Deep Learning: Making It Trainable

Geometry, Dynamics, and the Machine

Heman Shakeri

2026-08-02

Table of contents

Deep Learning: Making It Trainable 1
How the book moves . . . 1
The route at a glance . . . 2
Why this book . . . 2
How to read this book . . . 2
Proof modes and reading paths . . . 3
The computational harness . . . 4
Reading contract . . . 4
Exercise navigation . . . 5
Colophon and edition note . . . 5
Suggested citation . . . 6

On-Ramp: The Minimum Mechanics 7
The 20–30 minute diagnostic . . . 7
The minimal route through *Making It Learnable* . . . 7
One minimal training loop . . . 8
You are ready for Chapter 1 when… . . . 8

Act 0 — The computational arena 9

1. The Count Is Not the Cost: Arithmetic Intensity and the Memory Hierarchy 13
1.1. Two resource clocks . . . 13
1.1.1. Proof . . . 14
1.2. A zero-count kernel still consumes a clock . . . 15
1.3. The denominator is an algorithmic choice . . . 17
1.4. Hardware-claim boundary . . . 18
1.5. Check yourself . . . 18
1.6. Okay, so — . . . 18
1.7. Sources and further reading . . . 18
1.8. Exercises . . . 19

2. One Pass, Two Failures: Streaming State and Stable Variance 21
2.1. The impossible output . . . 21
2.1.1. The shift test . . . 23
2.2. Diagnose the subtraction . . . 25
2.3. Preserve the centered invariant . . . 26
2.3.1. Proof . . . 27
2.3.2. The mechanism in code . . . 28
2.4. One state must also combine . . . 29
2.4.1. Proof . . . 30
2.4.2. Associative in which arithmetic? . . . 31

2.5. Choosing the state is choosing the algorithm . . . 32
2.6. Okay, so — . . . 33
2.7. Sources and further reading . . . 33
2.8. Exercises . . . 33

3. The Grid Under the Update: Range, Resolution, and Rounding 35
3.1. The update that never lands . . . 35
3.2. A floating format is a changing grid . . . 37
3.2.1. Proof . . . 38
3.2.2. Three nearby quantities that are not synonyms . . . 38
3.3. Range and resolution are separate budgets . . . 39
3.4. Two losses that no longer look alike . . . 40
3.4.1. Proof . . . 41
3.5. Resolution must be allocated . . . 41
3.6. Changing the rounding rule . . . 42
3.6.1. Proof . . . 43
3.7. Diagnose before prescribing . . . 45
3.8. Okay, so — . . . 46
3.9. Sources and further reading . . . 46
3.10. Exercises . . . 46

4. One Step, Many Clocks: Quadratic Stability and Conditioning 49
4.1. Rotate the trace, not the story . . . 49
4.2. The update matrix is the instrument . . . 52
4.2.1. Proof . . . 52
4.3. The condition number is a waiting-time tax . . . 53
4.3.1. Proof . . . 54
4.4. Change the geometry seen by the update . . . 54
4.5. A stochastic update has no point endpoint . . . 56
4.5.1. Proof . . . 57
4.6. Okay, so — . . . 60
4.7. Sources and further reading . . . 60
4.8. Exercises . . . 61

Act I — Geometry of high dimensions 63

5. The Safe Zone Is Logarithmic: Sub-Gaussian Concentration and Simultaneous Control 67
5.1. A million chances to fail . . . 67
5.2. Exponentiate, then optimize . . . 69
5.2.1. Proof . . . 70
5.3. Coefficient energy controls a sum . . . 70
5.3.1. Proof . . . 71
5.4. Pay only a logarithm for simultaneous control . . . 71
5.4.1. Proof . . . 71
5.5. Four instruments for the same safe zone . . . 72
5.6. Width without scale explosion . . . 73
5.7. Geometry meets the finite grid . . . 76
5.8. Okay, so — . . . 76
5.9. Sources and further reading . . . 76

5.10. Exercises . . . 77

6. One Square, Two Regimes: Sub-Exponential Tails and Clipping Bias 79
6.1. Inspect the domain, not only the variance . . . 79
6.2. The square keeps an exponential contract . . . 83
6.2.1. Proof . . . 83
6.3. A local parabola creates two regimes . . . 84
6.3.1. Proof . . . 85
6.4. Watch the optimizer hit its boundary . . . 85
6.5. When no positive MGF exists . . . 88
6.5.1. Proof . . . 89
6.6. Okay, so — . . . 92
6.7. Sources and further reading . . . 92
6.8. Exercises . . . 92

7. The Sphere Has a Finite Price: Epsilon-Nets and Uniform Operator Control 95
7.1. A sampled maximum is not a supremum . . . 95
7.2. Make the continuum finite . . . 99
7.2.1. Proof . . . 99
7.3. Fill the gaps deterministically . . . 100
7.3.1. Proof . . . 100
7.4. Tail budget plus interpolation . . . 100
7.4.1. Proof . . . 101
7.5. Which tails can afford the sphere? . . . 101
7.6. Okay, so — . . . 104
7.7. Sources and further reading . . . 104
7.8. Exercises . . . 104

8. One Average, Two Edges: Random Operators and Singular Values 107
8.1. The direction can adapt to the matrix . . . 107
8.2. Two edges define the stretch interval . . . 110
8.2.1. Proof . . . 110
8.3. From fixed rows to all directions . . . 111
8.3.1. Diagnostic proof sketch . . . 111
8.4. Aspect ratio opens the lower edge . . . 112
8.4.1. Proof with pointer . . . 112
8.5. One-sided instruments answer one-sided questions . . . 115
8.6. Okay, so — . . . 117
8.7. Sources and further reading . . . 118
8.8. Exercises . . . 118

9. The Bulk Is Not a Verdict: Marchenko–Pastur and Finite-Null Calibration 119
9.1. The outlier is made of noise . . . 119
9.2. Name the estimator before naming the law . . . 122
9.2.1. Proof . . . 122
9.3. The bulk law is asymptotic . . . 123
9.3.1. Proof with pointer . . . 123
9.4. A deterministic bulk can still be a poor covariance estimate . . . 124
9.4.1. Diagnostic proof sketch . . . 124

9.5. Calibrate the finite statistic . . . 124
9.5.1. Proof . . . 128
9.6. Signal needs an alternative model . . . 128
9.6.1. Proof with pointer . . . 128
9.7. Okay, so — . . . 132
9.8. Sources and further reading . . . 132
9.9. Exercises . . . 132

10. Full Rank, Few Directions: Effective Rank and Covariance Complexity 135
10.1. Same rank, different sample demand . . . 135
10.2. Replace the coordinate count by spectral mass . . . 138
10.2.1. Proof . . . 139
10.2.2. Proof . . . 139
10.3. Why this quantity enters covariance error . . . 140
10.3.1. Proof . . . 140
10.4. A usable covariance theorem . . . 140
10.4.1. Proof with pointer . . . 141
10.4.2. Diagnostic proof sketch . . . 141
10.5. Five dimensions that must not be conflated . . . 141
10.6. The Act I handoff . . . 142
10.7. Okay, so — . . . 142
10.8. Sources and further reading . . . 142
10.9. Exercises . . . 143

Act II — Dynamics of optimization 145

11. The Gradient Has a Memory: Reverse Accumulation and Checkpointing 149
11.1. One node, two debts . . . 149
11.2. One derivative, two orientations . . . 152
11.2.1. Proof . . . 153
11.3. The reverse rule on a graph . . . 153
11.3.1. Proof . . . 153
11.4. Instructions are cheap; saved values are not . . . 154
11.5. Trade retained states for replay . . . 154
11.5.1. Proof . . . 155
11.6. Replay is part of the numerical contract . . . 157
11.7. The next diagnostic object . . . 157
11.8. Okay, so — . . . 157
11.9. Sources and further reading . . . 158
11.10Exercises . . . 158

12. Curvature Is Not One Matrix: Hessian, Generalized Gauss–Newton, Fisher, and Matrix-Free Products 159
12.1. Hold the problem fixed; change the object . . . 159
12.2. The term a positive-semidefinite surrogate drops . . . 162
12.2.1. Proof . . . 162
12.3. Three expectations hiding behind two words . . . 162
12.3.1. Proof of the score identity . . . 163

12.4. Ask for an action, not a dense matrix . . . 164
12.4.1. Proof . . . 164
12.5. Object, representation, and solver are separate choices . . . 166
12.6. One linear solver, two lenses . . . 167
12.7. Okay, so — . . . 168
12.8. Sources and further reading . . . 168
12.9. Exercises . . . 169

13. The Mean Gradient Is Not the Regime: Conditional Noise and Stochastic Dynamics 171
13.1. Declare the estimator before its variance . . . 171
13.1.1. Proof . . . 171
13.2. Watch the regime change while the objective stays fixed . . . 173
13.3. Batch size is an assumption-bearing lever . . . 177
13.4. Clipping repairs a tail by changing the target . . . 177
13.5. Acceleration is a branch, not a survey . . . 178
13.6. Check yourself . . . 178
13.7. Okay, so — . . . 178
13.8. Sources and further reading . . . 178
13.9. Exercises . . . 178

14. Critical on Average, Broken by Direction: Initialization and Dynamical Isometry 181
14.1. Criticality is a recursion with assumptions . . . 181
14.1.1. Diagnostic proof . . . 182
14.2. The moment survived; the distinction did not . . . 182
14.2.1. Diagnostic derivation . . . 182
14.3. Width is relative to depth . . . 183
14.3.1. Proof . . . 183
14.4. Average control does not survive multiplication . . . 185
14.5. Trace control is not edge control . . . 187
14.5.1. Proof . . . 187
14.6. The de-branded object first . . . 188
14.7. Check yourself . . . 188
14.8. Okay, so — . . . 189
14.9. Sources and further reading . . . 189
14.10Exercises . . . 189

15. Normalization Chooses an Invariance: Coordinate Statistics and Derivative Paths 191
15.1. Two maps, two quotients . . . 191
15.1.1. Proof . . . 192
15.2. Epsilon changes the map . . . 193
15.3. Train state and evaluation state are different estimators . . . 194
15.4. Placement changes the derivative path . . . 196
15.5. Brand bridge and claim boundary . . . 197
15.6. Check yourself . . . 197
15.7. Okay, so — . . . 197
15.8. Sources and further reading . . . 197
15.9. Exercises . . . 198

16. The Boundary Moves: Progressive Sharpening and Edge-of-Stability Diagnostics **199**
16.1. The fixed control . . . 199
16.1.1. Proof . . . 199
16.2. Instrument the path, not one endpoint . . . 200
16.3. One boundary, two diagnostic objects . . . 202
16.4. Local Taylor information is historical . . . 203
16.5. From witness to empirical phenomenon . . . 203
16.6. Numerical stability returns . . . 204
16.7. Check yourself . . . 204
16.8. Okay, so — . . . 204
16.9. Sources and further reading . . . 204
16.10Exercises . . . 205

17. The Norm Chooses the Update: Matrix Geometry and Orthogonalized Optimization **207**
17.1. One linearization, three unit balls . . . 207
17.1.1. Proof . . . 207
17.2. A polynomial is an approximation contract . . . 209
17.3. The update spectrum is not the weight spectrum . . . 210
17.4. One complete matrix-block step . . . 213
17.5. Brand bridge . . . 214
17.6. The terminal capability: autopsy before adoption . . . 214
17.7. Check yourself . . . 215
17.8. Okay, so — . . . 215
17.9. Sources and further reading . . . 215
17.10Exercises . . . 216

One Spike, Four Suspects **217**
Open the panel . . . 217
Suspect 1: precision . . . 219
Suspect 2: curvature . . . 219
Suspect 3: implementation state . . . 220
Normalization state . . . 220
Update geometry . . . 220
Suspect 4: estimator and data . . . 220
Completed incident report . . . 220

Beyond This Volume **223**
Similarity-weighted aggregation and kernel geometry . . . 223
IO-aware exact algorithms . . . 223
Parallel recurrence and associativity . . . 223
Lazy and feature-learning regimes . . . 224

Provenance and Acknowledgements **225**

References **227**

Appendices **233**

A. Notation Covenant **233**
A.1. Inherited core . . . 233
A.2. New in this volume . . . 233

B. Rosetta: Primitive to Literature Name **235**

C. Executable Claim Contract **237**
C.1. Reading a provenance strip . . . 237

D. The Incident Card **239**
D.1. Three cumulative act assignments . . . 239
D.1.1. Act 0 — execution contract . . . 239
D.1.2. Act I — geometric locator . . . 239
D.1.3. Act II — causal incident . . . 239
D.2. Blank incident report . . . 240
D.3. Compact instrument panel . . . 241

Deep Learning: Making It Trainable

Training fails in ways that look deceptively alike.

A loss spike may be caused by a step that crosses a curvature boundary. It may be caused by a value that leaves the representable range. It may be caused by a noisy direction that only looks like signal. It may be caused by an implementation that moves the right mathematical object through the machine in the wrong way.

The trace alone does not name the cause. We need instruments.

This book develops those instruments through three lenses:

- **geometry**, which describes the high-dimensional arena;
- **dynamics**, which describes how the update moves through it;
- **algorithms and systems**, which describe the arithmetic, state, and data movement that make the update physically executable.

The lenses are coupled. A repair can move the failure instead of removing it: a larger step can accelerate a flat direction while destabilizing a sharp one; clipping can bound an extreme update while biasing the estimator; normalizing scale can change both the Jacobian and the curvature. The book therefore audits what each repair changes, what it preserves, and which new boundary it creates.

The governing question is:

> What mathematical and physical conditions make a modern learning system trainable, and which diagnostic tells us which condition failed?

How the book moves

Chapters begin with something that breaks. You will predict a cause before the diagnostic is revealed. The mathematics then arrives as the smallest language that separates the competing explanations, and the chapter closes with a repair and an audit of its boundary.

The recurring order is:

$$\text{Problem} \to \text{Prediction} \to \text{Instrumentation} \to \text{Diagnosis} \to \text{Theory} \to \text{Solution} \to \text{Audit}$$

The book is inductive on purpose. A finished term can make an object feel settled before its assumptions are visible. We therefore build the primitive first and supply the conventional literature name when that name becomes useful for search, comparison, and primary sources.

Each chapter closes its ledger under the heading "Okay, so —": what was inherited, what changed, what was instrumented, what is now established, and what remains unresolved. The unresolved entry is always a question; the next chapter opens with the phenomenon that forces it.

The route at a glance

Stage	Question the reader gains the instrument to answer
On-ramp	Do I have the minimum tensor, loss, reverse-accumulation, mini-batch, and update mechanics needed to begin?
Act 0 · C01–C04	What did the machine actually execute, retain, round, and move?
Act I · C05–C10	Which directions and spectra should high dimension make typical, and what finite evidence turns that expectation into a verdict?
Act II · C11–C17	How do derivatives, stochastic estimators, curvature, normalization, and matrix-valued updates change along a trajectory?
Coda	Which control identifies the cause of one incident rather than merely detecting it?
Beyond this volume	Where do these instruments return in IO-aware interaction, parallel recurrence, and feature-learning research?

The detailed contents locates every theorem and subsection. This table is the argument to remember before opening that index.

Why this book

Goodfellow et al. (2016) owns the broad mechanics and modeling vocabulary of modern deep learning; this volume assumes only the small on-ramp above and does not rebuild that survey. Vershynin (2026) owns the general high-dimensional probability theory, while Act I turns selected results into finite diagnostic controls. Bottou et al. (2018) and Schmidt (2026) map optimization theory and practice; this book follows quantities across an actual trajectory instead of cataloging methods. Roberts and Yaida (2022) is the derivational companion for wide and finite-width statistical models, whereas the witnesses here measure where a declared approximation succeeds or fails. Austin et al. (2025) and Tazi et al. (2025) are systems-side companions for accelerator topology and distributed execution. The division of labor is deliberate: this book connects arithmetic, geometry, and dynamics through differential diagnosis, typed evidence, and executable controls; it does not claim new research theorems in place of those sources.

How to read this book

As a student. Read in order, and read each chapter as an investigation rather than a summary target. Stop at the Prediction and commit to an answer specific enough to fail; the gap between your answer and the diagnostic is the chapter's actual content. Work the three exercise kinds as three different muscles: Pencil is derivation, Code is implementation, and Audit is judgment. Let none substitute for another. Keep your own closing ledger as you go: the object you can now name, the failure you can now detect,

the rival you can now acquit, and the question still open. By the Coda, that notebook should read as an instrument panel, not a collection of formulas.

As a researcher or practitioner. Enter through your incident, not through Chapter 1. Complete the first fields of the Incident Card: symptom, ranked suspects, and declared contract. Do that before shopping for a remedy, then let the route table above send you to the missing instrument. A result from a clean example transfers exactly as far as its contract survives: the same object, the same estimator, the same axes and denominator, a comparable numerical contract, and a control that still separates the same rivals. When one of these breaks, the conclusion has quietly moved from identification to detection. Say so, and measure again.

As an instructor. The chapter structure is a lesson plan. Assign the opening phenomenon and require predictions before class; spend the class deriving the minimum theory that separates the rivals; put the deliberate error in the computational session and let it produce a reasonable-looking answer; grade the declared contract and the control, not the polish of the final artifact. A companion instructor note on assessment patterns, broken implementations, and assistance policy lives with the course materials rather than in this volume.

With a language model beside you. This book assumes you may study it with a model open in the next window, and treats that neither as cheating nor as a substitute for the work. What has changed is the price of fluency: a restatement, a derivation, an implementation, or a plausible diagnosis now costs seconds. What has not changed is the price of judgment: deciding what was actually computed, which object it is, which assumptions carry the conclusion, and what the evidence permits you to say. The reading contract below was written for exactly that judgment, and every clause doubles as a rule for model-assisted work.

Predict before you ask, and keep your prediction distinct from the model's. Make generated code declare its object, shapes, reduction axes, denominator, and dtype before you accept it. Treat it as a proposed experiment, and add the assertion that would fail if the contract were violated. Use the model as tutor, translator, and implementer where translation is the bottleneck; use it above all as an adversary. Ask it for the rival mechanism, the hidden assumption, the counterexample, and the control under which your preferred explanation should fail. Do not let it serve as an authority: fluency is not evidence, and a polished explanation does not enlarge the permission granted by the proof or experiment beneath it. The test of any session is whether you could defend the result after the conversation disappeared: name the object, state the contract, run the control, and say what remains uncertain. That test, not possession of an answer, is what this book trains.

Proof modes and reading paths

Three proof labels carry different permissions. A **Proof** supplies the logical argument under the displayed assumptions. A **Diagnostic proof sketch** exposes the mechanism and the exact usable consequence while marking the technical steps it omits. A **Proof with pointer** states the theorem in the form used here and locates a complete external proof; the pointer is part of the contract, not permission to drop assumptions.

The acts deliberately change the lens balance. Act 0 is algorithmic-primary: formulas meet representation, state, and traffic. Act I is geometric-primary by construction. Act II turns the dynamic lens forward and uses geometry and systems as controls. The three lenses therefore recur across the book without occupying equal space in every act.

Read in chapter order for the full course argument. For practitioner triage, begin with C01, C03, C13, and C16 before returning to the supporting theory. For a theory-first route, read Act I after C04, then enter Act II through C12. All routes should end with C17 and the Coda, where the separate instruments must diagnose one event together.

The computational harness

One codebase grows with the argument. The first capability is deliberately small: a state that can carry stable moments through a stream and merge partial states without revisiting the data. Later chapters add precision diagnostics, spectral instruments, curvature products, and update audits.

Visible code is divided into:

1. the **equation**;
2. a 10–25 line **mechanism kernel** whenever the object permits one;
3. two or three **critical assertions** that protect the interpretation;
4. a compact **provenance strip** naming the claim, seed, dtype, device, estimator, and artifact;
5. the full **harness**, linked rather than repeatedly printed.

Every published numerical result is either produced on the page or linked to a machine-readable provenance record. Hash verification, manifest activation, path handling, and JSON orchestration still execute, but their centralized implementation lives in the Executable Claim Contract. Printed code emphasizes the semantic change from one chapter to the next.

Reading contract

The book assumes graduate linear algebra and probability, plus working Python and NumPy. It does **not** assume that you have completed a deep-learning course. Before Chapter 1, the 20–30 minute on-ramp diagnoses the few mechanics used here and provides one minimal training-loop notebook. Any gap is repaired through three selected stops in the sibling volume, *Deep Learning: Making It Learnable*, not by requiring the entire earlier course.

The destination is not recall of a catalog. It is the ability to read a current optimization or systems paper and identify:

- the assumption carrying the claim;
- the quantity that was actually measured;
- the estimator, reduction axes, and denominator behind that quantity;
- the numerical and hardware contract;
- the regime in which the conclusion survives;
- the control that would distinguish the proposed mechanism from a plausible rival.

Exercise navigation

The three tags state the expected deliverable: **Pencil** ends in a derivation or counterexample, **Code** in an executable artifact plus a short interpretation, and **Audit** in a claim/assumption/control verdict. Within each chapter, Exercises 1–3 are **Core**, Exercise 4 is **Extension**, and Exercise 5 is **Research**. Any additional exercise carries an explicit route label. Unless a prompt says otherwise:

Route	Typical time	Prerequisite and tool	Hint availability
Core	15–35 minutes each	the chapter itself; paper and pencil or its first mechanism kernel	no hidden prerequisite; a marked hint may be opened after a written prediction
Extension	45–75 minutes	the current act and its pinned harness	the relevant invariant or control is named in the prompt
Research	60–120 minutes	the Sources reading order and Paper Autopsy fields named in the prompt	the protocol supplies structure, not a conclusion
Act checkpoint	60–90 minutes	the completed act	the blank Incident Card supplies the scaffold

The final exercise of C04, C10, and C17 is a cumulative Incident Card. By the Coda, numerical state, spectral evidence, and stochastic estimators have already been audited separately. The final card asks them to work together.

Colophon and edition note

The HTML edition is canonical; the PDF is a derived, content-equivalent print conversion. Differences are limited to pagination, float placement, line breaking, and other presentation requirements.

Two PDF layouts carry the same content:

- print PDF, with recto chapter and act openings;
- continuous screen PDF, without intentionally blank versos.

This volume is developed for **DS 6210 — Computation II: Numerical Analysis & Optimization**, also known as *Algorithms for Deep Learning*, in the School of Data Science at the University of Virginia. It is the advanced companion to *Deep Learning: Making It Learnable*.

This is a public working `v0.x` edition. A numbered chapter enters the public draft only after its mathematics, code, provenance, HTML, and PDF pass the repository audits. The present C01–C17 core, Coda, interludes, On-Ramp, Incident Card, and continuation map are author accepted; later revisions must pass the same gates again.

Suggested citation

Shakeri, Heman. 2026. *Deep Learning: Making It Trainable*. Public draft 0.2. School of Data Science, University of Virginia. https://shakeri-lab.github.io/opt-book/.

On-Ramp: The Minimum Mechanics

This book does not require a prior deep-learning course. It does require a small operational vocabulary: tensors have named axes, a scalar loss is differentiated through a composition, a mini-batch defines an estimator, and parameters are updated outside the derivative graph. This on-ramp lets you test that vocabulary before Chapter 1.

It is deliberately not a primer on architectures. If the diagnostic exposes a gap, the repair is a short route through selected sections of the earlier volume, not the whole book.

The 20–30 minute diagnostic

Work without searching for a finished training script. Explanations matter more than syntax.

1. **Shapes, 4 minutes.** Let $\mathbf{X} \in \mathbb{R}^{32\times 8}$, $\mathbf{W} \in \mathbb{R}^{8\times 3}$, and $\mathbf{b} \in \mathbb{R}^3$. State the shape of $\mathbf{XW} + \mathbf{b}$, identify the broadcast axis, and name the reduction that would produce one scalar mean loss.
2. **Composition, 4 minutes.** For $\mathbf{h} = \phi(\mathbf{XW}_1)$ and $\hat{\mathbf{y}} = \mathbf{hW}_2$, draw the dependency graph and mark which primal values a reverse derivative calculation may need.
3. **Loss and estimator, 5 minutes.** Distinguish the full-data objective from the mean loss on a mini-batch. Under what sampling rule is the latter an unbiased estimator of the former?
4. **Reverse accumulation, 5 minutes.** If one intermediate value feeds two children, explain why its two cotangent contributions must add rather than overwrite one another.
5. **Update boundary, 4 minutes.** Put these operations in a valid order: clear gradients, evaluate predictions, form a scalar loss, propagate derivatives, update parameters. Which operation must occur without being recorded as part of the next derivative graph?
6. **Numerical contract, 4 minutes.** Name the dtype and device of the parameters, loss accumulation, and update. Explain why "the model uses FP16" is not a complete answer.

