Nonsmooth Implicit Differentiation for Machine Learning and Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

In view of training increasingly complex learning architectures, we establish a nonsmooth implicit function theorem with an operational calculus. Our result applies to most practical problems (i.e., definable problems) provided that a nonsmooth form of the classical invertibility condition is fulfilled. A major feature of our formula, based on conservative Jacobians, is its compatibility with algorithmic differentiation (e.g., backpropagation). We provide several applications of our results: training deep equilibrium networks, training neural nets with conic optimization layers, and hyperparameter tuning for nonsmooth Lasso-type models. To show the sharpness of our assumptions, we present numerical experiments showcasing the extremely pathological gradient dynamics one can encounter when applying implicit algorithmic differentiation without any hypothesis.

1 Introduction

Differentiable programming. The recent introduction of deep equilibrium networks [7], the increasing importance of bilevel programming (e.g., hyperparameter optimization) [41] and the ubiquity of differentiable programming (e.g., TensorFlow [1], PyTorch [40], JAX [16]) in modern optimization call for the development of a versatile theory of nonsmooth differentiation. Our focus is on nonsmooth implicit differentiation. There are currently two practices lying at the crossroads of mathematics and computer science: on the one hand the use of the standard smooth implicit function theorem "almost everywhere" [29, 28] and on the other hand the development of algorithmic differentiation tools [2, 3, 50]. The empirical use of the latter in the nonsmooth world has shown surprisingly efficient results [50], but the current theories cannot explain this success. We bridge this gap by providing nonsmooth implicit differentiation results and illustrating their impact on the training of neural networks and hyperparameter optimization.

Backpropagation: a formal differentiation approach. Let us consider z implicitly defined through $F(z(x)) = h(x)$ where F and h have full domain and adequate dimensions. How does autograd apply to evaluating the "derivative" of the implicitly defined function z? Regardless of differentiability or nonsmoothness, and provided that inversion is possible, one commonly uses (or dynamically approximates) this derivative by

$$(\mathrm{backprop}_F(z(x)))^{-1}\,\mathrm{backprop}_h x,$$

where $\mathrm{backprop}$ outputs the result of formal backpropagation, see e.g., [43]. This identity[1] is used to provide efficient training despite the fact that the rules of classical nonsmooth calculus are transgressed [7, 50]. Note that spurious outputs may be created by this approach, but on a

[1]The notation $\mathrm{backprop}_z$ instead of $\mathrm{backprop}(z)$ is indicative of the fact that $\mathrm{backprop}$ is an operator that does not act on functions themselves but rather on the program used to represent them, see [15].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

negligible set. Consider for example the simple implicit problem $x = f(z(x))$ where $f(z) := \tanh(z) + \mathrm{relu}(-z) + z - \mathrm{relu}(z)$, whose solution is $z(x) = \tanh x$. Yet applying the implicit differentiation framework of [7] using `JAX` library, as presented in [50], provides inconsistency of the derivative at the origin, see Figure 1. As mentioned above, despite these unpredictable outputs, propagating derivatives leads to an undeniable efficiency. But can we parallel these propagation ideas with a simple mathematical counterpart? Is there a rigorous theory backing up *formal differentiation* or *formal propagation*? The answer is positive and was initiated in [14, 15] through conservative Jacobians (see also [35, 23]).

A mathematical model for propagating derivatives. Conservative calculus models nonsmooth algorithmic differentiation faithfully and allows for a sharp study of training methods in Deep Learning [14, 15]. It involves a new class of derivatives, generalizing Clarke Jacobians [19]. A distinctive feature of conservative calculus is that it is preserved by Jacobian multiplication. Consider for example a feed forward network combining analytic or relu activations and max pooling. A conservative Jacobian for this network can be obtained by using Clarke Jacobians formally as classical Jacobians, regardless of qualification conditions. For instance, Figure 1 depicts a selection in a conservative Jacobian. This approach is general enough to handle spurious points such as in Figure 1 while keeping the essence of the properties one expects from a derivative. It was proved in [14] that $\mathrm{backprop}$, applied to any reasonable program of a function, is a conservative Jacobian for this function; in contrast, $\mathrm{backprop}$ cannot be modelled by some subdifferential operator. For instance for the fixed point problem above, given conservative Jacobians J_F and J_h (e.g., Clarke Jacobians) for F and h one obtains a new conservative Jacobian J_z implicitly defined through

$$J_F(z(x))J_z(x) = J_h(x).$$

This property exactly parallels the idea of "propagating derivatives" in practice. It gives a strong meaning to the formal use of Jacobians proposed in [7], and many empirical approaches [30, 2, 29, 28].

An implicit differentiation theorem and its consequences. Our contributions are as follows:

— We establish a nonsmooth conservative implicit function theorem that comes with an *implicit calculus* which is the central focus of this paper. As in the smooth case, the theorem relies on an invertibility condition with respect to the solution variable.

— We study a wide range of applications of our implicit differentiation theorem, covering deep equilibrium problems [7], conic optimization layers [2], and hyperparameter optimization for the Lasso [9]. Each case is detailed and its specificities are discussed.

— As a consequence, we obtain convergence guarantees for minibatching stochastic algorithms with vanishing step size for training wide classes of Neural Nets, or for Lasso hyperparameter selection. The assumptions needed for our results are mild and fulfilled by most losses occurring in ML in the spirit of [15, 34]: elementary log-exp functions [15], semialgebraic functions [12], all being subclasses of definable functions [22, 46]. The use of such structural classes has become standard in nonsmooth optimization and is more and more common in ML (see, e.g., [18, 15, 34, 32]).

— As in the smooth implicit function theorem, the invertibility condition is not avoidable in general. We provide various examples for which the assumption is not satisfied; this results in severe failures for the corresponding gradient methods. In Figure 1, one sees how lack of invertibility on an otherwise ordinary problem may provide totally unpredictable behavior for smooth quadratic optimization.

