decomposition case also arises in inverse option pricing problems in math finance (see [20] and [24, Sect.4]), where the ill-posed compact linear integration operator occurs as inner operator and a well-posed Nemytskii operator as outer operator.

Remark 2.3 (Second decomposition case) Regularization methods for solving operator equations with operators satisfying the second order decomposition case, see Eq. (2.1), were probably first analyzed in [9] under the name of *state space regularization*. They considered for instance Tikhonov-type regularization methods, consisting in minimization of

$$J_\lambda(\vec{p}\,) = \|F\Psi(\vec{p}\,) - \mathbf{y}\|_{\mathbf{Y}}^2 + \lambda\,\|\Psi(\vec{p}\,) - \tilde{\mathbf{x}}\|_{\mathbf{X}}^2\,, \tag{2.2}$$

where $\tilde{\mathbf{x}}$ is a prior and $\lambda > 0$ is a regularization parameter. In [9] they derived estimates for the second derivative of $J_\lambda(\vec{p}_\lambda)(\mathbf{h}, \mathbf{h})$, where $\mathbf{h} \in \vec{P}$, that is for the curvature of J_λ. If the curvature can be bounded from below by some term $\|\mathbf{h}\|_{\vec{P}}^2$, then, for instance, a locally unique minimizer of J_λ can be guaranteed and also domains can be specified where the functional is convex. Conditions, which guarantee convexity are called *curvature to size conditions*. Subsequently, these decomposition cases have been studied exemplarily in [22]. The theory developed there directly applies to Eq. (1.3).

Instead of J_λ researchers often study direct regularization with respect to $\vec{p}$. For instance in [14] functionals of the form

$$J_\lambda(\vec{p}\,) = \|F\Psi(\vec{p}\,) - \mathbf{y}\|_{\mathbf{Y}}^2 + \lambda\mathcal{L}(\vec{p}\,), \tag{2.3}$$

where $\mathcal{L}$ is some functional directly regularizing the parameter space. Typically $\mathcal{L}$ is chosen to penalize for sparsity of parameters. The main difference between Eqs. (2.2) and (2.3) is that in the prior regularization is performed with respect to the image space $\mathbf{X}$ and in the later with respect to the parameter space $\vec{P}$. Well-posedness of the functional J_λ in Eq. (2.2) follows if $F \circ \Psi$ is lower-semicontinuous, which in turn follows if Ψ is invertible.

In the following we study the solution of decomposable operator equations, such as Eq. (1.3), with Gauss–Newton methods. Decomposition cases have been used in the analysis of iterative regularization methods as well (see [30]):

Definition 2.4 (*Strong tangential cone condition*) Let $N : \mathcal{D}(N) \subset \vec{P} \to \mathbf{Y}$ with $\mathcal{D}(N)$ its domain be a nonlinear operator.

(i) Then N is said to satisfy the strong tangential cone condition, originally introduced in [18], if

$$N'(\vec{p}_2) = R_{\vec{p}_2,\vec{p}_1} N'(\vec{p}_1) \quad \text{for all } \vec{p}_1, \vec{p}_2 \in \mathcal{D}(N). \tag{2.4}$$

where

$$\left\| R_{\vec{p}_2,\vec{p}_1} - I \right\| \leq C_T \left\| \vec{p}_2 - \vec{p}_1 \right\|_{\vec{P}} . \tag{2.5}$$

(ii) In [5] the *range invariance condition*,

$$N'(\vec{p}_2) = N'(\vec{p}_1) R_{\vec{p}_2,\vec{p}_1} \quad \text{for all } \vec{p}_1, \vec{p}_2 \in \mathcal{D}(N), \tag{2.6}$$

together with Eq. (2.5), has been introduced.

Remark 2.5 Equation (2.4) has been used for analyzing *gradient descent methods* (see for instance [18, 30]). For the analysis of Newton methods Eq. (2.6) has been used (see [5, 30]).

The relation to the decomposition cases is as follows:

Lemma 2.6 *Let $N : \mathcal{D}(N) \subseteq \vec{P} \to \mathbf{Y}$ with $\mathcal{D}(N) = \mathcal{B}(\vec{p}^{\,\dagger}; \rho)$ satisfy the second decomposition case and assume that $\Psi'(\vec{p}\,)$ is invertible for $\vec{p} \in \mathcal{D}(N)$. Then N satisfies Eq. (2.6).*

Proof If N satisfies the second decomposition case, Eq. (2.1), then $N'(\vec{p}\,) = F\Psi'(\vec{p}\,)$ for all $\vec{p} \in \mathcal{D}(N)$. Note, that because F is defined on the whole space $\mathbf{X}$, $\mathcal{D}(N) = \mathcal{D}(\Psi)$. By using the invertability assumption on Ψ we get

$$N'(\vec{p}_2) = F\Psi'(\vec{p}_2) = F\Psi'(\vec{p}_1) \underbrace{\Psi'(\vec{p}_1)^{-1}\Psi'(\vec{p}_2)}_{=:R_{\vec{p}_2,\vec{p}_1}} = N'(\vec{p}_1) R_{\vec{p}_2,\vec{p}_1},$$

which gives the assertion. □

As we have shown, decomposition cases have been extensively studied in the regularization literature. One conclusion out of these studies is that the range invariant condition Eq. (2.6) is suitable for analyzing Newton methods [5, 30] and thus in turn for the coded linear operator Eq. (1.3) because of Lemma 2.6. The standard tool for analyzing Newton methods is the Newton–Mysovskii condition as discussed below.

3 The Newton–Mysovskii conditions

In this section we put abstract convergence conditions for Newton type methods in context with decoding. We consider first Newton methods for solving the *general* operator Eq. (1.1). Decomposition cases of the operator N will be considered afterwards.

3.1 Newton method with invertible linearizations

For Newton methods *local convergence* is guaranteed under *affine covariant* condition. We refer to them as generalized *Newton–Mysovskii* to attribute to the inventors, which figured out that structural properties of operators are important for proving

convergence. For comparison reasons, we first recall a simple Newton method analysis in finite dimensional spaces if the nonlinear operator has derivatives which are invertible. The proof of more general results, such as Theorem 3.6 below, is contained in the paper, and thus for this standard result the proof is omitted. Several variants of Newton–Mysovskii conditions have been proposed in the literature (see for instance [12, 13, 38]). Analysis of Newton method was an active research area in the last century, see for instance [40, 47].

Theorem 3.1 (Finite dimensional Newton method) *Let $N : \mathcal{D}(N) \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be continuously Fréchet-differentiable on a non-empty, open and convex set $\mathcal{D}(N)$. Let $\vec{p}^{\,\dagger} \in \mathcal{D}(N)$ be a solution of Eq. (1.1). Moreover, we assume that*

(i) *$N'(\vec{p}\,)$ is invertible for all $\vec{p}\, \in \mathcal{D}(N)$ and that*
(ii) *the* Newton–Mysovskii condition *holds: That is, there exist some $C_N > 0$ such that*

$$\begin{aligned} &\left\| N'(\vec{q}\,)^{-1}(N'(\vec{p}\, + s(\vec{q}\, - \vec{p}\,)) - N'(\vec{p}\,))(\vec{q}\, - \vec{p}\,) \right\|_{\vec{P}} \\ &\quad leqsC_N \, \|\vec{p}\, - \vec{q}\, \|_{\vec{P}}^2 \;\; \text{for all } \vec{p}\,, \vec{q}\, \in \mathcal{D}(N), s \in [0,1]. \end{aligned} \tag{3.1}$$

Let $\vec{p}^{\,0} \in \mathcal{D}(N)$ which satisfies

$$\overline{\mathcal{B}(\vec{p}^{\,0}; \rho)} \subseteq \mathcal{D}(N) \text{ with } \rho := \left\| \vec{p}^{\,\dagger} - \vec{p}^{\,0} \right\|_{\vec{P}} \text{ and } h := \frac{\rho C_I C_L}{2} < 1. \tag{3.2}$$

Then the Newton iteration with starting point $\vec{p}^{\,0}$,

$$\vec{p}^{\,k+1} = \vec{p}^{\,k} - N'(\vec{p}^{\,k})^{-1}(N(\vec{p}^{\,k}) - \mathbf{y}) \quad k \in \mathbb{N}_0, \tag{3.3}$$

satisfies that the iterates $\left\{\vec{p}^{\,k} : k = 0, 1, 2, \ldots\right\}$ belong to $\overline{\mathcal{B}(\vec{p}^{\,0}; \rho)}$ and converge quadratically to $\vec{p}^{\,\dagger} \in \overline{\mathcal{B}(\vec{p}^{\,0}, \rho)}$.

Now, we turn to the case that N is a decomposition operator.

3.2 Newton–Mysovskii conditions with composed operator

Now, we study the case of convergence of Gauss–Newton methods where $N : \vec{P} \to \mathbf{Y}$ with $\vec{P} = \mathbb{R}^{n_*}$ and $\mathbf{Y}$ is an infinite dimensional Hilbert space, where $F : \mathbf{X} \to \mathbf{Y}$ is linear and bounded and $\Psi : \vec{P} = \mathbb{R}^{n_*} \to \mathbf{X}$ is differentiable. In this case the Moore–Penrose inverse, or even more general the outer inverse, replaces the inverse in a classical Newton method (see Eq. (3.3)), because linearizations of N cannot be invertible as a simple count of dimensions show. We refer now to Gauss–Newton methods if the linearizations might not be invertible to distinguish between classical Newton methods also by name.

Before we phrase a convergence result for Gauss–Newton methods we recall and introduce some definitions:

Notation 3.2 (*Inner, outer and Moore–Penrose inverse*) (see [35, 37]) Let $L : \vec{P} \to \mathbf{Y}$ be a linear and bounded operator mapping between two vector spaces $\vec{P}$ and $\mathbf{Y}$. Then

(i) the operator $B : \mathbf{Y} \to \vec{P}$ is called a *left inverse* to L if

$$BL = I.$$

(ii) $B : \mathbf{Y} \to \vec{P}$ is called a *right inverse* to L if

$$LB = I.$$

Left and right inverses are used in different context:

- For a left inverse the nullspace of L has to be trivial, in contrast to B.
- For a right inverse the nullspace of B has to be trivial.