If you can answer five or six items precisely, begin Chapter 1. If you can answer three or four, run the notebook below and read only the indicated sibling sections. If you can answer fewer than three, take the minimal route in order before returning.

The minimal route through *Making It Learnable*

The required route has three stops:

1. Nonlinearity and the MLP: read "Your first real MLP" to see a tensor composition assembled once.
2. Training: Loss and Stochastic Gradient Descent: read through the mini-batch and learning-rate sections; optimizer catalogs are optional for this book.
3. Backpropagation: read the one-chain derivation, the autograd rules, and "Keep updates outside the graph."

If Item 1 was the obstacle, begin instead with Tensors in Practice, then return to the three stops. The sibling volume is a repair shelf, not a prerequisite course.

One minimal training loop

Open or download the on-ramp notebook. It uses one linear map and one mean-squared loss so architecture cannot hide the mechanics. The notebook makes five objects inspectable:

- tensor shapes and the batch axis;
- prediction and scalar loss;
- reverse accumulation into parameter gradients;
- mini-batch sampling and averaging;
- an update performed outside the derivative graph.

The notebook is complete when you can change the batch size and learning rate, predict which quantities change, and explain which target the batch loss estimates.

You are ready for Chapter 1 when...

- you can state tensor shapes before a matrix product and name every reduction axis;
- you can distinguish a full objective from its mini-batch estimator;
- you can read a scalar loss, call reverse accumulation, and find each parameter gradient;
- you can explain why gradient contributions add at fan-out;
- you can update parameters without accidentally differentiating through the update;
- you can name storage, compute, and accumulation dtypes separately.

Chapter 1 starts from there. Everything beyond this list is built when the diagnosis needs it.

Act 0 — The computational arena

A formula becomes an algorithm only after its state, arithmetic, and traffic are declared. This act builds that contract from the machine upward: first the bytes that must move, then the state a stream must retain, then the grid on which arithmetic lands, and finally the mode-by-mode dynamics of a solvable quadratic control. The result can distinguish a bandwidth ceiling from a floating-point failure and a flat direction from a sharp one. It still cannot say what high dimension makes typical. Act I turns from one declared execution to the geometry of many possible directions.

1. The Count Is Not the Cost: Arithmetic Intensity and the Memory Hierarchy

Geometric quiet · Dynamic supporting · Algorithmic primary

A copy performs no arithmetic. A fused affine transform performs $2N$ operations. Their compulsory traffic — and therefore their bandwidth time bound — is the same.

Both computations act on N stored FP32 values. The copy moves at least $8N$ bytes: one read and one write. The fused transform $y_i = ax_i + b$ has the same dominant traffic when a and b remain in a fast store. One operation count is zero; that count does not identify elapsed time.

Now compare a square matrix product of order $n = 512$. The conventional count is $2n^3$ operations. Its compulsory traffic under an idealized one-read, one-write model is $12n^2$ bytes, so its arithmetic intensity is $n/6 = 85.33$ operations per byte. The affine transform has intensity $2/8 = 0.25$. On a hypothetical machine whose peak arithmetic-to-bandwidth ratio is 20, one falls below the balance point and the other lies above it. The larger operation count can be the computation with more opportunity to reuse data.

> **!** Prediction
>
> On one fixed machine, a proposed method halves the multiplications and doubles compulsory traffic. Choose a prediction before continuing: **faster**, **slower**, or **undetermined**. Which traffic boundary and machine balance would let you defend that choice?

1.1. Two resource clocks

Let F be the required operations, Q the bytes transferred through a declared boundary, $P_{\max}$ peak arithmetic rate, and B peak bandwidth. The two unavoidable time lower bounds are

$$T \geq \frac{F}{P_{\max}}, \qquad T \geq \frac{Q}{B}. \tag{1.1}$$

With arithmetic intensity $I = F/Q$, achieved performance $P = F/T$ obeys the following deliberately small theorem.

Theorem 1.1 (Two-resource performance ceiling)**.** *Under the declared operation count, traffic boundary, peak arithmetic rate, and peak bandwidth,*

$$P \leq \min(P_{\max}, BI). \tag{1.2}$$

1.1.1. Proof

Invert each inequality in Equation 1.1 and multiply by F. The arithmetic bound gives $P \le P_{\max}$; the traffic bound gives $P \le BF/Q = BI$. Both must hold.

The bend occurs at the **machine balance** $I_* = P_{\max}/B$. This diagram is the roofline model (Williams et al. 2009). It is an upper envelope, not a prediction that a kernel reaches either roof.

1. Read the committed C01 model calculation.
2. Declare one hypothetical machine balance.
3. Plot the bandwidth and arithmetic ceilings.
4. Place the vector transform and matrix product on the same axes.
5. Check the registered arithmetic intensities before interpretation.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]
claim = verify_claim("c01-roofline-001")
# [2]
balance = claim["result"]["machine_balance_flop_per_byte"]
# [3]
intensity = np.logspace(-2, 3, 300)
ceiling = np.minimum(intensity / balance, 1.0)
fig, axis = plt.subplots(figsize=(7.2, 3.8))
axis.loglog(intensity, ceiling, color="#232D4B", linewidth=2)
# [4]
points = {
    "fused affine": claim["result"]["vector_affine_intensity"],
    "512×512 product": claim["result"]["gemm_intensity"],
}
for label, value in points.items():
    axis.scatter(value, min(value / balance, 1.0), s=55)
    axis.annotate(
        label,
        (value, min(value / balance, 1.0)),
        xytext=(5, 8),
        textcoords="offset points",
    )
axis.set(
    xlabel="arithmetic intensity (operation/byte)",
    ylabel="fraction of arithmetic peak",
)
axis.grid(True, which="both", alpha=0.25)
fig.tight_layout()
# [5]
assert np.isclose(points["fused affine"], 0.25)
assert np.isclose(points["512×512 product"], 512 / 6)
print("claim c01-roofline-001: verified")
```

```
claim c01-roofline-001: verified
```

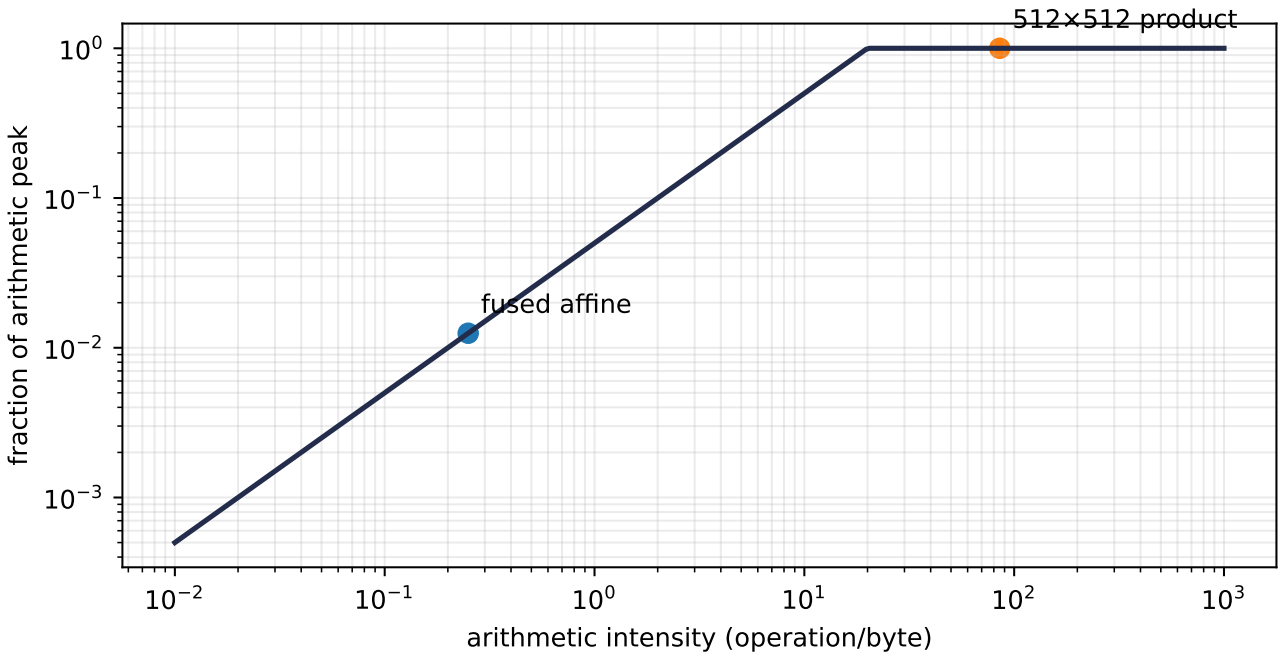


Figure 1.1.: One operation count is not one cost. Under the declared two-resource model, the vector transform lies below the balance point while the reused matrix product lies above it.

1.2. A zero-count kernel still consumes a clock

The model now needs a measured control. The committed observation times a 64 MiB FP32 payload copy and an order-512 FP32 matrix product on one Apple M4 Max CPU. It also sweeps nearby sizes to establish two **local empirical ceilings** in the same process. Those ceilings are not vendor peaks. They are reference rates that make the achieved fractions inspectable without claiming that a laptop CPU represents an accelerator.

1. Read the pinned local measurement and its protocol.
2. Extract the selected copy and matrix-product rates.
3. Normalize each rate by its own local empirical ceiling.
4. Plot resource-specific fractions rather than a false common unit.
5. Check that the displayed values match the committed observation.

```
# [1]
measurement = verify_claim("c01-local-measurement-001")
protocol = measurement["measurement_contract"]

# [2]
copy_result = measurement["result"]["copy_by_mib"]["64"]
matmul_result = measurement["result"]["matmul_by_order"]["512"]

# [3]
```

```
fractions = np.array([
    measurement["result"]["copy_64_fraction_of_local_bandwidth_ceiling"],
    measurement["result"]["matmul_512_fraction_of_local_arithmetic_ceiling"],
])

# [4]
fig, axis = plt.subplots(figsize=(7.2, 2.8))
axis.barh(
    ["64 MiB copy", "order-512 product"],
    fractions,
    color=["#9C2F2F", "#232D4B"],
)
axis.set(
    xlim=(0, 1.08),
    xlabel="fraction of the matching local empirical ceiling",
)
axis.grid(axis="x", alpha=0.22)
fig.tight_layout()

# [5]
assert protocol["statistic"] == "median elapsed time"
assert np.isclose(copy_result["median_seconds"], 0.000947375)
assert np.isclose(matmul_result["median_seconds"], 0.0001039795)
print(
    f"copy: {1e3 * copy_result['median_seconds']:.3f} ms; "
    f"{copy_result['achieved_gbyte_per_second']:.1f} GB/s"
)
print(
    f"matmul: {1e3 * matmul_result['median_seconds']:.3f} ms; "
    f"{matmul_result['achieved_gflop_per_second']:.1f} GFLOP/s"
)
```

```
copy: 0.947 ms; 141.7 GB/s
matmul: 0.104 ms; 2581.6 GFLOP/s
```

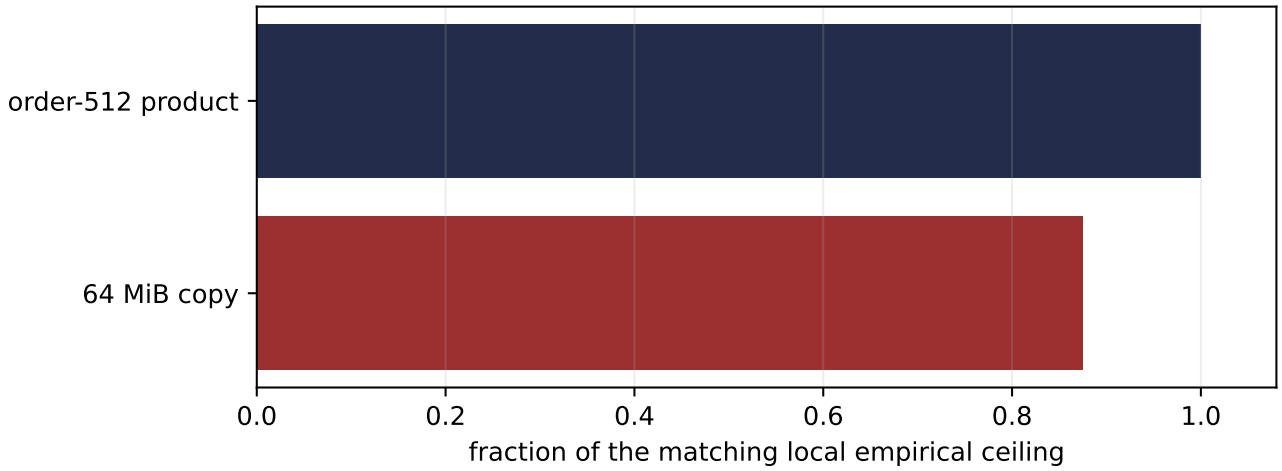


Figure 1.2.: A zero-FLOP copy has a positive measured cost. Each bar is normalized by a separately measured local ceiling for its own resource, so the figure supports a resource diagnosis rather than a cross-kernel speed ranking.

The copy moved a 64 MiB payload across the declared read-plus-write boundary, so the traffic count was 128 MiB. Its median time was about 0.947 ms and its achieved bandwidth was 141.7 GB/s, or 87.5% of the best copy rate in the declared size sweep. The matrix product reached 2581.6 GFLOP/s in the same software process. These observations do not say that one kernel is generally faster: they show that zero counted arithmetic does not mean zero elapsed time, and that each kernel must be compared with the roof for the resource it actually demands.

1.3. The denominator is an algorithmic choice

Traffic has no meaning until its boundary is named. Bytes between registers and a first-level cache, between cache and main memory, and between host and accelerator are different denominators. Tiling changes Q by retaining a working set near arithmetic units. Fusion changes Q by keeping an intermediate value from making a round trip. Recompute can deliberately increase F to reduce Q. C11 will return to exactly that trade when it checkpoints derivative state.

This also foreshadows the numerical-stability thread that C02 seeds. A fused operation can change both traffic and rounding: eliminating an intermediate store may eliminate one rounding step. A performance transformation is therefore part of the numerical contract, not merely a scheduling detail.

> ⚠ Named wrong answer: fewer operations means faster
>
> Fewer operations is evidence only after the limiting resource is identified. It may improve a compute-limited kernel, do nothing to a bandwidth-limited kernel, or make the kernel slower by destroying reuse or fusion.

1.4. Hardware-claim boundary

The roofline is useful because it makes omissions visible. Peak values may be unreachable; compulsory traffic can underestimate actual traffic; latency, launch overhead, synchronization, occupancy, instruction mix, and access regularity remain outside Equation 1.2. CPU simulation of FP16 behavior is not evidence of accelerator timing. Any measured claim later in this book must name the device, software stack, warmup, repetitions, statistic, and traffic boundary—or be labeled a model calculation or pinned artifact.

> Field note: read a speedup claim in three passes
>
> First locate the mathematical work and the denominator behind its reported rate. Then locate the resource boundary: bytes, arithmetic, latency, communication, or some mixture. Only then compare measured elapsed time, including baseline, device, precision, warmup, repetitions, statistic, and failure accounting. A speedup is an observation about that full comparison, not a synonym for a smaller operation count.

1.5. Check yourself

A kernel performs 10^{10} operations and moves 10^9 bytes. On a device with $P_{\max} = 10^{13}$ operation/s and $B = 5 \times 10^{11}$ byte/s, its intensity is what? What are the machine balance, active roofline branch, and model ceiling? If F is halved while Q is unchanged, which lower bound changes and which one does not?

1.6. Okay, so —

- **Inherited:** asymptotic operation counts remain indispensable scaling summaries.
- **Changed:** elapsed-time reasoning now requires a declared traffic denominator and machine balance.
- **Instrumented:** one exact roofline claim and one pinned laptop measurement separate modeled ceilings from achieved rates.
- **Established:** $P \leq \min(P_{\max}, BI)$ follows from two resource lower bounds.
- **Unresolved:** Which state must an online computation retain when real executions move data through multiple memory levels?

1.7. Sources and further reading

The roofline formulation and its machine-balance interpretation follow Williams et al. (2009). For systems-scale companions that carry rooflines into accelerator topology, collectives, and large-model execution, see Austin et al. (2025) and Tazi et al. (2025).

Reading order. Start with Williams et al. (2009) for the two-resource diagram, then return to this chapter's denominator and measurement sections before using Austin et al. (2025) or Tazi et al. (2025) for multi-device scale.

1.8. Exercises

1. **(Pencil.)** Recompute the matrix-product intensity if one input matrix is reread four times. State the traffic boundary.
2. **(Code.)** For three hypothetical machine balances, redraw the figure and report which classifications change. Do not time the plotting code.
3. **(Pencil.)** A checkpointing rule adds 30% arithmetic and removes half the retained-state traffic. Derive the intensities before and after, then state the machine-balance range in which the trade can improve the roofline bound.
4. **(Code.)** Re-run the local protocol with three payload sizes and three matrix orders. Declare device, stack, warmup, repetitions, statistic, and traffic boundary. Compare medians and dispersion; do not call the best observed rate a vendor peak.
5. **(Audit.)** **Paper audit:** Find one reported speedup and complete the **Claim**, **Mathematical object**, **Evidence culture and interface**, **Numerical and hardware contract**, and **Transfer verdict** fields of the Paper Autopsy Protocol. Separate operation count, modeled traffic, measured elapsed time, hardware identity, and baseline. Which claim is actually supported?

2. One Pass, Two Failures: Streaming State and Stable Variance

NS · Numerical stability Geometric quiet · Dynamic quiet · Algorithmic primary

A variance cannot be negative. Yet the following one-pass calculation returns one.

An impossible output is more than a failed test: it is a diagnostic instrument.

It tells us that algebraic correctness did not survive execution.

The setting is ordinary. We receive a stream $x_1, \dots, x_n$, and we want its population variance

$$\sigma_n^2 := \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_n)^2, \qquad \mu_n := \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{2.1}$$

The formula appears to demand two passes. We need μ_n before we can form the centered deviations. If the stream cannot be revisited, we must either store it or find a different state.

An algebraic identity seems to remove that obstacle:

$$\sigma_n^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \mu_n^2. \tag{2.2}$$

Now a single pass can carry three scalars: n, $\sum_i x_i$, and $\sum_i x_i^2$. The memory problem looks solved.

! Prediction

We will hold the deviations fixed and add the same constant to every value. Exact variance is unchanged by this translation.
Which computation should fail first as the constant grows: the raw-moment formula in Equation 2.2, a centered two-pass calculation, or a centered online calculation? Name the operation that you expect to fail.

2.1. The impossible output

The chapter setup verifies its content-addressed harness once, outside the printed mechanism kernel. The random values are cast to FP32 *before* the FP64 reference is computed. Every method therefore receives the same represented inputs.

1. Load the chapter-pinned variance instruments.
2. Generate one seeded FP32 stream with a large common offset.
3. Compare three variance methods with a same-input FP64 reference.

4. Assert the impossible symptom and the centered controls.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import (
    comparison_audit,
    merge_moments,
    observation_ledger,
    stable_online_moments,
)

# [2]
seed = 6210
rng = np.random.default_rng(seed)
values = (20_000.0 + rng.normal(size=200_000)).astype(np.float32)

# [3]
audit = comparison_audit(values, dtype=np.float32)
ledger = observation_ledger(seed=seed, dtype=np.float32)
raw_second_moment = np.mean(values * values, dtype=np.float32)
mean_square = np.mean(values, dtype=np.float32) ** np.float32(2)

# [4]
assert audit["naive"]["population_variance"] < 0.0
assert audit["two_pass"]["relative_error"] < 1e-5
assert audit["welford"]["relative_error"] < 5e-3

print(f"harness={manifest['harness_ref']} wheel={manifest['wheel_sha256'][:12]}")
print(f"seed={ledger['seed']} device={ledger['device']} dtype={ledger['dtype']}")
print(f"reference variance={audit['reference']['population_variance']:.9f}")
for method in ("naive", "two_pass", "welford"):
    result = audit[method]
    print(
        f"{method:>8}: variance={result['population_variance']:.9f}, "
        f"relative error={result['relative_error']:.3e}"
    )
```

```
harness=ch-02 wheel=253a9367089f
seed=6210 device=cpu dtype=float32
reference variance=1.002515215
   naive: variance=-32.000000000, relative error=3.292e+01
two_pass: variance=1.002516150, relative error=9.331e-07
 welford: variance=1.004139766, relative error=1.620e-03
```

The raw-moment estimate is -32, even though the FP64 reference on those same FP32 values is 1.002515. The population denominator is not the culprit: changing n to $n-1$ would rescale an already corrupted

centered sum. Random sampling is not the culprit either: sampling can move a variance away from one, but not below zero.

The impossible answer is not an arbitrary negative number. The two rounded terms are 400000192 and 400000224; their local grid spacing is $\mathrm{ulp}(\mu^2) = 32$. The result therefore lands one grid step below zero: $-\mathrm{ulp}(\mu^2) =$ -32. The scale model in Equation 2.3 predicts roughly 23.8; it locates the grid scale without pretending to predict a particular reduction path exactly.

The common offset is about 19,975 reference standard deviations. That ratio points toward the dangerous subtraction.

2.1.1. The shift test

Use the same deviations at every offset. This removes a tempting alternate explanation: changes in the graph cannot be attributed to a new random sample.

1. Hold one random deviation vector fixed.
2. Add offsets without changing the exact centered variance.
3. Evaluate all methods on the same represented inputs.
4. Plot relative error and mark impossible negative results.

```
# [1]
offset_rng = np.random.default_rng(6210)
deviations = offset_rng.normal(size=200_000)

# [2]
offsets = np.array([0, 100, 1_000, 3_000, 10_000, 20_000, 30_000, 100_000])

# [3]
records = []
for offset in offsets:
    represented = (offset + deviations).astype(np.float32)
    result = comparison_audit(represented, dtype=np.float32)
    scale = result["reference"]["population_variance"] ** 0.5
    records.append(
        {
            "ratio": max(float(offset / scale), 1.0),
            "naive": result["naive"]["relative_error"],
            "two_pass": result["two_pass"]["relative_error"],
            "welford": result["welford"]["relative_error"],
            "naive_negative": result["naive"]["population_variance"] < 0,
        }
    )

# [4]
fig, ax = plt.subplots(figsize=(7.0, 4.2))
styles = {
    "naive": ("Raw moments", "#9C2F2F", "o"),
    "two_pass": ("Centered two-pass", "#232D4B", "s"),
    "welford": ("Centered online", "#2E7D32", "^"),
```

```
}
for key, (label, color, marker) in styles.items():
    ax.plot(
        [row["ratio"] for row in records],
        [max(row[key], 1e-9) for row in records],
        label=label,
        color=color,
        marker=marker,
        linewidth=2,
    )
for row in records:
    if row["naive_negative"]:
        ax.scatter(
            row["ratio"],
            max(row["naive"], 1e-9),
            color="#9C2F2F",
            marker="v",
            s=90,
            zorder=4,
        )
ax.set(xscale="log", yscale="log")
ax.set_xlabel("common offset / reference standard deviation")
ax.set_ylabel("relative error")
ax.grid(True, which="both", alpha=0.25)
ax.legend(frameon=False)
plt.show()
```

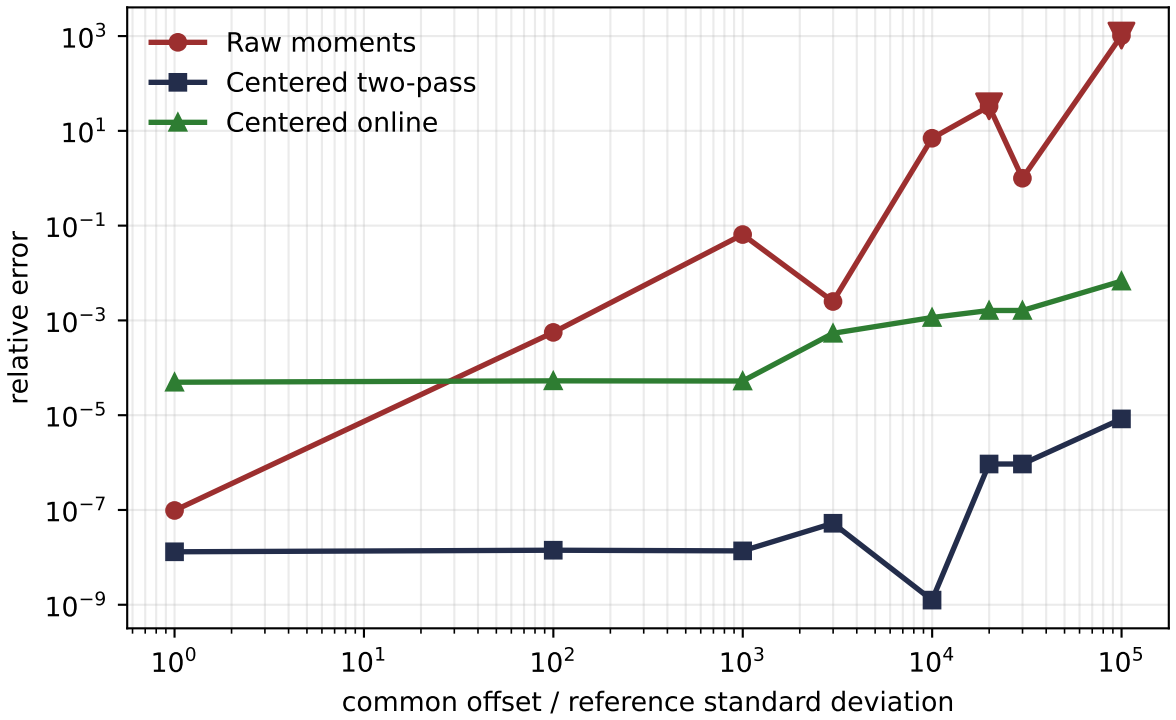


Figure 2.1.: The raw-moment formula degrades as a common offset grows, although exact variance is translation invariant. Downward triangles mark negative estimates. Centered methods avoid the dominant cancellation, but the figure does not claim that they eliminate all rounding or input-quantization error.