Definitions and Notations. A function $F : \mathbb{R}^n \to \mathbb{R}^m$ is *locally Lipschitz* if, for each $x \in \mathbb{R}^n$, there exists a neighborhood $\mathcal{U}$ of x such that F is Lipschitz on $\mathcal{U}$. Given matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{n\times p}$, $[A\ B] \in \mathbb{R}^{n\times(m+p)}$ denotes their concatenation; Id_n denotes the $n \times n$ identity matrix. For $q \in \mathbb{R}^n$, $\mathrm{diag}\,(q) \in \mathbb{R}^{n\times n}$ denotes the diagonal matrix whose diagonal entries are given by the q_i; $\mathrm{sign}\,(q) \in \{-1, 0, 1\}^n$ denotes the componentwise sign function. The *convex hull* of $\mathcal{U}$ is denoted $\mathrm{conv}\ \mathcal{U}$. The projection onto a closed convex set $\mathcal{C} \in \mathbb{R}^n$ is given, for each $x \in \mathbb{R}^n$, by $P_{\mathcal{C}}\,(x) := \mathrm{argmin}\{\frac{1}{2}\,\|u - x\|^2 : u \in \mathcal{C}\}$. Given a convex proper lower semicontinuous function $f : \mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$, we define its proximal operator through $x \in \mathbb{R}^n$, $\mathrm{prox}_f\,(x) := \mathrm{argmin}\{f\,(u) + \frac{1}{2}\,\|u - x\|^2\ u \in \mathbb{R}^n\}$. Set-valued maps are denoted by $\rightrightarrows$, for example the subgradient $\partial f \colon \mathbb{R}^n \rightrightarrows \mathbb{R}^n$. Additional details and notations are provided in Appendix A.

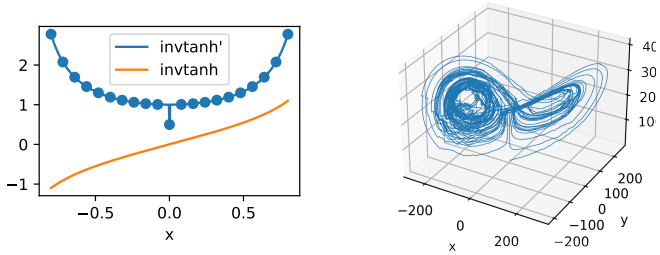


Figure 1: Left: Inconsistencies due to combination of implicit differentiation and algorithmic differentiation. Right: A gradient trajectory of an implicitly defined quadratic function.

2 Implicit Differentiation with Conservative Jacobians

Definitions and conservativity. Conservative Jacobians are generalized forms of Jacobians well suited for automatic differentiation, introduced in [14]. Given a locally Lipschitz continuous function $F : \mathbb{R}^n \to \mathbb{R}^m$, we say that $J_F : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n\times m}$ is a *conservative mapping* or a *conservative Jacobian* for F if J_F has a closed graph, is locally bounded, and is nonempty with

$$\frac{\mathrm{d}}{\mathrm{d}t} F(\gamma(t)) \in J_F(\gamma(t))\dot{\gamma}(t) \text{ a.e.} \tag{1}$$

whenever γ is an absolutely continuous curve in $\mathbb{R}^n$. When $m = 1$, the corresponding vectors are called *conservative gradient fields*. Note that when J_F is conservative, so is its pointwise convexified extension $\operatorname{conv} J_F$.

A locally Lipschitz function is called *path differentiable* if it has a conservative Jacobian. Recall that the *Clarke Jacobian* is defined as

$$\operatorname{Jac}^c F(x) = \operatorname{conv}\left\{ \lim_{k\to+\infty} \operatorname{Jac} F(x_k) : x_k \in \operatorname{diff}_F, x_k \underset{k\to+\infty}{\to} x \right\}$$

where diff_F is the full measure set of points where F is differentiable and $\operatorname{Jac} F$ is the standard Jacobian of F. Path differentiability is equivalent to having a chain rule as in (1) for the Clarke subdifferential, see [14, 24].

Examples of path differentiable functions and conservative Jacobians. (a) Convex functions and concave functions are path differentiable, see [14]. This implies that their subdifferential in the sense of convex analysis is a conservative field.
(b) The vast class of definable functions are path differentiable [24, 14]. As a result, the Clarke Jacobian of a Lipschitz definable mapping is a conservative Jacobian. Definable functions (see [5, 24, 18, 14] for an optimization context and [22] for a foundational work) encompass semi-algebraic functions [12], elementary log-exp selection [14], PAP [34] (restricted to analytic functions with full domain), and many others, see [46] and references therein. This includes networks with common nonlinearities: for example analytic with full domain (e.g., square, exponential, logistic loss, hyperbolic tangent, sigmoid), relu, max pooling, sort, (see Appendix A.2 for more detail).
(c) The backpropagation can be seen as an oracle (in the optimization sense) for a conservative Jacobian. Let P_F be a numerical program for a function F, aggregating elementary functions, for instance, relu, max pooling, affine mappings, polynomials (in general, any definable function). Then the backpropagation algorithm applied to P_F, which we denote (abusively) by $\operatorname{backprop} P_F := \operatorname{backprop}_F$, outputs an element of a conservative Jacobian [14, Theorem 8] which depends on P_F and can be constructed by a closure procedure [15, definition 5]. As described in [15], due to spurious behaviors, $\operatorname{backprop}_F$ is not in general an element of the Clarke Jacobian of F.

The structure of conservative Jacobians. As established in [35] in a semialgebraic context, the discrepancy between conservative gradients and Clarke subdifferentials is somehow negligible. Let us provide a version of that result matching our concerns. We call conservative mappings of the null function *residual* or *residual conservative*. Such a mapping R has the property that $R(x + tv)v = 0$ for almost all t in $\mathbb{R}$ and all x, v in $\mathbb{R}^n \times \mathbb{R}^n$. The following theorem and proposition (partially) extend results from [14] and [35], their proof is given in Appendix B.

