Automatic Local Inverse Calculation for Change of Variables

Automatic Local Inverse Calculation for Change of Variables

Elias Rojas Collins
Massachusetts Institute of Technology
Cambridge, USA

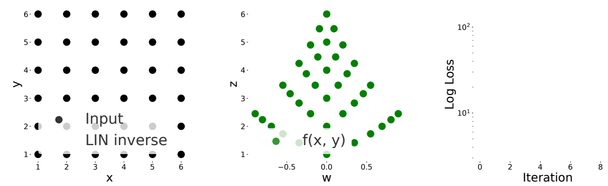
Abstract

Inversion is a fundamental operation that arises frequently in probabilistic inference and computer graphics. For example, inversion is used to decrease variance and to enable differentiation in variational inference (e.g., reparameterization trick) and in differentiable rendering (e.g., to integrate over object boundaries). Existing approaches to inversion limit the class of functions inverted, for example, to affine functions, or require a user-specified inverse. We study when a *local inverse*—an inverse that is valid in a neighborhood of a point—exists. We provide an algorithm to approximate the local inverse and give the convergence rate of the solver. We present `LIN`, a system that automatically computes the local inverse of a function using a fixed-point solver. We implement `LIN` in Python and use it to automatically compute the local inverse of affine, polar, and hyperbolic changes of variables arising in image stylization.

CCS Concepts: • **Mathematics of computing** → **Solvers**; • **Computing methodologies** → Rendering.

Keywords: Inversion, Differentiable Programming, Differentiable Rendering, Probabilistic Programming

ACM Reference Format:
Elias Rojas Collins. 2024. Automatic Local Inverse Calculation for Change of Variables. In *Companion Proceedings of the 2024 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity (SPLASH Companion '24), October 20–25, 2024, Pasadena, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3689491.3689970



(a) Inputs (black), `LIN` inverse (orange). **(b)** The lattice under the mapping f. **(c)** Log loss over iterations

Figure 1. `LIN` local inverse matches input under hyperbolic change of variables: $w, z = f(x, y) = \left(\log\left(\sqrt{\frac{x}{y}}\right), \sqrt{xy}\right)$.

1 Introduction and Background

Inversion is a fundamental mathematical primitive critical in solving problems arising in domains such as computer graphics and probabilistic inference. For example, changes of variables often require inversion and are used to reduce variance and to enable differentiation in variational inference (e.g., reparameterization trick) [6, 7] and in differentiable rendering (to integrate over object boundaries) [3, 11].

To meet this challenge, researchers have developed invertible programming languages that given the forward execution of a program from a restricted grammar, can automatically deduce an inverse [1, 3, 10, 14]. If for every input the forward execution has a unique inverse then the function it denotes has a *global inverse*.

A key gap in the existing systems is support for functions with a *local inverse*—an inverse that need only be correct in a neighborhood of a query point. Automated solvers fail to (locally) invert these functions [13] or require handwritten piecewise inverses [3, 11].

In this paper, we build a system, `LIN`, which solves the problem of approximate, automatic inversion for functions that have local inverses. `LIN` uses a fixed-point solver that given y, produces a sequence $(x_n)_{n\in\mathbb{N}}$ that approaches the inverse $x = f^{-1}(y)$ near a given initialization point x_0. The solver satisfies useful theoretical properties such as being correct in a neighborhood of the point and approaching the exact inverse exponentially quickly.

2 Theory of Local Inversion

In this section, we present the theory and analysis that underlies the local inversion algorithm used in `LIN`. We start by defining differentiability and the local inverse of a function.

Definition 2.1 (Lee [9, Definition C.1])**.** A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is *differentiable at a point t* if there exists a linear map $Df(t) : \mathbb{R}^n \to \mathbb{R}^m$ such that

$$\lim_{x\to 0} \frac{\|f(x+t) - f(t) - Df(t)\cdot(x)\|}{\|x\|} = 0. \qquad \triangle$$

A function Df that satisfies the above property is the *total derivative* of f. If $n = m = 1$ then the total derivative equals

SPLASH Companion '24, October 20–25, 2024, Pasadena, CA, USA

ACM ISBN 979-8-4007-1214-2/24/10
https://doi.org/10.1145/3689491.3689970

the usual derivative: $Df(t) = \frac{df}{dx}(t)$. A function f is *differentiable* if it is differentiable everywhere and is *continuously differentiable* if the total derivative is continuous.