The relevant observation is not that one curve has a particular shape. It is that a transformation which preserves exact variance changes the raw-moment answer dramatically. The experiment has isolated sensitivity to translation.

2.2. Diagnose the subtraction

Write each observation as

$$x_i = \mu + \varepsilon_i, \qquad \frac{1}{n}\sum_i \varepsilon_i = 0, \qquad \frac{1}{n}\sum_i \varepsilon_i^2 = \sigma^2.$$

Then

$$\frac{1}{n}\sum_i x_i^2 = \mu^2 + \sigma^2, \qquad \left(\frac{1}{n}\sum_i x_i\right)^2 = \mu^2.$$

The raw-moment formula asks floating-point arithmetic to construct two values on the scale of μ^2, then subtract them to recover one on the scale of σ^2.

Suppose, as a diagnostic model calculation, that the two large terms acquire absolute errors of order $u\mu^2$, where u is the unit roundoff. The absolute error in their difference is then also on that scale. Relative to the desired variance, the amplification is approximately

$$\frac{u\mu^2}{\sigma^2} = u\left(\frac{\mu}{\sigma}\right)^2. \tag{2.3}$$

This is not a complete forward-error bound for a particular summation kernel. It deliberately suppresses the number of additions, reduction order, compensation, and intermediate precision. It does explain the observed control: translating the data leaves exact variance alone while increasing the scale of the two terms that must cancel.

> ⚠ Named wrong answer: 'FP32 cannot represent the noise'
>
> At the opening offset, FP32 still represents many distinct unit-scale deviations. We know because the FP64 reference is computed *after* the FP32 cast and remains near one. The dominant opening failure is algorithmic cancellation, not already-lost input information.
> At a sufficiently larger offset, the spacing between representable FP32 values does become comparable with the deviations. That is a different failure, and no variance algorithm can recover distinctions absent from its input.

The two failures must stay separate:

Failure	Where information is lost	Can centered state repair it?
Raw-moment cancellation	During accumulation and subtraction of large moments	It avoids the dominant subtraction
Input quantization	When real values are rounded to the input dtype	No; the distinctions never reach the algorithm

The next chapter develops the precision language needed to quantify the second row. For now, this separation is enough to design the state.

2.3. Preserve the centered invariant

The desired state after n values is

$$h_n = (n, \mu_n, M_{2,n}), \qquad M_{2,n} := \sum_{i=1}^{n}(x_i - \mu_n)^2. \tag{2.4}$$

The final population variance is $M_{2,n}/n$. The usual unbiased sample variance is $M_{2,n}/(n-1)$ when $n > 1$. The state does not need to commit to either denominator while the stream is arriving.

When x_n arrives, define its deviation from the *old* mean:

$$\delta_n := x_n - \mu_{n-1}.$$

The mean update follows directly from the definition:

$$\mu_n = \mu_{n-1} + \frac{\delta_n}{n}. \tag{2.5}$$

The centered sum of squares uses both the old and new means:

$$M_{2,n} = M_{2,n-1} + (x_n - \mu_{n-1})(x_n - \mu_n). \tag{2.6}$$

Theorem 2.1 (Centered-state invariant)**.** *Assume the state after $n-1$ values satisfies*

$$M_{2,n-1} = \sum_{i=1}^{n-1} (x_i - \mu_{n-1})^2.$$

Together, the mean update and the centered-sum update above produce

$$M_{2,n} = \sum_{i=1}^{n} (x_i - \mu_n)^2$$

in exact arithmetic.

2.3.1. Proof

Let $\delta = \delta_n$. The mean moves by δ/n, so for each old observation,

$$x_i - \mu_n = (x_i - \mu_{n-1}) - \frac{\delta}{n}.$$

Expand the contribution from the first $n-1$ values:

$$\begin{aligned}\sum_{i=1}^{n-1} (x_i - \mu_n)^2 = &\sum_{i=1}^{n-1} (x_i - \mu_{n-1})^2 \\ &- \frac{2\delta}{n} \sum_{i=1}^{n-1} (x_i - \mu_{n-1}) + (n-1)\frac{\delta^2}{n^2}.\end{aligned}$$

The middle term vanishes because deviations from μ_{n-1} sum to zero. The new observation contributes

$$(x_n - \mu_n)^2 = \left(\delta - \frac{\delta}{n}\right)^2 = \left(\frac{n-1}{n}\delta\right)^2.$$

Adding the old and new contributions gives

$$\begin{aligned}M_{2,n} &= M_{2,n-1} + (n-1)\frac{\delta^2}{n^2} + (n-1)^2\frac{\delta^2}{n^2} \\ &= M_{2,n-1} + \frac{n-1}{n}\delta^2.\end{aligned}$$

Finally,

$$(x_n - \mu_{n-1})(x_n - \mu_n) = \delta\left(\frac{n-1}{n}\delta\right),$$

which is exactly the increment added by the centered-sum update. □

The proof does more than validate a formula. It identifies the invariant the state represents after every prefix. That invariant lets us test the kernel on any prefix and gives us the object that partial computations must preserve.

2.3.2. The mechanism in code

1. Initialize the empty centered state.
2. Update the mean with the deviation from the old mean.
3. Update the centered sum using deviations from both means.
4. Check the kernel against the pinned harness and a direct calculation.

```
def welford_kernel(stream, dtype=np.float64):
    scalar = np.dtype(dtype).type
    count, mean, m2 = 0, scalar(0), scalar(0)  # [1]
    for value in np.asarray(stream, dtype=dtype):
        count += 1
        delta = scalar(value - mean)
        mean = scalar(mean + delta / scalar(count))  # [2]
        delta_after = scalar(value - mean)
        m2 = scalar(m2 + delta * delta_after)  # [3]
    return count, float(mean), float(m2)


small = np.array([4.0, 7.0, 13.0, 16.0, 20.0], dtype=np.float64)
kernel_state = welford_kernel(small)
harness_state = stable_online_moments(small, dtype=np.float64)
direct_m2 = float(np.sum((small - small.mean()) ** 2))

assert kernel_state == (
    harness_state.count,
    harness_state.mean,
    harness_state.m2,
)
assert np.isclose(kernel_state[2], direct_m2)  # [4]
print(
    f"count={kernel_state[0]}, mean={kernel_state[1]:.1f}, "
    f"M2={kernel_state[2]:.1f}, variance={kernel_state[2] / kernel_state[0]:.1f}"
)
```

```
count=5, mean=12.0, M2=170.0, variance=34.0
```

The update remains a floating-point recurrence. Centering removes the particular subtraction diagnosed by Equation 2.3; it does not make every update exact. "Stable" always needs an object and a failure mode. Here it means that the algorithm avoids forming raw moments on the scale of μ^2 merely to recover a centered quantity on the scale of σ^2.

2.4. One state must also combine

Sequential state solves the one-pass problem for one stream. It does not yet solve the execution problem for many blocks. If block A and block B are processed independently, each produces a valid state:

$$h_A = (n_A, \mu_A, M_{2,A}), \qquad h_B = (n_B, \mu_B, M_{2,B}).$$

Adding $M_{2,A}$ and $M_{2,B}$ is wrong whenever the block means differ. The smallest counterexample is:

$$A = \{0, 0\}, \qquad B = \{10, 10\}.$$

Both within-block variances are zero. The union has mean five and population variance 25. The missing quantity is variation *between* the block means.

Let

$$\delta := \mu_B - \mu_A, \qquad n := n_A + n_B.$$

The combined mean is

$$\mu_{A\cup B} = \mu_A + \delta\frac{n_B}{n}. \tag{2.7}$$

The centered sum combines as

$$M_{2,A\cup B} = M_{2,A} + M_{2,B} + \delta^2\frac{n_A n_B}{n}. \tag{2.8}$$

Theorem 2.2 (Pairwise centered-state merge)**.** *If h_A and h_B satisfy the centered-state invariant on disjoint collections A and B, then Equation 2.7 and Equation 2.8 satisfy it on $A \cup B$.*

2.4.1. Proof

Recenter block A around the combined mean:

$$\begin{aligned}\sum_{i\in A}(x_i-\mu_{A\cup B})^2 &= \sum_{i\in A}\left[(x_i-\mu_A)+(\mu_A-\mu_{A\cup B})\right]^2\\ &= M_{2,A}+n_A(\mu_A-\mu_{A\cup B})^2.\end{aligned}$$

The cross term vanishes because deviations from μ_A sum to zero. The same argument for B gives

$$M_{2,B}+n_B(\mu_B-\mu_{A\cup B})^2.$$

From Equation 2.7,

$$\mu_A-\mu_{A\cup B}=-\delta\frac{n_B}{n},\qquad \mu_B-\mu_{A\cup B}=\delta\frac{n_A}{n}.$$

The two recentering costs therefore sum to

$$\delta^2\left(\frac{n_An_B^2}{n^2}+\frac{n_Bn_A^2}{n^2}\right)=\delta^2\frac{n_An_B}{n}.$$

That is exactly the correction in Equation 2.8. □

The correction measures how much centered energy appears when two local origins are replaced by one global origin.

1. Build one centered state per block.
2. Merge the states with the between-block correction.
3. Compare the merged result with the direct union.
4. Reject the tempting sum of within-block centered sums.

```
# [1]
left = stable_online_moments([0.0, 0.0], dtype=np.float64)
right = stable_online_moments([10.0, 10.0], dtype=np.float64)

# [2]
combined = merge_moments(left, right)

# [3]
direct = np.var(np.array([0.0, 0.0, 10.0, 10.0]), dtype=np.float64)
assert np.isclose(combined.population_variance, direct)

# [4]
without_correction = (left.m2 + right.m2) / combined.count
assert without_correction == 0.0
print(
    f"within-only={without_correction:.1f}, "
    f"with correction={combined.population_variance:.1f}"
)
```

```
within-only=0.0, with correction=25.0
```

2.4.2. Associative in which arithmetic?

In exact arithmetic, the merge operation is associative on valid states. A state represents the count, mean, and centered sum of squares of one underlying multiset. Both

$$(h_A \oplus h_B) \oplus h_C \quad \text{and} \quad h_A \oplus (h_B \oplus h_C)$$

represent the same union, whose three exact statistics are unique.

That statement gives us a parallel reduction tree. It does **not** say that floating-point parenthesizations are bitwise identical. Each merge rounds, so the tree becomes part of the numerical algorithm.

1. Form the same FP32 block states once.
2. Merge them left-to-right, right-to-left, and in a balanced tree.
3. Retain a sequential state and same-input FP64 reference as controls.
4. Report the spread without declaring one order universally best.

```
# [1]
order_rng = np.random.default_rng(6211)
order_values = (20_000.0 + order_rng.normal(size=131_072)).astype(np.float32)
blocks = [
    stable_online_moments(block, dtype=np.float32)
    for block in np.array_split(order_values, 128)
]

def left_fold(states):
    state = states[0]
    for next_state in states[1:]:
        state = merge_moments(state, next_state)
    return state


def balanced_fold(states):
    level = list(states)
    while len(level) > 1:
        level = [
            merge_moments(level[index], level[index + 1])
            if index + 1 < len(level)
            else level[index]
            for index in range(0, len(level), 2)
        ]
    return level[0]

# [2]
orders = {
    "left": left_fold(blocks),
```

```
    "right": left_fold(list(reversed(blocks))),
    "balanced": balanced_fold(blocks),
}

# [3]
orders["sequential"] = stable_online_moments(order_values, dtype=np.float32)
order_reference = float(np.var(order_values.astype(np.float64)))

# [4]
for name, state in orders.items():
    print(
        f"{name:>10}: variance={state.population_variance:.9f}, "
        f"error={abs(state.population_variance - order_reference):.3e}"
    )
print(f"{'reference':>10}: variance={order_reference:.9f}")
```

```
      left: variance=0.998615682, error=1.937e-04
     right: variance=0.998613298, error=1.913e-04
  balanced: variance=0.998613000, error=1.910e-04
sequential: variance=1.000126481, error=1.704e-03
 reference: variance=0.998421996
```

The three merge trees differ in their final bits, as expected. The balanced tree is not promoted here as a universal accuracy theorem: data order, partial-state quality, accumulator width, and the implementation of the combine all matter. The supported claim is narrower. A mathematical combine law creates parallel structure; floating-point execution still requires a reduction-order audit.

2.5. Choosing the state is choosing the algorithm

The original question was not "which variance formula should a library use?" The useful question is "what constraints does this execution have?"

Constraint	Honest baseline	Principal cost or boundary
Data can be revisited	Centered two-pass variance	Reads the input at least twice
True one-pass stream	Sequential centered state	Dependency depth grows with stream length
Independent blocks	Centered partial states plus pairwise merge	Floating-point tree order affects final bits
Input distinctions already lost	Wider storage or an upstream scaling change	No downstream state can reconstruct missing values

The raw-moment formula remains an algebraic identity. Its failure does not make the identity false. It makes the evaluation path unsuitable in the observed regime.

The same centered state later appears inside batch-coordinate normalization: its running statistics are mergeable estimates whose axes, denominator, update rule, and accumulator precision must be declared explicitly.

Field note: what a trace can and cannot tell you

A negative variance strongly implicates numerical execution because exact variance is nonnegative. A merely inaccurate positive variance is less diagnostic. It can come from cancellation, input quantization, accumulation order, a mismatched denominator, or a changing target. Record the state, dtype, axes, denominator, and same-input reference before assigning a cause.

Check yourself

You have two blocks whose internal variances are both zero, but whose means differ. Why is adding the two M_2 values wrong? Which single term repairs the result, and what does that term measure?

2.6. Okay, so —

- **Inherited:** A formula has a data-movement schedule; rereading a tensor is an algorithmic decision.
- **Changed:** Variance is no longer just a definition. It is a state-design problem constrained by passes, rounding, and parallel structure.
- **Instrumented:** The harness now records the execution contract, compares methods on the same represented inputs, maintains centered moments, and merges partial states.
- **Established:** The centered online recurrence and the pairwise correction preserve M_2 in exact arithmetic. The seeded FP32 witness establishes that the raw-moment evaluation can return an impossible negative result.
- **Unresolved:** Which precision contract separates values lost at storage from errors introduced by a non-associative reduction?

2.7. Sources and further reading

The corrected online update traces to Welford's short note (Welford 1962). Chan, Golub, and LeVeque compare variance algorithms, analyze roundoff, and develop pairwise computation (Chan et al. 1983).

Hennig, Pförtner, and Weiland frame limited computation as limited information (Hennig et al. 2026). Hennig makes one version exact for iterative linear solvers under a declared Gaussian model (Hennig 2015). That work motivates Exercise 7; it does not make the centered-moment recurrence a Bayesian update.

For floating-point format anatomy, mixed-precision roles, and introductory rounding examples, see the sibling volume's precision and hardware appendix. This chapter instead owns the centered invariant, merge law, and same-input diagnostic.

Reading order. Start with Welford (1962) for the recurrence and Chan et al. (1983) for the stability and merge analysis; read Hennig (2015) only after the chapter's state-versus-belief boundary is clear.

2.8. Exercises

1. **(Pencil.) The invariant from another direction.** Starting from $M_{2,n} = \sum_{i=1}^{n} x_i^2 - n\mu_n^2$, derive the centered-sum update stated in this chapter. Mark the step that is safe as an exact

identity but would reintroduce the raw-moment hazard if used as the implementation.

2. **(Pencil.) Population versus sample.** Prove that the same state $(n, \mu_n, M_{2,n})$ can produce either the population variance or the usual unbiased sample variance. Explain why changing the final denominator cannot repair catastrophic cancellation already present in $M_{2,n}$.

3. **(Code.) Crash map.** Hold one represented deviation vector fixed and sweep both offset and accumulator dtype. Report the first offset at which the raw-moment estimate becomes negative and the first at which the centered methods exceed one percent relative error. Keep the FP64 reference on the same represented inputs.

4. **(Code.) Profile: one pass versus two.** On a laptop CPU, compare bytes implied by one-pass and two-pass schedules for a contiguous FP32 array. Time the implementations only after stating warmup, sample count, and estimator. Do not interpret CPU timing as accelerator performance.

5. **(Audit.) The clipped repair.** A code review proposes `variance = max(raw_variance, 0)`. Predict which visible symptom this removes, then design a test showing what error it conceals. State the strongest claim the clipped result can support.

6. **(Audit.) Merge contract.** An implementation all-reduces only (n, μ, M_2) by componentwise sum. Use two constant blocks to identify the defect. Write the missing correction and explain why the tempting componentwise answer is wrong.

7. **(Audit.) Paper audit: state versus belief.** Complete the **Mathematical object**, **Evidence culture and interface**, **Estimator and comparison contract**, **Assumption stress test**, and **Transfer verdict** fields for Hennig's Gaussian linear-solver construction (Hennig 2015). Compare its state $(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with Welford's $(n, \mu_n, M_{2,n})$: record the target, observation, prior, rank-one update, stopping output, and uncertainty claim. Identify the shared state-machine pattern, then explain why Welford is not Bayesian conditioning without an additional probability model.

3. The Grid Under the Update: Range, Resolution, and Rounding

NS · Numerical stability Geometric supporting · Dynamic supporting · Algorithmic primary

A parameter is stored as FP16 at the value 1024. An algorithm asks for the same update, +0.25, four hundred times. Exact arithmetic ends at 1124. The stored parameter never moves.

Nothing overflowed. The update was not zero. The code executed all four hundred additions. Yet every requested change disappeared.

! Prediction

Which boundary swallowed the update?

1. The value left the format's range.
2. The input 0.25 could not be represented.
3. The addition produced a result that rounded back to 1024.
4. Positive and negative quantities catastrophically cancelled.

Choose one, then name the control that would separate it from the others.

3.1. The update that never lands

The opening witness is a format simulation on a CPU. It makes a statement about FP16 rounding, not about CPU or accelerator speed. The chapter verifies and imports its own vendored harness wheel, so later changes to the package cannot silently alter this result.

1. Load the chapter-pinned precision instruments.
2. Request four hundred +0.25 updates from the same initial value.
3. Record the local spacing and the full precision contract.
4. Assert the stalled stored path against the exact path.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import (
    float_contract,
    local_spacing,
    repeated_update,
```

```
    stochastic_update_trials,
    symmetric_quantize,
)

# [2]
initial, increment, steps = 1024.0, 0.25, 400
trace = repeated_update(initial, increment, steps, dtype=np.float16)

# [3]
spacing = local_spacing(initial, dtype=np.float16)
precision_ledger = {
    "storage": "float16",
    "operands": "exactly representable float16 inputs",
    "compute": "correctly rounded binary addition",
    "accumulator": "stored state itself",
    "output": "float16 after every step",
    "rounding": "nearest, ties to even",
}

# [4]
assert trace.exact_final == 1124.0
assert trace.final == 1024.0
assert trace.changed_steps == 0

fig, ax = plt.subplots(figsize=(7.0, 3.8))
ax.plot(trace.exact, color="#232D4B", linewidth=2.2, label="Exact requested path")
ax.plot(trace.stored, color="#9C2F2F", linewidth=2.2, label="Stored FP16 path")
ax.set(xlabel="update step", ylabel="state")
ax.grid(alpha=0.25)
ax.legend(frameon=False)
plt.show()

print(f"harness={manifest['harness_ref']} wheel={manifest['wheel_sha256'][:12]}")
print(f"local spacing={spacing:g}; half spacing={spacing / 2:g}")
print(f"exact endpoint={trace.exact_final:g}; stored endpoint={trace.final:g}")
```

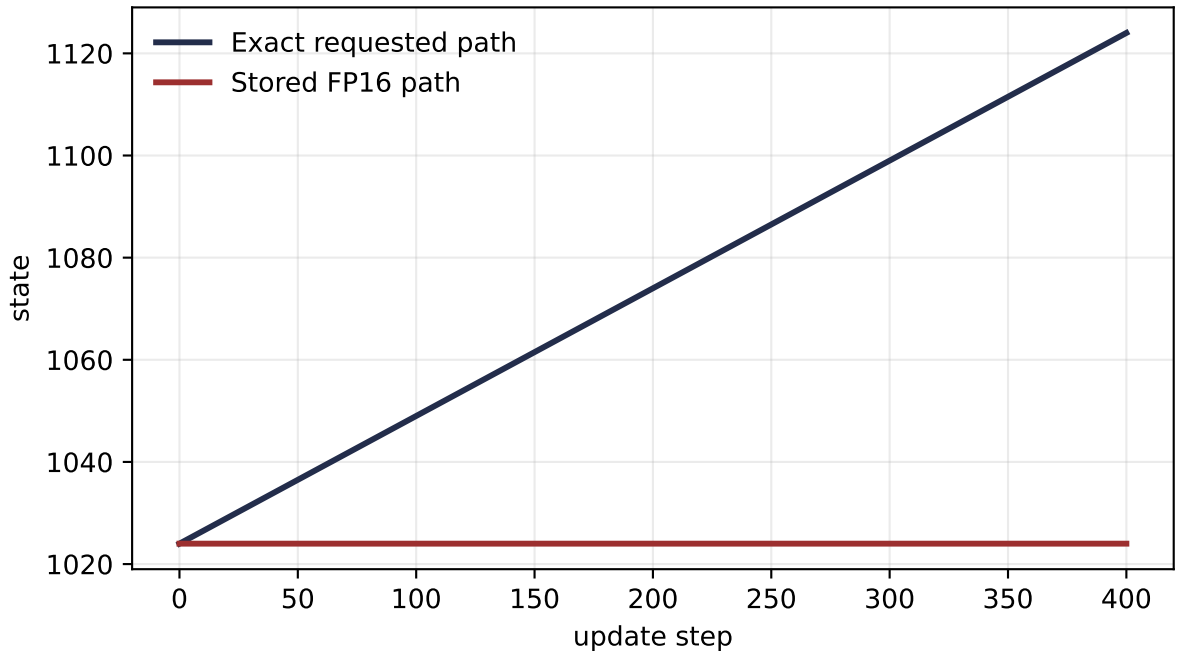


Figure 3.1.: A requested update of 0.25 advances the exact trajectory but never changes an FP16 value stored at 1024. The figure supports a local-resolution diagnosis under the declared round-after-each-step contract; it is not a hardware timing result.