Theorem 1 (The Clarke Jacobian is a minimal conservative Jacobian) *Given a nonempty open subset $\mathcal{U}$ of $\mathbb{R}^n$ and $F : \mathcal{U} \subset \mathbb{R}^n \to \mathbb{R}^m$ locally Lipschitz, let J_F be a convex-valued conservative Jacobian for F. Then for almost all $x \in \mathcal{U}, J_F(x) = \{\mathrm{Jac}\, F\}$ and, for all $x \in \mathcal{U}$, $\mathrm{Jac}^c F(x) \subset J_F(x)$.*

Proposition 1 (Decomposition of conservative fields) *Let J_F be a conservative Jacobian for F, then there is a residual R such that*

$$J_F \subset \mathrm{Jac}^c F + R.$$

Note that the above may not hold with equality. Consider $F(x) = |x|$ and $J_F(0) = [-1, 1] \cup [2, 3]$, $J_F(x) = \mathrm{sign}\,(x)$ otherwise. One cannot write $J_F = \mathrm{Jac}^c F + R$ with a residual operator R.

Formal differentiation in a nonsmooth setting. Propagating derivatives within a nonsmooth function finds its justification in the following:

Proposition 2 (Stability by composition, [14]) *Let $F : \mathbb{R}^n \to \mathbb{R}^m$ and $G : \mathbb{R}^m \to \mathbb{R}^l$ be two locally path differentiable functions having respective conservative Jacobians J_F and J_G. Then $F \circ G$ is path differentiable and the point-to-set matrix-valued $x \rightrightarrows J_F(G(x))J_G(x)$ is conservative.*

A conservative Implicit Function Theorem. There is already a long tradition of nonsmooth implicit function theorems, e.g., [19, 6, 42, 25]. What makes the following theorem useful is that it comes with a qualification-free calculus. The proofs are given in Appendix B.

Theorem 2 (Implicit differentiation) *Let $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ be path differentiable on $\mathcal{U} \times \mathcal{V} \subset \mathbb{R}^n \times \mathbb{R}^m$ an open set and $G : \mathcal{U} \to \mathcal{V}$ a locally Lipschitz function such that, for each $x \in \mathcal{U}$,*

$$F(x, G(x)) = 0. \tag{2}$$

Furthermore, assume that for each $x \in \mathcal{U}$, for each $[A\ B] \in J_F(x, G(x))$, the matrix B is invertible where J_F is a conservative Jacobian for F. Then, $G : \mathcal{U} \to \mathcal{V}$ is path differentiable with conservative Jacobian given, for each $x \in \mathcal{U}$, by

$$J_G \colon x \rightrightarrows \left\{-B^{-1}A : [A\ B] \in J_F(x, G(x))\right\}.$$

Corollary 1 (Path differentiable implicit function theorem) *Let $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ be path differentiable with conservative Jacobian J_F. Let $(\hat{x}, \hat{y}) \in \mathbb{R}^n \times \mathbb{R}^m$ be such that $F(\hat{x}, \hat{y}) = 0$. Assume that $J_F(\hat{x}, \hat{y})$ is convex and that, for each $[A\ B] \in J_F(\hat{x}, \hat{y})$, the matrix B is invertible. Then, there exists an open neighborhood $\mathcal{U} \times \mathcal{V} \subset \mathbb{R}^n \times \mathbb{R}^m$ of $(\hat{x}, \hat{y})$ and a path differentiable function $G : \mathcal{U} \to \mathcal{V}$ such that the conclusion of Theorem 2 holds.*

Corollary 2 (Path differentiable inverse function theorem) *Let $\mathcal{U}$ and $\mathcal{V}$ be open neighborhoods of 0 in $\mathbb{R}^n$ and $\Phi : \mathcal{U} \to \mathcal{V}$ path differentiable with $\Phi(0) = 0$. Assume that Φ has a conservative Jacobian J_Φ such that $J_\Phi(0)$ contains only invertible matrices. Then, locally, Φ has a path differentiable inverse Ψ with a conservative Jacobian given by*

$$J_\Psi(y) = \left\{A^{-1} : A \in J_\Phi(\Psi(y))\right\}.$$

Remark 1 (On the necessity of conservativity) (a) Example 1 in Appendix B shows that one cannot hope for the formulas in Corollaries 1 & 2 to provide Clarke Jacobians in general, even if the input(s) are Clarke Jacobians themselves.
(**Lipchitz definable implicit and inverse functions theorems**) (b) See Theorem 4 and 5 in the appendix

3 Nonsmooth implicit differentiation in Machine Learning

Detailed proof arguments for all considered models are given in Appendix C.

Monotone deep equilibrium networks. Deep Equilibrium Networks (DEQs) [7] are specific neural network architectures including layers whose input-output relation is implicitly defined through a fixed point equation of the form

$$z = f(z, x) \tag{3}$$

where $x \in \mathbb{R}^p$ is a given input and $z \in \mathbb{R}^m$ is the corresponding output. We may consider that the variable x represents both the input layer and layer parameters. Assuming that, for each $x \in \mathbb{R}^p$, there is a unique $z \in \mathbb{R}^m$ satisfying the relation (3), this defines an input-output relation $z\colon \mathbb{R}^p \to \mathbb{R}^m$. Furthermore, if f is path differentiable with convex-valued conservative Jacobian $J_f\colon \mathbb{R}^m \times \mathbb{R}^p \rightrightarrows \mathbb{R}^{m\times(m+p)}$ whose projection on the first m columns are all invertible, then the function z itself admits a conservative Jacobian which can be computed from Theorem 2.

We now focus on monotone operator implicit layers [49] for which assumptions are easily stated. Our method applies to other similar architectures, e.g., DEQs [7] or implicit graph neural networks [30]. Let $\sigma\colon \mathbb{R}^m \to \mathbb{R}^m$ be the proximal operator of a convex function and assume σ is path differentiable with conservative Jacobian $J_\sigma\colon \mathbb{R}^m \rightrightarrows \mathbb{R}^{m\times m}$, assumed to be convex-valued. This encompasses the majority of activation functions used in practice [20]. Let $W \in \mathbb{R}^{m\times m}$ be a matrix such that $W + W^T \succeq 2\theta I$ with $\theta > 0$. Under these assumptions the implicit equation

$$z = \sigma(Wz + b) \tag{4}$$

has a unique output $z(W, b)$ [49, Theorem 2]. The transformation $(W, b) \mapsto z(W, b)$ is a *monotone implicit layer.*

The set-valued mapping obtained from Theorem 2 provides a conservative Jacobian for $(W, z) \mapsto z(W, z)$. A similar expression was described in [49, Theorem 2], without using conservativity and using the Clarke Jacobian formally as a classical Jacobian. The proposition below provides a full justification of this heuristic and ensures convergence of algorithmic differentiation based training.

