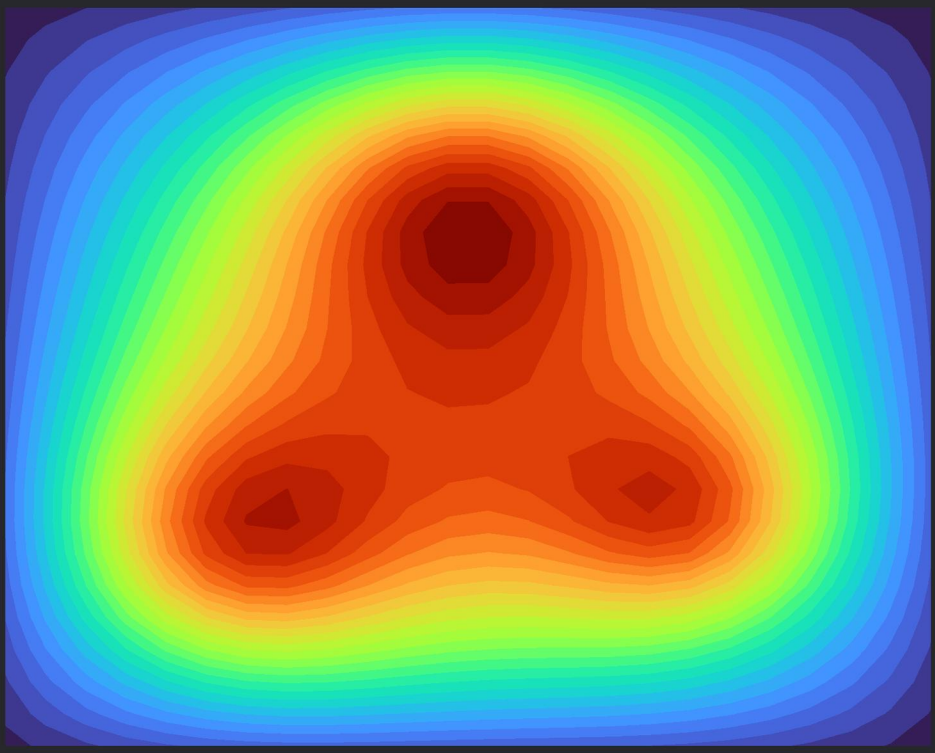

The forward equilibrium of a differentiable simulator — the state an adjoint gradient differentiates.

Differentiable Programming for Inverse Problems

Adjoint Gradients and Reproducible Solvers in PyTorch

Lena Ostrowski

ODIN PRESS

Differentiable Programming for Inverse Problems

Paperback version, First edition: 2026.
ISBN: forthcoming (Bowker allocation pending)
Trim Size: 6×9 inch. Paper color: White.

Published by Odin Press.
`https://odinpress.org`
This book was typeset in LaTeX by Odin Press from a manuscript submitted by the author.

Printed and Bound in the United States of America.

9 8 7 6 5 4 3 2 1 0

Preface

The gradient of an inverse problem used to cost a season of a graduate student's life. To recover a subsurface velocity model, a tissue's stiffness, or a contaminant's release history, you derived the adjoint equations by hand, implemented them, and then spent months hunting the single sign error that made the gradient check fail three chapters into the project. I did this for two decades, across seismic exploration, medical imaging, and environmental flow, and the hardest part of every project was the same: the adjoint.

Reverse-mode automatic differentiation changed that. The gradient that once cost a season now comes, exact, for the price of a backward pass — a small constant times one forward evaluation, independent of how many parameters you are solving for. This is not a machine-learning fashion. It is the quiet completion of the adjoint-state method the inverse-problems community has used since the 1970s: autodiff is the adjoint, computed mechanically, and a working scientist can now obtain it by differentiating the forward code rather than re-deriving the mathematics by hand. The mathematics did not change. The labor did.