```
harness=ch-03 wheel=833cfe618f0c
local spacing=1; half spacing=0.5
exact endpoint=1124; stored endpoint=1024
```

The value 0.25 is exactly representable in binary. The exact sum 1024.25 is also far inside the finite range of FP16. The failure occurs when that sum must be stored back on the local FP16 grid. Its neighbors are 1024 and 1025; round-to-nearest returns 1024. The next step starts from the same state and repeats the same loss.

This is stagnation by local resolution. Describing the computation only as “FP16” does not reveal it. We needed the state value, update size, storage point, and rounding rule.

3.2. A floating format is a changing grid

A normalized radix-two value with p significand bits has the form

$$x = \pm \left(1.b_1 b_2 \cdots b_{p-1}\right)_2 2^e. \tag{3.1}$$

The leading one is fixed for normal values. Incrementing the last stored bit changes the significand by $2^{-(p-1)}$, then the exponent scales that change by 2^e.

Theorem 3.1 (Binade spacing)**.** *In a normalized radix-two format with p significand bits, consecutive representable values in the binade $[2^e, 2^{e+1})$ differ by*

$$\Delta_e = 2^{e-(p-1)}. \tag{3.2}$$

Under round-to-nearest, an update from a representable value cannot change the stored result unless the exact result reaches a midpoint between grid points, with the midpoint itself decided by the tie rule.

3.2.1. Proof

Within one binade, the exponent e is fixed. The stored significands are

$$1 + \frac{k}{2^{p-1}}, \qquad k = 0, 1, \ldots, 2^{p-1} - 1.$$

Consecutive values therefore differ by

$$2^e \left[\left(1 + \frac{k+1}{2^{p-1}}\right) - \left(1 + \frac{k}{2^{p-1}}\right) \right] = 2^{e-(p-1)}.$$

Round-to-nearest assigns every exact value between two adjacent midpoints to the enclosed grid point. A smaller perturbation cannot cross that cell's boundary. □

For FP16, $p = 11$. At $1024 = 2^{10}$, Equation 3.2 gives

$$\mathrm{ulp}(1024) = 2^{10-(11-1)} = 1.$$

The update 0.25 is only one quarter of the local spacing and one half of the distance to the rounding boundary.

3.2.2. Three nearby quantities that are not synonyms

- $\mathrm{ulp}(x)$ is a **local** grid spacing near x.
- Machine epsilon, $\varepsilon_{\mathrm{mach}}$, is the gap from 1 to the next larger representable value.
- For round-to-nearest in a normal binade, the unit roundoff is $u = \varepsilon_{\mathrm{mach}}/2$.

The conventional one-operation model is

$$\mathrm{fl}(z) = z(1 + \delta), \qquad |\delta| \le u. \tag{3.3}$$

It applies when one exact result z is correctly rounded to a normal finite value. It is not a license to ignore overflow, gradual underflow, multiple operations, an already-rounded input, or an unspecified reduction.

⚠ Named wrong answer: ‘Machine epsilon is the smallest number the format can store’

Machine epsilon describes the gap after one. The smallest positive normal and the smallest positive subnormal describe range near zero. Local spacing grows with the exponent. One number cannot stand in for all three.

3.3. Range and resolution are separate budgets

The next instrument queries the format instead of relying on remembered decimal approximations.

1. Query the FP16 and FP32 format contracts at runtime.
2. Separate significand allocation from exponent allocation.
3. Report epsilon, unit roundoff, and both lower range boundaries.
4. Verify the observed spacing at the opening state.

```
# [1]
contracts = {
    "FP16": float_contract(np.float16),
    "FP32": float_contract(np.float32),
}

# [2]
for name, contract in contracts.items():
    print(
        f"{name}: significand={contract['significand_bits']} bits, "
        f"exponent={contract['exponent_bits']} bits"
    )

# [3]
for name, contract in contracts.items():
    print(
        f"{name}: eps={contract['epsilon']:.8g}, "
        f"u={contract['unit_roundoff']:.8g}"
    )
    print(
        f"  normal floor={contract['smallest_normal']:.8g}, "
        f"smallest subnormal={contract['smallest_subnormal']:.8g}, "
        f"largest finite={contract['largest_finite']:.8g}"
    )

# [4]
assert contracts["FP16"]["unit_roundoff"] == 2**-11
assert local_spacing(1024.0, dtype=np.float16) == 1.0
```

```
FP16: significand=11 bits, exponent=5 bits
FP32: significand=24 bits, exponent=8 bits
FP16: eps=0.0009765625, u=0.00048828125
```

```
  normal floor=6.1035156e-05, smallest subnormal=5.9604645e-08, largest finite=65504
FP32: eps=1.1920929e-07, u=5.9604645e-08
  normal floor=1.1754944e-38, smallest subnormal=1.4012985e-45, largest finite=3.4028235e+38
```

Increasing exponent bits extends the range; increasing significand bits refines the grid within a binade. Those are different resources. A value can be finite but too coarsely resolved for the update applied to it. Conversely, a computation can have fine relative resolution throughout its normal range and still overflow because that range is too short.

Subnormal values extend the grid below the smallest normal with reduced relative precision. A system may also change their execution behavior. The portable claim is therefore not merely a smallest-positive number: the contract must say whether subnormals are supported or flushed.

> **i** Field note: a dtype name leaves four blanks
>
> Record **storage**, **compute**, **accumulation**, and **output** dtypes separately. Then record the rounding rule and exceptional-value behavior. A reduction can read narrow operands, multiply in one format, accumulate in a wider format, and round to a narrow output. Calling the entire path "FP16" hides the operation most likely to fail.

This chapter simulates the declared rounding contract on a CPU. It never uses CPU FP16 timing as evidence about accelerator performance.

3.4. Two losses that no longer look alike

Chapter 2 separated two failures:

1. distinctions can disappear when real inputs are represented on a grid;
2. represented inputs can survive, while the evaluation path destroys the desired result.

Local spacing now makes the first boundary quantitative. If two real values round to the same stored value, no later algorithm can infer which one arrived. Wider accumulation begins too late.

Cancellation is different. Suppose x and y are close, but the represented operands satisfy

$$\hat{x} = x(1+\delta_x), \qquad \hat{y} = y(1+\delta_y), \qquad |\delta_x|, |\delta_y| \le u.$$

Theorem 3.2 (Cancellation amplification)**.** *If $x \neq y$, then even before rounding the subtraction itself,*

$$\frac{|(\hat{x}-\hat{y})-(x-y)|}{|x-y|} \le u\frac{|x|+|y|}{|x-y|}. \tag{3.4}$$

3.4.1. Proof

Substitute the represented operands:

$$(\hat{x} - \hat{y}) - (x - y) = x\delta_x - y\delta_y.$$

The triangle inequality gives

$$|x\delta_x - y\delta_y| \le u(|x| + |y|).$$

Divide by the nonzero exact difference $|x - y|$. □

When $|x - y|$ is tiny compared with $|x| + |y|$, the subtraction exposes errors that were small relative to the operands but large relative to the answer. Performing that final subtraction in a wider format does not restore bits already lost while the two operands were formed.

This closes the first promise from Chapter 2. Its raw moments occupied the scale of μ^2, and their rounded values differed by exactly one $\mathrm{ulp}(\mu^2) = 32$. The impossible variance was therefore -32, while the centered algorithms avoided constructing those two large nearby operands. At a still larger offset, distinct deviations eventually map to the same FP32 input; that later representation loss is irrecoverable.

The bound in Equation 3.4 is a diagnostic sensitivity statement. It does not include the addition count, reduction tree, fused operations, or accumulator format of a complete kernel. Those belong in the precision contract.

3.5. Resolution must be allocated

A finite grid is not merely chosen; it is placed. Consider a signed b-bit symmetric integer grid with

$$q_{\max} = 2^{b-1} - 1, \qquad s = \frac{\max_i |x_i|}{q_{\max}}, \qquad Q_s(x_i) = s\ \mathrm{clip}\left(\mathrm{round}\left(\frac{x_i}{s}\right), -q_{\max}, q_{\max}\right). \tag{3.5}$$

One scale for $[0.1, 0.2, 0.3, 100]$ must cover the outlier. The following model uses eight signed bits. It is a uniform integer quantizer, not a floating-point format.

1. Quantize all four coordinates with one declared symmetric scale.
2. Quantize the first three coordinates as one block and the outlier as another.
3. Compare integer codes and reconstructed values.
4. Assert that only the global scale erases every small coordinate.

```
values = np.array([0.1, 0.2, 0.3, 100.0])

# [1]
global_grid = symmetric_quantize(values, bits=8)

# [2]
blocked_grid = symmetric_quantize(values, bits=8, block_size=3)
```

```
# [3]
for name, result in (("global", global_grid), ("blocked", blocked_grid)):
    print(
        f"{name:>7}: scales={np.array2string(result.scales, precision=6)}, "
        f"codes={result.integer_codes.tolist()}"
    )
    print(
        " " * 9
        + f"reconstructed={np.array2string(result.dequantized, precision=6)}"
    )

# [4]
assert np.all(global_grid.integer_codes[:3] == 0)
assert np.all(blocked_grid.integer_codes[:3] != 0)
```

```
 global: scales=[0.787402], codes=[0, 0, 0, 127]
         reconstructed=[  0.   0.   0. 100.]
blocked: scales=[0.002362 0.787402], codes=[42, 85, 127, 127]
         reconstructed=[9.921260e-02 2.007874e-01 3.000000e-01 1.000000e+02]
```

The global step is 0.787402; every small coordinate lies below half that step and rounds to zero. The first block's step is 0.002362, so those coordinates receive useful codes. Nothing about the bit count alone predicted the difference. Scale granularity was part of the algorithm.

Blocking is not free. It stores more scales, adds grouping rules, and can introduce discontinuities at block boundaries. The supported conclusion is not that smaller blocks are always better. It is that a claim about quantization error is incomplete until scale selection and block axes are specified.

3.6. Changing the rounding rule

The opening update is always pushed toward the old state by deterministic rounding. A different rounding rule can remove that one-sided local error.

Let $a < x < b$ be adjacent grid points. Define stochastic adjacent rounding by

$$\Pr(Q_{\mathrm{sr}}(x) = a) = \frac{b - x}{b - a}, \qquad \Pr(Q_{\mathrm{sr}}(x) = b) = \frac{x - a}{b - a}. \tag{3.6}$$

Theorem 3.3 (Unbiased adjacent stochastic rounding)**.** *Under Equation 3.6,*

$$\mathbb{E}[Q_{\mathrm{sr}}(x)] = x, \qquad \mathbb{E}[(Q_{\mathrm{sr}}(x) - x)^2] = (x - a)(b - x). \tag{3.7}$$

3.6.1. Proof

The expectation is

$$a\frac{b-x}{b-a} + b\frac{x-a}{b-a} = x.$$

The error equals $-(x-a)$ when rounding to a and $b-x$ when rounding to b. Its second moment is

$$(x-a)^2\frac{b-x}{b-a} + (b-x)^2\frac{x-a}{b-a} = (x-a)(b-x).$$

Because the mean error is zero, this second moment is also the variance. □

At the opening state, each requested value is an integer plus 0.25 and the grid spacing remains one. Each step therefore moves upward with probability 0.25. Under independent draws, the endpoint has the model

$$1024 + \text{Binomial}(400, 0.25),$$

with mean 1124 and standard deviation $\sqrt{400(0.25)(0.75)} \approx 8.66$.

1. Run seeded stochastic-rounding trials under the opening contract.
2. Estimate the endpoint mean, standard deviation, and standard error.
3. Compare the mean with the exact endpoint and the binomial model.
4. Plot the endpoint distribution with both reference locations.

```
# [1]
stochastic_seed, trials = 6212, 20_000
endpoints = stochastic_update_trials(
    initial,
    increment,
    steps,
    trials,
    dtype=np.float16,
    seed=stochastic_seed,
).astype(np.float64)

# [2]
endpoint_mean = float(endpoints.mean())
endpoint_sd = float(endpoints.std(ddof=1))
endpoint_se = endpoint_sd / np.sqrt(trials)

# [3]
model_sd = np.sqrt(steps * 0.25 * 0.75)
assert abs(endpoint_mean - trace.exact_final) < 4 * endpoint_se
assert abs(endpoint_sd - model_sd) < 0.2

# [4]
fig, ax = plt.subplots(figsize=(7.0, 3.8))
```

```
ax.hist(endpoints, bins=np.arange(endpoints.min() - 0.5, endpoints.max() + 1.5),
        color="#5379AA", alpha=0.82)
ax.axvline(trace.exact_final, color="#2E7D32", linewidth=2.2,
           label="Exact endpoint")
ax.axvline(trace.final, color="#9C2F2F", linewidth=2.2, linestyle="--",
           label="Deterministic endpoint")
ax.set(xlabel="stored endpoint", ylabel="trial count")
ax.legend(frameon=False)
plt.show()

print(
    f"mean={endpoint_mean:.5f}, sd={endpoint_sd:.5f}, "
    f"SE(mean)={endpoint_se:.5f}"
)
print(
    f"normal 95% interval for mean="
    f"[{endpoint_mean - 1.96 * endpoint_se:.5f}, "
    f"{endpoint_mean + 1.96 * endpoint_se:.5f}]"
)
```

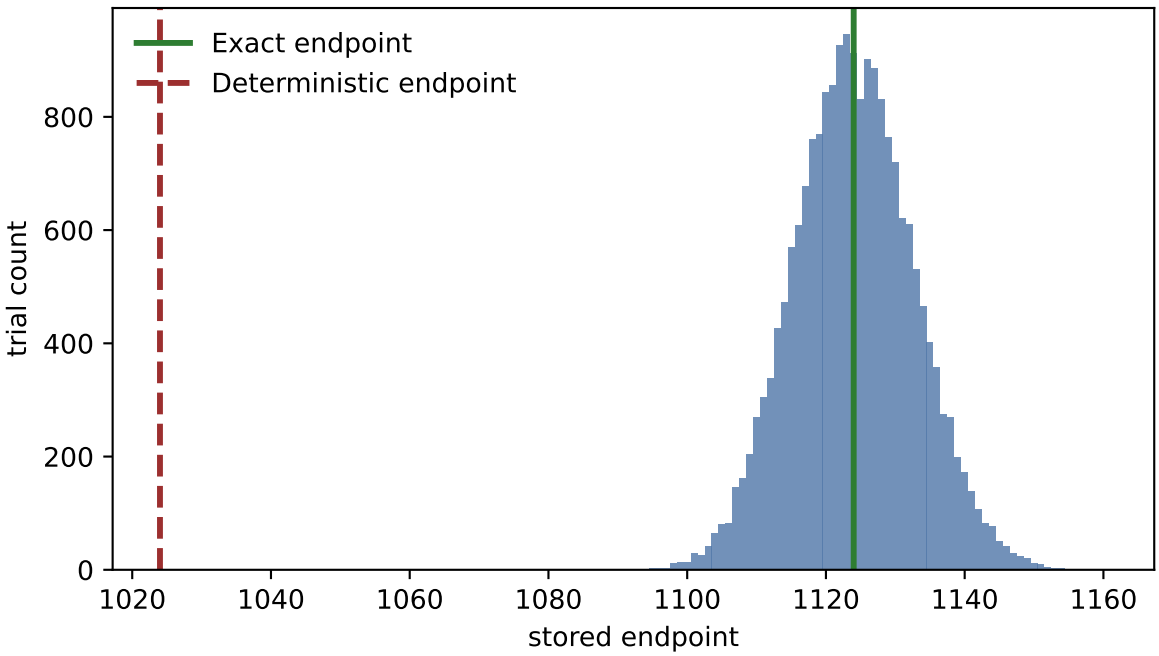


Figure 3.2.: Across 20,000 seeded trials, stochastic adjacent rounding centers the endpoint distribution near the exact endpoint 1124, whereas deterministic rounding ends at 1024. The figure supports an expectation claim for this declared model, not exact trajectories or general training convergence.

```
mean=1124.00655, sd=8.64855, SE(mean)=0.06115
normal 95% interval for mean=[1123.88669, 1124.12641]
```

The estimated mean is 1124.00655, with standard error 0.06115. The exact endpoint lies inside the displayed normal-approximation interval for the mean. Individual trajectories remain noisy; their standard deviation is about 8.65.

⚠ Named wrong answer: 'Unbiased rounding makes the training run unbiased'

The theorem concerns one rounding event conditional on its input. A later state depends on earlier random outcomes, and nonlinear maps do not generally preserve unbiasedness. Stochastic rounding prevents this deterministic sub-grid stagnation in expectation. It does not prevent overflow, choose a good scale, stabilize an update rule, or guarantee convergence.

3.7. Diagnose before prescribing

The same symptom—no visible progress—admits different repairs.

Diagnostic	Instrument	Matched repair	Repair cannot do
Value exceeds range	Peak and exceptional-value trace	Rescale, widen exponent range, or change the evaluation	Recover an overflowed intermediate after the fact
Update lies below local midpoint	ulp(w) versus $\lvert\Delta w\rvert$	Retain a wider state, rescale, accumulate increments, or use a scoped rounding change	Make the mathematical update direction correct
Inputs already coincide on the grid	Same-input wider reference	Change upstream storage or scaling	Infer distinctions absent from the input
Nearby large operands expose prior errors	Cancellation ratio and wider controls	Reformulate or form operands more accurately	Repair arbitrary ill-conditioning
One outlier sets a global quantization step	Per-block maxima and zero-code rate	Allocate scales by a declared block or channel	Eliminate scale metadata and boundary costs
Long reduction loses increments	Accumulator dtype and reduction-order audit	Widen or compensate the accumulation	Establish accelerator speed from CPU timing

This table is the precision language promised by Chapter 2. The dtype is one field inside the diagnosis, not the diagnosis itself.

The same discipline will return in later acts. A variance-preserving initialization must keep values in range as well as control exact moments. Batch-coordinate normalization inherits the centered-state estimator and its accumulator contract. An edge-of-stability trace needs a precision control before a spike is assigned to curvature. A quantized update must declare where its scale came from.

Check yourself

A stored parameter is 2048 in FP16 and a computation requests +0.75. Without running code, compute the local spacing, identify the two adjacent stored candidates, and predict the round-to-nearest result. Which part of the answer would change under stochastic adjacent rounding?

3.8. Okay, so —

- **Inherited:** Chapter 2 showed that algebraically equal formulas can have different execution behavior and that wider downstream arithmetic cannot reconstruct absent inputs.
- **Changed:** "FP16" is no longer an explanation. The operative object is a contract over storage, compute, accumulation, output, rounding, range, and scale.
- **Instrumented:** The harness now measures local spacing, queries format envelopes, simulates round-after-each-step trajectories, audits block quantizers, and estimates stochastic endpoints with uncertainty.
- **Established:** Binade spacing predicts the stalled update; cancellation amplifies operand errors relative to a small difference; stochastic adjacent rounding is locally unbiased with an exact error variance.
- **Unresolved:** Is an observed training failure numerical, dynamic, or geometric, and which control separates those explanations?

3.9. Sources and further reading

IEEE 754 specifies floating-point formats and operations (IEEE Standard for Floating-Point Arithmetic 2019). Higham develops the standard rounding model, cancellation, stability, and error analysis in detail (Higham 2002). Connolly, Higham, and Mary analyze stochastic rounding and its probabilistic backward-error behavior (Connolly et al. 2021). The FP8 proposal by Micikevicius and collaborators is useful modern format context, but it does not make a format name a complete execution contract (Micikevicius et al. 2022).

The Probabilistic Numerics tutorial contrasts uncertainty from limited computation with uncertainty from limited data (Hennig et al. 2026). Exercise 7 uses that analogy as an audit boundary: neither a deterministic rounding bound nor local unbiasedness is automatically a calibrated posterior over an unfinished computation.

For introductory format anatomy, mixed-precision roles, and the hardware motivation, see the sibling volume's precision and hardware appendix. This chapter owns the local-grid diagnosis, the cancellation boundary, resolution allocation, and the rounding-rule audit.

Reading order. Start with Higham (2002) for the error-analysis language, consult (IEEE Standard for Floating-Point Arithmetic 2019) for the exact format contract, and then use Connolly et al. (2021) to audit what stochastic rounding changes.

3.10. Exercises

1. **(Pencil.) Binade boundaries.** For FP16, compute the spacing immediately below 2^{10}, at 2^{10}, and at 2^{11}. Explain why $\mathrm{ulp}(x)$ jumps at a power of two even though relative precision remains of the same order.
2. **(Pencil.) Cancellation control.** Let $x = 1 + \eta$ and $y = 1$, with $\eta > 0$. Use Equation 3.4 to identify the amplification factor. Then explain why the bound diagnoses sensitivity without predicting the exact sign of the error.

3. **(Code.) Stagnation map.** Sweep represented FP16 states across several binades and requested positive updates. Predict stagnation from the local midpoint before running the harness. Report a confusion matrix between the prediction and observed state changes.

4. **(Code.) Scale allocation.** Add one adjustable outlier to a vector of small coordinates. Compare global, fixed-block, and per-coordinate symmetric scales using reconstruction error and zero-code rate. Count scale metadata explicitly; do not rank methods on error alone.

5. **(Audit.) The incomplete precision claim.** A report says, "The model was trained in FP16 and remained stable." Write the missing storage, compute, accumulation, output, rounding, exceptional-value, and scaling questions. For each question, name one rival explanation it could eliminate.

6. **(Audit.) Paper audit: FP8 formats.** Complete the **Claim**, **Mathematical object**, **Evidence culture and interface**, **Numerical and hardware contract**, and **Transfer verdict** fields for the primary FP8 format proposal cited above. Extract one claim about a format, one about an execution path, and one empirical result. For each, record its assumptions and identify whether the paper's evidence is a proof, model calculation, simulation, or hardware measurement. Do not convert a paper-reported endpoint into a reproduced book claim.

7. **(Audit.) Computational uncertainty.** Compare three objects: a deterministic roundoff bound, stochastic adjacent rounding under the model in this chapter, and a probabilistic numerical posterior (Hennig et al. 2026). For each, record the source of randomness, what is conditioned on, what event the uncertainty statement covers, and what calibration evidence would be needed. Explain why local unbiasedness of a rounding error does not by itself quantify uncertainty in the exact computation.

4. One Step, Many Clocks: Quadratic Stability and Conditioning

LG · Landscape and update geometry Geometric supporting · Dynamic primary · Algorithmic supporting

One update rule acts on the two coordinates of

$$\mathcal{L}(\mathbf{e}) = \frac{1}{2}\mathbf{e}^{\mathsf{T}} \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix} \mathbf{e}, \qquad \mathbf{e}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{4.1}$$

The step size is $\alpha = 0.019$. It is below the classical stability limit. The objective decreases at every step. Yet the second coordinate crosses zero on every update while the first coordinate appears almost frozen.

! Prediction

Before running the recurrence, decide:

1. Which coordinate limits the largest safe step?
2. Which coordinate determines the time needed to reach a small error?
3. Can a decreasing scalar objective reveal the sign alternation?

Name the quantity you would record to distinguish the three answers.

4.1. Rotate the trace, not the story

A first optimization course supplies the update $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha\nabla\mathcal{L}(\mathbf{w}_t)$. The diagnostic question begins after that line: what does the update do in each direction?

The opening study uses a centered quadratic, so $\mathbf{e}_t$ is both the iterate and the error from the minimizer. It verifies and activates its exact vendored harness wheel before importing the quadratic instruments.

1. Load the chapter-pinned quadratic instruments.
2. Run four hundred exact quadratic steps.
3. Rotate the iterates into Hessian eigencoordinates.
4. Assert the closed form and classify the mode factors.
5. Plot signed and absolute mode coordinates.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import (
    classify_mode_factors,
    quadratic_spectrum,
    quadratic_trace,
    scalar_noisy_quadratic_trials,
)

# [2]
hessian = np.diag([1.0, 100.0])
initial_error = np.array([1.0, 1.0])
step_size, steps = 0.019, 400
trace = quadratic_trace(
    hessian,
    initial_error,
    step_size=step_size,
    steps=steps,
)

# [3]
mode_coordinates = trace.mode_coordinates
eigenvalues = trace.eigenvalues

# [4]
factors, labels = classify_mode_factors(
    eigenvalues,
    step_size=step_size,
)
closed_form = factors[None, :] ** np.arange(steps + 1)[:, None]
assert np.allclose(mode_coordinates, closed_form)
assert np.all(np.diff(trace.objectives) < 0)
tolerance = 1e-3
waits = [
    int(np.ceil(np.log(tolerance) / np.log(abs(factor))))
    for factor in factors
]

# [5]
fig, axes = plt.subplots(1, 2, figsize=(8.2, 3.5))
for index, (eigenvalue, label) in enumerate(zip(eigenvalues, labels)):
    axes[0].plot(
        mode_coordinates[:90, index],
        linewidth=2,
        label=fr"$\lambda={eigenvalue:g}$: {label}",
    )
```

```
    axes[1].semilogy(
        np.abs(mode_coordinates[:, index]),
        linewidth=2,
        label=fr"$\lambda={eigenvalue:g}$",
    )
axes[0].axhline(0, color="black", linewidth=0.8)
axes[0].set(xlabel="step", ylabel="signed eigencoordinate")
axes[1].axhline(tolerance, color="black", linestyle="--", linewidth=1)
axes[1].set(xlabel="step", ylabel="absolute eigencoordinate")
for ax in axes:
    ax.grid(alpha=0.25)
    ax.legend(frameon=False)
plt.tight_layout()
plt.show()

print(f"harness={manifest['harness_ref']} wheel={manifest['wheel_sha256'][:12]}")
for eigenvalue, factor, label, wait in zip(eigenvalues, factors, labels, waits):
    print(
        f"lambda={eigenvalue:g}: factor={factor:.3f}, "
        f"{label}, steps-to-1e-3={wait}"
    )
```