Proposition 3 (Path differentiation through monotone layers) *Assume that J_σ is convex-valued and that, for all $J \in J_\sigma(Wz(W, b) + b)$, the matrix $(\mathrm{Id}_m - JW)$ is invertible. Consider a loss-like function $\ell\colon \mathbb{R}^m \to \mathbb{R}$ with conservative gradient $D_\ell\colon \mathbb{R}^m \rightrightarrows \mathbb{R}^m$, then $g : (W, z) \mapsto \ell(z(W, b))$ is path differentiable and has a conservative gradient D_g defined through*

$$D_g\colon (W, b) \rightrightarrows \left\{ J^T(\mathrm{Id}_m - JW)^{-T} v z^T, J^T(\mathrm{Id}_m - JW)^{-T} v) : \ J \in J_\sigma(Wz + b),\ v \in D_\ell(z) \right\}.$$

Remark 2 Convexity and invertibility assumptions are satisfied when J_σ is the Clarke Jacobian [49].

Optimization layers: the conic program case. Optimization layers in deep learning may take many forms; we consider here those based on conic programming [17, 3, 2, 4]. We follow [3], simplifying the analysis by ignoring infeasability certificates, which correspond to the absence of a primal-dual solution [17], in line with the implementation described in [2, Appendix B]. Consider a conic problem (P) and its dual (D):

$$\begin{array}{llll} \text{(P)} & \inf & c^T x & \\ & \text{subject to} & Ax + s = b & \\ & & s \in \mathcal{K} & \end{array} \qquad \begin{array}{llll} \text{(D)} & \inf & b^T y & \\ & \text{subject to} & A^T y + c = 0 & \\ & & y \in \mathcal{K}^*, & \end{array} \tag{5}$$

with primal variable $x \in \mathbb{R}^n$, dual variable $y \in \mathbb{R}^m$, and primal slack variable $s \in \mathbb{R}^m$. The set $\mathcal{K} \subset \mathbb{R}^m$ is a nonempty closed convex cone and $\mathcal{K}^* \subset \mathbb{R}^m$ is its dual cone. The problem parameters are the matrix $A \in \mathbb{R}^{m\times n}$ and the vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$; the cone $\mathcal{K}$ is fixed. Under the assumption that there is a unique primal-dual solution (x, y, s), we study the path differentiability of the solution mapping as a function of its parameters:

$$(A, b, c) \mapsto \mathrm{sol}(A, b, c) = (x, y, s).$$

For this, let us interpret the solution mapping as a composition mapping involving equation-like implicit formulations. Set $N = n + m$, given $A, b, c \in \mathbb{R}^{m\times n} \times \mathbb{R}^m \times \mathbb{R}^n$, define

$$Q(A, b, c) = \begin{bmatrix} 0 & A^T \\ -A & 0 \end{bmatrix} \in \mathbb{R}^{N\times N} \qquad V(b, c) = \begin{bmatrix} c \\ b \end{bmatrix} \in \mathbb{R}^N.$$

Consider a vector $z = (u, v) \in \mathbb{R}^n \times \mathbb{R}^m$, denote by Π the projection onto $\mathbb{R}^n \times \mathcal{K}^*$ and define the *residual map* $\mathcal{N} : \mathbb{R}^N \times \mathbb{R}^{m\times n} \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^N$ as

$$\mathcal{N}(z, A, b, c) = (Q(A, b, c) - \mathrm{Id}_N)\Pi z + V(b, c) + z$$

The mapping $\mathcal{N}$ is a synthetic form of optimality measure for (P) and (D), capturing KKT conditions. To simplify the presentation, we ignore the extreme cases of infeasibility and unboundedness which correspond to an absence of solution in [17].

Define the function $\phi : \mathbb{R}^N \to \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m$ through $\phi(u, v) := (u, P_{\mathcal{K}^*}(v), P_{\mathcal{K}^*}(v) - v)$. As shown in Appendix C.2, $\phi(u, v)$ provides a primal-dual KKT solution of problems (P) and (D) if and only if $\mathcal{N}(z, A, b, c) = 0$. When we assume that, for fixed A, b, and c, there is a unique $z \in \mathbb{R}^N$ such that $\mathcal{N}(z, A, b, c) = 0$, we have an implicitly defined a function $z = \nu(A, b, c)$, such that

$$\mathrm{sol}(A, b, c) = [\phi \circ \nu]\,(A, b, c). \tag{6}$$

The following result extends the discussion in [17, 3], limited to situations where Π is differentiable at the proposed solution z, to a fully nonsmooth setting; its proof is postponed to Appendix C.2.

Proposition 4 (Path differentiation through cone programming layers) *Assume that $P_{\mathcal{K}^*}$, $\mathcal{N}$ are path differentiable, denote respectively by $J_{P_{\mathcal{K}^*}}$, $J_{\mathcal{N}}$ corresponding convex-valued conservative Jacobians. Assume that, for all $A, b, c \in \mathbb{R}^{m\times n} \times \mathbb{R}^m \times \mathbb{R}^n$, $z = \nu(A, b, c) \in \mathbb{R}^n \times \mathbb{R}^m$ is the unique solution to $\mathcal{N}(z, A, b, c) = 0$ and that all matrices formed from the N first columns of $J_{\mathcal{N}}(z, A, b, c)$ are invertible. Then, ϕ, ν, and* sol *are path differentiable functions with conservative Jacobians:*

$$\begin{aligned}
J_\nu(A, b, c) &:= \left\{-U^{-1}V : [U\ V] \in J_{\mathcal{N}}(\nu(A, b, c), A, b, c)\right\},\\
J_\phi(z) &:= \begin{pmatrix} \mathrm{Id}_n & 0 \\ 0 & J_{P_{\mathcal{K}^*}}(v) \\ 0 & (J_{P_{\mathcal{K}^*}}(v) - \mathrm{Id}_m) \end{pmatrix}\\
J_{\mathrm{sol}}(A, b, c) &:= J_\phi(\nu(A, b, c))J_\nu(A, b, c).
\end{aligned}$$