Definition 2.2. A function $F : U \longrightarrow V$ is said to have a *local inverse* near t if there exist connected (see Janich [5, Pg. 14]) open sets U_0, V_0 such that $t \in U_0 \subseteq U$ and $F(t) \in V_0 \subseteq V$ and $F|_{U_0} : U_0 \longrightarrow V_0$ is a bijection with a continuously differentiable inverse. △

The following theorem states when a local inverse exists.

Theorem 2.3 (Inverse Function Theorem Lee [9, Thm C.34])**.** *If $F : U \longrightarrow V$ is continuously differentiable and if the derivative $DF(t)$ is invertible at t, then F has a local inverse at t.* «

If the determinant of the total derivative at a point is nonzero, then the condition of Theorem 2.3 holds and thus a local inverse exists at that point. We will now show how to generate a local inverse using *fixed points*.

Definition 2.4. A *fixed point* of a function $F : S \longrightarrow S$ is a point $x^* \in S$ such that $F(x^*) = x^*$. △

3 Automatic Inversion in LIN

In this section, we present the algorithm for automatic inversion in LIN and discuss the properties of the algorithm including the region of existence of the local inverse and the convergence rate.

Algorithm. Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ be a locally invertible function satisfying the conditions of Theorem 2.3. LIN estimates the local inverse of f around a given input point t in the domain by taking in a query point y in the co-domain and using a fixed-point solver that approaches $f^{-1}(y)$ as the number of iterations approaches infinity. The fixed-point iterator, adapted from [4], is defined as:

$$x_0 = t \quad \text{and} \quad x_{n+1} = x_n - [Df(t)]^{-1}(f(x_n) - y). \tag{1}$$

The total derivative of f is a matrix that depends on t, and its inverse is the matrix inverse, which is easy to calculate. Since t satisfies the condition of 2.3 the matrix inverse exists. The inverse defined by the iterator above is correct in regions around t and $f(t)$ respectively, as provided by Theorem 2.3.

The following theorem shows that the fixed-point of the iterator is the local inverse of f in a neighborhood near t.

Theorem 3.1 (Edwards [4, Pg. 166])**.** *If $f : S \longrightarrow S$ is such that $Df(t)$ is invertible, then at a point $y \in S$ the inverse $f^{-1}(y)$ is the fixed point of:*

$$T(\lambda) := \lambda - [Df(t)]^{-1}\left(f(\lambda) - y\right). \tag{2}$$

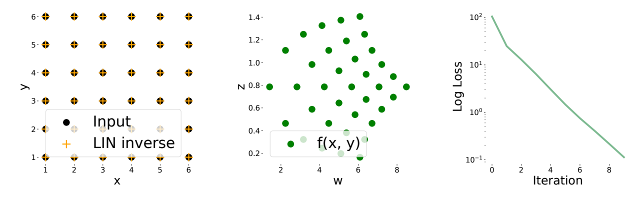
«

It is known that the fixed point iteration method for inversion *converges linearly*.

Theorem 3.2 (Atkinson [2, Pg. 79])**.** *Eq. 1 converges linearly on U_0, meaning that the loss decays exponentially.* «

4 Results

We develop a benchmark suite for coordinate changes, a step in image stylization [3, Figure 8].



(a) Inputs (black), LIN inverse (orange). **(b)** The lattice under the mapping f. **(c)** Log loss over iterations.

Figure 2. LIN local inverse matches input under polar change of variables: $w, z = f(x, y) = \left(\sqrt{x^2 + y^2}, \text{atan2}(y, x)\right)$.

Table 1. Mean ± standard deviation time (ms) per sample.

	LIN	Michel et al. [11]
Affine	$1.20 \pm 6.50 \times 10^{-2}$	$1.43 \times 10^{-2} \pm 9.00 \times 10^{-4}$
Polar	$1.40 \pm 1.50 \times 10^{-2}$	$2.02 \times 10^{-2} \pm 6.53 \times 10^{-3}$
Hyperbolic	$1.26 \pm 1.66 \times 10^{-1}$	$2.11 \times 10^{-2} \pm 5.28 \times 10^{-3}$

We evaluate LIN by calculating a local inverse for change of variables from Cartesian coordinates to: (1) an affine transformation of the coordinates, (2) polar coordinates, and (3) hyperbolic coordinates as in [3, Figures 8d, 5, 8e].