(iii) $B : \vec{P} \to \vec{P}$ is called a *inverse* to L if B is a right and a left inverse.
(iv) $B : \vec{P} \to \mathbf{Y}$ is an *outer inverse* to L if

$$BLB = B. \tag{3.4}$$

(v) Let $\vec{P}$ and $\mathbf{Y}$ be Hilbert-spaces, $L : \vec{P} \to \mathbf{Y}$ be a linear bounded operator. We denote the orthogonal projections P and Q onto $\mathcal{N}(L)$, the nullspace of L (which is closed), and $\overline{\mathcal{R}(L)}$, the closure of the range of L: That is for all $\vec{p} \in \vec{P}$ and $\mathbf{y} \in \mathbf{Y}$ we have

$$\begin{aligned} P\vec{p} &= \operatorname{argmin}\left\{\|\vec{p}_1 - \vec{p}\,\|_{\vec{P}} : \vec{p}_1 \in \mathcal{N}(L)\right\} \text{ and} \\ Q\mathbf{y} &= \operatorname{argmin}\left\{\|\mathbf{y}_1 - \mathbf{y}\|_{\mathbf{Y}} : \mathbf{y}_1 \in \overline{\mathcal{R}(L)}\right\}. \end{aligned} \tag{3.5}$$

We therefore have

$$\begin{aligned} P : \vec{P} &\to \mathcal{N}(L)\dot{+}\mathcal{N}(L)^{\perp} \\ \vec{p} &\mapsto P\vec{p} + 0 \end{aligned} \quad \text{and} \quad \begin{aligned} Q : \mathbf{Y} &\to \overline{\mathcal{R}(L)}\dot{+}\mathcal{R}(L)^{\perp}. \\ \mathbf{y} &\to Q\mathbf{y} + 0 \end{aligned}$$

$B : \mathcal{D}(B) \subseteq \mathbf{Y} \to \vec{P}$ with $\mathcal{D}(B) := \mathcal{R}(L)\dot{+}\mathcal{R}(L)^{\perp}$ is called the *Moore–Penrose inverse* of L if the following identities hold

$$\begin{aligned} LBL &= L, \\ BLB &= B, \\ BL &= I - P, \\ LB &= Q|_{\mathcal{D}(B)}. \end{aligned} \tag{3.6}$$

In coding theory it is often stated that the range of a neural network operator Ψ forms a manifold in $\mathbf{X}$, a space, which contains the natural images. This is the basis of the following definition making use of the Moore–Penrose inverse.

Definition 3.3 (Lipschitz-differentiable immersion) Let $\Psi : \mathcal{D}(\Psi) \subseteq \vec{P} = \mathbb{R}^{n_*} \to \mathbf{X}$ where $\mathcal{D}(\Psi)$ is open, non-empty, convex and $\mathbf{X}$ is a separable (potentially infinite dimensional) Hilbert-space.

(i) We assume that $\mathcal{M} := \Psi(\mathcal{D}(\Psi))$ is a n_*-dimensional *submanifold* in $\mathbf{X}$:

- Let for all $\vec{p} = (p_i)_{i=1}^{n_*} \in \mathcal{D}(\Psi)$ denote with $\Psi'(\vec{p})$ the Fréchet-derivative of Ψ:

$$\begin{aligned} &\Psi'(\vec{p}) : \vec{P} \to \mathbf{X}, \\ &\vec{q} = (q_i)_{i=1}^{n_*} \mapsto \left(\partial_{p_i} \Psi(\vec{p})\right)_{i=1,\dots,n_*} \vec{q}\,. \end{aligned}$$

  Here $\left(\partial_{p_i} \Psi(\vec{p})\right)_{i=1,\dots,n_*}$ denotes the vector of functions consisting of all partial derivatives with respect to $\vec{p}$. In differential geometry notation this coincides with the *tangential mapping* $T_{\vec{p}}\Psi$. However, the situation is slightly different here because $\mathbf{X}$ can be infinite dimensional.
- The *representation mapping* of the derivative

$$\begin{aligned} \Psi' : \mathcal{D}(\Psi) &\to \mathbf{X}^{n_*}, \\ \vec{p} &\mapsto \left(\partial_{p_i} \Psi(\vec{p})\right)_{i=1,\dots,n_*}. \end{aligned}$$

  has always the same rank n_* in $\mathcal{D}(\Psi)$, meaning that all elements of $\partial_{\vec{p}} \Psi(\vec{p})$ are linearly independent. This assumption means, in particular, that Ψ is an *immersion* and $\mathcal{M}$ is a submanifold.

(ii) We define

$$\begin{aligned} P_{\vec{p}} : \mathbf{X} \to \mathbf{X}_{\vec{p}} &:= \operatorname{span}\left\{\partial_{p_i} \Psi(\vec{p}) : i = 1, \dots, n_*\right\}, \\ \mathbf{x} \mapsto P_{\vec{p}}\,\mathbf{x} &:= \operatorname{argmin}\left\{\|\mathbf{x}_1 - \mathbf{x}\|_{\mathbf{X}} : \mathbf{x}_1 \in \mathbf{X}_{\vec{p}}\right\} \end{aligned} \tag{3.7}$$

as the projection from

$$\mathbf{X} = \mathbf{X}_{\vec{p}} \dot{+} \mathbf{X}_{\vec{p}}^{\perp}$$

onto $\mathbf{X}_{\vec{p}}$, which is well-defined by the closedness of the finite dimensional subspace $\mathbf{X}_{\vec{p}}$.
Next we define the inverse of $\Psi'(\vec{p})$ on $\mathbf{X}_{\vec{p}}$:

$$\begin{aligned} &\Psi'(\vec{p})^{-1} : \operatorname{span}\left\{\partial_{p_i} \Psi(\vec{p}) : i = 1, \dots, n_*\right\} \to \vec{P}, \\ &\mathbf{x} = \sum_{i=1}^{n_*} x_i \partial_{p_i} \Psi(\vec{p}) \mapsto (x_i)_{i=1}^{n_*} \end{aligned}$$

which is extended to $\mathbf{X}$ as follows

$$\begin{aligned} &\Psi'(\vec{p})^{\dagger} : \mathbf{X} = \mathbf{X}_{\vec{p}} \dot{+} \mathbf{X}_{\vec{p}}^{\perp} \to \vec{P}, \\ &\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \mapsto \Psi'(\vec{p})^{-1} \mathbf{x}_1 \end{aligned} \tag{3.8}$$

which are both well-defined because we assume that Ψ is an immersion. Note that x_i, $i = 1, \ldots, n_*$ are not necessarily the coordinates with respect to an orthonormal system in span $\{\partial_{p_i}\Psi(\vec{p}\,) : i = 1, \ldots, n_*\}$. We also note that a-priori we do not know that this is a Moore–Penrose inverse, but since this is proven below (see Lemma 3.4), we use the notation of a Moore–Penrose inverse already here.

(iii) Finally, we assume that the operators $\Psi'(\vec{p}\,)$ are locally bounded and locally Lipschitz-continuous in $\mathcal{D}(\Psi)$. That is

$$\begin{aligned} \left\| \Psi'(\vec{p}\,) - \Psi'(\vec{q}\,) \right\|_{\vec{P}\to\mathbf{X}} &\le C_L \left\| \vec{p} - \vec{q} \right\|_{\vec{P}} \\ \left\| \Psi'(\vec{p}\,) \right\|_{\vec{P}\to\mathbf{X}} &\le C_I \text{ for } \vec{p}\,, \vec{q} \in \mathcal{D}(\Psi). \end{aligned} \tag{3.9}$$

If Ψ satisfies these three properties we call it a *Lipschitz-differentiable immersion*.

The following lemma is proved by standard means:

Lemma 3.4 *For a Lipschitz-differentiable immersion*

- *the function $\Psi'(\vec{p}\,)^\dagger : \mathbf{X} \to \vec{P}$ is in fact the Moore–Penrose inverse of $\Psi'(\vec{p}\,)$ and*
- *for every point $\vec{p} \in \mathcal{D}(\Psi) \subseteq \vec{P}$ there exists a non-empty closed neighborhood where $\Psi'(\vec{p}\,)^\dagger$ is uniformly bounded and it is Lipschitz-continuous; That is*

$$\begin{aligned} \left\| \Psi'(\vec{p}\,)^\dagger - \Psi'(\vec{q}\,)^\dagger \right\|_{\mathbf{X}\to\vec{P}} &\le C_L \left\| \vec{p} - \vec{q} \right\|_{\vec{P}}, \\ \left\| \Psi'(\vec{p}\,)^\dagger \right\|_{\mathbf{X}\to\vec{P}} &\le C_I \text{ for } \vec{p}\,, \vec{q} \in \mathcal{D}(\Psi). \end{aligned} \tag{3.10}$$

- *Moreover, the operator $P_{\vec{p}}$ from Eq. (3.7) is bounded.*

Proof • We verify the four conditions Eq. (3.6) with

- $L = \Psi'(\vec{p}\,) : \vec{P} = \mathbb{R}^{n_*} \to \mathbf{X}$, $B = \Psi'(\vec{p}\,)^\dagger : \mathbf{X} \to \vec{P}$, with $\mathcal{D}(B) = \mathcal{D}(\Psi'(\vec{p}\,)^\dagger) = \mathbf{X}$ and
- $P : \vec{P} \to \vec{P}$ the zero-operator and $Q = P_{\vec{p}} : \mathbf{X} \to \mathbf{X}_{\vec{p}}$, the projection operator onto $\mathbf{X}_{\vec{p}}$ (see Eq. (3.7)).
- First we prove the third identity with $P = 0$ in Eq. (3.6): This follows from the fact that for all $\vec{q} = (q_i)_{i=1}^{n_*} \in \vec{P}$ we have

$$\Psi'(\vec{p}\,)^\dagger \Psi'(\vec{p}\,)\vec{q} = \Psi'(\vec{p}\,)^{-1} \left(\sum_{i=1}^{n_*} q_i \partial_{p_i} \Psi(\vec{p}\,) \right) = (q_i)_{i=1}^{n_*} = \vec{q}\,. \tag{3.11}$$

- For the forth identity we see that for all $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbf{X}$ there exists x_i, $i = 1, \ldots, n_*$ (because $\partial_{p_i}\Psi(\vec{p}\,)$, $i = 1, \ldots, n_*$ is a basis) such that

$$\mathbf{x} = \sum_{i=1}^{n_*} x_i \partial_{p_i} \Psi(\vec{p}\,) + \mathbf{x}_2 \quad \text{with } \mathbf{x}_2 \in \mathbf{X}_{\vec{p}}^{\perp}$$

and thus

$$P_{\vec{p}}\,\mathbf{x} = \sum_{i=1}^{n_*} x_i\,\partial_{p_i}\Psi(\vec{p}\,)$$

and therefore

$$\Psi(\vec{p}\,)^\dagger \mathbf{x} = (x_i)_{i=1}^{n_*} = \vec{x}. \tag{3.12}$$

Consequently, we have

$$\Psi'(\vec{p}\,)\Psi'(\vec{p}\,)^\dagger \mathbf{x} = \Psi'(\vec{p}\,)\vec{x} = P_{\vec{p}}\,\mathbf{x}. \tag{3.13}$$

– For the second identity we use that for all $\mathbf{x} \in \mathbf{X}$

$$\Psi'(\vec{p}\,)^\dagger\Psi'(\vec{p}\,)\Psi'(\vec{p}\,)^\dagger \mathbf{x} \underbrace{=}_{Eq.\ (3.12)} \Psi'(\vec{p}\,)^\dagger\Psi'(\vec{p}\,)\,(\vec{x}) \underbrace{=}_{Eq.\ (3.11)} \vec{x} \underbrace{=}_{Eq.\ (3.12)} \Psi(\vec{p}\,)^\dagger \mathbf{x}. \tag{3.14}$$

– The first identity is proven analogously.

Thus the Moore–Penrose inverse exists.

