



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

Copyright © 2026 by **Lena Ostrowski**.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, recording, or otherwise—without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

Paperback version, First edition: 2026.

ISBN: forthcoming (Bowker allocation pending)

Trim Size: 6 × 9 inch. Paper color: White.

Source-code copyright. All Python source code listings reproduced in this volume—including those in the Appendix, the chapter bodies, and any auxiliary file distributed alongside this book—are the copyrighted intellectual property of **Lena Ostrowski**. No reproduction, redistribution, adaptation, translation into another programming language, or incorporation into any derivative work, whether academic, commercial, or otherwise, is permitted without the prior written permission of the copyright holder. Single-machine execution by the individual purchaser of this volume for the sole purpose of reproducing a figure or table from this book is granted as a limited, non-transferable, non-sublicensable use and does not constitute redistribution.

Published by Odin Press.

<https://odinpress.org>

This book was typeset in L^AT_EX 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

Preface	vii
List of Figures	xv
I The Machinery of Differentiable Inversion	1
1 The Inverse Problem and Why Gradients Are the Bottleneck	3
1.1 From data to parameters: the discrete inverse problem	4
1.2 Why the gradient is the bottleneck: the cost of a derivative	7
1.3 Exactness: what a gradient computed by differentiation actually equals	14
1.4 The gradient in the optimization loop, and what ill-posedness still costs	18
2 Reverse-Mode Automatic Differentiation: Exact Gradients at Constant Cost	29
2.1 The computational graph and the chain rule as a sweep	30
2.2 Vector-Jacobian products: the primitive operation	32
2.3 The constant-overhead theorem	35
2.4 Forward mode and when it wins	37
2.5 Memory: checkpointing and recomputation	40
2.6 A working engine on the running problem	43
3 The Adjoint Method: Autodiff Versus Hand-Derived Adjoints	49
3.1 The Lagrangian and the adjoint-state equation	50
3.2 The adjoint gradient formula	53
3.3 Autodiff is the adjoint, computed mechanically	58

3.4	Discrete versus continuous adjoints	61
3.5	When the hand adjoint still earns its keep	65
3.6	A side-by-side gradient check on the running problem	70
4	Optimization Machinery: Descent, L-BFGS, Conditioning, and Convergence	75
4.1	Steepest descent and why step size matters	76
4.2	Line search and the Wolfe conditions	80
4.3	L-BFGS: curvature without the Hessian	83
4.4	The Gauss-Newton and Levenberg-Marquardt steps	87
4.5	Conditioning and preconditioning	90
4.6	The full solver on the running problem	93
5	Ill-Posedness, Regularization, and Identifiability	97
5.1	Ill-posedness in the singular value spectrum	98
5.2	Tikhonov regularization and filter factors	101
5.3	The discrete Picard condition	104
5.4	Choosing the parameter: the L-curve and discrepancy	107
5.5	Identifiability and the null space	110
5.6	Regularization on the running problem	113
6	PDE-Constrained Inversion: Differentiating Through a Solver	119
6.1	The PDE-constrained misfit	120
6.2	The continuous adjoint PDE	123
6.3	Discretize-then-optimize versus optimize-then-discretize	128
6.4	Differentiating the solver with autodiff	131
6.5	Cost and memory of a PDE gradient	133
6.6	A worked PDE inversion	136
7	Simulator-Based Inversion: Differentiable Forward Models	141
7.1	When the forward model contains an inner solve	142
7.2	Differentiating a converged fixed point	144
7.3	Custom vector-Jacobian rules	148
7.4	Pitfalls: unrolling, stop-gradient, and convergence	151
7.5	A differentiable physics simulator	154
7.6	Inversion through the simulator	157
8	Uncertainty Quantification: Gauss–Newton and the Hessian via Autodiff	163

8.1	The Bayesian inverse problem	164
8.2	The Gauss-Newton Hessian	167
8.3	The Laplace approximation	170
8.4	Computing posterior covariance at scale	172
8.5	Limits of the linearized posterior	176
8.6	Uncertainty on the running problem	179
II	Inverse Problems in Practice	185
9	Gravity and Potential-Field Inversion	187
9.1	The forward gravity operator and its linearity	188
9.2	The adjoint gradient and the inversion algorithm	191
9.3	Resolution, nonuniqueness, and the null space	196
9.4	Regularization: depth weighting and Tikhonov	201
9.5	The full inversion and the autodiff-vs-classical benchmark	205
10	Subsurface Permeability Identification in Darcy Flow	211
10.1	The Darcy-flow forward model	212
10.2	Parameterizing and regularizing the permeability	216
10.3	The adjoint gradient through the flow solver	218
10.4	Resolution and depth of investigation	221
10.5	The full permeability inversion	224
10.6	Benchmark and failure modes	229
11	Advection–Diffusion Source Inversion	233
11.1	The advection-diffusion forward model	234
11.2	Why source inversion is ill-posed in time	238
11.3	The time-reversed adjoint gradient	241
11.4	Regularization in time	244
11.5	The full source reconstruction	246
11.6	Sensitivity to sensor placement	248
12	4D-Var Data Assimilation for Shallow-Water Flow	253
12.1	The shallow-water forward model	254
12.2	The 4D-Var cost functional	258
12.3	The adjoint model and the gradient	260
12.4	The incremental Gauss-Newton outer loop	263
12.5	The full assimilation	266
12.6	Window length and observation density	270