This book exists because the two literatures that should have met have not. The texts that teach automatic differentiation explain the machine beautifully and then stop at the gradient, as though a gradient were a reconstruction. The texts that teach inverse problems were written before cheap differentiation and still hand-derive every adjoint, treating the gradient as the expensive part. Neither shows the whole pipeline on real problems: the exact gradient, yes, but then the regularization that the gradient cannot supply, the uncertainty quantification that turns an estimate into a result, the resolution analysis that says honestly what the data can and cannot recover — and runnable code that reproduces every number and figure. The gap between "I can compute the gradient" and "I have a trustworthy reconstruction with an error bar" is where the real work lives,

and it is the subject of this book.

What I have tried to deliver is a complete, honest account. The opening chapters prove the equivalence of autodiff and the hand adjoint and price it precisely — a seventy-five-fold advantage over finite differences at a few hundred parameters, growing with dimension. They then build the optimizer, the regularization that stabilizes an ill-posed inverse, and the Gauss–Newton uncertainty that quantifies the answer, all driven by the cheap gradient and all carried on a single running deconvolution so the machinery is never abstract. The second half spends that machinery on seven real inverse problems — gravity and subsurface flow, contaminant transport, atmospheric data assimilation, coherent-diffraction phase retrieval, and medical elastography — and reports each one's limits without flinching: the null space that sparse wells leave unrecoverable, the diffusion that erases a release's fine structure, the nonconvexity that traps phase retrieval, the stiffness contrast that elastography saturates. The closing chapter draws the boundary the rest of the book earns: when autodiff wins, and when — on nonsmooth code, behind a legacy solver, or against ill-posedness — a correct gradient is necessary and nowhere near sufficient.

Every claim that a method helps is paired with the experiment or the derivation that backs it, never with an adjective. Every numerical result is computed by the code in the appendix, reported with its uncertainty, and checked against finite differences before it is trusted. A reconstruction that arrives without the gradient check, the convergence study, and the honest error bar is, in this book's accounting, not yet a result.

The book is built in two movements. The first — The Machinery of Differentiable Inversion (Chapters 1–8) — establishes the tools: the cheap-gradient principle and reverse mode, the adjoint method and its identity with autodiff, the optimizer, the regularization that no gradient repairs, the adjoint through PDE solvers and nonlinear simulators, and the uncertainty quantification that earns the answer. The second — Inverse Problems in Practice (Chapters 9–15) — applies all of it, problem by problem, from gravity inversion to the honest boundary of the method.

I have written for the scientist who wants the method and the runnable code in one place, with nothing hand-waved and every number reproduced by the code that computed it. The gradient is free now. The inverse problem is still hard, and that is where the work — and the pleasure — lies.

Lena Ostrowski

Contents

Chapter 1

The Inverse Problem and Why Gradients Are the Bottleneck

Across the quantitative sciences the same problem recurs: recover parameters $m \in \mathbb{R}^p$ — a discretized field — from indirect, noisy measurements d_{obs} produced by a known forward model. A geophysicist wants a subsurface velocity field from surface seismograms; a radiologist wants a tissue stiffness map from boundary displacement; an environmental engineer wants a contaminant's release history from downstream concentrations. In every case the forward direction — parameters to data — is a single simulation, and the inverse direction is what we are after.

Every method that solves such a problem at scale, from steepest descent through Gauss–Newton to the stochastic optimizers of machine learning, consumes one object: the gradient of a data-misfit functional with respect to the parameters. This chapter makes a precise claim and proves it: for any forward model of realistic dimension it is the *gradient*, not the forward solve, that has historically been the bottleneck — and reverse-mode automatic differentiation removes that bottleneck because it computes the exact gradient at a cost independent of p. The first section builds the misfit from the statistics of the data and fixes notation; the second prices the gradient, derives the reverse sweep by hand, and proves the decisive scaling; the third settles exactness; the fourth places the gradient in an optimization loop, benchmarks the whole pipeline against finite differences, and proves

the one limit no gradient repairs — ill-posedness.

1.1 From data to parameters: the discrete inverse problem

We begin not with a formula but with the question the formula answers: given noisy data, which parameters are most consistent with them? The data-misfit is the answer, and it is worth deriving rather than asserting, because the derivation fixes the meaning of every term — including the regularizer that later sections lean on.