- For the proof of the boundedness we argue as follows: First note that

$$\begin{aligned}\left\|\Psi'(\vec{p}\,)^\dagger\right\|_{\mathbf{X}\to\vec{P}} &= \sup_{\{\mathbf{x}\in\mathbf{X}:\|\mathbf{x}\|_{\mathbf{X}}=1\}} \left\|\Psi'(\vec{p}\,)^\dagger\mathbf{x}\right\|_{\vec{P}} = \sup_{\{\mathbf{x}_1\in\mathbf{X}_{\vec{p}}:\|\mathbf{x}_1\|=1\}} \left\|\Psi'(\vec{p}\,)^{-1}\mathbf{x}_1\right\|_{\vec{P}} \\ &= \sup_{\{\mathbf{x}_1\in\mathbf{X}_{\vec{p}}:\|\mathbf{x}_1\|=1\}} \|\vec{x}_1\|_{\vec{P}}\,,\end{aligned}$$

where we used the representation

$$\begin{aligned}\mathbf{x}_1 &= \sum_{i=1}^{n_*} x_i\,\mathbf{e}_i \in \mathbf{X}_{\vec{p}} \ \text{ with } \mathbf{E} \\ &:= (\mathbf{e}_1 = \partial_{p_1}\Psi(\vec{p}\,), \ldots, \mathbf{e}_{n_*} = \partial_{p_{n_*}}\Psi(\vec{p}\,)).\end{aligned} \tag{3.15}$$

It remains to derive a uniform bound and the continuity for $\|\vec{x}_1\|_{\vec{P}}$ with respect to variations of $\vec{p}$. For this purpose we use the Gram–Schmidt inductive procedure to obtain the QR-decomposition of the function valued matrix $\mathbf{E}$.

– Because of the assumption Eq. (3.9), each vector $\mathbf{e}_j$, $j = 1, \ldots, n_*$, depends continuously of $\vec{p}$.
– The Gram–Schmidt procedure performs only additions, multiplications, division and inner products calculations of vectors of functions, which are all locally Lipschitz-continuous on the parameter $\vec{p}$ as long as the resulting output vectors of each Gram–Schmidt iteration are linearly independent. Thus by

chain-rule the Gram–Schmidt algorithm is Lipschitz-continuous with respect to $\vec{p}$ as long as the vectors $\mathbf{e}_j$, $j = 1, \dots, n_*$ are linearly dependent. Or in other word, Gram–Schmidt produces an orthogonal family $\hat{\mathbf{e}}_j$, $j = 1, \dots, n_*$, which are Lipschitz-continuously dependent on $\vec{p}$.

– Calculating the QR-decomposition of $\mathbf{E}$ we get

$$\mathbf{x}_1 = \vec{x}^T \mathbf{E} = \vec{x}^T Q R \text{ where } R = \begin{pmatrix} \langle \hat{\mathbf{e}}_1, \mathbf{e}_1 \rangle_{\mathbf{Y}} & \cdots & \cdots & \cdots & \langle \hat{\mathbf{e}}_1, \mathbf{e}_n \rangle_{\mathbf{Y}} \\ 0 & \langle \hat{\mathbf{e}}_2, \mathbf{e}_2 \rangle_{\mathbf{Y}} & \cdots & \cdots & \langle \hat{\mathbf{e}}_2, \mathbf{e}_n \rangle_{\mathbf{Y}} \\ \vdots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \\ 0 & \cdots & \cdots & 0 & \langle \hat{\mathbf{e}}_n, \mathbf{e}_n \rangle_{\mathbf{Y}} \end{pmatrix},$$

or in other word

$$\mathbf{x}_1 R^{-1} Q^T = \vec{x}^T.$$

The QR-algorithm is also Lipschitz-continuous with respect to perturbations in the entries of the matrix to be decomposed, that means with respect to $\vec{p}$, as long as the vectors do not get linearly dependent during the decomposition algorithm, which cannot happen because of our assumption on the linear dependence of our vectors $\mathbf{e}_i, i = 1, \dots, n_*$. Therefore Q and R are Lipschitz-continuous as well, and since R is a triangular matrix with diagonal entries different from 0, also R^{-1} is Lipschitz-continuous with respect to $\vec{p}$. This guarantees a local uniformly bound and local Lipschitz-continuity for Ψ.

- The boundedness of $P_{\vec{p}}$ is obvious because it is an orthogonal projector.

□

In the following we study a Gauss–Newton method for solving Eq. (1.3), where $\Psi : \mathcal{D}(\Psi) \subseteq \mathbb{R}^{n_*} \to \mathbf{X}$ is a Lipschitz-continuous immersion (see Definition 3.3), $F : \mathbf{X} \to \mathbf{Y}$ is bounded and $N = F \circ \Psi$.

Lemma 3.5 *Let $F : \mathbf{X} \to \mathbf{Y}$ be linear, bounded, with trivial nullspace and dense range. Moreover, let $\Psi : \mathcal{D}(\Psi) \subseteq \mathbb{R}^{n_*} \to \mathbf{X}$ be a Lipschitz-differentiable immersion; see Definition 3.3, and let $N = F \circ \Psi$.*

Note that by the definition of N, $\mathcal{D}(N) = \mathcal{D}(\Psi)$, and therefore for every $\vec{p} \in \mathcal{D}(N)$ the derivative of the operator N at a point $\vec{p}$ has a Moore–Penrose inverse $N'(\vec{p})^\dagger$, which satisfies:

- *Decomposition property of the Moore–Penrose inverse:*

$$N'(\vec{p})^\dagger \mathbf{z} = \Psi'(\vec{p})^\dagger F^{-1} \mathbf{z} \quad \text{for all } \vec{p} \in \mathcal{D}(N), \mathbf{z} \in \mathcal{R}(F) \subseteq \mathbf{Y}. \tag{3.16}$$

In particular this means that

$$N'(\vec{p})^\dagger N'(\vec{p}) = I \text{ on } \mathbb{R}^{n_*} \quad \text{and} \quad N'(\vec{p}) N'(\vec{p})^\dagger = Q|_{\mathcal{R}(F P_{\vec{p}})}, \tag{3.17}$$

where I denotes the identity operator on $\mathbb{R}^{n*}$ and $Q : \mathbf{Y} \to \overline{\mathcal{R}(FP_{\vec{p}})} \dot{+} \mathcal{R}(FP_{\vec{p}})^{\perp}$ is the orthogonal projection operator onto $\overline{\mathcal{R}(FP_{\vec{p}})}$. This means that $P \equiv 0$ in Eq. (3.6).

- *Generalized Newton–Mysovskii condition:*

$$\left\| N'(\vec{p})^{\dagger}(N'(\vec{q}+s(\vec{p}-\vec{q})-N'(\vec{q}))(\vec{p}-\vec{q}) \right\|_{\vec{P}} \leq sC_I C_L \|\vec{p}-\vec{q}\|_{\vec{P}}^2 \quad \vec{p},\vec{q} \in \mathcal{D}(N), s \in [0,1]. \tag{3.18}$$

We recall that the Lipschitz-constants C_I and C_L are defined in Eq. (3.9).

Proof First of all, we note that by the chain-rule

$$N'(\vec{p}) = F\Psi'(\vec{p}) \quad \text{for all } \vec{p} \in \mathcal{D}(\Psi) = \mathcal{D}(N).$$

To prove Eq. (3.16) we have to verify Eq. (3.6) with

$$L := N'(\vec{p}) = F\psi'(\vec{p}) : \vec{P} \to \mathbf{Y} \text{ and } B := \Psi'(\vec{p})^{\dagger}F^{-1} : \mathcal{R}(F) \subseteq \mathbf{Y} \to \vec{P}.$$

Note that since we assume that F has dense range we do not need to define and consider B on $\mathcal{R}(F) \dot{+} \underbrace{\mathcal{R}(F)^{\perp}}_{=\{0\}}$.

We state that with the notation of Eq. (3.6) we have for fixed $\vec{p}$:

$$\mathcal{D}(B) = \mathcal{D}(\Psi'(\vec{p})^{\dagger}F^{-1}) = \mathcal{R}(F) \quad \text{and} \quad \mathcal{R}(L) = \left\{F\Psi'(\vec{p})\vec{q} : \vec{q} \in \mathbb{R}^{n*}\right\} = \mathcal{R}(FP_{\vec{p}}).$$

Since Q is the orthogonal projector onto $\overline{\mathcal{R}(FP_{\vec{p}})}$ we see that for $\mathbf{z} = F\mathbf{x} = FP_{\vec{p}}\mathbf{x} + F(I-P_{\vec{p}})\mathbf{x}$ we have

$$Q\mathbf{z} = Q(FP_{\vec{p}}\mathbf{x} + F(I-P_{\vec{p}})(\mathbf{x})) = FP_{\vec{p}}\mathbf{x}. \tag{3.19}$$

Applying Lemma 3.4 and the invertability of F on the range of F shows that

$$\begin{aligned}
LBL &= F\Psi'(\vec{p})\Psi'(\vec{p})^{\dagger}F^{-1}F\Psi'(\vec{p}) \\
&= F\Psi'(\vec{p})\Psi'(\vec{p})^{\dagger}\Psi'(\vec{p}) \underbrace{=}_{Eq.\ (3.11)} F\Psi'(\vec{p}) = L, \\
BLB &= \Psi'(\vec{p})^{\dagger}F^{-1}F\Psi'(\vec{p})\Psi'(\vec{p})^{\dagger}F^{-1} \\
&= \Psi'(\vec{p})^{\dagger}\Psi'(\vec{p})\Psi'(\vec{p})^{\dagger}F^{-1} \underbrace{=}_{Eq.\ (3.14)} \Psi'(\vec{p})^{\dagger}F^{-1} = B, \\
BL &= \Psi'(\vec{p})^{\dagger}F^{-1}F\Psi'(\vec{p}) \underbrace{=}_{Eq.\ (3.11)} I - P = I \text{ on } \mathbb{R}^{n*}.
\end{aligned}$$

For the forth identity we take some $\mathbf{z} \in \mathcal{R}(F)$. Therefore, there exists some $\mathbf{x} \in \mathbf{X}$ such that $F\mathbf{x} = \mathbf{z}$ and thus

$$LB\mathbf{z} = F\Psi'(\vec{p}\,)\Psi'(\vec{p}\,)^\dagger F^{-1}\mathbf{z} = F\Psi'(\vec{p}\,)\Psi'(\vec{p}\,)^\dagger \mathbf{x} \underbrace{=}_{Eq.\ (3.13)} FP_{\vec{p}}\,\mathbf{x} \underbrace{=}_{Eq.\ (3.19)} Q\mathbf{z}.$$

Thus Eq. (3.6) holds and we know that $N'(\vec{p}\,)^\dagger$ from Eq. (3.16) is the Moore–Penrose of $N'(\vec{p}\,)$. The last two identities in fact show Eq. (3.17).