In practice, the path differentiability of conic projections is pervasive since they are generally semialgebraic (orthant, second-order cone, PSD cone). See [31, 37, 33, 37] for the computations of the corresponding Clarke Jacobians (which are conservative). Note that a conservative Jacobian for $\mathcal{N}$ may be obtained from $J_{P_{\mathcal{K}^*}}$ using Proposition 2.

Hyperparameter selection for Lasso type problems. Implicit differentiation can be used to tune hyperparameters via first-order methods optimizing some measure of task performance, see [10] and references therein. In a nonsmooth context, we recall the formulation in [9] of the general hyperparameter optimization problem as a bi-level optimization problem:

$$\min_{\lambda\in\mathbb{R}^m} C(\hat\beta(\lambda)) \quad \text{such that} \quad \hat\beta(\lambda) \in \operatorname*{argmin}_{\beta\in\mathbb{R}^p} \psi(\beta, \lambda)$$

where $C : \mathbb{R}^p \to \mathbb{R}$ is continuously differentiable (e.g., test loss) and $\psi\colon \mathbb{R}^p \times \mathbb{R}^m \to \mathbb{R}$ is a possibly nonsmooth training loss, convex in β, with hyperparameter $\lambda \in \mathbb{R}^m$. We seek a subgradient type method for this problem with convergence guaranties; our nonsmooth implicit differentiation results can be used for this purpose. We demonstrate this approach on the Lasso problem [44]

$$\hat\beta\,(\lambda) \in \operatorname{argmin}\left\{\frac{1}{2}\,\|y - X\beta\|_2^2 + e^\lambda\,\|\beta\|_1 : \beta \in \mathbb{R}^p\right\} \tag{7}$$

where $y \in \mathbb{R}^n$ is the vector of observations, $X = [X_1, \dots, X_p] \in \mathbb{R}^{n\times p}$ is the design matrix with columns $X_j \in \mathbb{R}^n$, $j \in \{1, \dots p\}$, and $\lambda \in \mathbb{R}$ is the hyperparameter. Define $F : \mathbb{R} \times \mathbb{R}^p \to \mathbb{R}^p$ to be

$$F\,(\lambda, \beta) := \beta - \mathrm{prox}_{e^\lambda\|\cdot\|_1}\left(\beta - X^T\,(X\beta - y)\right)$$

and recall that, for each $i \in \{1, \dots, p\}$, $[\mathrm{prox}_{e^\lambda\|\cdot\|_1}(\beta)]_i = \mathrm{sign}(\beta_i)\max\{|\beta_i| - e^\lambda, 0\}$. The function $F(\lambda, \beta)$ is thus nonsmooth but locally Lipschitz on $\mathbb{R} \times \mathbb{R}^p$. An optimal $\hat\beta(\lambda)$ for (7) must satisfy $F(\lambda, \hat\beta(\lambda)) = 0$ [21, Prop. 3.1]. For a given solution $\hat\beta(\lambda)$, we introduce the equicorrelation set by $\mathcal{E} := \{j \in \{1, \dots, p\} : |X_j^T(y - X\hat\beta(\lambda))| = e^\lambda\}$ which contains the support set $\mathrm{supp}\,\hat\beta := \{i \in \{1, \dots, p\} : \hat\beta_i \neq 0\}$. In fact, $\mathcal{E}$ does not depend on the choice of the solution $\hat\beta$, see [45, Lemma 1]. The proof of the following result is given in Appendix C.3.

Proposition 5 (Conservative Jacobian for the solution mapping) *For all $\lambda \in \mathbb{R}$, assume $X_{\mathcal{E}}^T X_{\mathcal{E}}$ is invertible where $X_{\mathcal{E}}$ is the submatrix of X formed by taking the columns indexed by $\mathcal{E}$. Then $\hat{\beta}(\lambda)$ is single valued, path differentiable with conservative Jacobian, $J_{\hat{\beta}}(\lambda)$, given for all λ as*

$$\left\{ \left[-e^{\lambda} \left(\mathrm{Id}_p - \mathrm{diag}\,(q) \left(\mathrm{Id}_p - X^T X \right) \right)^{-1} \mathrm{diag}\,(q)\, \mathrm{sign} \left(\hat{\beta} - X^T \left(X\hat{\beta} - y \right) \right) \right] \; : q \in \mathcal{M}(\lambda) \right\}$$

where $\mathcal{M}(\lambda) \subset \mathbb{R}^p$ is the set of vectors q such that $q_i = 1$ if $i \in \operatorname{supp} \hat{\beta}$, $q_i = 0$ if $i \notin \mathcal{E}$ and $q_i \in [0,1]$ if $i \in \mathcal{E} \setminus \operatorname{supp} \hat{\beta}$.

Taking, in Proposition 5, $q_i = 1$ for all $i \in \mathcal{E}$ corresponds to the directional derivative given by LARS algorithm [27], see also [36]. Alternatively, taking $q_i = 0$ for $i \notin \operatorname{supp} \hat{\beta}$ gives the weak derivative described by [9]. Both are particular selections in $J_{\hat{\beta}}$, which is the underlying conservative field.

4 Optimizing implicit problems with gradient descent

We establish the convergence of gradient descent algorithms for compositional learning problems involving implicitly defined functions. The result follows from the previous section and the general convergence results of [15].