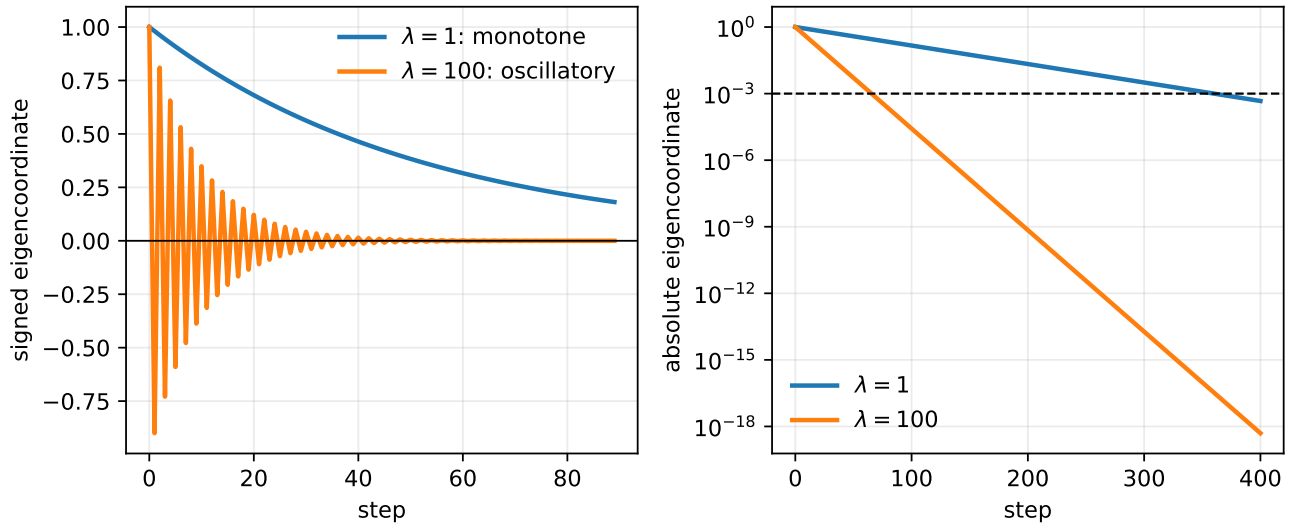


Figure 4.1.: The same scalar step creates two clocks. The flat mode stays positive and decays by 0.981 per step; the sharp mode alternates sign and decays in magnitude by 0.9. The figure establishes the exact behavior of this fixed quadratic recurrence, not the behavior of a changing training landscape.

```
harness=ch-04 wheel=a3478fd77e8c
lambda=1: factor=0.981, monotone, steps-to-1e-3=361
lambda=100: factor=-0.900, oscillatory, steps-to-1e-3=66
```

The sharp mode limits the step because its multiplier reaches magnitude one first. It does **not** set the stopping time here. Its magnitude falls below 10^{-3} after 66 steps, whereas the flat mode waits 361 steps.

The scalar objective is monotone because it squares the coordinates; it cannot show that the sharp coordinate changes sign.

4.2. The update matrix is the instrument

Let $\mathbf{Q} \in \mathbb{R}^{d\times d}$ be symmetric positive definite and

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}(\mathbf{w} - \mathbf{w}^\star)^\mathsf{T}\mathbf{Q}(\mathbf{w} - \mathbf{w}^\star).$$

With $\mathbf{e}_t := \mathbf{w}_t - \mathbf{w}^\star$, one gradient step becomes

$$\mathbf{e}_{t+1} = (\mathbf{I} - \alpha\mathbf{Q})\mathbf{e}_t. \tag{4.2}$$

Theorem 4.1 (Exact quadratic mode dynamics)**.** *Write* $\mathbf{Q} = \mathbf{U}\,\mathrm{diag}(\lambda_1, \dots, \lambda_d)\mathbf{U}^\mathsf{T}$ *with* $0 < \lambda_{\min} \le \lambda_k \le \lambda_{\max}$ *and define* $\mathbf{z}_t = \mathbf{U}^\mathsf{T}\mathbf{e}_t$*. Then*

$$z_{k,t} = (1 - \alpha\lambda_k)^t z_{k,0}. \tag{4.3}$$

All errors converge to zero exactly when

$$0 < \alpha < \frac{2}{\lambda_{\max}}. \tag{4.4}$$

4.2.1. Proof

Left-multiply Equation 4.2 by $\mathbf{U}^\mathsf{T}$:

$$\mathbf{z}_{t+1} = (\mathbf{I} - \alpha\,\mathrm{diag}(\lambda_1, \dots, \lambda_d))\,\mathbf{z}_t.$$

The recurrence is diagonal, so the kth coordinate multiplies by $1 - \alpha\lambda_k$ at every step, which gives Equation 4.3. That coordinate converges for every initial value exactly when $|1-\alpha\lambda_k| < 1$, or $0 < \alpha\lambda_k < 2$. Satisfying the inequality for the largest eigenvalue satisfies it for all modes. □

The same recurrence is explicit Euler applied to the quadratic gradient flow

$$\dot{\mathbf{e}}(t) = -\mathbf{Q}\mathbf{e}(t),$$

with time step $h = \alpha$. On an eigenmode of curvature λ_k, the continuous flow multiplies by $e^{-h\lambda_k}$ while explicit Euler uses its stability polynomial $R(-h\lambda_k) = 1 - h\lambda_k$. Thus the optimization boundary $0 < h\lambda_k < 2$ is also a discretization-stability boundary. This identity is exact for the fixed quadratic control. It does not say that a nonlinear stochastic training trajectory is merely its continuous gradient flow sampled on a grid.

Mode factor $r_k = 1 - \alpha\lambda_k$	What the coordinate does
$0 < r_k < 1$	decays without changing sign
$-1 < r_k < 0$	decays while alternating sign
$r_k = 0$	vanishes in one step
$\lvert r_k\rvert = 1$	remains on the stability boundary
$\lvert r_k\rvert > 1$	grows in magnitude

⚠ Named wrong answer: 'The largest eigenvalue controls convergence'

It controls the largest stable scalar step. Once that step is chosen, the smallest eigenvalue often controls the slowest decay. "Controls convergence" collapses two different diagnostic jobs into one phrase.

This is the first public object in the landscape/update-geometry thread: the landscape supplies eigendirections and curvature; the update assigns a scalar multiplier to each one.

4.3. The condition number is a waiting-time tax

A smaller step avoids divergence but does not make all clocks agree. Define

$$\kappa := \frac{\lambda_{\max}}{\lambda_{\min}}.$$

For a constant scalar step, the worst mode contracts by

$$\rho(\mathbf{I} - \alpha\mathbf{Q}) = \max_k |1 - \alpha\lambda_k|.$$

Theorem 4.2 (Best constant scalar step)**.** *Among positive constant scalar steps, the smallest worst-mode contraction is attained at*

$$\alpha^\star = \frac{2}{\lambda_{\max} + \lambda_{\min}},$$

and equals

$$\rho^\star = \frac{\kappa - 1}{\kappa + 1}. \tag{4.5}$$

4.3.1. Proof

Because $\lambda \mapsto |1 - \alpha\lambda|$ is convex, its maximum on $[\lambda_{\min}, \lambda_{\max}]$ occurs at an endpoint. At the minimax step, the endpoint errors have equal magnitude and opposite signs:

$$1 - \alpha^\star \lambda_{\min} = -\left(1 - \alpha^\star \lambda_{\max}\right).$$

Solving gives $\alpha^\star = 2/(\lambda_{\max} + \lambda_{\min})$. Substitution gives

$$1 - \alpha^\star \lambda_{\min} = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{\kappa - 1}{\kappa + 1}. \qquad \square$$

When κ is large, $\rho^\star$ is close to one. A scalar step cannot simultaneously erase a flat mode and remain stable in a sharp mode.

The word *best* needs an object.

The step above is minimax over every eigenvalue in the interval and remains fixed for every iteration. If the question is instead "which step minimizes the objective along the current negative-gradient direction?", the answer changes. With $\mathbf{g}_t = \mathbf{Q}\mathbf{e}_t$,

$$\mathcal{L}(\mathbf{w}_t - \alpha\mathbf{g}_t) = \mathcal{L}(\mathbf{w}_t) - \alpha\|\mathbf{g}_t\|_2^2 + \frac{\alpha^2}{2}\mathbf{g}_t^\mathsf{T}\mathbf{Q}\mathbf{g}_t.$$

The one-step line minimizer is therefore

$$\alpha_t^{\text{line}} = \frac{\|\mathbf{g}_t\|_2^2}{\mathbf{g}_t^\mathsf{T}\mathbf{Q}\mathbf{g}_t}. \tag{4.6}$$

Its denominator is curvature in the direction the update actually uses, not the largest curvature available anywhere. The value lies between $1/\lambda_{\max}$ and $1/\lambda_{\min}$ and can exceed $2/\lambda_{\max}$ while still decreasing this quadratic on the current step. That does not repeal Equation 4.4: it answers a different question with a state-dependent step.

We now have three distinct optimization targets: a worst-direction fixed step, a current-direction one-step choice, and a sequence chosen for progress over a horizon. C16 will add the missing complication that the curvature and the active directions can move with the trajectory.

4.4. Change the geometry seen by the update

Consider a positive-definite matrix $\mathbf{P}$ and the update

$$\mathbf{e}_{t+1} = (\mathbf{I} - \alpha\mathbf{P}\mathbf{Q})\,\mathbf{e}_t. \tag{4.7}$$

This is a matrix change of scale, or **preconditioning**. The relevant eigenvalues are those of the effective operator. For positive $\mathbf{P}$ and $\mathbf{Q}$, $\mathbf{PQ}$ is similar to the symmetric positive matrix $\mathbf{P}^{1/2}\mathbf{Q}\mathbf{P}^{1/2}$, so its eigenvalues are positive. The repair succeeds when those eigenvalues are more tightly clustered.

The next control does not use the oracle inverse. It scales the sharp coordinate by $1/25$, changing effective eigenvalues $(1, 100)$ into $(1, 4)$.

1. Run thirty unscaled steps under the opening contract.
2. Declare a diagonal preconditioner and its effective spectrum.
3. Use the optimal scalar step for the compressed spectrum.
4. Compare error norms and assert the predicted separation.

```
# [1]
raw = quadratic_trace(
    hessian,
    initial_error,
    step_size=step_size,
    steps=30,
)

# [2]
preconditioner = np.diag([1.0, 1 / 25])
effective_eigenvalues = np.linalg.eigvalsh(preconditioner @ hessian)
effective_kappa = effective_eigenvalues[-1] / effective_eigenvalues[0]

# [3]
compressed_step = 2 / effective_eigenvalues.sum()
compressed = quadratic_trace(
    hessian,
    initial_error,
    step_size=compressed_step,
    steps=30,
    preconditioner=preconditioner,
)

# [4]
raw_norms = np.linalg.norm(raw.iterates, axis=1)
compressed_norms = np.linalg.norm(compressed.iterates, axis=1)
assert np.isclose(effective_kappa, 4.0)
assert compressed.final_error_norm < 1e-6
assert raw.final_error_norm > 0.5

fig, ax = plt.subplots(figsize=(7.0, 3.7))
ax.semilogy(raw_norms, linewidth=2.2, label=r"raw spectrum: $\kappa=100$")
ax.semilogy(
    compressed_norms,
    linewidth=2.2,
    label=r"effective spectrum: $\kappa=4$",
)
ax.set(xlabel="step", ylabel=r"$\|\mathbf{e}_t\|_2$")
ax.grid(alpha=0.25)
ax.legend(frameon=False)
plt.show()

print(
    f"effective eigenvalues={effective_eigenvalues.tolist()}, "
```

```
    f"kappa={effective_kappa:g}, step={compressed_step:g}"
)
print(
    f"step 30: raw norm={raw.final_error_norm:.6f}, "
    f"preconditioned norm={compressed.final_error_norm:.9f}"
)
```

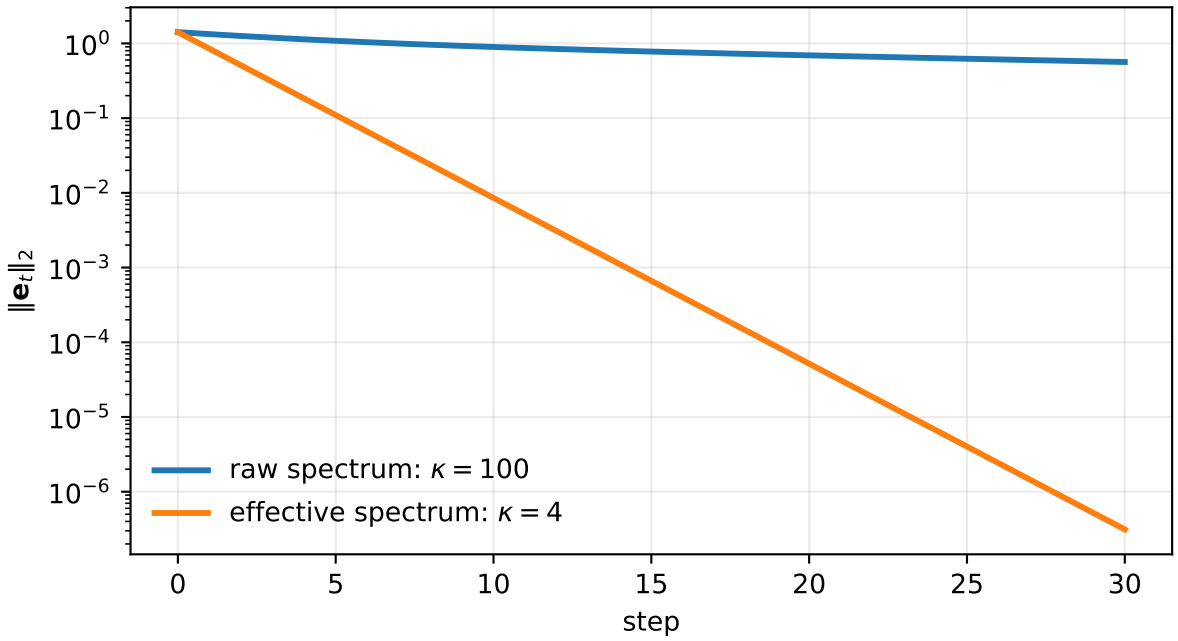


Figure 4.2.: A declared diagonal preconditioner compresses the effective spectrum from condition number 100 to 4. After 30 iterations the raw error norm is about 0.564, while the preconditioned error is about 3.13e-7. This is an exact quadratic model comparison; it does not establish the quality or cost of estimating a preconditioner in a changing problem.

```
effective eigenvalues=[1.0, 4.0], kappa=4, step=0.4
step 30: raw norm=0.564028, preconditioned norm=0.000000313
```

> **Field note: count the cost of the repair**
>
> An iteration reduction is not yet a wall-clock speedup. Applying or estimating $\mathbf{P}$ moves bytes and performs work. The book will not convert a condition number into accelerator time without a hardware contract.

4.5. A stochastic update has no point endpoint

Now isolate one scalar mode of curvature $\lambda > 0$. Suppose the observed gradient contains additive noise:

$$e_{t+1} = (1 - \alpha\lambda)e_t - \alpha\xi_t, \qquad \mathbb{E}[\xi_t] = 0, \qquad \mathbb{E}[\xi_t^2] = \tau^2. \tag{4.8}$$

The noise is independent across time and independent of the current state. These assumptions are the model, not decorative caveats.

Theorem 4.3 (Exact second moment under additive noise)**.** *If* $|1-\alpha\lambda|<1$*, then*

$$\mathbb{E}[e_t^2] = (1-\alpha\lambda)^{2t}e_0^2 + \frac{\alpha\tau^2}{\lambda(2-\alpha\lambda)}\left[1-(1-\alpha\lambda)^{2t}\right]. \tag{4.9}$$

Consequently,

$$\lim_{t\to\infty}\mathbb{E}[e_t^2] = \frac{\alpha\tau^2}{\lambda(2-\alpha\lambda)}. \tag{4.10}$$

4.5.1. Proof

Let $r = 1-\alpha\lambda$. Squaring Equation 4.8 and taking expectations removes the cross term by independence and zero mean:

$$\mathbb{E}[e_{t+1}^2] = r^2\mathbb{E}[e_t^2] + \alpha^2\tau^2.$$

Unrolling the scalar recurrence gives

$$\mathbb{E}[e_t^2] = r^{2t}e_0^2 + \alpha^2\tau^2\sum_{j=0}^{t-1} r^{2j}.$$

The geometric sum is $(1-r^{2t})/(1-r^2)$, and $1-r^2 = \alpha\lambda(2-\alpha\lambda)$. Substitution proves the result. □

1. Simulate independent noisy scalar trajectories.
2. Estimate mean squared error over the trial axis.
3. Compute the exact finite-time and stationary predictions.
4. Assert agreement using the endpoint standard error.
5. Plot the transient and the floor.

```
# [1]
noisy = scalar_noisy_quadratic_trials(
    curvature=1.0,
    step_size=0.1,
    noise_standard_deviation=1.0,
    initial_error=3.0,
    steps=120,
    trials=20_000,
    seed=6213,
)

# [2]
estimated_mse = noisy.mean_squared_error
endpoint_se = noisy.endpoint_standard_error
```

```
# [3]
time = np.arange(121)
noise_factor = 1 - noisy.step_size * noisy.curvature
stationary = noisy.stationary_second_moment
predicted_mse = (
    noise_factor ** (2 * time) * noisy.initial_error**2
    + stationary * (1 - noise_factor ** (2 * time))
)

# [4]
assert abs(estimated_mse[-1] - predicted_mse[-1]) < 4 * endpoint_se

# [5]
fig, ax = plt.subplots(figsize=(7.0, 3.7))
ax.semilogy(time, estimated_mse, linewidth=1.8, label="Monte Carlo mean")
ax.semilogy(time, predicted_mse, linewidth=2.2, label="exact finite-time moment")
ax.axhline(stationary, color="black", linestyle="--", label="stationary moment")
ax.set(xlabel="step", ylabel=r"$\widehat{\mathbb{E}}[e_t^2]$")
ax.grid(alpha=0.25)
ax.legend(frameon=False)
plt.show()

print(
    f"endpoint MSE={estimated_mse[-1]:.8f}, "
    f"SE={endpoint_se:.8f}"
)
print(
    f"finite-time prediction={predicted_mse[-1]:.8f}, "
    f"stationary={stationary:.8f}"
)
```

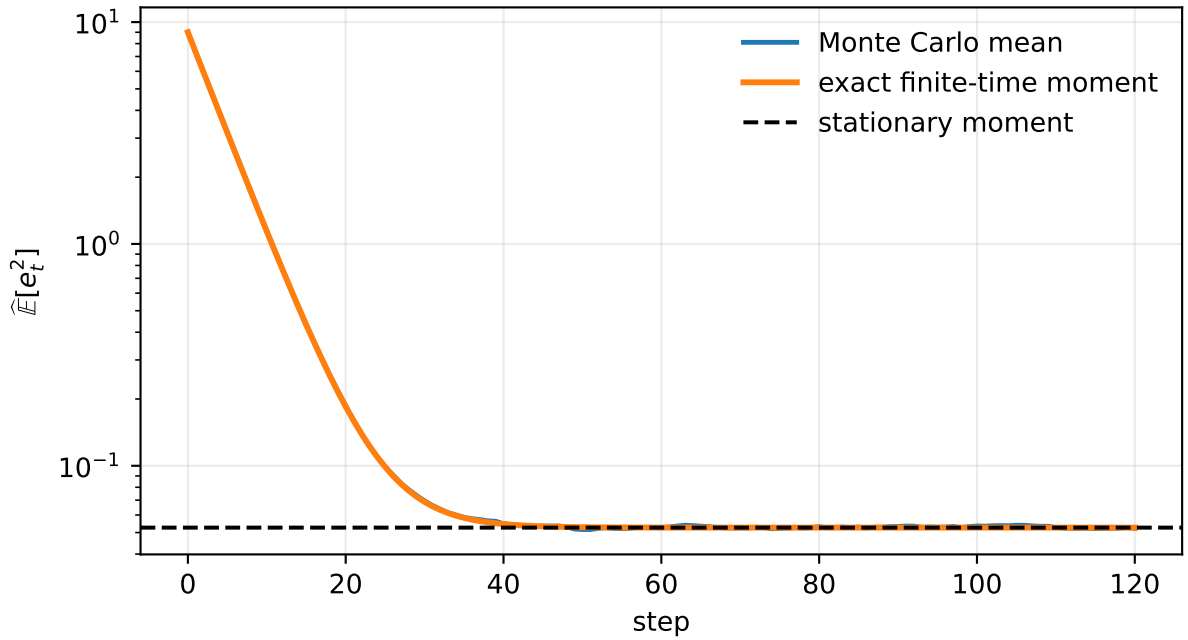


Figure 4.3.: Across 20,000 independent trials, the estimated squared error approaches the exact finite-time prediction and the stationary value 0.05263. The figure supports the additive, independent, finite-variance model in Equation 4.8; it does not establish a universal stochastic-gradient floor.

```
endpoint MSE=0.05229016, SE=0.00052526
finite-time prediction=0.05263158, stationary=0.05263158
```

This floor is algorithmic. C03's stalled update was representational. A coordinate can also appear stationary because its deterministic multiplier is near one. These diagnoses require different controls:

Symptom	Instrument	Supported diagnosis
coordinate alternates sign	signed eigencoordinate	negative stable mode factor
coordinate decays very slowly	$\lvert 1-\alpha\lambda_k\rvert$	conditioning tax
mean squared error plateaus across independent trials	estimator plus exact recurrence	stochastic floor under the declared model
requested change is below a local midpoint	local ULP audit	finite-grid stagnation
measured time remains large after iteration count falls	bytes/work profile	implementation cost

The scalar model keeps τ^2 and the curvature fixed, so it cannot diagnose when a stochastic run changes regime. C13 is obligated to replace ξ_t by a declared conditional gradient estimator, compare gradient signal with estimator variation, and test what happens when the second moment is not finite.

The fixed boundary in Equation 4.4 is now a registered control. C16 is obligated to compare it with local top curvature, curvature in the current gradient direction, and a setting in which the curvature itself moves during training.

Check yourself

For eigenvalues $(2, 20)$ and step $\alpha = 0.075$, what are both mode factors? How do you classify their sign behavior, which is the slow mode, and does reducing the step to 0.025 remove oscillation, improve the slowest contraction, both, or neither?