From Eq. (3.9) it follows that

$$\begin{aligned}
&\left\| N'(\vec{p}\,)^\dagger (N'(\vec{q}\, + s(\vec{p}\, - \vec{q}\,) - N'(\vec{q}\,))(\vec{p}\, - \vec{q}\,) \right\|_{\vec{P}} \\
&= \left\| \Psi'(\vec{p}\,)^\dagger F^{-1} (F\Psi'(\vec{q}\, + s(\vec{p}\, - \vec{q}\,) - F\Psi'(\vec{q}\,))(\vec{p}\, - \vec{q}\,) \right\|_{\vec{P}} \\
&= \left\| \Psi'(\vec{p}\,)^\dagger (\Psi'(\vec{q}\, + s(\vec{p}\, - \vec{q}\,) - \Psi'(\vec{q}\,))(\vec{p}\, - \vec{q}\,) \right\|_{\vec{P}} \\
&\leq C_I C_L s \, \|\vec{p}\, - \vec{q}\,\|_{\vec{P}}^2 \text{ for all } \vec{p}\,, \vec{q}\, \in \mathcal{D}(\Psi) = \mathcal{D}(N),
\end{aligned}$$

thus Eq. (3.18). □

We have now all ingredients to prove a local convergence rates result for a Gauss–Newton method, where the operator N is the composition of a linear bounded operator and a Lipschitz-differentiable immersions:

Theorem 3.6 *Let $F : \mathbf{X} \to \mathbf{Y}$ be linear, bounded, with trivial nullspace and dense range. Moreover, let $\Psi : \mathcal{D}(\Psi) \subseteq \vec{P} \to \mathbf{X}$ be a Lipschitz-differentiable immersion with $\mathcal{D}(\Psi)$ open, non-empty, and convex. Moreover, $N = F \circ \Psi : \mathcal{D}(\Psi) \to \mathbf{Y}$. We assume that there exist $\vec{p}^{\,\dagger} \in \mathcal{D}(\Psi)$ that satisfies*

$$N(\vec{p}^{\,\dagger}) = \mathbf{y}. \tag{3.20}$$

Moreover, we assume that there exists $\vec{p}^{\,0} \in \mathcal{D}(\Psi)$, which satisfies Eq. (3.2). Then, the iterates of the Gauss–Newton iteration,

$$\vec{p}^{\,k+1} = \vec{p}^{\,k} - N'(\vec{p}^{\,k})^\dagger (N(\vec{p}^{\,k}) - \mathbf{y}) \quad k \in \mathbb{N}_0 \tag{3.21}$$

are well-defined elements in $\overline{\mathcal{B}(\vec{p}^{\,0}, \rho)}$ and converge quadratically to $\vec{p}^{\,\dagger}$.

Proof First of all note, that $\mathcal{D}(\Psi) = \mathcal{D}(N)$ since F is defined all over $\mathbf{X}$.

Let $\rho = \left\| \vec{p}^{\,\dagger} - \vec{p}^{\,0} \right\|_{\vec{P}}$: We prove by induction that $\vec{p}^{\,k} \in \overline{\mathcal{B}(\vec{p}^{\,\dagger}; \rho)}$ for all $k \in \mathbb{N}_0$.

- For $k = 0$ the assertion is satisfied by assumption Eq. (3.2).
- Let $\vec{p}^{\,k} \in \overline{\mathcal{B}(\vec{p}^{\,\dagger}; \rho)}$. Using the first condition of Eq. (3.6), which a Moore–Penrose inverse satisfies, we see that

$$N'(\vec{p}^{\,k})N'(\vec{p}^{\,k})^\dagger N'(\vec{p}^{\,k})(\vec{p}^{\,k+1} - \vec{p}^{\,\dagger}) = N'(\vec{p}^{\,k})(\vec{p}^{\,k+1} - \vec{p}^{\,\dagger}).$$

The definition of Gauss–Newton method, Eq. (3.21), and Eq. (3.20) then imply that

$$N'(\vec{p}^{\,k})(\vec{p}^{\,k+1}-\vec{p}^{\,\dagger})=N'(\vec{p}^{\,k})N'(\vec{p}^{\,k})^{\dagger}(N(\vec{p}^{\,\dagger})-N(\vec{p}^{\,k})-N'(\vec{p}^{\,k})(\vec{p}^{\,\dagger}-\vec{p}^{\,k})),$$

and consequently, using the third identity of Eq. (3.6) (note that under the assumptions of this theorem $P = 0$, see the proof prior to Eq. (3.11)), the second identity of Eq. (3.6) and that F is injective, we get

$$\begin{aligned}\vec{p}^{\,k+1}-\vec{p}^{\,\dagger} &= N'(\vec{p}^{\,k})^{\dagger}N'(\vec{p}^{\,k})(\vec{p}^{\,k+1}-\vec{p}^{\,\dagger})\\ &= N'(\vec{p}^{\,k})^{\dagger}(N(\vec{p}^{\,\dagger})-N(\vec{p}^{\,k})-N'(\vec{p}^{\,k})(\vec{p}^{\,\dagger}-\vec{p}^{\,k}))\\ &= \Psi'(\vec{p}^{\,k})^{\dagger}(\Psi(\vec{p}^{\,\dagger})-\Psi(\vec{p}^{\,k})-\Psi'(\vec{p}^{\,k})(\vec{p}^{\,\dagger}-\vec{p}^{\,k})).\end{aligned}$$

From the Newton–Mysovskii condition Eqs. (3.18) and (3.2) it then follows that

$$\begin{aligned}&\left\|\vec{p}^{\,k+1}-\vec{p}^{\,\dagger}\right\|_{\vec{P}} \le \frac{C_I C_L}{2}\left\|\vec{p}^{\,k}-\vec{p}^{\,\dagger}\right\|_{\vec{P}}^2 \le \frac{C_I C_L \rho}{2}\left\|\vec{p}^{\,k}-\vec{p}^{\,\dagger}\right\|_{\vec{P}} < \left\|\vec{p}^{\,k}-\vec{p}^{\,\dagger}\right\|_{\vec{P}}\\ &\text{or } \left\|\vec{p}^{\,k+1}-\vec{p}^{\,\dagger}\right\|_{\vec{P}} = \left\|\vec{p}^{\,k}-\vec{p}^{\,\dagger}\right\|_{\vec{P}} = 0.\end{aligned} \tag{3.22}$$

This, in particular shows that $\vec{p}^{\,k+1} \in \mathcal{B}(\vec{p}^{\,\dagger};\rho)$, thus the well-definedness of the Gauss–Newton iterations in the closed ball.

- Using Eq. (3.22) we then get, since $h = C_I C_L \rho/2 < 1$, that

$$\left\|\vec{p}^{\,k+1}-\vec{p}^{\,\dagger}\right\|_{\vec{P}} \le h^{k+1}\left\|\vec{p}^{\,0}-\vec{p}^{\,\dagger}\right\|_{\vec{P}} \le h^{k+1}\rho,$$

  which converges to 0 for $k \to \infty$.
- Convergence and the first inequality of Eq. (3.22) imply quadratic convergence.

□

Remark 3.7 Based on the assumption of an immersion we have shown in Lemma 3.5 that $\Psi(\vec{p}\,)^{\dagger}F^{-1}$ is the Moore–Penrose inverse of $N = F\Psi(\vec{p}\,)$. In order to prove (quadratic) convergence of Gauss–Newton methods one only requires an *outer inverse* (see Notation 3.2). Following [38] (see also [19]) the analysis of Gauss–Newton method could be based on *outer inverses*, which is more general than for the Moore–Penrose inverse (compare Eqs. (3.4) and (3.6)). However, it is nice to actually see that $N'(\vec{p}\,)^{\dagger}$ is a Moore–Penrose inverse, which is the novelty compared to the analysis of [38]. For excellent expositions on Kantorovich and Mysovskii theory see [31, 41, 47]—here we replace the Newton–Mysovskii conditions by properties of an immersion. For aspects related to Newton methods for singular points see [11, 17]. For applications of generalized inverses in nonlinear analysis see [35, 36].

3.3 Neural networks

We want to apply the decomposition theory to Gauss–Newton methods for solving Eq. (1.3), where Ψ is a *shallow neural network operator*.

Definition 3.8 (*Shallow neural network operator*) Let $N_1 \in \mathbb{N}$ be fixed. We consider the operator

$$\Psi : \vec{P} := \mathbb{R}^{N_1} \times \mathbb{R}^{n\times N_1} \times \mathbb{R}^{N_1} \to C^1([0,1]^n) \subseteq \mathbf{X} := L^2([0,1]^n),$$
$$(\vec{\alpha}, \mathbf{w}, \vec{\theta}) \mapsto \left(\vec{x} \to \sum_{j=1}^{N_1} \alpha_j \sigma\left(\mathbf{w}_j^T \vec{x} + \theta_j\right)\right) \tag{3.23}$$

where $\alpha_j, \theta_j \in \mathbb{R}$ and $\vec{x}, \mathbf{w}_j \in \mathbb{R}^n$.

Here $\sigma : \mathbb{R} \to \mathbb{R}$ is called *activation function* (see Example 3.9 below).

Note, that with our previous notation, for instance in Definition 3.3, we have $n_* = (n+2) * N_1$.

We summarize the notation, because it is quite heavy:

(i) $\vec{\cdot}$ denotes a vector in $\mathbb{R}^n$ or $\mathbb{R}^{N_1}$,
(ii) $\mathbf{w}$ denotes a matrix: The only exception is Definition 3.10, where it is a general tensor. $\mathbf{w}_j$ denotes a vector, aside from Definition 3.10, where it is again a tensor.

Example 3.9 (Examples of activation functions) documented in the literature:

- the *sigmoid function*, defined by

$$\sigma(t) = \frac{1}{1 + \mathrm{e}^{-\frac{1}{\varepsilon}t}} \quad \text{for all } t \in \mathbb{R}. \tag{3.24}$$

  Note, we omit the ε dependence for notational convenience.
- The *hyperbolic tangent*

$$t \to \tanh(t) = \frac{\mathrm{e}^{2t} - 1}{\mathrm{e}^{2t} + 1}. \tag{3.25}$$

- The *ReLU* activation function,

$$\sigma(t) = \max\{0, t\} \quad \text{for all } t \in \mathbb{R}. \tag{3.26}$$

- The *step function*, which is the pointwise limit of the sigmoid function, with respect to $\varepsilon \to 0$,

$$\sigma(t) = \begin{cases} 0 & \text{for } t < 0, \\ \frac{1}{2} & \text{for } t = 0, \ t \in \mathbb{R}. \\ 1 & \text{for } t > 0. \end{cases} \tag{3.27}$$

In this paper ReLU and step activation functions are not used.