Definition 1.1 (Forward map and data-misfit)**.** The *forward map* $F : \mathbb{R}^p \to \mathbb{R}^d$ sends a parameter vector m to the predicted observations $F(m)$. Given measured data $d_{\text{obs}} \in \mathbb{R}^d$ with noise covariance Σ, the *data-misfit functional* is

$$J(m) \;=\; \tfrac{1}{2}\,\big\|F(m) - d_{\text{obs}}\big\|^2_{\Sigma^{-1}} \;+\; \alpha\, R(m), \tag{1.1}$$

where $\|r\|^2_{\Sigma^{-1}} = r^\top \Sigma^{-1} r$, R is a regularizer, and $\alpha \geq 0$ its weight. The *discrete inverse problem* is $\min_m J(m)$.

The two terms have a statistical origin. Suppose the measurement noise is Gaussian, $d_{\text{obs}} = F(m) + \eta$ with $\eta \sim \mathcal{N}(0, \Sigma)$. The likelihood of the data given the parameters is

$$p(d_{\text{obs}} \mid m) \;=\; (2\pi)^{-d/2} |\Sigma|^{-1/2} \exp\Big(-\tfrac{1}{2} \big(F(m) - d_{\text{obs}}\big)^\top \Sigma^{-1} \big(F(m) - d_{\text{obs}}\big) \Big), \tag{1.2}$$

so the maximum-likelihood estimate maximizes the exponent, i.e. minimizes $\frac{1}{2}\|F(m) - d_{\text{obs}}\|^2_{\Sigma^{-1}}$. That is the first term of J: the weighted least-squares misfit is exactly the negative log-likelihood, and the weight Σ^{-1} is the inverse noise covariance, so that precisely-measured components count more. The second term is a prior. If we believe m is itself drawn from $\mathcal{N}(0, \tau^2 I)$, Bayes' rule gives the posterior $p(m \mid d_{\text{obs}}) \propto p(d_{\text{obs}} \mid m)\, p(m)$, and maximizing it (the maximum-a-posteriori, or MAP, estimate) minimizes

$$\tfrac{1}{2}\|F(m) - d_{\text{obs}}\|^2_{\Sigma^{-1}} + \tfrac{1}{2\tau^2}\|m\|^2, \tag{1.3}$$

which is J with $R(m) = \frac{1}{2}\|m\|^2$ and $\alpha = 1/\tau^2$. The regularizer is therefore not an algebraic convenience but the encoding of prior belief, and its weight

α is the inverse prior variance: a strong prior (small τ) means a large α. We will see in Section 1.4 that this same α is what makes an unstable problem stable, so the statistical reading and the numerical reading of the regularizer are one and the same.

Why minimize at all, rather than solve $F(m) = d_{\text{obs}}$? Because that equation generally has no solution: F is nonlinear, the data are noisy so d_{obs} need not lie in the range of F, and with $d \neq p$ the system is not square. Even when a solution exists it need not be unique or stable, and these three failures have names.

Definition 1.2 (Well-posedness)**.** A problem is *well-posed* in the sense of Hadamard if a solution exists, is unique, and depends continuously on the data. A problem that fails any of these is *ill-posed*. Discrete inverse problems are typically ill-posed in the third sense: small data perturbations produce large solution changes.

In the linear case the minimizer is explicit, and its derivation already exhibits the adjoint operator that the rest of the chapter is about.

Proposition 1.3 (Regularized normal equations)**.** Let $F(m) = Am$ be linear and $R(m) = \frac{1}{2}\|m\|^2$. Then J is convex, every minimizer satisfies

$$(A^\top \Sigma^{-1} A + \alpha I)\, m \;=\; A^\top \Sigma^{-1} d_{\text{obs}}, \tag{1.4}$$

and the minimizer is unique if and only if $A^\top \Sigma^{-1} A + \alpha I$ is positive definite — in particular whenever $\alpha > 0$.