4.6. Okay, so —

- **Inherited:** C03 taught us to separate a missing update from a representational failure; basic gradient mechanics are assumed.
- **Changed:** A gradient trajectory is now an operator recurrence with a separate clock in every eigendirection.
- **Instrumented:** The harness records spectra, mode factors, exact quadratic traces, preconditioned traces, and stochastic second moments.
- **Established:** The stability interval, optimal constant-step contraction, and additive-noise floor follow exactly under their stated models.
- **Unresolved:** What probabilistic control survives when there are many possible directions and the stochastic perturbation is not fixed, additive, or light-tailed?

4.7. Sources and further reading

Nesterov develops gradient methods for smooth, strongly convex objectives and the role of conditioning (Nesterov 2004). Robbins and Monro supply the foundational stochastic-approximation setting (Robbins and Monro 1951). This chapter uses those sources as context, but derives its three exact recurrences on the page. Schmidt's ICML 2026 tutorial motivates the global, local, directional, and finite-horizon questions used to delimit "best" (Schmidt 2026); it is an intellectual map, not the technical authority for the equations.

Reading order. Start with Nesterov (2004) for the fixed smooth, strongly-convex control, then read Robbins and Monro (1951) for the stochastic branch and Schmidt (2026) for the research-question map.

4.8. Exercises

1. **(Pencil.) Mode phase diagram.** For a scalar curvature λ, draw the five regimes of $1 - \alpha\lambda$ as a function of α. Mark monotone decay, one-step annihilation, oscillatory decay, boundary behavior, and divergence.

2. **(Pencil.) Bits and conditioning.** Starting from Equation 4.5, derive an exact ceiling formula for the number of iterations required to reduce every eigencoordinate by a factor 2^{-b}. Then give its large-κ approximation and state where the approximation enters.

3. **(Code.) A misleading loss curve.** Construct two quadratic trajectories whose scalar objective curves are nearly indistinguishable for 50 steps but whose signed eigencoordinates have different mode classifications. Report the spectra and the control that separates them.

4. **(Code.) Imperfect preconditioning.** Sweep the second diagonal entry of $\mathbf{P} = \mathrm{diag}(1, p)$ for the opening Hessian. For each value, choose the optimal constant step for the effective spectrum and report iteration count, matrix-application cost model, and final error.

5. **(Audit.) The precision diagnosis.** A report says, "The flat coordinate stopped changing, so the optimizer reached its noise floor." List the eigencoordinate, repeated-trial, and local-ULP controls required before that conclusion is admissible. Predict what each rival explanation would do under those controls.

6. **(Audit.) Paper audit: stochastic approximation.** Complete the **Claim**, **Resolution of the theory**, **Dynamic regime**, **Evidence culture and interface**, and **Assumption stress test** fields for Robbins and Monro's primary paper. Identify which step-size, noise, and convergence claims differ from the constant-step finite recurrence in this chapter. Label each extracted result as theorem, asymptotic statement, or heuristic interpretation.

7. **(Audit.) Act checkpoint — open an Incident Card.** Complete the Act 0 assignment for a numerical or performance failure. Fill Symptom, Prediction, Contract, Precision control, and Corrective control before reading the result. **Route:** Act checkpoint. **Estimated time:** 60 minutes. **Deliverable:** a partial card and one paired rerun. **Hint:** a speedup without a traffic boundary and a reduction over the wrong axis are different contract failures.

Act I — Geometry of high dimensions

Act 0 made execution inspectable: traffic has a boundary, streaming has a state, precision has a local grid, and a quadratic update has independent spectral clocks. What it cannot yet decide is which directions a large random system is likely to expose. Act I builds that arena's instruments: a tail class that survives a union bound, the price of a continuum, two singular edges, a calibrated bulk, and a dimension that counts spectral mass instead of coordinates. The machine does not disappear. Its finite grids and finite samples remain the boundaries against which the geometric claims are audited. For general high-dimensional probability, Vershynin (2026) is the reference spine. This act's narrower job is to turn that theory into calibrated nulls, finite-sample witnesses, and controls that can diagnose a training run.

5. The Safe Zone Is Logarithmic: Sub-Gaussian Concentration and Simultaneous Control

SG · The sub-Gaussian safe zone Geometric primary · Dynamic quiet · Algorithmic supporting

Draw one standard-normal vector of length 100, then one of length one million. In the seeded witness below, their largest magnitudes are 2.50 and 4.71.

The number of opportunities grew by ten thousand. The maximum did not.

! Prediction

If the number of coordinates grows by a factor of ten, should a 99%-simultaneous deviation threshold:

1. grow by a factor of ten;
2. grow by a factor of $\sqrt{10}$; or
3. increase only through a logarithm?

Choose one, then name the tail assumption your answer needs.

5.1. A million chances to fail

The study uses one independent random stream per vector length. Each maximum is a single seeded witness, not an estimate of the expected maximum. The comparison curve is a theorem-derived threshold at failure probability $\delta = 0.01$.

1. Load the chapter-pinned concentration instruments.
2. Draw one independent standard-normal vector at each declared size.
3. Compute the simultaneous sub-Gaussian threshold.
4. Assert the witness remains inside the threshold.
5. Plot observed maxima against logarithmic and polynomial scales.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import (
    empirical_tail_profile,
    moment_growth_profile,
    rademacher_projection_trials,
```

```
    standard_normal_maxima,
    subgaussian_max_threshold,
)

# [2]
counts = np.array([100, 1_000, 10_000, 100_000, 1_000_000])
maxima = standard_normal_maxima(counts, seed=6214)
observed = np.array([maxima[int(count)] for count in counts])

# [3]
delta = 0.01
safe_threshold = np.array(
    [subgaussian_max_threshold(int(count), delta) for count in counts]
)
variance_only_threshold = np.sqrt(counts / delta)

# [4]
assert np.all(observed < safe_threshold)
billion_threshold = subgaussian_max_threshold(1_000_000_000, delta)

# [5]
fig, ax = plt.subplots(figsize=(7.0, 3.8))
ax.loglog(counts, observed, "o-", linewidth=2.2, label="single seeded maximum")
ax.loglog(counts, safe_threshold, linewidth=2.2, label="sub-Gaussian threshold")
ax.loglog(
    counts,
    variance_only_threshold,
    linestyle="--",
    linewidth=1.8,
    label="variance-only threshold",
)
ax.set(xlabel="number of coordinates", ylabel="simultaneous magnitude")
ax.grid(alpha=0.25, which="both")
ax.legend(frameon=False)
plt.show()

print(f"harness={manifest['harness_ref']} wheel={manifest['wheel_sha256'][:12]}")
for count, maximum, threshold in zip(counts, observed, safe_threshold):
    print(f"m={count:>7}: max={maximum:.4f}, 99%-threshold={threshold:.4f}")
print(f"billion-coordinate model threshold={billion_threshold:.4f}")
```

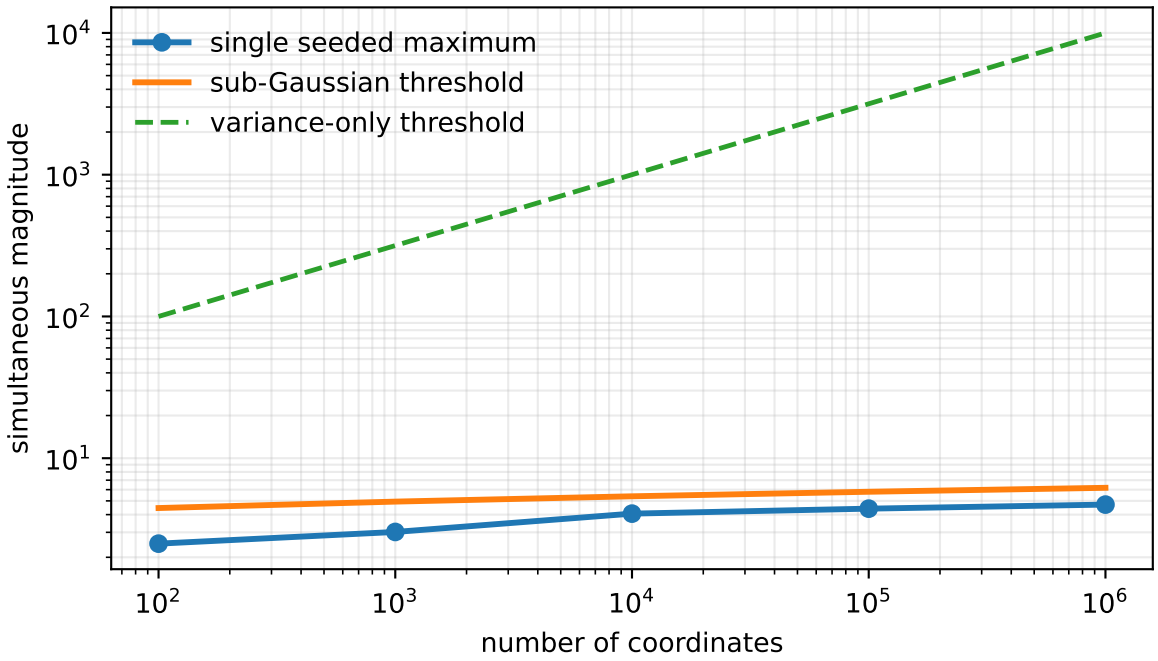


Figure 5.1.: Observed coordinate maxima grow slowly across vector sizes from 100 to one million. The sub-Gaussian 99%-simultaneous threshold grows as the square root of a logarithm, while a variance-only union-bound threshold grows as the square root of the count. The observed line is one seeded witness; it does not estimate the expected maximum or certify an unknown distribution's tail class.

```
harness=ch-05 wheel=316959af6007
m=    100: max=2.4959, 99%-threshold=4.4505
m=   1000: max=3.0185, 99%-threshold=4.9409
m=  10000: max=4.0644, 99%-threshold=5.3868
m= 100000: max=4.4066, 99%-threshold=5.7985
m=1000000: max=4.7068, 99%-threshold=6.1829
billion-coordinate model threshold=7.2141
```

The bound pays for one million opportunities through $\sqrt{\log(10^6)}$, not $\sqrt{10^6}$. The final printed number, 7.2141, is a model calculation for one billion independent standard-normal coordinates under the same proxy scale. The code did not generate a billion values.

The geometric problem is now precise: we need a tail envelope strong enough to survive a union over many quantities, and we need a scale that composes under weighted sums.

5.2. Exponentiate, then optimize

For a centered real random variable X, suppose

$$\mathbb{E}[e^{\lambda X}] \le \exp\left(\frac{\sigma^2\lambda^2}{2}\right) \qquad \text{for every } \lambda \in \mathbb{R}. \tag{5.1}$$

We call σ a **sub-Gaussian MGF proxy scale**. It is an admissible envelope, not automatically the smallest such scale and not automatically the standard deviation.

Theorem 5.1 (MGF envelope implies a Gaussian tail)**.** *Under Equation 5.1, for every $t \geq 0$,*

$$\Pr(|X| \geq t) \leq 2\exp\left(-\frac{t^2}{2\sigma^2}\right). \tag{5.2}$$

5.2.1. Proof

For any $\lambda > 0$, monotonicity of the exponential and Markov's inequality give

$$\Pr(X \geq t) = \Pr(e^{\lambda X} \geq e^{\lambda t}) \leq e^{-\lambda t}\mathbb{E}[e^{\lambda X}] \leq \exp\left(-\lambda t + \frac{\sigma^2\lambda^2}{2}\right).$$

The exponent is a quadratic in λ, minimized at $\lambda^\star = t/\sigma^2$. Substitution gives $\Pr(X \geq t) \leq e^{-t^2/(2\sigma^2)}$. Apply the same argument to $-X$ and add the two tail probabilities. □

This is the exponential lift: a probability question becomes a one-dimensional optimization problem. The theorem does not manufacture an MGF bound. It tells us what the bound buys once we establish it.

For a Rademacher sign $\varepsilon \in \{-1, +1\}$ with equal probabilities,

$$\mathbb{E}[e^{\lambda\varepsilon}] = \cosh(\lambda) \leq e^{\lambda^2/2}. \tag{5.3}$$

Thus a random sign has proxy scale one. The inequality can be proved by comparing the power series or by checking that $e^{-\lambda^2/2}\cosh(\lambda)$ decreases for $\lambda > 0$.

5.3. Coefficient energy controls a sum

The usefulness of Equation 5.1 is compositional.

Theorem 5.2 (Independent weighted sums)**.** *Let $X_1, \dots, X_d$ be independent, centered random variables with MGF proxy scales $\sigma_1, \dots, \sigma_d$. For fixed coefficients $\mathbf{a} \in \mathbb{R}^d$,*

$$S := \sum_{j=1}^{d} a_j X_j$$

has proxy variance

$$\sigma_S^2 = \sum_{j=1}^{d} a_j^2\sigma_j^2. \tag{5.4}$$

5.3.1. Proof

Independence factorizes the MGF:

$$\begin{aligned}\mathbb{E}[e^{\lambda S}] &= \prod_{j=1}^{d} \mathbb{E}[e^{\lambda a_j X_j}] \\ &\le \prod_{j=1}^{d} \exp\left(\frac{\lambda^2 a_j^2 \sigma_j^2}{2}\right) \\ &= \exp\left(\frac{\lambda^2}{2} \sum_{j=1}^{d} a_j^2 \sigma_j^2\right).\end{aligned}$$

This is the required envelope. □

Independence is doing identifiable work: it turns one MGF into a product. Without it, Equation 5.4 is not licensed.

If the X_j are Rademacher and $a_j = 1/\sqrt{d}$, then $\sum_j a_j^2 = 1$. The sum keeps proxy scale one at every width. Width alone does not cause the scalar projection to explode; coefficient energy decides.

5.4. Pay only a logarithm for simultaneous control

Theorem 5.3 (Simultaneous sub-Gaussian control)**.** *Suppose $X_1, \dots, X_m$ each satisfy Equation 5.2 with the same proxy scale σ. Mutual independence is not required. For any $0 < \delta < 1$, with probability at least $1 - \delta$,*

$$\max_{1 \le j \le m} |X_j| \le \sigma \sqrt{2 \log\left(\frac{2m}{\delta}\right)}. \tag{5.5}$$

5.4.1. Proof

For any $t \ge 0$, the union bound gives

$$\Pr\left(\max_j |X_j| \ge t\right) \le \sum_{j=1}^{m} \Pr(|X_j| \ge t) \le 2me^{-t^2/(2\sigma^2)}.$$

Set the last expression equal to δ and solve for t. □

Dependence can make the union bound loose, but it does not invalidate the guarantee. Independence was essential for closing weighted sums; it is not essential for union-bounding already available marginal tails.

> ⚠ Named wrong answer: 'A billion variables require a billion-sigma threshold'
>
> The number of opportunities appears inside a logarithm under a sub-Gaussian tail envelope. The threshold at $m = 10^9$ and $\delta = 0.01$ is about 7.21σ, not $10^9\sigma$. Change the tail class, however, and the scaling can change with it.

Theorem 5.4 (Gaussian input, nonlinear diagnostic)**.** *Let* $\mathbf{G} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ *and let* $f : \mathbb{R}^d \to \mathbb{R}$ *be* L*-Lipschitz in Euclidean distance, with* $L > 0$*. There are universal constants* $c, C > 0$ *such that*

$$\|f(\mathbf{G}) - \mathbb{E}f(\mathbf{G})\|_{\psi_2} \le CL,$$

and therefore

$$\Pr(|f(\mathbf{G}) - \mathbb{E}f(\mathbf{G})| \ge t) \le 2\exp\left(-\frac{ct^2}{L^2}\right), \qquad t \ge 0. \tag{5.6}$$

Proof with pointer. This is Gaussian concentration, stated in the Orlicz form of Vershynin's second-edition Theorem 5.2.3 (Vershynin 2026). The tail form follows from the ψ_2 characterization below. The useful extension is not a new exponent but a new object: f can be nonlinear. For example, $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x}\|_2$ is $\|\mathbf{A}\|_{\mathrm{op}}$-Lipschitz, so its fluctuation around its mean inherits a sub-Gaussian scale.

The theorem controls fluctuation around a declared center; it does not compute that center. Its standard-Gaussian input is also part of the contract. The statement is not a dependence theorem for arbitrary inputs and does not extend unchanged to heavy-tailed data.

5.5. Four instruments for the same safe zone

Sometimes the MGF is easy to calculate. Sometimes moments or a tail bound are what the problem exposes. The following theorem makes those entry points interchangeable at the level of tail class, not at the level of identical constants.

Theorem 5.5 (Equivalent sub-Gaussian diagnostics)**.** *For a centered real random variable* X*, the following properties are equivalent up to universal constant factors:*

1. $\Pr(|X| \ge t) \le 2e^{-t^2/K_1^2}$ *for all* $t \ge 0$*;*
2. $(\mathbb{E}|X|^p)^{1/p} \le K_2\sqrt{p}$ *for every* $p \ge 1$*;*
3. $\mathbb{E}e^{X^2/K_3^2} \le 2$*;*
4. $\mathbb{E}e^{\lambda X} \le e^{K_4^2\lambda^2}$ *for every real* λ*.*

The best admissible constants $K_1, \dots, K_4$ *are comparable up to universal factors.*

The full equivalence is proved in Vershynin's second-edition Proposition 2.6.1 (Vershynin 2026). One direction is immediate here: property 3 and Markov's inequality give

$$\Pr(|X| \ge t) = \Pr\left(e^{X^2/K_3^2} \ge e^{t^2/K_3^2}\right) \le 2e^{-t^2/K_3^2}.$$

The exponential-square characterization defines the Orlicz norm

$$\|X\|_{\psi_2} := \inf\left\{K > 0 : \mathbb{E}e^{X^2/K^2} \le 2\right\}. \tag{5.7}$$

It is a population quantity. An empirical moment curve may reveal a violation or suggest a regime, but a finite sample cannot certify what happens beyond its largest observation.

Available object	Diagnostic question	Boundary
tail probabilities	does $-\log \Pr(\lvert X\rvert \geq t)$ grow quadratically?	rare tails may be unseen
moments	does $\lVert X\rVert_p / \sqrt{p}$ remain bounded?	high-order estimates are unstable
exponential-square moment	is a candidate K integrable?	sample averages can hide divergence
MGF	is log-MGF bounded by a quadratic for all λ?	local-in-λ control gives local tails

> **Field note: proxy variance is not variance**
>
> Differentiating an exact MGF at zero connects its second derivative to variance, so every valid proxy variance is at least the actual variance. Equality is special. Record "MGF proxy scale" when that is the object used by the proof.

5.6. Width without scale explosion

The next study uses independent Rademacher signs and the normalized coefficients $a_j = 1/\sqrt{d}$. For each width, 20,000 independent projections estimate the mean, sample standard deviation, and the two-sided event $|S| \geq 2$. Each width receives a separate named seed.

1. Declare widths, trial count, coefficient norm, and seed policy.
2. Generate normalized Rademacher projections.
3. Estimate mean, standard deviation, and a two-sided tail probability.
4. Report estimator uncertainty and assert order-one scale.
5. Plot scale and tail behavior across widths.

```
# [1]
widths = np.array([16, 64, 256, 1024])
trials = 20_000
base_seed = 6215

# [2]
projection_samples = {}
for index, width in enumerate(widths):
    coefficients = np.ones(width) / np.sqrt(width)
    assert np.isclose(np.linalg.norm(coefficients), 1.0)
    projection_samples[int(width)] = rademacher_projection_trials(
        coefficients,
        trials=trials,
        seed=base_seed + index,
    )

# [3]
means = np.array([projection_samples[int(width)].mean() for width in widths])
standard_deviations = np.array(
```

```
    [projection_samples[int(width)].std(ddof=1) for width in widths]
)
tails = np.array(
    [
        empirical_tail_profile(
            projection_samples[int(width)],
            [2.0],
        )[2.0]
        for width in widths
    ]
)

# [4]
mean_standard_errors = standard_deviations / np.sqrt(trials)
sd_standard_errors = standard_deviations / np.sqrt(2 * (trials - 1))
tail_standard_errors = np.sqrt(tails * (1 - tails) / trials)
assert np.all(np.abs(means) < 4 * mean_standard_errors)
assert np.all(np.abs(standard_deviations - 1) < 0.04)
tail_upper_bound = 2 * np.exp(-2)

# [5]
fig, axes = plt.subplots(1, 2, figsize=(8.2, 3.5))
axes[0].errorbar(
    widths,
    standard_deviations,
    yerr=1.96 * sd_standard_errors,
    marker="o",
    linewidth=2,
    capsize=3,
)
axes[0].axhline(1.0, color="black", linestyle="--", label="theoretical scale")
axes[0].set(xlabel="width", ylabel="sample standard deviation")
axes[1].errorbar(
    widths,
    tails,
    yerr=1.96 * tail_standard_errors,
    marker="o",
    linewidth=2,
    capsize=3,
    label=r"empirical $\Pr(|S|\geq2)$",
)
axes[1].axhline(
    tail_upper_bound,
    color="black",
    linestyle="--",
    label="sub-Gaussian upper bound",
)
axes[1].set(xlabel="width", ylabel="probability")
for ax in axes:
```

```
    ax.set_xscale("log", base=2)
    ax.grid(alpha=0.25)
    ax.legend(frameon=False)
plt.tight_layout()
plt.show()

for index, width in enumerate(widths):
    print(
        f"d={width:>4}: mean={means[index]: .5f} "
        f"(SE {mean_standard_errors[index]:.5f}), "
        f"sd={standard_deviations[index]:.5f}, "
        f"P(|S|>=2)={tails[index]:.5f}"
    )
```

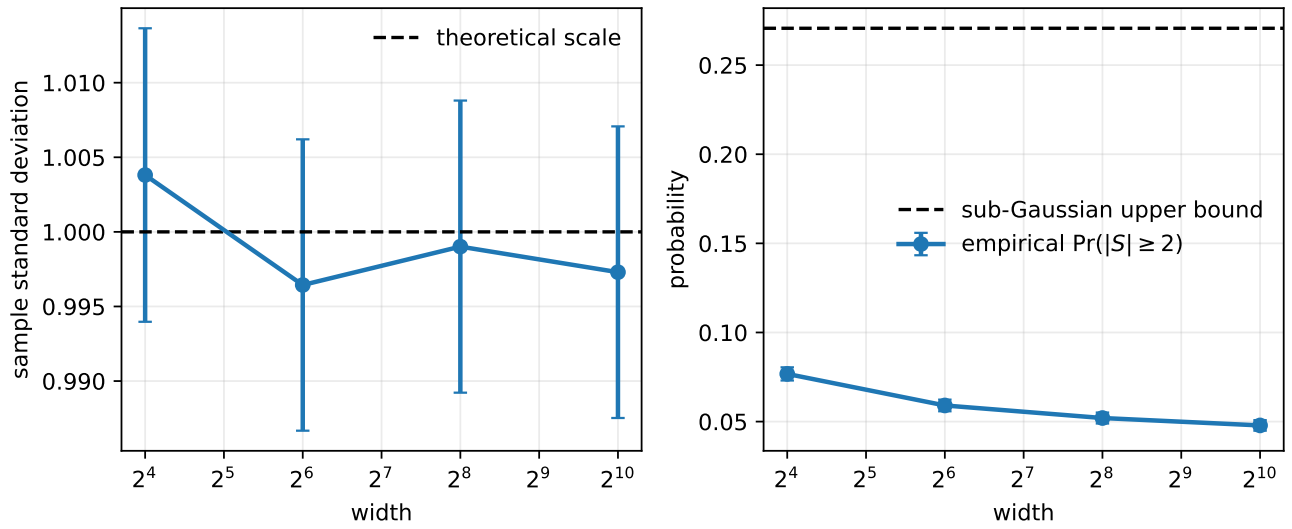


Figure 5.2.: Normalized Rademacher projections retain sample standard deviation near one from width 16 through 1024, as predicted by coefficient energy. The empirical event probability for absolute projection at least two varies with the finite lattice and approaches the Gaussian-like regime; the loose sub-Gaussian upper bound is shown only as a guarantee.