We only consider shallow neural networks in contrast to *deep neural networks*, which consist of several layers of shallow neural networks (see for instance [27]):

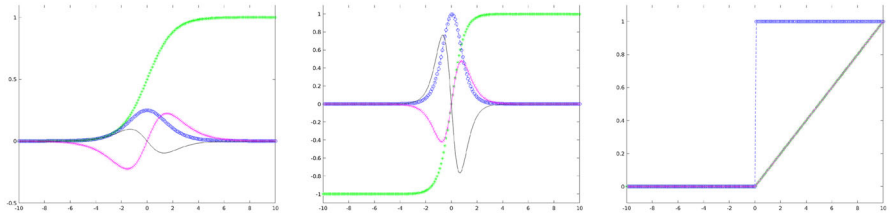

Fig. 1 Three different activation functions: Sigmoid, tanh and ReLU. O-th derivative with green $*$, first derivative with blue $-\circ$ and with magenta $\times$ is the derivative multiplied by x, 2nd derivative with continuous black line. The ReLU function is scaled by a factor $1/10$ and the derivative is plotted in original form, derivative times x is again scaled by $1/10$ for visualization purposes; 2nd derivative is of course omitted

Definition 3.10 (Deep neural networks) Let

$$\vec{P}_l := \mathbb{R}^{N_l} \times \mathbb{R}^{n \times N_l} \times \mathbb{R}^{N_l} \quad \text{for } l = 1, \dots, L \text{ and } \vec{P} := \prod_{l=1}^{L} \vec{P}_l.$$

Then a deep neural network consisting of L layers is written as

$$\begin{aligned} &\Psi : \vec{P} \to L^2([0,1]^n), \\ &(\vec{\alpha}_l, \mathbf{w}_l, \vec{\theta}_l)_{l=1}^L \mapsto \left(\vec{x} \to \sum_{j_L=1}^{N_L} \alpha_{j_L,L} \sigma_{\varepsilon_L,L} \left(p_{j_L,L} \left(\sum_{j_{L-1}=1}^{N_{L-1}} \cdots \left(\sum_{j_1=1}^{N_1} \alpha_{j_1,1} \sigma_{\varepsilon_1,1} \left(p_{j_1,1}(\vec{x}) \right) \right) \right) \right) \right), \end{aligned} \tag{3.28}$$

where

$$p_{j,l}(\vec{x}) = \mathbf{w}_{j,l}^T \vec{x} + \theta_{j,l} \text{ with } \alpha_{j,l}, \theta_{j,l} \in \mathbb{R} \text{ and } \vec{x}, \mathbf{w}_{j,l} \in \mathbb{R}^n \text{ for all } l = 1, \dots, L.$$

Note that the values ε_k, $k = 1, \dots, L$ can be chosen differently for activation functions at different levels of the deep network (cf. Eq. (3.24)).

The success of neural network is due to the universal approximation properties, proven for the first time in [10, 28]. The universal approximation result states that shallow neural networks are universal, that is, that each continuous function can be approximated arbitrarily well by a neural network function. We review this result now.

Theorem 3.11 ([27]) *In dependence of the smoothness of the activation function σ there exist two classes of results.*

- *Theorem 2 from* [27]*: Let $\sigma : \mathbb{R} \to \mathbb{R}^+$ be a* **continuous, bounded and non-constant** *function. Then, for every function $g \in C(\mathbb{R}^n)$ and every $\nu > 0$, there exists a function*

$$\vec{x} \to G(\vec{x}) = \sum_{j=1}^{N_1} \alpha_j \sigma(\mathbf{w}_j^T \vec{x} + \theta_j) \quad \textit{with } N \in \mathbb{N}, \alpha_j, \theta_j \in \mathbb{R}, \mathbf{w}_j \in \mathbb{R}^n, \tag{3.29}$$

satisfying

$$|G(\vec{x}) - g(\vec{x})| < \nu \text{ uniformly for all compact subsets } K \subseteq \mathbb{R}^n.$$

- *Theorem 1 from* [27]: *Let* $\sigma : \mathbb{R} \to \mathbb{R}^+$ *be* **unbounded and non-constant**. *Then for every measure* μ *on* $\mathbb{R}^n$ *and every constant* $\nu > 0$ *and* $p \geq 1$ *there exists a function* G *of the form Eq.* (3.29) *that satisfies*

$$\int_{\mathbb{R}^n} |G(\vec{x}) - g(\vec{x})|^p d\mu(\vec{x}) < \nu.$$

The first result applies for instance to the sigmoid and hyperbolic tangent function (see Eqs. (3.24) and (3.25)). The second result applies to the ReLU function (see Eq. (3.26)). In particular all approximation properties also hold on the compact set $[0, 1]^n$, which we are considering.

3.4 Newton–Mysovskii condition with shallow neural networks coders

In the following we verify Newton–Mysovskii conditions for Ψ being the encoder of Eq. (3.23). First we calculate the first and second derivatives of Ψ with respect to $\vec{\alpha}$, $\mathbf{w}$ and $\vec{\theta}$. The computations can be performed for deep neural network encoders as defined in Eq. (3.28) in principle analogously, but are technically and notationally more complicated. To make the notation consistent we define

$$\vec{p} := (\vec{\alpha}, \mathbf{w}, \vec{\theta}) \in \mathbb{R}^{N_1} \times \mathbb{R}^{n \times N_1} \times \mathbb{R}^{N_1} = \mathbb{R}^{n_*}.$$

Lemma 3.12 *Let* $\sigma : \mathbb{R} \to \mathbb{R}$ *be a two times differentiable function with uniformly bounded function values and first, second order derivatives, such as the sigmoid, hyperbolic tangent functions* (*see Fig.* 1).[1] *Then, the derivatives of* Ψ *with respect to the coefficients* $\vec{p}$ *are given by the following formulas:*

- *Derivative with respect to* α_s, $s = 1, \ldots, N_1$*:*

$$\frac{\partial \Psi}{\partial \alpha_s}[\vec{p}\,](\vec{x}) = \sigma\left(\sum_{i=1}^{n} w_s^i x_i + \theta_s\right) \quad \text{for } s = 1, \ldots, N_1. \tag{3.30}$$

- *Derivative with respect to* w_s^t *where* $s = 1, \ldots, N_1$, $t = 1, \ldots, n$*:*

$$\frac{\partial \Psi}{\partial w_s^t}[\vec{p}\,](\vec{x}) = \sum_{j=1}^{N_1} \alpha_j \sigma'\left(\sum_{i=1}^{n} w_j^i x_i + \theta_j\right) \delta_{s=j} x_t = \alpha_s \sigma'\left(\sum_{i=1}^{n} w_s^i x_i + \theta_s\right) x_t. \tag{3.31}$$

[1] This assumption is actually too restrictive, and only used to see that $\Psi \in L^2([0, 1]^n)$.

- *Derivative with respect to θ_s where $s = 1, \dots, N_1$:*

$$\frac{\partial \Psi}{\partial \theta_s}[\vec{p}\,](\vec{x}) = \sum_{j=1}^{N_1} \alpha_j \sigma'\left(\sum_{i=1}^{n} w_j^i x_i + \theta_j\right)\delta_{s=j} = \alpha_s \sigma'\left(\sum_{i=1}^{n} w_s^i x_i + \theta_s\right). \tag{3.32}$$

Note, that all the derivatives above are functions in $\mathbf{X} = L^2([0,1]^n)$. In particular, maybe in a more intuitive way, we have

$$D\Psi[\vec{p}\,](\vec{x})\vec{h} = \left(\frac{\partial \Psi}{\partial \vec{\alpha}}[\vec{p}\,](\vec{x})\ \frac{\partial \Psi}{\partial \mathbf{w}}[\vec{p}\,](\vec{x})\ \frac{\partial \Psi}{\partial \vec{\theta}}[\vec{p}\,](\vec{x})\right)^T \vec{h} \text{ for all } \vec{h} = \begin{pmatrix} \vec{h}_{\vec{\alpha}} \\ \mathbf{h}_{\mathbf{w}} \\ \vec{h}_{\vec{\theta}} \end{pmatrix} \in \mathbb{R}^{n_*} \text{ and } \vec{x} \in \mathbb{R}^n. \tag{3.33}$$

Moreover, let $s_1, s_2 = 1, \dots, N_1$, $t_1, t_2 = 1, \dots, n$, then we have in a formal way:

$$\begin{aligned}
\frac{\partial^2 \Psi}{\partial \alpha_{s_1} \partial \alpha_{s_2}}(\vec{x}) &= 0, \\
\frac{\partial^2 \Psi}{\partial \alpha_{s_1} \partial w_{s_2}^{t_1}}(\vec{x}) &= \sigma'\left(\sum_{i=1}^{n} w_{s_1}^i x_i + \theta_{s_1}\right) x_{t_1} \delta_{s_1=s_2}, \\
\frac{\partial^2 \Psi}{\partial \alpha_{s_1} \partial \theta_{s_2}}(\vec{x}) &= \sigma'\left(\sum_{i=1}^{n} w_{s_1}^i x_i + \theta_{s_1}\right) \delta_{s_1=s_2}, \\
\frac{\partial^2 \Psi}{\partial w_{s_1}^{t_1} \partial w_{s_2}^{t_2}}(\vec{x}) &= \alpha_{s_1} \sigma''\left(\sum_{i=1}^{n} w_{s_1}^i x_i + \theta_{s_1}\right) x_{t_1} x_{t_2} \delta_{s_1=s_2}, \\
\frac{\partial^2 \Psi}{\partial w_{s_1}^{t_1} \partial \theta_{s_2}}(\vec{x}) &= \alpha_{s_1} \sigma''\left(\sum_{i=1}^{n} w_{s_1}^i x_i + \theta_{s_1}\right) x_{t_1} \delta_{s_1=s_2}, \\
\frac{\partial^2 \Psi}{\partial \theta_{s_1} \partial \theta_{s_2}}(\vec{x}) &= \alpha_{s_1} \sigma''\left(\sum_{i=1}^{n} w_{s_1}^i x_i + \theta_{s_1}\right) \delta_{s_1=s_2},
\end{aligned} \tag{3.34}$$

where $\delta_{a=b} = 1$ if $a = b$ and 0 else, that is the Kronecker-delta.

The notation of directional derivatives with respect to parameters might be confusing. Note, that for instance $\frac{\partial \Psi}{\partial \theta_s}[\vec{p}\,](\vec{x})$ denotes a directional derivative of the functional Ψ with respect to the variable θ_s and this derivative is a function, which depends on $\vec{x} \in \mathbb{R}^n$.

Remark 3.13 • In particular Eq. (3.34) shows that

$$\left(\vec{h}_{\vec{\alpha}}\ \mathbf{h}_{\mathbf{w}}\ \vec{h}_{\vec{\theta}}\right) D^2\Psi[\vec{p}\,](\vec{x}) \begin{pmatrix} \vec{h}_{\vec{\alpha}} \\ \mathbf{h}_{\mathbf{w}} \\ \vec{h}_{\vec{\theta}} \end{pmatrix} \text{ is continuously dependent on } \vec{p} \text{ for fixed } \vec{x}. \tag{3.35}$$

- We emphasize that under the assumptions of Lemma 3.12 the linear space (for fixed $\vec{p}$)

$$\mathcal{R}(D\Psi[\vec{p}\,]) = \left\{ D\Psi[\vec{p}\,]\vec{h} : \vec{h} = (\vec{h}_{\vec{\alpha}}, \mathbf{h}_{\mathbf{w}}, \vec{h}_{\vec{\theta}}) \in \mathbb{R}^{(n+2)*N_1} \right\} \subseteq L^2([0,1]^n).$$

- In order to prove convergence of the Gauss–Newton method, Eq. (3.21), by applying Theorem 3.1, we have to prove that Ψ is a Lipschitz-continuous immersion. Below we lack proving one important property so far, namely, that

$$\partial_k \Psi[\vec{p}\,], \quad k = 1, \ldots, n_* = (n+2) * N_1 \tag{3.36}$$

  are linearly independent functions. In this paper, this will remain open as a conjecture, and the following statements are valid modulo this conjecture.