Proof. With $F(m) = Am$, $J(m) = \frac{1}{2}(Am - d_{\text{obs}})^\top \Sigma^{-1}(Am - d_{\text{obs}}) + \frac{\alpha}{2} m^\top m$. Expanding and differentiating term by term, $\nabla J(m) = A^\top \Sigma^{-1}(Am - d_{\text{obs}}) + \alpha m$, and the Hessian is $H = A^\top \Sigma^{-1} A + \alpha I$, constant in m. Since $\Sigma^{-1} \succ 0$, for any x we have $x^\top A^\top \Sigma^{-1} A x = (Ax)^\top \Sigma^{-1}(Ax) \geq 0$, so $A^\top \Sigma^{-1} A \succeq 0$ and $H \succeq \alpha I \succeq 0$; thus J is convex and its stationary points are global minima. Setting $\nabla J = 0$ gives the stated equation. If $H \succ 0$ it is invertible and the minimizer is the unique $m = H^{-1} A^\top \Sigma^{-1} d_{\text{obs}}$. If H is singular, a null vector v satisfies $Hv = 0$, hence $\nabla J(m + tv) = \nabla J(m) + tHv = \nabla J(m)$, so $m + tv$ is also a minimizer for every t and uniqueness fails. As $\alpha > 0$ forces $H \succeq \alpha I \succ 0$, regularization restores both invertibility and uniqueness. □

Two features of this result drive the whole book. First, the data enter only through $A^\top \Sigma^{-1} d_{\text{obs}}$ — the *adjoint* $A^\top$ applied to the weighted residual; we will see in Section 1.2 that this adjoint is exactly what reverse-mode

Chapter 3

The Adjoint Method: Autodiff Versus Hand-Derived Adjoints

Chapter 2 showed that reverse-mode automatic differentiation returns the exact gradient of a numeric program at a constant multiple of one evaluation, and that the transpose $A^\top$ of a linear forward operator falls out of the matrix-apply vjp with no separate code to write. The inverse-problems literature reached the same gradient three decades earlier by a different route: the adjoint-state method, derived by hand from a Lagrangian, an object Lena spent years implementing and sign-checking one project at a time. This chapter proves the two routes deliver the identical gradient. We set up the constrained misfit, derive the adjoint-state equation from the stationarity of its Lagrangian, and read off the gradient as a single inner product of the adjoint state with the parameter sensitivity of the forward model. We then prove that reverse-mode differentiation of the solver computes exactly this hand-derived adjoint — adjoint, reverse mode, and backpropagation are one method under three names. We separate the discrete adjoint a program actually computes from the continuous adjoint of the underlying equations, state when they agree, and identify the cases where a hand-coded adjoint still beats naive autodiff on memory. The chapter closes on the running deconvolution problem with a three-way gradient check: hand adjoint, autodiff, and central differences, agreeing to working precision. Every number comes from the engine listed in Appendix A.

3.1 The Lagrangian and the adjoint-state equation

The forward map sends a parameter field $m \in \mathbb{R}^p$ to a state $u \in \mathbb{R}^n$ through a state equation, and the inverse problem minimizes a misfit between the predicted and observed data over m. Writing the state equation as a residual that must vanish makes the dependence of u on m implicit, and that implicit dependence is exactly what the adjoint method is built to differentiate without ever forming it.

Definition 3.1 (The constrained inverse problem)**.** A *constrained inverse problem* consists of a *state equation* $g(u, m) = 0$ with $g : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$, defining the state $u = u(m)$ implicitly as a function of the parameter, together with an *objective* $J(u, m) : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ to be minimized over m. The *reduced objective* is $\mathcal{J}(m) = J(u(m), m)$, the objective evaluated along the solution manifold of the state equation.

The difficulty is visible already: to differentiate $\mathcal{J}(m)$ we need $\mathrm{d}u/\mathrm{d}m$, the sensitivity of the implicitly-defined state, a dense $n \times p$ object we can neither store nor want. The adjoint method removes it. The device is a Lagrangian that adjoins the state equation to the objective with a multiplier free to be chosen.