The minimization problem. The applications considered in the previous section all yield minimization problems of the type

$$\min_{w \in \mathbb{R}^p} \ell(w) := \frac{1}{N} \sum_{i=1}^{N} \ell_i(w) \;\text{ with }\; \ell_i = g_{i,L} \circ g_{i,L-1} \circ \ldots \circ g_{i,1} \tag{8}$$

where, for each $i \in \{1, \ldots, N\}$, $\ell_i : \mathbb{R}^p \to \mathbb{R}$ is a composition of functions having appropriate input and output dimensions. The indices i correspond in practice to learning samples while the loss ℓ embodies an empirical expectation, as for instance in deep learning. We will enforce the following structural condition.

Assumption 1 *For $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, L\}$, the function $g_{i,j}$ is locally Lipschitz with conservative Jacobian $J_{i,j}$ and one of the following holds*

- *$g_{i,j}$ and $J_{i,j}$ are semialgebraic (or, more generally, definable).*
- *$g_{i,j}$ is defined as G in Theorem 2, with F and J_F semialgebraic (or, more generally, definable).*

Actually, in Assumption 1 the second point implies the first point; we list both for clarity. More details on semialgebraicity and definability are given in Appendix A.2. Let us stress that virtually all elements entering the definition of neural networks are semialgebraic or, more generally, definable, see for example [15] for a constructive model. In particular, beyond classical networks with usual nonlinearities (e.g., relu, sigmoid, max pooling ...), this setting encompasses (through Corollary 1):

(a) Deep equilibrium networks: each $g_{i,j}$ may correspond to usual explicit layers or an implicit layer involving a fixed point mapping and a learning sample i as in (4) or (3).

(b) Training with optimization layers: similarly, the inner maps $g_{i,j}$ may also be solution mapping to convex conic programs and related to the sol function (6) of conic problems.

(c) One may assume that $N = 1, L = 2$ and retrieve the hyperparameter tuning for Lasso in its implicit formulation.

SGD with backpropagation. Algorithmic differentiation (AD) is an automated application of the chain rule of differential calculus. When applied to ℓ_i, it amounts to computing one element of the product $J_i := \prod_{j=1}^{L} J_{i,j}$ by choosing one element in each $J_{i,j}$ with appropriate inputs given by intermediate results kept in memory during a forward computation of the composition.

In this context AD stochastic gradient descent requires an initial $w_0 \in \mathbb{R}^p$ and a sequence of *i.i.d.* random indices uniform in $\{1, \ldots, N\}$, $(I_k)_{k \in \mathbb{N}}$. It gives:

$$w_{k+1} = w_k - s\alpha_k v_k \tag{9}$$

$$v_k \in J_{I_k}(w_k), \qquad \text{(given by backprop)}, \tag{10}$$

where $(\alpha_k)_{k\in\mathbb{N}}$ is a sequence of positive step sizes and $s \in (s_{\min}, s_{\max})$ is a scaling factor where $s_{\max} > s_{\min} > 0$. A simpler choice could be $v_k \in \partial^c \ell_{I_k}(w_k)$, however, the chain rule used within algorithmic differentiation routines does not produce subgradients (see, e.g., Figure 1). In contrast, conservative Jacobians are faithful models of AD outputs. The asymptotic behavior of the above algorithm depends on the variational properties of the conservative Jacobian $J := \frac{1}{N}\sum_{i=1}^{N} J_i$.

Theorem 3 (Convergence result) *Consider minimizing ℓ given in* (8) *using algorithm* (10) *under Assumption 1. Assume furthermore the following*

- ***Step size:*** $\sum_{k=1}^{+\infty} \alpha_k = +\infty$ *and* $\alpha_k = o(1/\log(k))$.
- ***Boundedness:*** *there exists $M > 0$, and $K \subset \mathbb{R}^p$ open and bounded, such that, for all $s \in (s_{\min}, s_{\max})$ and $w_0 \in \operatorname{cl} K$, $\|w_k\| \leq M$ almost surely.*

For almost all $w_0 \in K$ and $s \in (s_{\min}, s_{\max})$, the objective value $\ell(w_k)$ converges and all accumulation points $\bar{w}$ of w_k are Clarke-critical in the sense that $0 \in \partial^c \ell(\bar{w})$.

This result shows that AD SGD may be applied successfully to all problems described in Section 3, combining algorithmic differentiation with implicit differentiation. Its proof may be adapted directly from [14, 11]; details are given in Appendix D.

5 Numerical experiments

Using implicit differentiation when the invertibility condition in Theorem 2 does not hold can result in absurd training dynamics.

A cyclic gradient dynamic via fixed-point/optimization layer. Consider the bilevel problem:

$$\begin{aligned} \min_{x,y,s} \quad & \ell(x,y,s) := (x-s_1)^2 + 4(y-s_2)^2 \\ \text{s.t.} \quad & s \in s(x,y) := \arg\max \left\{(a+b)(-3x+y+2) : \, a \in [0,3], b \in [0,5]\right\}. \end{aligned} \tag{11}$$

Problem (11) has an equivalent fixed-point formulation using projected gradient descent on the inner problem (Appendix E.1.1). Backpropagation applied to (11) associates to (x, y) an element of the form:

$$\nabla_{(x,y)} \ell(x,y,s(x)) + \tilde{J}_s(x,y)^T \nabla_s \ell(x,y,s(x)) \tag{12}$$

where $\tilde{J}_s$ is piecewise derivative. We implement gradient descent for (11), evaluating (12) either using `cvxpylayers` [2] or the `JAX` tutorial [50] for fixed point layers. In both cases, the invertibility condition in Theorem 2 fails when $-3x + y + 2 = 0$, resulting in discontinuity of s, affecting the dynamics globally: the gradient trajectory converges to a limit cycle of non critical points (Figure 2a); see Appendix E.1 for details.

Persistence under small perturbations: The limit cycle remains (Figure 2b) when running the same experiment on a slightly perturbed version of Problem (11) with perturbed initializations as well (Appendix E.1.2).