13 Phase Retrieval and Coherent Diffraction Imaging	275
13.1 Coherent diffraction and the lost phase	276
13.2 The Fourier-magnitude forward operator	280
13.3 The autodiff gradient of the intensity misfit	282
13.4 Constraints: support and oversampling	285
13.5 The full reconstruction	287
13.6 Nonconvexity: basins, restarts, and stagnation	291
14 Elastography: Recovering a Spatially-Varying Modulus Field	297
14.1 The elastic-wave forward model	298
14.2 Parameterizing the modulus field	301
14.3 The adjoint gradient through the wave solver	303
14.4 Resolution and the stiffness contrast	306
14.5 The full elastography inversion	308
14.6 Uncertainty on the recovered stiffness	313
15 When Not to Use Autodiff: The Honest Boundary	319
15.1 What the running problem taught us	320
15.2 When autodiff wins, and the cost model	322
15.3 When it does not: nonsmoothness, memory, and legacy code	326
15.4 What no gradient repairs: ill-posedness revisited	330
15.5 A decision checklist and the road ahead	333
A Source Code	339
A.1 Chapter 1 — The Inverse Problem and Why Gradients Are the Bottleneck	339
A.2 Chapter 2 — Reverse-Mode Automatic Differentiation: Exact Gradients at Constant Cost	350
A.3 Chapter 3 — The Adjoint Method: Autodiff Versus Hand-Derived Adjoints	359
A.4 Chapter 4 — Optimization Machinery: Descent, L-BFGS, Conditioning, and Convergence	369
A.5 Chapter 5 — Ill-Posedness, Regularization, and Identifiability	378
A.6 Chapter 6 — PDE-Constrained Inversion: Differentiating Through a Solver	387
A.7 Chapter 7 — Simulator-Based Inversion: Differentiable Forward Models	396

A.8	Chapter 8 — Uncertainty Quantification: Gauss–Newton and the Hessian via Autodiff	405
A.9	Chapter 9 — Gravity and Potential-Field Inversion . . .	413
A.10	Chapter 10 — Subsurface Permeability Identification in Darcy Flow	426
A.11	Chapter 11 — Advection–Diffusion Source Inversion . .	436
A.12	Chapter 12 — 4D-Var Data Assimilation for Shallow-Water Flow	445
A.13	Chapter 13 — Phase Retrieval and Coherent Diffraction Imaging	455
A.14	Chapter 14 — Elastography: Recovering a Spatially-Varying Modulus Field	463
A.15	Chapter 15 — When Not to Use Autodiff: The Honest Boundary	472
 Index		 485
 About the Author		 489

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* $F : \mathbb{R}^p \rightarrow \mathbb{R}^d$ 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

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

where $\|r\|_{\Sigma^{-1}}^2 = r^\top \Sigma^{-1} r$, 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

$$p(d_{\text{obs}} | m) = (2\pi)^{-d/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2} (F(m) - d_{\text{obs}})^\top \Sigma^{-1} (F(m) - d_{\text{obs}})\right), \quad (1.2)$$

so the maximum-likelihood estimate maximizes the exponent, i.e. minimizes $\frac{1}{2} \|F(m) - d_{\text{obs}}\|_{\Sigma^{-1}}^2$. 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 $p(m | d_{\text{obs}}) \propto p(d_{\text{obs}} | m) p(m)$, and maximizing it (the maximum-a-posteriori, or MAP, estimate) minimizes

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

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

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

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. \square

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 Adjoint

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 $g : \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}^n$, defining the state $u = u(m)$ implicitly as a function of the parameter, together with an *objective* $J(u, m) : \mathbb{R}^n \times \mathbb{R}^p \rightarrow \mathbb{R}$ 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 du/dm , 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

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

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 du/dm 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*

$$\left(\frac{\partial g}{\partial u}\right)^\top \lambda = -\left(\frac{\partial J}{\partial u}\right)^\top. \quad (3.2)$$

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

$$\frac{d\mathcal{J}}{dm} = \frac{\partial J}{\partial m} + \frac{\partial J}{\partial u} \frac{du}{dm}. \quad (3.3)$$

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

$$\frac{d\mathcal{J}}{dm} = \frac{\partial J}{\partial m} - \frac{\partial J}{\partial u} \left(\frac{\partial g}{\partial u}\right)^{-1} \frac{\partial g}{\partial m}. \quad (3.4)$$

Define $\lambda^\top = -\frac{\partial J}{\partial u} \left(\frac{\partial g}{\partial u}\right)^{-1}$, the choice that collects the bracket; transposing gives $\left(\frac{\partial g}{\partial u}\right)^\top \lambda = -\left(\frac{\partial J}{\partial u}\right)^\top$, 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. \square

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 $\left(\frac{\partial g}{\partial u}\right)^\top \lambda = -\left(\frac{\partial J}{\partial u}\right)^\top$ has system matrix $\left(\frac{\partial g}{\partial u}\right)^\top \in \mathbb{R}^{n \times n}$, 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 $\left(\frac{\partial g}{\partial u}\right)^\top = U^\top L^\top$ 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. \square

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

An engine monograph.