```
d=  16: mean= 0.01333 (SE 0.00710), sd=1.00381, P(|S|>=2)=0.07675
d=  64: mean=-0.00676 (SE 0.00705), sd=0.99644, P(|S|>=2)=0.05905
d= 256: mean= 0.00569 (SE 0.00706), sd=0.99901, P(|S|>=2)=0.05200
d=1024: mean= 0.00829 (SE 0.00705), sd=0.99730, P(|S|>=2)=0.04785
```

The sample standard deviations range from about 0.996 to 1.004. That is evidence for this seeded protocol, with approximate standard error about 0.005 for each standard deviation. The theorem explains why the scale should remain order one: $\|\mathbf{a}\|_2 = 1$ at every width.

The tail bound $2e^{-2} \approx 0.271$ is deliberately loose. It is a valid envelope, not a prediction of the exact tail probability.

5.7. Geometry meets the finite grid

Suppose a computation produces m centered scalar quantities with proxy scale σ. Equation Equation 5.5 gives a high-probability envelope for their maximum. C03 then asks whether that envelope fits the declared storage range and whether the sums forming each quantity use an adequate accumulator.

For $m = 10^9$, $\delta = 0.01$, and $\sigma = 1$, the model threshold 7.21 is far below FP16's largest finite value. That statement alone does **not** guarantee safe training:

- the proxy scale may drift;
- coordinates may cease to be sub-Gaussian;
- depth can multiply scales;
- intermediate reductions can overflow even when outputs do not;
- an implementation can change the accumulation contract.

The safe zone is a diagnostic hypothesis with a numerical consequence, not a branding phrase. C14 will derive when initialization keeps the coefficient energy controlled across depth. C06 has the nearer obligation: it will square a sub-Gaussian quantity and show why the same tail model no longer applies.

Check yourself

Let $m = 10^7$, $\delta = 10^{-3}$, and proxy scale $\sigma = 0.5$. Compute the simultaneous threshold from Equation 5.5. If all variables are identical copies of the same random variable rather than independent, does the theorem remain valid? Which earlier proof step changes?

5.8. Okay, so —

- **Inherited:** C03 supplied the range/accumulation contract, and C04 showed why controlling one direction is not enough when many modes coexist.
- **Changed:** Randomness is now described by a tail class and proxy scale that can survive simultaneous control.
- **Instrumented:** The harness computes simultaneous thresholds, seeded maxima, weighted-sign projections, empirical tails, and moment-growth profiles.
- **Established:** A quadratic MGF envelope yields Gaussian tail decay, independent weighted sums add proxy variances, a union bound charges only $\sqrt{\log m}$, and Lipschitz diagnostics of a standard Gaussian input concentrate around their own means.
- **Unresolved:** Which tail class replaces global sub-Gaussian control after a square or product changes the moment-generating function?

5.9. Sources and further reading

Chernoff's exponential method supplies the optimization view of tail bounds (Chernoff 1952). Hoeffding develops probability inequalities for sums of bounded independent variables (Hoeffding 1963). Vershynin gives the modern sub-Gaussian equivalence, Orlicz norm, and Gaussian Lipschitz concentration used here (Vershynin 2026).

For the general theory, use Vershynin (2026); this chapter's contribution is the precision-facing safe-zone diagnostic and its explicit failure boundary.

Reading order. Start with Vershynin (2026) for the reusable equivalences, then read Chernoff (1952) and Hoeffding (1963) for the original exponential-method and bounded-sum viewpoints.

5.10. Exercises

1. **(Pencil.) Optimize the exponent.** Repeat the proof of Theorem 5.1 when the MGF bound is known only for $|\lambda| \le \lambda_0$. Derive the range of t in which the quadratic optimizer remains admissible and state what happens outside it.
2. **(Pencil.) Dependence audit.** Give two examples with identical sub-Gaussian marginals: one with independent coordinates and one with all coordinates equal. Compare their exact maxima and the common union-bound guarantee. Explain why validity and sharpness are separate questions.
3. **(Code.) Maxima as an estimator.** Replace the single opening witness with repeated maxima at each feasible count. Predeclare trial counts, report the empirical median and a quantile interval, and keep the total draw budget fixed across design choices.
4. **(Code.) Coefficient geometry.** Compare three coefficient sequences: uniform $1/\sqrt{d}$, one-spike, and geometrically decaying, all normalized to the same ℓ_2 norm. Measure tails and explain what the MGF theorem predicts identically and what the exact distributions retain.
5. **(Audit.) The empirical certificate.** A report computes $\|X\|_p / \sqrt{p}$ for $p \le 8$ on 10,000 observations and declares the population sub-Gaussian. Predict a contamination distribution that is likely to evade the sample, then state a defensible empirical conclusion.
6. **(Audit.) Paper audit: Hoeffding's bounded-sum inequality.** Complete the **Mathematical object**, **Evidence culture and interface**, **Assumption stress test**, and **Transfer verdict** fields for the primary paper and extract the independence assumption, the bounded-range quantity, and the tail exponent for a weighted sum. Map each object to the MGF proxy language of this chapter and identify any constant or scope that cannot be transferred verbatim.

6. One Square, Two Regimes: Sub-Exponential Tails and Clipping Bias

SG · The sub-Gaussian safe zone Geometric primary · Dynamic quiet · Algorithmic supporting

Let $Z \sim \mathcal{N}(0,1)$. C05 puts Z safely inside a global quadratic MGF envelope. Now perform one of the most ordinary operations in computation:

$$X = Z^2 - 1.$$

The centering makes $\mathbb{E}X = 0$, and $\mathrm{Var}(X) = 2$. Neither fact saves the old tail contract. On the declared threshold grid, the exact upper tail at $t = 4$ is about 0.02535, already larger than the variance-matched quadratic candidate $e^{-t^2/4} \approx 0.01832$. At $t = 8$, the gap is no longer close.

! Prediction

Which diagnosis survives a stronger test?

1. The variance proxy merely needs a slightly larger constant.
2. Centering should eventually restore a global quadratic envelope.
3. Squaring changed the domain on which the MGF exists.

Choose before inspecting the generating function. Then predict which step of C05's Chernoff optimization will become constrained.

6.1. Inspect the domain, not only the variance

For this distribution the MGF is available in closed form. The opening study places its log beside a variance-matched quadratic candidate and then compares the exact upper-tail rate with quadratic and linear guides. A guide is not a probability bound; its job is to expose the shape that eventually becomes impossible.

1. Load the chapter-pinned tail instruments.
2. Evaluate the exact log-MGF below its finite boundary.
3. Evaluate exact upper tails on a declared threshold grid.
4. Locate the first failure of a variance-matched quadratic candidate.
5. Plot the MGF domain and tail-rate change.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import (
    bernstein_two_regime_rate,
    centered_chi_square_log_mgf,
    centered_chi_square_sum_trials,
    centered_chi_square_sum_upper_tail,
    centered_chi_square_upper_tail,
    centered_pareto_clip_bias,
)

# [2]
mgf_parameters = np.linspace(-0.45, 0.49, 300)
log_mgf = centered_chi_square_log_mgf(mgf_parameters)
log_mgf_values = np.array([log_mgf[float(value)] for value in mgf_parameters])

# [3]
tail_thresholds = np.geomspace(0.25, 32.0, 100)
exact_tail = centered_chi_square_upper_tail(tail_thresholds)
tail_values = np.array(
    [exact_tail[float(value)] for value in tail_thresholds]
)

# [4]
audit_thresholds = np.array([1.0, 2.0, 4.0, 8.0, 16.0, 32.0])
audit_tail = centered_chi_square_upper_tail(audit_thresholds)
quadratic_candidate = np.exp(-(audit_thresholds**2) / 4.0)
violations = [
    float(threshold)
    for threshold, candidate in zip(
        audit_thresholds,
        quadratic_candidate,
        strict=True,
    )
    if audit_tail[float(threshold)] > candidate
]
assert violations[0] == 4.0

# [5]
fig, axes = plt.subplots(1, 2, figsize=(8.2, 3.5))
axes[0].plot(mgf_parameters, log_mgf_values, linewidth=2.2, label="exact log-MGF")
axes[0].plot(
    mgf_parameters,
    mgf_parameters**2,
    linestyle="--",
    linewidth=1.8,
```

```
    label="variance-matched quadratic",
)
axes[0].axvline(0.5, color="black", linestyle=":", label="MGF boundary")
axes[0].set(xlabel=r"$\lambda$", ylabel="log-MGF", ylim=(-0.02, 1.7))

tail_rate = -np.log(tail_values)
axes[1].plot(tail_thresholds, tail_rate, linewidth=2.2, label="exact tail rate")
axes[1].plot(
    tail_thresholds,
    tail_thresholds**2 / 4.0,
    linestyle="--",
    linewidth=1.8,
    label="quadratic guide",
)
axes[1].plot(
    tail_thresholds,
    tail_thresholds / 2.0,
    linestyle=":",
    linewidth=1.8,
    label="linear guide",
)
axes[1].set(
    xlabel=r"threshold $t$",
    ylabel=r"$-\log\Pr(X\geq t)$",
    xscale="log",
    yscale="log",
)
for axis in axes:
    axis.grid(alpha=0.25)
    axis.legend(frameon=False)
plt.tight_layout()
plt.show()

print(f"harness={manifest['harness_ref']} wheel={manifest['wheel_sha256'][:12]}")
for threshold in (4.0, 8.0):
    candidate = np.exp(-(threshold**2) / 4.0)
    print(
        f"t={threshold:g}: exact tail={audit_tail[threshold]:.8g}, "
        f"variance-matched candidate={candidate:.8g}"
    )
print("exact MGF domain: lambda < 0.5")
```

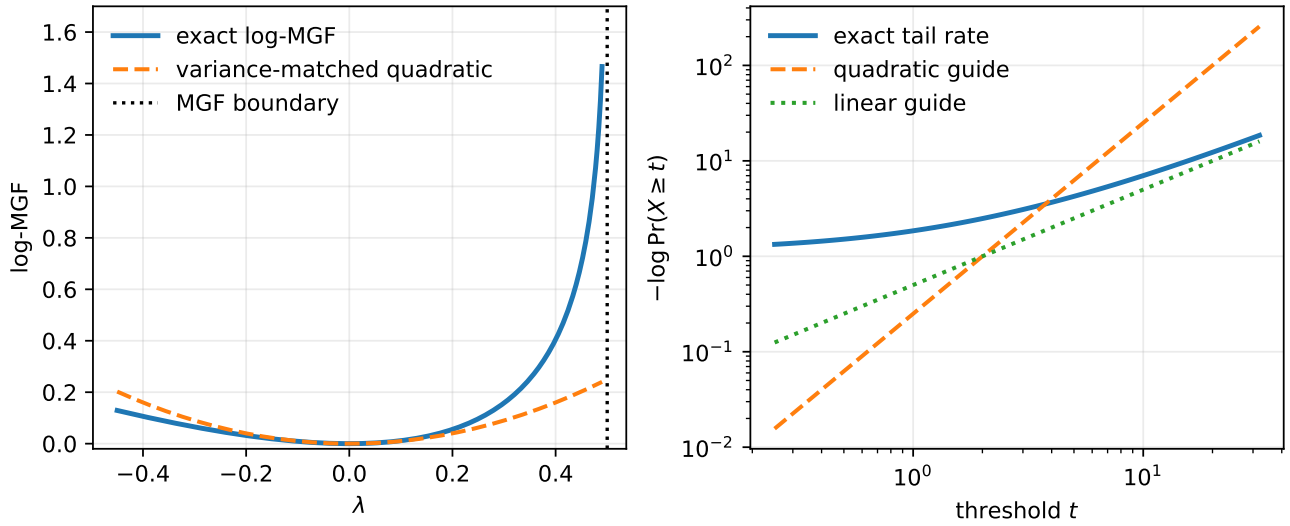


Figure 6.1.: A square breaks the global quadratic contract. At left, the exact log-MGF of the centered Gaussian square rises toward a singularity at one half, so no finite parabola can dominate it on all positive parameters. At right, the exact upper-tail rate bends toward linear growth while the variance-matched quadratic candidate becomes far too optimistic. The guides diagnose shape; only the exact MGF domain proves non-sub-Gaussianity.

```
harness=ch-06 wheel=160afa2bff70
t=4: exact tail=0.025347319, variance-matched candidate=0.018315639
t=8: exact tail=0.0026997961, variance-matched candidate=1.1253517e-07
exact MGF domain: lambda < 0.5
```

The decisive observation is not the crossing at $t = 4$. A different finite quadratic could postpone that crossing. The decisive observation is that

$$\mathbb{E}e^{\lambda(Z^2-1)} = \frac{e^{-\lambda}}{\sqrt{1-2\lambda}} \tag{6.1}$$

is infinite for every $\lambda \geq 1/2$. No finite quadratic envelope can dominate an infinite MGF. The square is not sub-Gaussian.

For $\lambda < 1/2$, Equation 6.1 follows by inserting the standard normal density:

$$\begin{aligned}\mathbb{E}e^{\lambda(Z^2-1)} &= \frac{e^{-\lambda}}{\sqrt{2\pi}} \int_{\mathbb{R}} \exp\left[-\frac{1-2\lambda}{2} z^2\right] dz \\ &= \frac{e^{-\lambda}}{\sqrt{1-2\lambda}}.\end{aligned}$$

At $\lambda \geq 1/2$, the integrand no longer decays at infinity. Centering removes the linear term near the origin; it cannot reopen the MGF after the quadratic transform has closed it.

6.2. The square keeps an exponential contract

The safe zone did not disappear. It became smaller. Define

$$\|X\|_{\psi_1} := \inf\left\{K > 0 : \mathbb{E}e^{|X|/K} \le 2\right\}. \tag{6.2}$$

A random variable with finite ψ_1 norm is called **sub-exponential**. The name describes an exponential tail class; it does not mean that the random variable itself is an exponential distribution.

Theorem 6.1 (Squares and products remain sub-exponential)**.** *If X is sub-Gaussian, then*

$$\left\|X^2\right\|_{\psi_1} = \|X\|_{\psi_2}^2. \tag{6.3}$$

Moreover,

$$\left\|X^2 - \mathbb{E}X^2\right\|_{\psi_1} \le 2\|X\|_{\psi_2}^2. \tag{6.4}$$

If X and Y are sub-Gaussian, without any independence requirement, then

$$\|XY\|_{\psi_1} \le \|X\|_{\psi_2}\|Y\|_{\psi_2}. \tag{6.5}$$

6.2.1. Proof

Let $a = \|X\|_{\psi_2}$. The defining condition $\mathbb{E}e^{X^2/a^2} \le 2$ is exactly the condition that $\left\|X^2\right\|_{\psi_1} \le a^2$. Taking the infimum in both definitions gives equality in Equation 6.3.

Jensen's inequality also gives

$$\exp\left(\frac{\mathbb{E}X^2}{a^2}\right) \le \mathbb{E}\exp\left(\frac{X^2}{a^2}\right) \le 2,$$

so $\mathbb{E}X^2 \le a^2 \log 2$. A constant c has ψ_1 norm $|c|/\log 2$. The triangle inequality for the Orlicz norm therefore yields Equation 6.4.

For the product, put $a = \|X\|_{\psi_2}$ and $b = \|Y\|_{\psi_2}$. The elementary inequality

$$\frac{|XY|}{ab} \le \frac{1}{2}\frac{X^2}{a^2} + \frac{1}{2}\frac{Y^2}{b^2}$$

and Cauchy–Schwarz give

$$\mathbb{E}e^{|XY|/(ab)} \le \left(\mathbb{E}e^{X^2/a^2}\right)^{1/2}\left(\mathbb{E}e^{Y^2/b^2}\right)^{1/2} \le 2.$$

This proves Equation 6.5. □

For a standard normal variable, the definitions even expose an exact scale:

$$\|Z\|_{\psi_2}^2 = \|Z^2\|_{\psi_1} = \frac{8}{3}.$$

Indeed, $\mathbb{E}e^{Z^2/K^2} = (1-2/K^2)^{-1/2} \le 2$ exactly when $K^2 \ge 8/3$. This number is a definition-dependent scale, not the variance of Z^2.

> ⚠ Named wrong answer: 'Squaring destroys concentration'
>
> Squaring destroys the **global quadratic MGF envelope**. It leaves an exponential-moment contract, which is strong enough for useful concentration but changes the deviation rate and the diagnostic scale.

6.3. A local parabola creates two regimes

A centered sub-exponential variable still looks quadratic near $\lambda = 0$. The difference from C05 is the quantifier:

Tail contract	Quadratic log-MGF bound	Chernoff optimizer
sub-Gaussian	every real λ	unconstrained
sub-exponential	only $\lvert\lambda\rvert \le c/K$	constrained
no positive MGF	no interval to the right of zero	unavailable

The sub-exponential equivalence theorem makes the middle row precise: for a centered X, there are universal constants $c_0, C_0 > 0$ such that

$$\mathbb{E}e^{\lambda X} \le \exp\left(C_0\lambda^2 \|X\|_{\psi_1}^2\right) \quad \text{when} \quad |\lambda| \le \frac{c_0}{\|X\|_{\psi_1}}. \tag{6.6}$$

As in C05, tail, moment, exponential-moment, and local-MGF characterizations are equivalent up to universal constants. The new fact is that the admissible interval is finite.

Theorem 6.2 (Bernstein's two-regime inequality)**.** *Let $X_1, \dots, X_n$ be independent, centered, sub-exponential random variables. Write*

$$V = \sum_{i=1}^n \|X_i\|_{\psi_1}^2, \qquad B = \max_i \|X_i\|_{\psi_1}.$$

There is a universal constant $c > 0$ such that, for every $t \ge 0$,

$$\Pr\left(\left|\sum_{i=1}^n X_i\right| \ge t\right) \le 2\exp\left[-c\min\left(\frac{t^2}{V}, \frac{t}{B}\right)\right]. \tag{6.7}$$

6.3.1. Proof

For $\lambda > 0$, exponential Markov and independence give

$$\Pr\left(\sum_i X_i \geq t\right) \leq \exp(-\lambda t) \prod_i \mathbb{E}e^{\lambda X_i}.$$

When $\lambda \leq c_0/B$, Equation 6.6 implies

$$\Pr\left(\sum_i X_i \geq t\right) \leq \exp(-\lambda t + C_0 \lambda^2 V)\,. \tag{6.8}$$

Without the constraint, the quadratic exponent is minimized at a constant multiple of t/V. Choose instead

$$\lambda = \min\left(\frac{t}{4C_0 V}, \frac{c_0}{2B}\right).$$

If the first argument is smaller, substitution into Equation 6.8 gives an exponent at most $-c_1 t^2/V$. If the second is smaller, the inequality defining that case lets the quadratic term absorb at most a fixed fraction of λt, leaving an exponent at most $-c_2 t/B$. Apply the same argument to $-X_i$, add the two tails, and take $c = \min(c_1, c_2)$. $\square$

The transition occurs near $t = V/B$. If all ψ_1 scales are comparable to K, then $V/B \asymp nK$. Averaging has not failed: increasing n extends the quadratic regime. What changes is the far tail, where the desired Chernoff parameter would lie outside the MGF's admissible interval.

6.4. Watch the optimizer hit its boundary

The next study uses

$$S_{16} = \sum_{i=1}^{16} (Z_i^2 - 1) = \chi^2_{16} - 16.$$

Because the degrees of freedom are even, its exact upper tail is a finite series. That exact curve is the control. Monte Carlo is shown only where its 500,000-trial denominator has resolution.

For this particular distribution,

$$\log \mathbb{E}e^{\lambda(Z_i^2 - 1)} \leq \frac{\lambda^2}{1 - 2\lambda}, \qquad 0 \leq \lambda < \frac{1}{2}. \tag{6.9}$$

Choosing $\lambda = t/[2(n + t)]$ for a sum of n terms gives the explicit one-sided bound

$$\Pr(S_n \geq t) \leq \exp\left[-\frac{t^2}{4(n + t)}\right]. \tag{6.10}$$

Its exponent is quadratic when $t \ll n$ and linear when $t \gg n$.

1. Generate centered-square sums in one declared random stream.
2. Count upper-tail exceedances at fixed thresholds.
3. Compute exact even-degree chi-square tails.
4. Compute distribution-specific Chernoff and Bernstein controls.
5. Plot probabilities, uncertainty, and the transition scale.

```
# [1]
terms, trials, seed = 16, 500_000, 6219
thresholds = np.array([2.0, 4.0, 6.0, 8.0, 12.0, 16.0, 20.0, 24.0, 32.0, 40.0])
samples = centered_chi_square_sum_trials(
    terms,
    trials=trials,
    seed=seed,
)

# [2]
exceedances = np.array(
    [np.count_nonzero(samples >= threshold) for threshold in thresholds]
)
empirical = exceedances / trials
standard_errors = np.sqrt(empirical * (1.0 - empirical) / trials)

# [3]
exact = centered_chi_square_sum_upper_tail(terms, thresholds)
exact_probabilities = np.array([exact[float(t)] for t in thresholds])
assert np.all(
    np.abs(empirical - exact_probabilities)
    <= 4.0 * standard_errors + 1.0 / trials
)

# [4]
exact_chernoff = (
    np.exp(-thresholds / 2.0)
    * (1.0 + thresholds / terms) ** (terms / 2.0)
)
explicit_bernstein = np.exp(
    -(thresholds**2) / (4.0 * (terms + thresholds))
)
piecewise = bernstein_two_regime_rate(
    thresholds,
    quadratic_scale=4.0 * terms,
    linear_scale=4.0,
)
piecewise_rates = np.array([piecewise[float(t)] for t in thresholds])
assert np.all(exact_probabilities <= exact_chernoff)
assert np.all(exact_probabilities <= explicit_bernstein)

# [5]
fig, axes = plt.subplots(1, 2, figsize=(8.2, 3.5))
```

```
positive_lower = np.maximum(empirical - 1.96 * standard_errors, 0.5 / trials)
positive_upper = empirical + 1.96 * standard_errors
axes[0].errorbar(
    thresholds,
    empirical,
    yerr=np.vstack([empirical - positive_lower, positive_upper - empirical]),
    fmt="o",
    capsize=3,
    label="Monte Carlo estimate",
)
axes[0].plot(thresholds, exact_probabilities, linewidth=2.2, label="exact tail")
axes[0].plot(thresholds, exact_chernoff, linestyle="--", label="exact-MGF Chernoff")
axes[0].plot(thresholds, explicit_bernstein, linestyle=":", label="explicit Bernstein")
axes[0].axhline(1.0 / trials, color="gray", linewidth=1, label="one-trial resolution")
axes[0].set(xlabel=r"threshold $t$", ylabel=r"$\Pr(S_{16}\geq t)$", yscale="log")

axes[1].plot(
    thresholds,
    -np.log(exact_probabilities),
    linewidth=2.2,
    label="exact tail rate",
)
axes[1].plot(thresholds, piecewise_rates, linestyle="--", label="two-regime guide")
axes[1].axvline(terms, color="black", linestyle=":", label="declared scale n")
axes[1].set(xlabel=r"threshold $t$", ylabel="negative log tail")
for axis in axes:
    axis.grid(alpha=0.25)
    axis.legend(frameon=False)
plt.tight_layout()
plt.show()

for threshold in (16.0, 32.0, 40.0):
    index = int(np.flatnonzero(thresholds == threshold)[0])
    print(
        f"t={threshold:g}: count={exceedances[index]}/{trials}, "
        f"estimate={empirical[index]:.8g}, "
        f"SE={standard_errors[index]:.3g}, "
        f"exact={exact_probabilities[index]:.8g}"
    )
```

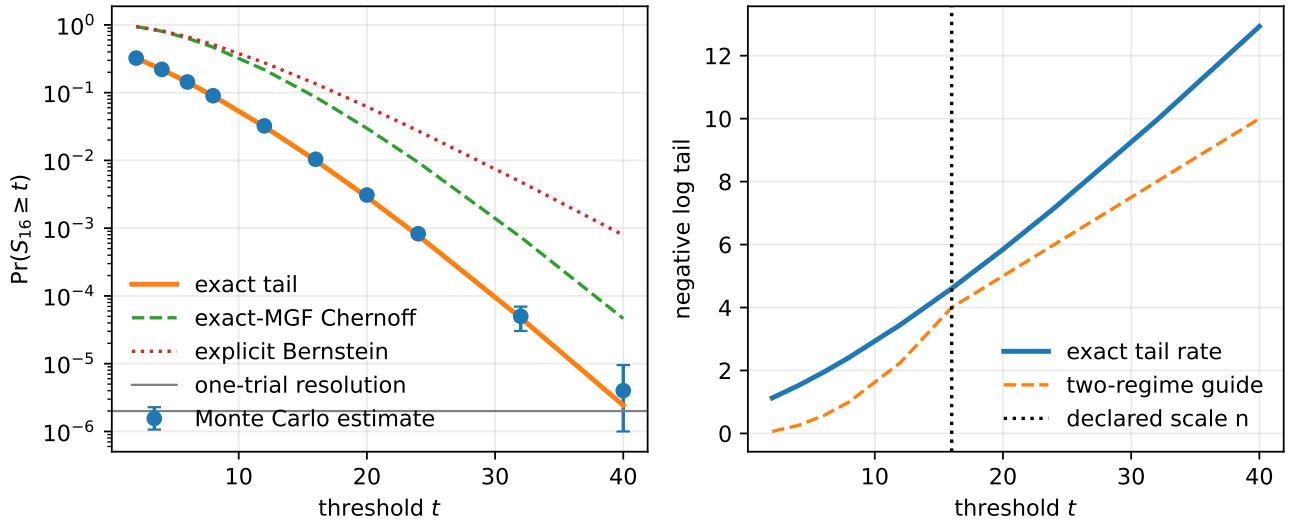


Figure 6.2.: A centered chi-square sum retains two distinct deviation scales. At left, 500,000 seeded trials track the exact upper tail until only a few exceedances remain; exact probabilities continue beyond simulation resolution. Both analytic controls remain upper bounds. At right, the exact negative log tail bends from a quadratic toward a linear rate around the declared scale n equals 16. The experiment validates this distribution-specific witness, not every sub-exponential sum.

```
t=16: count=5196/500000, estimate=0.010392, SE=0.000143, exact=0.009999781
t=32: count=25/500000, estimate=5e-05, SE=1e-05, exact=4.749992e-05
t=40: count=2/500000, estimate=4e-06, SE=2.83e-06, exact=2.4335719e-06
```

At $t = 40$, the simulation sees only two exceedances. The exact tail is about 2.43×10^{-6}, while the estimate is 4.00×10^{-6} with standard error about 2.83×10^{-6}. Reporting only the empirical curve would make the most interesting region the least trustworthy one. The exact control prevents a plotting artifact from becoming a tail claim.