A Lorenz-like attractor via implicit differentiation. Consider the Lorenz Ordinary Differential Equation (ODE) system:

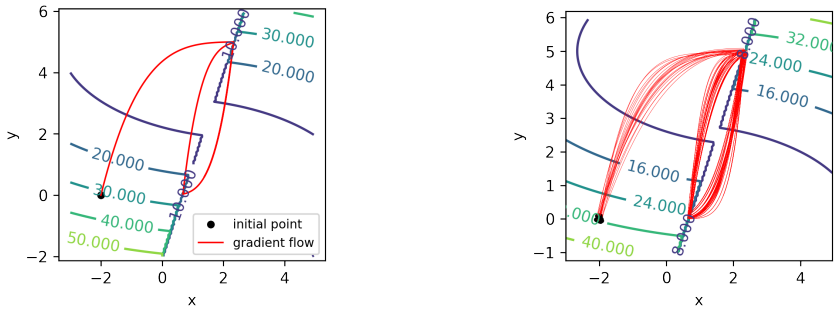
$$\dot{x} = \sigma(y-x), \qquad \dot{y} \;= x(\rho - z) - y, \quad \text{and} \quad \dot{z} = xy - \beta z. \tag{13}$$

It is well-known that taking $(\sigma, \rho, \beta) = (10, 28, 8/3)$, and $(x(0), y(0), z(0)) = (0, 1, 1.05)$ gives a chaotic trajectory, displayed in Figure 3a. Denoting $F : (x, y, z) \mapsto (\sigma(y-x), x(\rho-z)-y, xy-\beta z)$ the vector field of the Lorenz system (13), consider the optimization problem:

$$\max_{u\in\mathbb{R}^3} \quad u^T z \qquad \text{s.t.} \qquad z \in \arg\min_{s\in\mathbb{R}^3} \|s - F(u)\|^4 \tag{14}$$

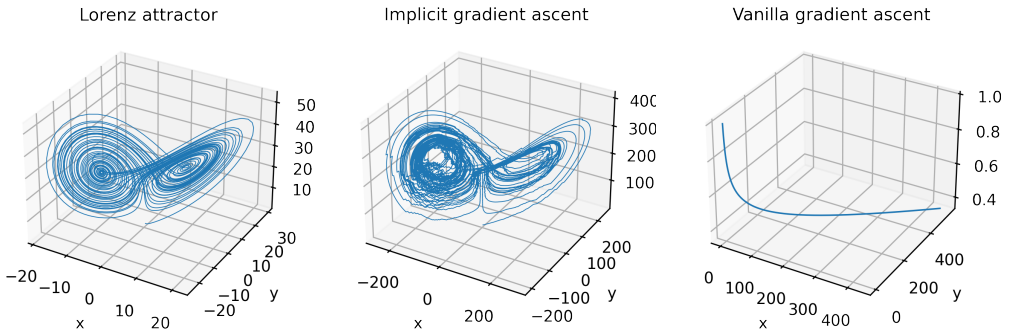
which is obviously equivalent to

$$\max_{u\in\mathbb{R}^3} \quad u^T F(u). \tag{15}$$



(a) (b)

Figure 2: (a) Gradient flow for all implementations. (b) Gradient flows for 20 perturbed experiments.



(a) (b) (c)

Figure 3: Implicit gradient ascent (b) outputs a pathological curve with some qualitative aspects of the Lorenz dynamic (a) and really different from a classical gradient (c).

The function $g : u \mapsto u^T F(u)$ is a nondegenerate quadratic function whose expression can be found in Appendix E.2.1. The function g has for unique critical point $(0, 0, 0)$ which is a strict saddle-point. We perform gradient ascent with implicit differentiation using `cvxpylayers` on (14), and the classical gradient ascent on the equivalent problem (15). The path obtained by implicit differentiation (Figure 3b) resembles the Lorenz attractor (Figure 3a), in stark contrast to the conventional method (Figure 3c). The chaotic dynamics are a consequence of the lack of invertibility, due to the power 4 in (14), and various numerical approximations related to optimization and implicit differentiation.

6 Conclusion and future work

This article provides a rigorous framework and calculus rules for nonsmooth implicit differentiation using the theory of conservative Jacobians. In particular, it describes precise conditions under which implicit differentiation can be used, in a way that is compatible with backpropagation and first-order algorithms.

We show the applicability of our results on practical machine learning problems including training of neural networks involving layers with implicitly defined outputs (deep equilibrium nets, networks with optimization layers) and nonsmooth hyperparameter optimization (Lasso-type models).

Finally, we demonstrate the necessity of a rigorous theory of nonsmooth implicit differentiation through multiple numerical experiments. These illustrate the range of extremely pathological gradient dynamics that can occur when algorithmic differentiation is combined with nonsmooth implicit differentiation outside the scope of our theorem, i.e., without satisfying the invertibility condition we specify.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[3] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi. Differentiating through a cone program. *J. Appl. Numer. Optim*, 1(2):107–115, 2019.

[4] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[5] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-łojasiewicz inequality. *Mathematics of operations research*, 35(2):438–457, 2010.

[6] H. Aubin, J.-P; Frankowska. On inverse function theorems for set-valued maps. *Journal de mathématiques pures et appliquées*, 1987.

[7] S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[8] M. Benaïm, J. Hofbauer, and S. Sorin. Stochastic approximations and differential inclusions. *SIAM Journal on Control and Optimization*, 44(1):328–348, 2005.

[9] Q. Bertrand, Q. Klopfenstein, M. Blondel, S. Vaiter, A. Gramfort, and J. Salmon. Implicit differentiation of lasso-type models for hyperparameter optimization. In *International Conference on Machine Learning*, pages 810–821. PMLR, 2020.

[10] Q. Bertrand, Q. Klopfenstein, M. Massias, M. Blondel, S. Vaiter, A. Gramfort, and J. Salmon. Implicit differentiation for fast hyperparameter selection in non-smooth convex learning. *arXiv preprint arXiv:2105.01637*, 2021.

[11] P. Bianchi, W. Hachem, and S. Schechtman. Convergence of constant step stochastic gradient descent for non-smooth non-convex functions. *arXiv preprint arXiv:2005.08513*, 2020.

[12] J. Bochnak, M. Coste, and M.-F. Roy. *Real algebraic geometry*, volume 36. Springer Science & Business Media, 2013.