In the following we survey some results on linear independence with respect to the coefficients $\vec{\alpha}, \mathbf{w}, \vec{\theta}$ of the functions $\vec{x} \to \sigma\left(\sum_{i=1}^n w_s^i x_i + \theta_s\right)$, which match the functions $\vec{x} \to \frac{\partial \Psi}{\partial \alpha_s}[\vec{p}\,](\vec{x})$, that is with respect to the first N_1 variables of the neural network operator.

3.5 Linear independence of activation functions and its derivatives

The universal approximation results from for instance [10, 27, 28] do not allow to conclude that neural networks function as in Eq. (3.23) are linearly independent. Linear independence is a non-trivial research question: We recall a result from [32] from which linear independence of a shallow neural network operator, as defined in Eq. (3.23), can be deduced for a variety of activator functions. Similar results on linear independence of shallow network functions based on sigmoid activation functions have been stated in [29, 48], but the discussion in [32] raises questions on the completeness of the proofs. In [32] it is stated that all activation functions from the *Pytorch library* [42] are linearly independent with respect to almost all parameters $\mathbf{w}$ and θ.

Theorem 3.14 ([32]) *For all activation functions* HardShrink, HardSigmoid, HardTanh, HardSwish, LeakyReLU, PReLU, ReLU, ReLU6, RReLU, SoftShrink, Threshold, LogSigmoid, Sigmoid, SoftPlus, Tanh, and TanShrink *and the* PyTorch *functions* CELU, ELU, SELU *the shallow neural network functions Eq.* (3.23) *formed by randomly generated vectors* $(\mathbf{w}, \vec{\theta})$ *are linearly independent.*

Remark 3.15 (i) Theorem 3.14 states that the functions $\frac{\partial \Psi}{\partial \alpha_s}$ (taking into account Eq. (3.30)) are linearly independent for *almost all* parameters $(\mathbf{w}, \vec{\theta}) \in \mathbb{R}^{n \times N_1} \times$

$\mathbb{R}^{N_1}$. In other words, the first block of the matrix is $D\Psi$ in Eq. (3.33) consists of functions, which are linearly independent for almost all parameters $(\mathbf{w}, \vec{\theta})$. For our results to hold we need on top that the functions $\frac{\partial \Psi}{\partial w_s^t}$ and $\frac{\partial \Psi}{\partial \theta_s}$ from the second and third block (see Eq. (3.34)) are linearly independent within the blocks, respectively, and also across the blocks. So far this has not been proven but can be conjectured already from Fig. 1.

(ii) For the sigmoid function we have *obvious symmetries* because

$$\sigma'\left(\mathbf{w}_j^T \vec{x} + \theta_j\right) = \sigma'\left(-\mathbf{w}_j^T \vec{x} - \theta_j\right) \quad \text{for every } \mathbf{w}_j \in \mathbb{R}^n, \vec{\theta} \in \mathbb{R}^{N_1}, j \in \{1, \dots, N_1\}, \tag{3.37}$$

or in other words for the function Ψ from Eq. (3.23) we have according to Eq. (3.32) that

$$\frac{\partial \Psi}{\partial \theta_s}[\vec{\alpha}, \mathbf{w}, \vec{\theta}](\vec{x}) = \alpha_s \sigma'(\mathbf{w}_j^T \vec{x} + \theta_j) = \alpha_s \sigma'(-\mathbf{w}_j^T \vec{x} - \theta_j) = \frac{\partial \Psi}{\partial \theta_s}[\vec{\alpha}, -\mathbf{w}, -\vec{\theta}](\vec{x}), \tag{3.38}$$

or again in other words $\frac{\partial \Psi}{\partial \theta_s}[\vec{\alpha}, \mathbf{w}, -\vec{\theta}]$ and $\frac{\partial \Psi}{\partial \theta_s}[\vec{\alpha}, -\mathbf{w}, -\vec{\theta}]$ are linearly dependent.

Conjecture 3.16 We define by $\mathcal{D}(\Psi)$ a *maximal set of vectors* $(\vec{\alpha}, \mathbf{w}, \vec{\theta})$ such that the $n_* = (n+2) * N_1$ functions in $\vec{x}$

$$\vec{x} \to \frac{\partial \Psi}{\partial \alpha_s}[\vec{\alpha}, \mathbf{w}, \vec{\theta}](\vec{x}), \quad \vec{x} \to \frac{\partial \Psi}{\partial w_s^t}[\vec{\alpha}, \mathbf{w}, \vec{\theta}](\vec{x}),$$
$$\vec{x} \to \frac{\partial \Psi}{\partial \theta_s}[\vec{\alpha}, \mathbf{w}, \vec{\theta}](\vec{x}), \quad s = 1, \dots, N_1, t = 1, \dots, n,$$

are linearly independent. We assume that $\mathcal{D}(\Psi)$ is open and dense in $\mathbf{X} \in L^2([0,1])^2$. The later is guaranteed by Theorem 3.11. Recall the discussion above: The differentiation variables and the arguments coincide notationally, but are different objects.

Remark 3.17

- It can be conjectured that for every element from $\mathcal{D}(\Psi)$ only one element in $\mathbb{R}^{n_*}$ exists, which satisfies *obvious symmetries* such as formulated in Eq. (3.38). These "mirrored" elements are a set of measure zero in $\vec{P}$. We conjecture that this corresponds to the set of measure zero as stated in [32].
- Equation (3.34) requires that all components of the vector $\vec{\alpha}$ are non-zero. This means in particular that for "sparse solutions", with less that $n_* = (n+2) * N_1$ coefficients, convergence is not guaranteed, because of a locally degenerating submanifold. We consider the manifold given by the function

$$F : \mathbb{R}^2 \to \mathbb{R}^2. \quad \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} xy \\ x^2 + y^2 \end{pmatrix} \tag{3.39}$$

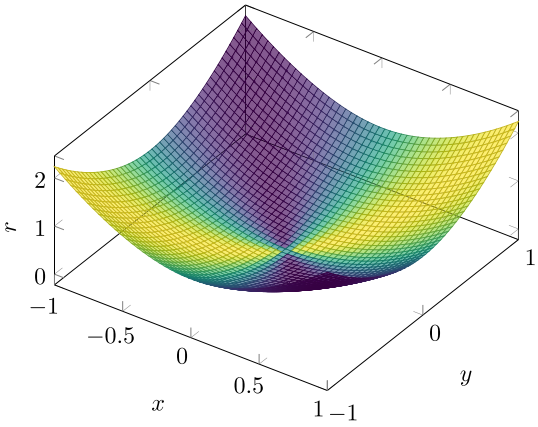

Fig. 2 The function F from Eq. (3.39). We have plotted $F(x, y)$ via its polar coordinates, I.e. $r = |F(x, y)|$ and $\theta = \tan^{-1}\left(\frac{xy}{x^2+y^2}\right)$. The colors correspond to identical angles θ (color figure online)

Then

$$\nabla F(x, y) = \begin{pmatrix} y & x \\ 2x & 2y \end{pmatrix}.$$

We have $\det \nabla F(x, y) = 2(y^2 - x^2)$, which vanishes along the diagonals in (x, y)-space. That is of the diagonals the function is locally a submanifold (see Fig. 2):

3.6 Local convergence of Gauss–Newton method with coding networks

In the following we prove a local convergence result for a Gauss–Newton method, for solving operator equations Eq. (1.3) where F is complemented by a shallow neural network coder Ψ. In order to apply Theorem 3.1 we have to verify that the shallow neural network operator (see Eq. (3.23)) is a Lipschitz-differentiable immersion.

Lemma 3.18 *Let $F : \mathbf{X} = L^2([0, 1]^n) \to \mathbf{Y}$ be linear, bounded, with trivial nullspace and closed range, and let σ be strictly monotonic (like sigmoid or hyperbolic tangent) and satisfy the assumptions of Lemma 3.12. Moreover, assume that Conjecture 3.16 holds. Then*

(i) *For every element $\vec{p} = (\vec{\alpha}, \mathbf{w}, \vec{\theta}) \in \mathbb{R}^{n_*}$ in the maximal set $\mathcal{D}(\Psi)$ (see Conjecture 3.16), $\mathcal{R}(D\Psi[\vec{p}\,])$ is a linear subspace of the space $\mathbf{X}$ of dimension $n_* = (n + 2) * N_1$.*

(ii) *There exists an open neighborhood $\mathcal{U} \subseteq \mathbb{R}^{(n+2)*N_1}$ of vectors $(\vec{\alpha}, \mathbf{w}, \vec{\theta})$ such that Ψ is a Lipschitz-differentiable immersion in $\mathcal{U}$.*

Proof • It is clear that for each fixed $\vec{p}$, $D\Psi[\vec{p}\,] \in L^2([0, 1]^n)$ because of the differentiability assumptions of σ, see Eq. (3.35). Conjecture 3.16 implies that $\mathcal{R}(D\Psi[\vec{p}\,])$ is a linear subspaces of $\mathbf{X}$ of dimension $(n+2)*N_1$ (note the elements are functions).

- $D^2\Psi[\vec{p}\,] : \mathbb{R}^{(n+2)*N_1} \to L^2([0,1]^n)$ is continuous (see Eq. (3.35)) since we assume that the activation function σ is twice differentiable. Now we consider a non-empty open neighborhood $\mathcal{U}$ of a vector $\vec{p}$, with a compact closure.
  - Then, from the continuity of $D^2\Psi$ with respect to $\vec{p}$, it follows that $D\Psi$ is a Frèchet-differentiable with Lipschitz-continuous derivative on $\mathcal{U}$. In particular this means that item (i) in Definition 3.3 holds. Moreover, Eq. (3.9) holds for Ψ'. That is, there exists constants C_L and C_I such that

$$\begin{aligned} &\left\| \Psi'(\vec{p}\,) - \Psi'(\vec{q}\,) \right\|_{\vec{P}\to\mathbf{Y}} \le C_L \left\| \vec{p} - \vec{q} \right\|_{\vec{P}} \quad \text{and} \\ &\left\| \Psi'(\vec{p}\,) \right\|_{\vec{P}\to\mathbf{Y}} \le C_I \text{ for } \vec{p}\,, \vec{q} \in \mathcal{D}(\Psi). \end{aligned} \tag{3.40}$$

□

Note that $\Psi'(p)^\dagger$ as defined in Eq. (3.8) is also uniformly bounded and Lipschitz-continuous as a consequence of Lemma 3.4.

Theorem 3.19 (Local convergence of Gauss–Newton method) *Let $F : \mathbf{X} = L^2([0,1]^n) \to \mathbf{Y}$ be a linear, bounded operator with trivial nullspace and dense range and let $N = F \circ \Psi$, where $\Psi : \mathcal{D}(\Psi) \subseteq \mathbb{R}^{(n+2)*N_1} \to \mathbf{X}$ is a shallow neural network operator generated by an activation function σ which satisfies the assumptions of Lemma 3.18 and Conjecture 3.16. Let $\vec{p}^{\,0} \in \mathcal{D}(\Psi)$ be the starting point of the Gauss–Newton iteration Eq. (3.21) and let $\vec{p}^{\,\dagger} \in \mathcal{D}(\Psi)$ be a solution of Eq. (3.20), which satisfy Eq. (3.2). Then the Gauss–Newton iterations are locally, that is if $\vec{p}^{\,0}$ is sufficiently close to $\vec{p}^{\,\dagger}$, and quadratically converging.*

Proof The proof is an immediate application of Lemma 3.18 to Theorem 3.6. □

Remark 3.20 We have shown that a nonlinear operator equation, where the operator is a composition of a linear compact operator and a shallow neural network operator, can be solved with a Gauss–Newton method with guaranteed local convergence in the parameter space.