Definition 3.2 (The Lagrangian and the adjoint state)**.** The *Lagrangian* of the constrained inverse problem is

$$\mathcal{L}(u, m, \lambda) = J(u, m) + \lambda^\top g(u, m), \tag{3.1}$$

with $\lambda \in \mathbb{R}^n$ the *adjoint state* (the Lagrange multiplier of the state equation). Because $g(u(m), m) = 0$ on the solution manifold, $\mathcal{L}(u(m), m, \lambda) = \mathcal{J}(m)$ for *every* λ, so λ may be chosen freely to simplify the gradient.

That freedom is the whole method. Differentiating the reduced objective directly forces $\mathrm{d}u/\mathrm{d}m$ into the open; differentiating the Lagrangian lets us choose λ to annihilate its coefficient.

Theorem 3.3 (The adjoint-state equation)**.** Let g and J be continuously differentiable and let $\partial g/\partial u$ be nonsingular at $(u(m), m)$. Stationarity of $\mathcal{L}$ with respect to the state u — the requirement $\partial \mathcal{L}/\partial u = 0$ that frees the gradient of the unknown sensitivity — holds iff the adjoint state solves the *adjoint-state equation*

$$\Big(\frac{\partial g}{\partial u}\Big)^{\top}\lambda = -\Big(\frac{\partial J}{\partial u}\Big)^{\top}. \tag{3.2}$$

Proof. The total derivative of the reduced objective is, by the chain rule through $u(m)$,

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}m} = \frac{\partial J}{\partial m} + \frac{\partial J}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}m}. \tag{3.3}$$

Differentiating the state equation $g(u(m), m) = 0$ totally in m gives $\frac{\partial g}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}m} + \frac{\partial g}{\partial m} = 0$, hence $\frac{\mathrm{d}u}{\mathrm{d}m} = -(\frac{\partial g}{\partial u})^{-1}\frac{\partial g}{\partial m}$, which exists because $\partial g/\partial u$ is nonsingular. Substituting,

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}m} = \frac{\partial J}{\partial m} - \frac{\partial J}{\partial u}\Big(\frac{\partial g}{\partial u}\Big)^{-1}\frac{\partial g}{\partial m}. \tag{3.4}$$

Define $\lambda^{\top} = -\frac{\partial J}{\partial u}(\frac{\partial g}{\partial u})^{-1}$, the choice that collects the bracket; transposing gives $(\frac{\partial g}{\partial u})^{\top}\lambda = -(\frac{\partial J}{\partial u})^{\top}$, the stated equation. This is precisely the condition $\partial\mathcal{L}/\partial u = \frac{\partial J}{\partial u} + \lambda^{\top}\frac{\partial g}{\partial u} = 0$ read as a column system, so stationarity of $\mathcal{L}$ in u and the adjoint-state equation are the same statement. □

The adjoint-state equation is a single linear system of the same size as the forward problem, and it inherits the forward operator's structure transposed.

Proposition 3.4 (Existence, uniqueness, and size of the adjoint solve)**.** When $\partial g/\partial u$ is nonsingular the adjoint state λ exists and is unique, the adjoint-state equation is an $n \times n$ linear system — the same dimension as the forward state equation — and its operator is the transpose of the forward Jacobian $\partial g/\partial u$. If the forward solve factorizes $\partial g/\partial u$, the adjoint solve reuses that factorization transposed.

Proof. The adjoint-state equation $(\frac{\partial g}{\partial u})^{\top}\lambda = -(\frac{\partial J}{\partial u})^{\top}$ has system matrix $(\frac{\partial g}{\partial u})^{\top} \in \mathbb{R}^{n\times n}$, which is nonsingular iff $\partial g/\partial u$ is, so λ exists and is unique exactly under the theorem's hypothesis. The system has n unknowns and n equations, matching the forward state dimension. If the forward solver computes an LU factorization $\partial g/\partial u = LU$, then $(\frac{\partial g}{\partial u})^{\top} = U^{\top}L^{\top}$ is a factorization of the adjoint operator into triangular factors already in hand, so the adjoint solve costs two triangular back-substitutions rather than a fresh factorization. □

For a linear forward model the adjoint-state equation loses its implicit character entirely.

An engine monograph.