Methodology. We implement LIN in PyTorch [12] using the fixed-point inversion algorithm specified in Equation 1. We run the benchmarks on a 14-inch MacBook Pro with an M3 Pro processor and 18GB of memory. To evaluate performance, we calculate the L^2 loss between the input and inverse that LIN computes for each point in the lattice. To select an initialization point for the fixed-point iteration we sweep the lattice for a point such that LIN converges to an inverse for all points on the lattice. We believe that future work could use results such as in Lang [8, Pg. 362] to more efficiently find a good initialization.

Results. In the affine case, $f(x, y) = (x + y - 8, y)$, there exists a global inverse and LIN has zero loss after a single iteration. Local inverses always agree with global inverses and overlapping local inverses as shown in Lee [9, Pg. 660].

Figure 2 shows the result of applying LIN to a polar coordinate transformation, $f(x, y) = \left(\sqrt{x^2 + y^2}, \text{atan2}(y, x)\right)$, which has a global inverse. Figure 2c shows the loss over several iterations. Figure 1 shows a Cartesian-to-hyperbolic transformation, $f(x, y) = \left(\log\left(\sqrt{\frac{x}{y}}\right), \sqrt{xy}\right)$, which lacks a global inverse but has a local one. Figure 1c shows the loss over several iterations.

Table 1 shows the mean and standard deviation of the time (in milliseconds) to calculate a local inverse at each of 36 lattice points (e.g., the points in Figures 1 and 2). We time how long it takes LIN to execute 100 iterations of the fixed-point solver. Michel et al. [11] requires hand-coded piecewise inverses, requiring more user-effort, but providing better performance as it directly evaluates the appropriate piecewise inverse.

References

[1] Luke Anderson, Tzu-Mao Li, Jaakko Lehtinen, and Frédo Durand. 2017. Aether: an embedded domain specific sampling language for Monte Carlo rendering. *Transactions on graphics* (2017). https://doi.org/10.1145/3072959.3073704

[2] Kendall Atkinson. 1989. *An Introduction to Numerical Analysis* (2 ed.). John Wiley & Sons.

[3] Sai Bangaru, Jesse Michel, Kevin Mu, Gilbert Bernstein, Tzu-Mao Li, and Jonathan Ragan-Kelley. 2021. Systematically Differentiating Parametric Discontinuities. *Special Interest Group on Computer Graphics and Interactive Techniques* (2021). https://doi.org/10.1145/3450626.3459775

[4] C. H. Edwards. 1994. *Advanced Calculus of Several Variables*. Dover Publications.

[5] Klaus. Janich. 1984. *Topology*. Springer-Verlag.

[6] Durk P Kingma, Tim Salimans, and Max Welling. 2015. Variational Dropout and the Local Reparameterization Trick. In *Neural Information Processing Systems*.

[7] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*.

[8] S. Lang. 1993. *Real and Functional Analysis* (3rd ed.). Springer.

[9] John M Lee. 2013. *Introduction to Smooth Manifolds* (2 ed.). Springer.

[10] Kazutaka Matsuda and Meng Wang. 2020. Sparcl: a language for partially-invertible computation. (2020). https://doi.org/10.1145/3409000

[11] Jesse Michel, Kevin Mu, Xuanda Yang, Sai Praveen Bangaru, Elias Rojas Collins, Gilbert Bernstein, Jonathan Ragan-Kelley, Michael Carbin, and Tzu-Mao Li. 2024. Distributions for Compositionally Differentiating Parametric Discontinuities. *Object-Oriented Programming, Systems, Languages, and Applications* (2024). https://doi.org/10.1145/3649843

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: an imperative style, high-performance deep learning library. In *Neural Information Processing Systems*.

[13] Chung-chieh Shan and Norman Ramsey. 2017. Exact Bayesian inference by symbolic disintegration. *Principles of Programming Languages* (2017). https://doi.org/10.1145/3093333.3009852

[14] Tetsuo Yokoyama, Holger Bock Axelsen, and Robert Glück. 2008. Principles of a reversible programming language. In *Conference on Computing Frontiers*. https://doi.org/10.1145/1366230.1366239

Received 2024-07-09; accepted 2024-08-19