4 Simple numerical experiments

In this section we present very simple, but hopefully illustrating numerical examples, where $N = F \circ \Psi$ with Ψ is a shallow neural network coder as in Eq. (3.23) with the sigmoid function from Eq. (3.24) with $\varepsilon = 1$,

$$x \in \mathbb{R} \to \sigma(x) = \frac{1}{1 - e^{-x}}.$$

We assumed in the analysis that the data $\mathbf{y}$ is attainable through the operator N (see Theorem 3.19) which means that we can write $\mathbf{y} = F(\Psi^\dagger)$ with $\Psi^\dagger = \Psi(\vec{p}^{\,\dagger})$, or in other words $\mathbf{y} = N(\vec{p}^{\,\dagger})$, such that Eq. (3.21) rewrites to

$$\vec{p}^{\,k+1} = \vec{p}^{\,k} - \Psi'(\vec{p}^{\,k})^\dagger(\Psi(\vec{p}^{\,k}) - \Psi^\dagger) \quad k \in \mathbb{N}_0; \tag{4.1}$$

Note that the operator F is annihilated because we assume it is injective. Therefore the Gauss–Newton finds the coefficients $\vec{p}^{\,\dagger}$ of the given neural network function $\Psi^\dagger$. In practice one would not work like this because the ill-posedness of inverting F hinders to calculate $\Psi^\dagger$ without regularization. Alternative strategies can be found in [30].

In the iterative reconstruction we represent a neural network function $\Psi(\vec{p}\,) : [0,1]^2 \to \mathbb{R}$ by a matrix of dimension 10×10. The vector $\vec{p}$ represents a shallow network with $N_1 = 2$ ansatz functions, thus has dimension 8. I.e.

$$\vec{p} = ((\alpha_1, \mathbf{w}_1, \theta_1); (\alpha_2, \mathbf{w}_2, \theta_2)) \quad \text{with } \mathbf{w}_i \in \mathbb{R}^2.$$

4.1 An example with the Gauss–Newton method

The first example concerns reconstructing the reconstruction of the coefficients of the 2D neural network function (see Eq. (3.29)) with

$$\vec{p}^{\,\dagger} = ((1.0, 1.0, 0.1, 0.1); (0.3, 0.1, 1.0, 0.8)).$$

That is,

$$\begin{aligned} \Psi^\dagger(\vec{x}) &= \Psi((1.0, 1.0, 0.1, 0.1); (0.3, 0.1, 1.0, 0.8))(\vec{x}) \\ &= 1.0\,\sigma((1.0, 0.1)^T\vec{x} + 0.1) + 0.3\,\sigma((0.1, 1.0)^T\vec{x} + 0.8), \end{aligned} \tag{4.2}$$

which is represented by the left image in the middle subimage of Fig. 3. The vector $\vec{p}^{\,\dagger}$ is therefore the ground truth and it is represented in the left image in the left subimage of Fig. 3.

The initialization vector $\vec{p}^{\,0} = ((0.8, 0.9, 0.05, 0.1), (0.7, 0.3, 0.5, 0.5)$ is plotted on the right hand side of the left subimage. The according 2D neural network functions are represented as the right images of the middle subimage.

The iterations of the Gauss–Newton iteration where stopped at $k_s + 1$ when the residual $\left\|\Psi(\vec{p}^{\,k_s+1}) - \Psi^\dagger\right\| < \delta$ with $\delta = 0.001$ (this is actually the stopping criterion in all examples). Plotted are the results for k_s, that is, $\vec{p}^{\,k_s}$ and $\Psi(\vec{p}^{\,k_s})$, which are represented in the middle of the two subimages on the left and in the middle of the row.

In this example the Gauss–Newton method terminates after 6 iterations. The right plot shows the residual decay $\left\|\Psi(\vec{p}^{\,k}) - \Psi^\dagger\right\|$ and the rate

$$\log\left(\frac{\left\|\Psi(\vec{p}^{\,k+1}) - \Psi^\dagger\right\|}{\left\|\Psi(\vec{p}^{\,k}) - \Psi^\dagger\right\|^2}\right), \tag{4.3}$$

which, if bounded by a constant, showing at least quadratic convergence (probably very much faster).

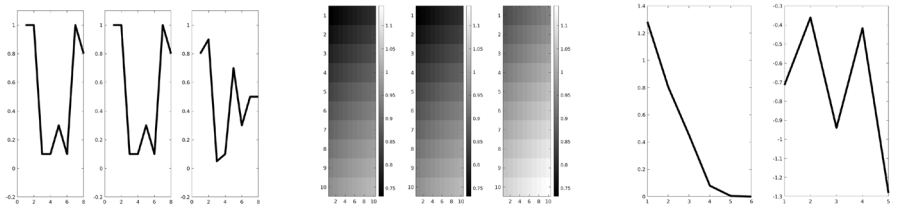

Fig. 3 Gauss–Newton method: Left: The neural network coefficients—ground truth ($\vec{p}^{\,\dagger}$), reconstruction ($\vec{p}^{\,k_S}$) and initial guess ($\vec{p}^{\,0}$) in the subimage. Middle: According neural network functions $\Psi^\dagger$, Ψ^{k_S}, Ψ^0. Right: residual error and rate over the number of iterations

4.2 A degenerated example with the Gauss–Newton method

The second example concerns the reconstruction of the coefficients of the 2D neural network function (see Eq. (3.29)) with

$$\vec{p}^{\,\dagger} = ((1.0, 1.0, 0.1, 0.1); (0, 0.1, 1.0, 0.8)).$$

That is,

$$\Psi^\dagger(\vec{x}) = \Psi((1.0, 1.0, 0.1, 0.1); (0, 0.1, 1.0, 0.8))(\vec{x}) = 1.0\sigma((1.0, 0.1)^T\vec{x} + 0.1), \tag{4.4}$$

which is represented by the left image in the middle subimage of Fig. 4. Note, because $\alpha_2 = 0$, the rest of the coefficients, $\mathbf{w}_2$ and θ_2 are irrelevant for representing Ψ. The vector $\vec{p}^{\,\dagger}$ is represented in the left image in the left subimage of Fig. 4. The result of the Gauss–Newton methods are represented in the middle of the two subimage in the left and the middle. The Gauss–Newton method terminates after 4 iterations (left plot in the right subimage). The right plot shows the rate from Eq. (4.3), which looks also faster than quadratic. However, note that the coefficients are not recovered, which is clear, because for $\alpha_2 = 0$, the remaining coefficients $\mathbf{w}_2$ and θ_2 can be selected arbitrary. We see that the first 5 coefficients can be reconstructed accurately, which are the coefficients $(\alpha_1, \mathbf{w}_1, \theta_1)$ and $\alpha_2 = 0$. This shows the necessity in Theorem 3.19 of the assumptions of Lemma 3.18, which in particular require that the coefficients of the $\vec{p}^{\,\dagger}$ do not vanish (see Remark 3.17). Clearly, if we have non-unique solutions, we should consider a set-valued convergence, which we have not in our analysis so far.

4.3 An examples with the Landweber method

We consider the same problem as in Sect. 4.1 but use for the numerical solution the *Landweber iteration*

$$\vec{p}^{\,k+1} = \vec{p}^{\,k} - \lambda\Psi'(\vec{p}^{\,k})^T(\Psi(\vec{p}^{\,k}) - \Psi^\dagger) \quad k \in \mathbb{N}_0. \tag{4.5}$$

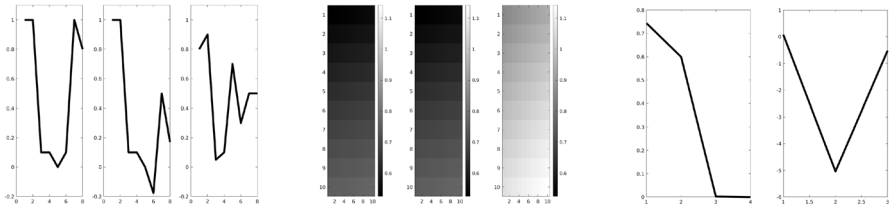

Fig. 4 Gauss–Newton method: Left: The neural network coefficients—ground truth, reconstruction, initial guess from left to right in the subimage. Middle: Neural network functions. Right: error and rate. What should be noted here is that the fifth coefficient is $\alpha_2 = 0$ in the ground truth and so as a consequence 6th to 8th coefficient are not uniquely determined. So it is a correct solution for $\vec{p}^{\,\dagger}$

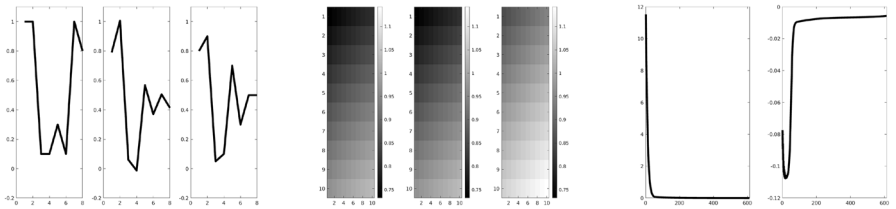

Fig. 5 Landweber method: Left: The neural network coefficients—ground truth, reconstruction, initial guess from left to right in the subimage. Middle: Neural network functions. Right: error and rate

The only formal difference is that instead of the Moore–Penrose inverse the transpose is used in the iteration. Moreover, a parameter λ had to be selected: A choice $\lambda = 1$ leads to divergence and a choice $\lambda = 0.02$ was sufficiently small to provide convergence. It was found by trial and error, but an appropriate choice is consistent with the literature (see for instance [30]). All other parameters where chosen as for the Gauss–Newton method. To satisfy the stopping criterion the Landweber iteration required about 500 iterations (Fig. 5).

Here we plotted the rate

$$\log\left(\frac{\left\|\Psi(\vec{p}^{\,k+1}) - \Psi^{\dagger}\right\|}{\left\|\Psi(\vec{p}^{\,k}) - \Psi^{\dagger}\right\|}\right), \tag{4.6}$$

which indicates at most linear convergence (see Fig. 5). Here an interesting observation is that the Landweber iteration is not converging to the same solution as the Gauss–Newton method in terms of parameters ($\vec{p}$), although the neural networks functions (Ψ) match quite well. Note however that Landweber iteration select solutions with negative coefficients, which might be due to the non-uniqueness of the neural network representation (see Remark 3.15).

We note that convergence of iterative methods, and in particular gradient descent methods, for large scale machine learning problems can be found for instance in [7].