[13] J. Bolte, A. Daniilidis, A. Lewis, and M. Shiota. Clarke subgradients of stratifiable functions. *SIAM Journal on Optimization*, 18(2):556–572, 2007.

[14] J. Bolte and E. Pauwels. Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Mathematical Programming*, pages 1–33, 2020.

[15] J. Bolte and E. Pauwels. A mathematical model for automatic differentiation in machine learning. In *Conference on Neural Information Processing Systems*, 2020.

[16] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[17] E. Busseti, W. M. Moursi, and S. Boyd. Solution refinement at regular points of conic problems. *Computational Optimization and Applications*, 74(3):627–643, 2019.

[18] C. Castera, J. Bolte, C. Févotte, and E. Pauwels. An inertial newton algorithm for deep learning. *arXiv preprint arXiv:1905.12278*, 2019.

[19] F. H. Clarke. *Optimization and nonsmooth analysis*. SIAM, 1983.

[20] P. L. Combettes and J.-C. Pesquet. Deep neural network structures solving variational inequalities. *Set-Valued and Variational Analysis*, pages 1–28, 2020.

[21] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[22] M. Coste. *An introduction to o-minimal geometry*. Istituti editoriali e poligrafici internazionali Pisa, 2000.

[23] D. Davis and D. Drusvyatskiy. Conservative and semismooth derivatives are equivalent for semialgebraic maps. *arXiv preprint arXiv:2102.08484*, 2021.

[24] D. Davis, D. Drusvyatskiy, S. Kakade, and J. D. Lee. Stochastic subgradient method converges on tame functions. *Foundations of computational mathematics*, 20(1):119–154, 2020.

[25] A. L. Dontchev and R. T. Rockafellar. *Implicit functions and solution mappings*, volume 543. Springer, 2009.

[26] L. Dries and C. Miller. On the real exponential field with restricted analytic functions. *Israel Journal of Mathematics*, 92:427, 1995.

[27] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *Annals of statistics*, 32(2):407–499, 2004.

[28] L. E. Ghaoui, F. Gu, B. Travacca, and A. Askari. Implicit deep learning. *CoRR*, abs/1908.06315, 2019.

[29] S. Gould, R. Hartley, and D. J. Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[30] F. Gu, H. Chang, W. Zhu, S. Sojoudi, and L. El Ghaoui. Implicit graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[31] S. Hayashi, N. Yamashita, and M. Fukushima. A combined smoothing and regularization method for monotone second-order cone complementarity problems. *SIAM Journal on Optimization*, 15(2):593–615, 2005.

[32] Z. Ji and M. Telgarsky. Directional convergence and alignment in deep learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[33] L. Kong, L. Tunçel, and N. Xiu. Clarke generalized jacobian of the projection onto symmetric cones. *Set-Valued and Variational Analysis*, 17(2):135–151, 2009.

[34] W. Lee, H. Yu, X. Rival, and H. Yang. On correctness of automatic differentiation for non-differentiable functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[35] A. Lewis and T. Tian. The structure of conservative gradient fields. *arXiv preprint arXiv:2101.00699*, 2021.

[36] J. Mairal and B. Yu. Complexity analysis of the lasso regularization path. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1835–1842, 2012.

[37] J. Malick and H. S. Sendov. Clarke generalized jacobian of the projection onto the cone of positive semidefinite matrices. *Set-Valued Analysis*, 14(3):273–293, 2006.

[38] J. J. Moreau. Décomposition orthogonale d'un espace hilbertien selon deux cônes mutuellement polaires. *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, 255:238–240, 1962.

[39] M. R. Osborne, B. Presnell, and B. A. Turlach. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337, 2000.

[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[41] F. Pedregosa. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pages 737–746. PMLR, 2016.

[42] S. M. Robinson. An implicit-function theorem for a class of nonsmooth functions. *Mathematics of operations research*, 16(2):292–309, 1991.

[43] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986.

[44] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[45] R. J. Tibshirani et al. The lasso problem and uniqueness. *Electronic Journal of statistics*, 7:1456–1490, 2013.

[46] L. van den Dries and C. Miller. Geometric categories and o-minimal structures. *Duke Math. J*, 84(2):497–540, 1996.

[47] J. Warga. Fat homeomorphisms and unbounded derivate containers. *Journal of Mathematical Analysis and Applications*, 81:545–560, 1981.

[48] A. Wilkie. A theorem of the complement and some new o-minimal structures. *Selecta Mathematica*, 5:397–421, 1999.

[49] E. Winston and J. Z. Kolter. Monotone operator equilibrium networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[50] M. J. Zico Kolter, David Duvenaud. Deep implicit layers - neural odes, deep equilibirum models, and beyond, 2020. NeurIPS Tutorial.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Rather less than more.
   (b) Did you describe the limitations of your work? [Yes] The main limitation lies in the assumptions of our theoretical results. Assumptions are always carefully described; they are discussed if they are not optimal. Many examples are given in the Appendix.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is theoretical and we do not foresee any such direct consequence.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We followed the guidelines as closely as we could.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] All our results are stated as Theorems, Propositions, Lemmas for which detailed assumptions are given, sometimes in a separate environement if details are required.
   (b) Did you include complete proofs of all theoretical results? [Yes] All proofs are given in appendices appendix B, C, D.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] A jupyter notebook is added in supplementary material. It allows to reproduce all our optimization experiments.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A] We do not perform training.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] We do not perform training and our experiments are only illustrative. We still conducted a sensitivity analysis experiment in the experimental section which illustrates the robustness of the observed behavior with respect to problem data perturbation.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We only conduct illustrative experiments in dimension 2, 3 and 4 which can be run in minutes on a basic laptop or any computer.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Creators and corresponding works are cited in Section 5
   (b) Did you mention the license of the assets? [Yes] The license of the assets is mentioned in appendix, all used assets are released under Apache License.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] A jupyter notebook is added in supplementary material. It allows to reproduce all our optimization experiments.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We do not use data.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We do not use data.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]