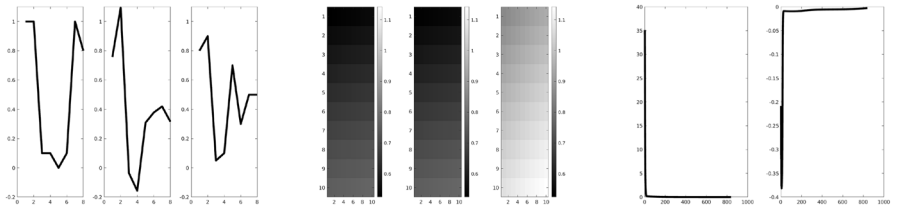

Fig. 6 Left: The neural network coefficients—ground truth, reconstruction, initial guess from left to right in the subimage. Middle: Neural network functions. Right: error and rate

4.4 A degenerated example with the Landweber method

The same observations as for the non-degenerated example Sect. 4.2 apply also here for the Landweber iteration (see Fig. 6).

Remark 4.1 Note that the Landweber iteration for solving Eq. (1.3) with a composed operator $N = F \circ \Psi$ reads as follows:

$$\vec{p}^{\,k+1} = \vec{p}^{\,k} - \lambda \Psi'(\vec{p}^{\,k})^T F^*(F\Psi(\vec{p}^{\,k}) - \mathbf{y}) \quad k \in \mathbb{N}_0. \tag{4.7}$$

Here F^* denotes the adjoint of F. In fact a pre-processing consisting of solving $\mathbf{y} = \Psi(\vec{p}^{\,\dagger})$ is not necessary. On the other hand a convergence analysis requires verification of a *weak-tangential cone condition* (see [18], which is not so adequate to Eq. (1.3) (see [30]).

Conclusion

We have shown that Gauss–Newton methods are efficient algorithms for solving linear inverse problems, where the solution can be encoded with a neural network. The convergence studies, however, are not complete, and are based on a conjecture on linear independence of activation functions and its derivatives.

Acknowledgements This research was funded in whole, or in part, by the Austrian Science Fund (FWF) P 34981—New Inverse Problems of Super-Resolved Microscopy (NIPSUM). For the purpose of open access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. Moreover, OS is supported by the Austrian Science Fund (FWF), with SFB F68 "Tomography Across the Scales", project F6807-N36 (Tomography with Uncertainties). The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged. BH is supported by the German Science Foundation (DFG) under the grant HO 1454/13-1 (Project No. 453804957).

Funding Open access funding provided by Austrian Science Fund (FWF).

Data availability The software and data are available by the first author on request.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Aspri, A., Frischauf, L., Korolev, Y., Scherzer, O.: Data driven reconstruction using frames and Riesz bases. In: Jadamba, B., Khan, A.A., Migórski, S., Sama, M. (eds.) Deterministic and Stochastic Optimal Control and Inverse Problems, pp. 303–318. CRC Press, Boca Raton (2021). https://doi.org/10.1201/9781003050575-13
2. Aspri, A., Korolev, Y., Scherzer, O.: Data driven regularization by projection. Inverse Probl. **36**(12), 125009 (2020). https://doi.org/10.1088/1361-6420/abb61b
3. Bakushinskii, A., Goncharskii, A.: Ill-Posed Problems: Theory and Applications. Kluwer Academic Publishers, Dordrecht (1994)
4. Bakushinskii, A., Goncharskii, A.: Iterative Methods for the Solution of Incorrect Problems. Nauka, Moscow (1989)
5. Blaschke, B.: Some Newton Type Methods for the Regularization of Nonlinear Ill-Posed Problems. Universitätsverlag Rudolf Trauner, Linz (1996)
6. Bora, A., Jalal, A., Price, E., Dimakis, A.G.: Compressed sensing using generative models. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 537–546. PMLR, New York (2017)
7. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Rev. **60**(2), 223–311 (2018). https://doi.org/10.1137/16m1080173
8. Braunsmann, J., Rajković, M., Rumpf, M., Wirth, B.P.: Learning low bending and low distortion manifold embeddings: theory and applications (2022). arxiv:2208.10193
9. Chavent, G., Kunisch, K.: Regularization in state space. Math. Model. Numer. Anal. **27**, 535–564 (1993)
10. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. **2**(4), 303–314 (1989). https://doi.org/10.1007/bf02551274
11. Decker, D.W., Keller, H.B., Kelley, C.T.: Convergence rates for Newton's method at singular points. SIAM J. Numer. Anal. **20**(2), 296–314 (1983). https://doi.org/10.1137/0720020
12. Deuflhard, P., Heindl, G.: Affine invariant convergence theorems for Newton's method and extensions to related methods. SIAM J. Numer. Anal. **16**(1), 1–10 (1979)
13. Deuflhard, P., Potra, F.A.: Asymptotic mesh independence of Newton–Galerkin methods via a refined Mysovskii theorem. SIAM J. Numer. Anal. **29**(5), 1395–1412 (1992). https://doi.org/10.1137/0729080
14. Duff, M., Campbell, N.D.F., Ehrhardt, M.J.: Regularising inverse problems with generative machine learning models (2021). arXiv:2107.11191
15. Engl, H.W., Hanke, M., Neubauer, A.: Regularization of inverse problems. Mathematics and its Applications, vol. 375. Kluwer Academic Publishers Group, Dordrecht. viii+321. ISBN:0-7923-4157-0 (1996)
16. Gorenflo, R., Hofmann, B.: On autoconvolution and regularization. Inverse Probl. **10**(2), 353–373 (1994). https://doi.org/10.1088/0266-5611/10/2/011
17. Grisvard, P.: Elliptic Problems in Nonsmooth Domains. Pitman, Boston (1985)
18. Hanke, M., Neubauer, A., Scherzer, O.: A convergence analysis of the Landweber iteration for nonlinear ill-posed problems. Numer. Math. **72**(1), 21–37 (1995). https://doi.org/10.1007/s002110050158
19. Häussler, W.: A Kantorovich-type convergence analysis for the Gauss–Newton-method. Numer. Math. **48**, 119–125 (1986)
20. Hein, T., Hofmann, B.: On the nature of ill-posedness of an inverse problem arising in option pricing. Inverse Probl. **19**(6), 1319–1338 (2003). https://doi.org/10.1088/0266-5611/19/6/006
21. Hofmann, B.: Mathematik Inverser Probleme. Teubner, Stuttgart (1999)

22. Hofmann, B.: On the degree of ill-posedness for nonlinear problems. J. Inverse Ill-Posed Probl. **2**, 61–76 (1994)
23. Hofmann, B. (ed.): Regularization for Applied Inverse and Ill-Posed Problems. Leipzig (1986). https://doi.org/10.1007/978-3-322-93034-7
24. Hofmann, B., Kaltenbacher, B., Pöschl, C., Scherzer, O.: A convergence rates result for Tikhonov regularization in Banach spaces with non-smooth operators. Inverse Probl. **23**(3), 987–1010 (2007). https://doi.org/10.1088/0266-5611/23/3/009
25. Hofmann, B., Scherzer, O.: Factors influencing the ill-posedness of nonlinear problems. Inverse Probl. **10**(6), 1277–1297 (1994). https://doi.org/10.1088/0266-5611/10/6/007
26. Hofmann, B., Tautenhahn, U.: On ill-posedness measures and space change in Sobolev scales. J. Anal. Appl. **16**(4), 979–1000 (1997). https://doi.org/10.4171/zaa/800
27. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Netw. **4**(2), 251–257 (1991). https://doi.org/10.1016/0893-6080(91)90009-t
28. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**, 359–366 (1989)
29. Huang, G.-B.: Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Trans. Neural Netw. **14**(2), 274–281 (2003). https://doi.org/10.1109/tnn.2003.809401
30. Kaltenbacher, B., Neubauer, A., Scherzer, O.: Iterative regularization methods for nonlinear ill-posed problems. Radon Series on Computational and Applied Mathematics, vol. 6. Walter de Gruyter, Berlin (2008). ISBN:978-3-11-020420-9. https://doi.org/10.1515/9783110208276
31. Kantorowitsch, L., Akilow, G.: Funktionalanalysis in normierten Räumen. Akademie Verlag, Berlin (1964)
32. Lamperski, A.: Neural network independence properties with applications to adaptive control. In: 2022 IEEE 61st Conference on Decision and Control (CDC) (2022). https://doi.org/10.1109/cdc51059.2022.9992994
33. Morozov, V.A.: Methods for Solving Incorrectly Posed Problems. Springer, New York (1984)
34. Nashed, M.Z.: A new approach to classification and regularization of ill-posed operator equations. Inverse Ill-Posed Probl. **4**, 53–75 (1987). https://doi.org/10.1016/b978-0-12-239040-1.50009-0
35. Nashed, M.Z.: Inner, outer, and generalized inverses in Banach and Hilbert spaces. Numer. Funct. Anal. Optim. **9**(3–4), 261–325 (1987). https://doi.org/10.1080/01630568708816235
36. Nashed, M.Z.: On the perturbation theory of generalized inverse operators in Banach spaces. In: Nashed, M.Z. (ed.) Functional Analysis Methods in Numerical Analysis, pp. 180–195. Springer, New York (1979)
37. Nashed, M. (ed.) Generalized Inverses and Applications. Academic Press [Harcourt Brace Jovanovich Publishers], New York, xiv+1054 (1976)
38. Nashed, M., Chen, X.: Convergence of Newton-like methods for singular operator equations using outer inverses. Numer. Math. **66**, 235–257 (1993)
39. Obmann, D., Schwab, J., Haltmeier, M.: Deep synthesis network for regularizing inverse problems. Inverse Probl. **37**(1), 015005 (2021). https://doi.org/10.1088/1361-6420/abc7cd
40. Ortega, J.M.: The Newton–Kantorovich theorem. Am. Math. Mon. **75**(6), 658 (1968). https://doi.org/10.2307/2313800
41. Ortega, J., Rheinboldt, W.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York (1970)
42. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, vol. 721. Curran Associates Inc. (2019)
43. Pöschl, C., Resmerita, E., Scherzer, O.: Discretization of variational regularization in Banach spaces. In: Inverse Problems 26.10 (2010), p. 105017. issn: 0266-5611. https://doi.org/10.1088/ 0266-5611/26/10/105017
44. Rifai, S., Mesnil, G., Vincent, P., Muller, X., Bengio, Y., Dauphin, Y., Glorot, X.: Higher order contractive auto-encoder. In: Machine Learning and Knowledge Discovery in Databases, pp. 645–660 (2011). https://doi.org/10.1007/978-3-642-23783-6_41
45. Scherzer, O., Grasmair, M., Grossauer, H., Haltmeier, M., Lenzen, F.: Variational methods in imaging. Applied Mathematical Sciences, vol. 167. Springer, New York (2009). ISBN:978-0-387-30931-6. https://doi.org/10.1007/978-0-387-69277-7

46. Schuster, T., Kaltenbacher, B., Hofmann, B., Kazimierski, K.S.: Regularization methods in Banach spaces. Radon Series on Computational and Applied Mathematics, vol. 10. De Gruyter, Berlin. xii+283 (2012). https://doi.org/10.1515/9783110255720
47. Schwetlick, H.: Numerische Lösung nichtlinearer Gleichungen, vol. 17. Mathematik für Naturwissenschaft und Technik [Mathematics for Science and Technology]. VEB Deutscher Verlag der Wissenschaften, Berlin, p. 346 (1979)
48. Tamura, S., Tateishi, M.: Capabilities of a four-layered feedforward neural network: four layers versus three. IEEE Trans. Neural Netw. **8**(2), 251–255 (1997). https://doi.org/10.1109/72.557662