6.5. When no positive MGF exists

Sub-exponential is not a synonym for every distribution heavier than a Gaussian. It is still an exponential-moment class. A Pareto variable with power-law survival has no finite positive MGF, so neither C05's global optimization nor Bernstein's local optimization begins.

One possible intervention maps a scalar x to a bounded interval:

$$T_C(x) = \operatorname{sign}(x) \min(|x|, C). \tag{6.11}$$

This operation is conventionally called **clipping**. It creates a new random variable. The distinction between new tail behavior and the old target is the whole contract.

Theorem 6.3 (Concentration and bias after clipping)**.** *Let X be integrable and $C > 0$. Then*

$$\mathbb{E}\exp[\lambda\,(T_C(X) - \mathbb{E}T_C(X))] \le \exp\left(\frac{\lambda^2 C^2}{2}\right) \tag{6.12}$$

for every real λ. Thus the centered clipped variable is sub-Gaussian with proxy variance at most C^2. Its bias for the original mean is

$$\mathrm{Bias}_C = \mathbb{E}T_C(X) - \mathbb{E}X. \tag{6.13}$$

6.5.1. Proof

$T_C(X)$ lies in an interval of length $2C$. Hoeffding's lemma applied after centering gives Equation 6.12. Equation Equation 6.13 is the difference between the expectation of the transformed estimator and the original target. Boundedness imposes no reason for that difference to vanish. □

The tradeoff is exact for a simple control. Let Y have Pareto shape $3/2$ and scale 1, so $\mathbb{E}Y = 3$ but $\mathrm{Var}(Y) = \infty$, and put $X = Y - 3$. For every $C \ge 2$,

$$\mathbb{E}T_C(X) = -\frac{2}{\sqrt{C+3}}. \tag{6.14}$$

Larger C reduces bias while widening the bounded variable's sub-Gaussian proxy. There is no distribution-free choice that makes both costs vanish.

1. Declare a centered Pareto control with infinite variance.
2. Evaluate its exact clipped-mean bias over fixed thresholds.
3. Pair each threshold with its bounded-variable proxy scale.
4. Plot the opposing concentration and target costs.

```
# [1]
pareto_shape, pareto_scale = 1.5, 1.0
clip_thresholds = np.array([2.0, 4.0, 8.0, 16.0, 32.0, 64.0, 128.0])

# [2]
bias = centered_pareto_clip_bias(
    pareto_shape,
    clip_thresholds,
    scale=pareto_scale,
)
bias_values = np.array([bias[float(value)] for value in clip_thresholds])
assert np.all(bias_values < 0)

# [3]
proxy_scales = clip_thresholds

# [4]
fig, axis = plt.subplots(figsize=(6.8, 3.6))
```

```
axis.loglog(
    clip_thresholds,
    np.abs(bias_values),
    "o-",
    linewidth=2.2,
    label="absolute target bias",
)
axis.loglog(
    clip_thresholds,
    proxy_scales,
    "s--",
    linewidth=2.0,
    label="proxy-scale upper bound",
)
axis.set(xlabel=r"clipping threshold $C$", ylabel="magnitude")
axis.grid(alpha=0.25, which="both")
axis.legend(frameon=False)
plt.tight_layout()
plt.show()

for threshold in (2.0, 32.0, 128.0):
    print(
        f"C={threshold:g}: exact bias={bias[threshold]:.6f}, "
        f"proxy-scale bound={threshold:g}"
    )
```

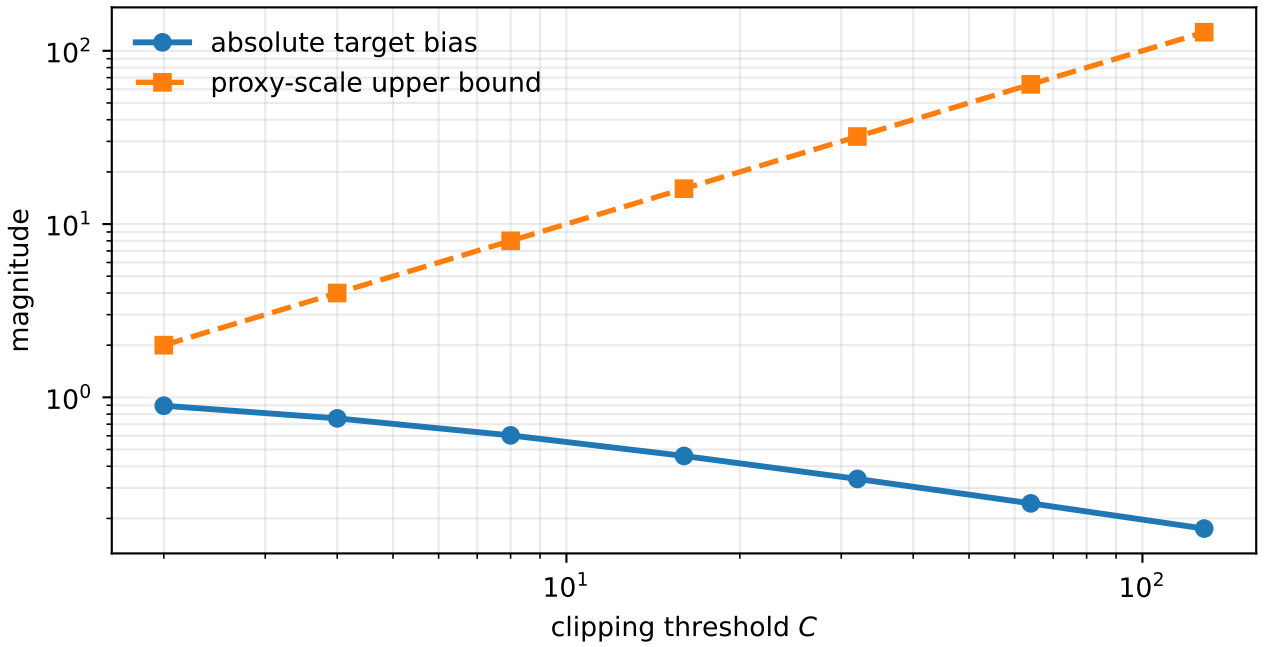


Figure 6.3.: Bounding a heavy-tailed variable creates opposing costs. Increasing the clipping threshold reduces the exact mean bias of the centered Pareto control, but the worst-case sub-Gaussian proxy scale of the centered clipped variable grows with that threshold. The curves establish a model tradeoff; they do not prescribe a training threshold.

```
C=2: exact bias=-0.894427, proxy-scale bound=2
C=32: exact bias=-0.338062, proxy-scale bound=32
C=128: exact bias=-0.174741, proxy-scale bound=128
```

i Field note: the guardrail must itself be computable

For vector norm clipping, the norm is usually formed from a reduction of squares. If that reduction overflows before the threshold comparison, clipping never gets a finite input. C03's precision ledger still applies: use a declared wider accumulator or a scaled sum-of-squares kernel. A post-overflow guardrail is not a guardrail.

Field	Clipping audit
Target	the original population mean or gradient
Estimator	the mean of bounded transformed observations
Reduction	over the declared sample or mini-batch
Validity	concentrated for the transformed target; useful for the original target only with a bias argument
Boundary	threshold choice, direction distortion, state dependence, and pre-clip numerical overflow remain

Empirical reports of heavy-tailed stochastic gradients are research evidence, not a universal law. Tail-index estimates depend on the model, data, mini-batch construction, training time, and estimator

(Simsekli et al. 2019). Convergence guarantees for clipped stochastic updates likewise depend on the threshold and noise assumptions (Koloskova et al. 2023). C13 will turn those dependencies into a dynamic audit.

Check yourself

Suppose independent centered variables have $\|X_i\|_{\psi_1} \le 3$ for $i = 1, \dots, 100$. Up to universal constants, identify the transition scale V/B. Which Bernstein exponent controls at $t = 30$ and at $t = 3000$? Then state why finite variance alone would not justify either answer.

6.6. Okay, so —

- **Inherited:** C05 supplied a global quadratic MGF envelope, a ψ_2 scale, and the Chernoff optimizer.
- **Changed:** one square restricts the MGF domain and replaces one global quadratic tail regime with Bernstein's quadratic-to-linear pair.
- **Instrumented:** the harness now evaluates exact centered-chi-square MGFs and tails, seeded sum tails, Bernstein rates, and exact clipping bias.
- **Established:** squares and products of sub-Gaussian variables are sub-exponential; independent centered sub-exponential sums obey a two-regime bound; bounded transforms concentrate around their own means.
- **Unresolved:** How can a finite tail budget control an uncountable supremum without sampling directions and hoping the maximum was found?

6.7. Sources and further reading

Vershynin develops sub-exponential equivalences, the square/product lemmas, and Bernstein's inequality (Vershynin 2026). Hoeffding's bounded-variable lemma supplies the clipped-variable guarantee (Hoeffding 1963). Simsekli, Sagun, and Gurbuzbalaban provide a primary empirical tail-index study (Simsekli et al. 2019), while Koloskova, Hendrikx, and Stich analyze the stochastic bias and threshold dependence of gradient clipping (Koloskova et al. 2023).

For the general theory, use Vershynin (2026); this chapter's contribution is the squared-quantity witness and the audit that clipping changes the target.

Reading order. Start with Vershynin (2026) for the square, product, and Bernstein machinery; use Simsekli et al. (2019) and Koloskova et al. (2023) to audit the heavy-tail evidence and the target changed by clipping.

6.8. Exercises

1. **(Pencil.) The exact Orlicz scale.** Derive $\|Z\|_{\psi_2}^2 = 8/3$ directly from the Gaussian integral. Then use the definitions, not a tail equivalence theorem, to obtain $\left\|Z^2\right\|_{\psi_1} = 8/3$.

2. **(Pencil.) Products without independence.** Rework the proof of Equation 6.5 and mark the exact line where Cauchy–Schwarz replaces independence. Construct dependent X, Y for which the conclusion remains useful.

3. **(Code.) Resolve the transition.** Repeat the centered-square sum study for $n \in \{4, 16, 64\}$ under one predeclared total draw budget. Plot thresholds as multiples of n and report exceedances, denominators, and binomial intervals. Separate failure to resolve a tail from evidence that its probability is zero.

4. **(Code.) A clipping target audit.** Compare symmetric clipping on a symmetric Student distribution and the skewed Pareto control. Use matched thresholds and sample counts. Predict the sign of the bias before running, then report the transformed target and the original target separately.

5. **(Audit.) The finite-sample certificate.** A report fits a straight line to a log-survival plot over two decades and declares the population sub-exponential. Name two heavier-tailed alternatives that can mimic that window, state what the plot can falsify, and design a held-out threshold audit.

6. **(Audit.) Paper audit: a tail-index claim.** Complete the **Mathematical object**, **Dynamic regime**, **Estimator and comparison contract**, **Assumption stress test**, and **Transfer verdict** fields for Simsekli, Sagun, and Gurbuzbalaban (Simsekli et al. 2019). Extract the random variable whose tail is modeled, the estimator used for its index, the sampling unit, and the architecture/data/training regimes tested. Identify which statements are empirical observations, which invoke an α-stable model, and which would be invalid if transferred as a universal law to a new training run.

7. The Sphere Has a Finite Price: Epsilon-Nets and Uniform Operator Control

SG · The sub-Gaussian safe zone Geometric primary · Dynamic quiet · Algorithmic supporting

Consider a linear map $\mathbf{A} : \mathbb{R}^2 \to \mathbb{R}^2$ with singular values 6 and 1. Rotate its most stretched direction away from both coordinate axes. The largest stretch seen by those two predeclared probes is about 5.606. The true maximum is 6.

Nothing random or numerically unstable happened. The probes answered the questions they were asked. The direction that maximizes $\|\mathbf{Ax}\|_2$ is selected *after* $\mathbf{A}$ is known, whereas the coordinate probes were selected before it.

> **!** Prediction
>
> Suppose a probability bound controls each one of one million predeclared directions. Does it control every direction on the unit sphere? If not, name the additional deterministic fact that would let the finite calculation reach the continuum.

7.1. A sampled maximum is not a supremum

C05 paid a logarithmic price for finitely many random variables. C06 showed that the price depends on the tail class. Neither chapter authorized a union bound over an uncountable sphere.

The opening study makes the missing step inspectable. An **epsilon-net** is a finite set of anchors close enough to every point in the target set. The largest anchor value can still underestimate the true maximum. The covering radius supplies a deterministic correction that turns the underestimate into an upper certificate.

1. Load the chapter-pinned covering instruments.
2. Construct a rotated map with known singular values.
3. Evaluate its stretch over the entire circle and on verified epsilon-nets.
4. Compute the sampled maximum and interpolation certificate separately.
5. Plot the hidden direction, anchors, and certificate gap.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import (
    circle_covering_radius,
    circle_epsilon_net,
```

```
    operator_net_certificate,
    operator_net_estimate,
    sphere_cover_log_upper,
    uniform_subgaussian_threshold,
)

# [2]
angle = 0.37
rotation = np.array(
    [[np.cos(angle), -np.sin(angle)], [np.sin(angle), np.cos(angle)]]
)
matrix = rotation @ np.diag([6.0, 1.0]) @ rotation.T
exact_norm = np.linalg.norm(matrix, ord=2)

# [3]
angles = np.linspace(0.0, 2.0 * np.pi, 721)
directions = np.column_stack((np.cos(angles), np.sin(angles)))
stretches = np.linalg.norm(directions @ matrix.T, axis=1)
display_net = circle_epsilon_net(0.25)
display_angles = np.mod(
    np.arctan2(display_net[:, 1], display_net[:, 0]),
    2.0 * np.pi,
)
display_stretches = np.linalg.norm(display_net @ matrix.T, axis=1)

# [4]
epsilons = np.array([0.5, 0.25, 0.125, 0.0625])
sampled, certified = [], []
for epsilon in epsilons:
    net = circle_epsilon_net(float(epsilon))
    maximum = operator_net_estimate(matrix, net)
    sampled.append(maximum)
    certified.append(operator_net_certificate(maximum, float(epsilon)))
sampled = np.asarray(sampled)
certified = np.asarray(certified)
assert np.all(sampled <= exact_norm + 1e-12)
assert np.all(certified >= exact_norm)

# [5]
figure = plt.figure(figsize=(8.2, 3.6))
polar = figure.add_subplot(1, 2, 1, projection="polar")
polar.plot(angles, stretches, linewidth=2.2, label="all directions")
polar.scatter(
    display_angles,
    display_stretches,
    s=24,
    color="#E57200",
    label=r"$\epsilon=0.25$ anchors",
    zorder=3,
```

```
)
coordinate_angles = np.array([0.0, np.pi / 2.0])
coordinate_stretches = np.linalg.norm(
    np.column_stack((np.cos(coordinate_angles), np.sin(coordinate_angles)))
    @ matrix.T,
    axis=1,
)
polar.scatter(
    coordinate_angles,
    coordinate_stretches,
    marker="s",
    s=38,
    color="black",
    label="coordinate probes",
    zorder=4,
)
polar.set_yticklabels([])
polar.legend(frameon=False, loc="lower left", bbox_to_anchor=(-0.25, -0.27))

axis = figure.add_subplot(1, 2, 2)
axis.plot(epsilons, sampled, "o-", linewidth=2.0, label="sampled maximum")
axis.plot(epsilons, certified, "s--", linewidth=2.0, label="certificate")
axis.axhline(exact_norm, color="black", linestyle=":", label="exact norm")
axis.invert_xaxis()
axis.set(xlabel=r"declared radius $\epsilon$", ylabel="stretch")
axis.grid(alpha=0.25)
axis.legend(frameon=False)
plt.tight_layout()
plt.show()

coordinate_maximum = operator_net_estimate(matrix, np.eye(2))
print(f"harness={manifest['harness_ref']} wheel={manifest['wheel_sha256'][:12]}")
print(
    f"coordinate probes={coordinate_maximum:.6f}; "
    f"exact operator norm={exact_norm:.6f}"
)
for epsilon, maximum, certificate in zip(
    epsilons,
    sampled,
    certified,
    strict=True,
):
    print(
        f"epsilon={epsilon:g}: sampled={maximum:.6f}, "
        f"certificate={certificate:.6f}"
    )
```

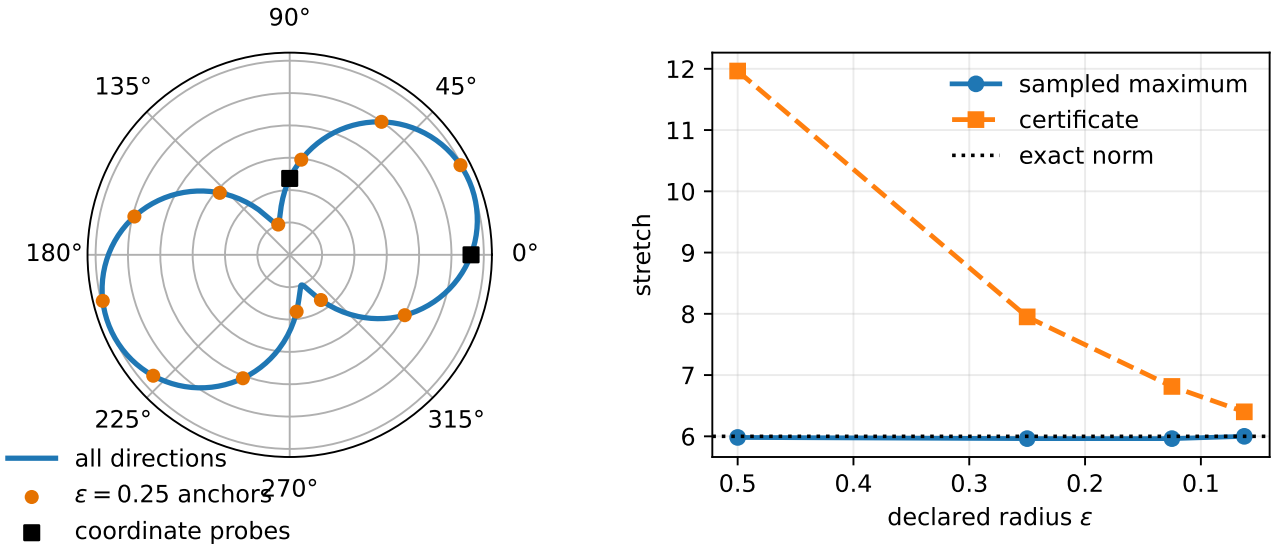


Figure 7.1.: A finite cover converts missed extrema into a certificate. At left, the stretch of a rotated map varies over the unit circle; the coordinate probes miss the peak, and a verified one-quarter net samples it closely. At right, finer nets close the sampled gap while the deterministic one-over-one-minus-epsilon certificate remains above the exact operator norm. The sampled maximum and the certificate are different quantities.

```
harness=ch-07 wheel=50543994a041
coordinate probes=5.605640; exact operator norm=6.000000
epsilon=0.5: sampled=5.981900, certificate=11.963800
epsilon=0.25: sampled=5.962588, certificate=7.950117
epsilon=0.125: sampled=5.962588, certificate=6.814386
epsilon=0.0625: sampled=6.000000, certificate=6.400000
```

The net with declared radius 0.25 uses 13 points. Its sampled maximum is about 5.963, still below 6. The theorem below turns that shortfall into a valid statement: $5.963/(1-0.25) \approx 7.950$ is conservative, but it cannot miss the true norm.

This is the book's recurring distinction in geometric form:

Quantity	Meaning
sampled maximum	largest value at the anchors actually evaluated
covering radius	largest permitted distance from any target point to an anchor
interpolation rule	deterministic control of change between nearby points
certificate	sampled maximum plus the declared interpolation cost
true supremum	control value for a small study; usually unavailable at scale

7.2. Make the continuum finite

Let (K, d) be a metric space. A finite subset $\mathcal{N}_\epsilon \subseteq K$ is an **epsilon-net** of K when every $x \in K$ has an anchor $y \in \mathcal{N}_\epsilon$ satisfying $d(x, y) \leq \epsilon$. The **covering number**

$$N(K, d, \epsilon) := \min\{|\mathcal{N}| : \mathcal{N} \text{ is an epsilon-net of } K\} \tag{7.1}$$

measures the smallest such price. Its logarithm is called the **metric entropy**.

The word *finite* matters. In $\mathbb{R}^d$, a closed and bounded set is compact, so every open cover has a finite subcover. Outside finite-dimensional Euclidean space, boundedness alone does not establish compactness.

Theorem 7.1 (Volumetric upper bound)**.** *For every $\epsilon > 0$, the Euclidean unit ball B_2^d has an epsilon-net with*

$$N(B_2^d, \|\cdot\|_2, \epsilon) \leq \left(1 + \frac{2}{\epsilon}\right)^d. \tag{7.2}$$

The same upper bound holds for the unit sphere S^{d-1}.

7.2.1. Proof

Choose a maximal epsilon-separated subset $\{x_1, \ldots, x_M\}$ of the ball: distinct centers are more than epsilon apart, and no new point can be added while preserving that property. The closed balls of radius $\epsilon/2$ centered at the x_i have disjoint interiors. Every one lies inside $(1 + \epsilon/2)B_2^d$. Comparing volumes gives

$$M \ \mathrm{Vol}((\epsilon/2)B_2^d) \leq \mathrm{Vol}((1 + \epsilon/2)B_2^d).$$

Euclidean volume scales as the dth power of the radius, hence

$$M \leq \left(\frac{1 + \epsilon/2}{\epsilon/2}\right)^d = \left(1 + \frac{2}{\epsilon}\right)^d.$$

Maximal separation also implies covering: if some point were farther than epsilon from every center, it could be added. The identical packing argument with centers restricted to the sphere proves the sphere upper bound. □

The count is exponential in d, but probability calculations pay its logarithm:

$$\log N(S^{d-1}, \|\cdot\|_2, \epsilon) \leq d \log\left(1 + \frac{2}{\epsilon}\right). \tag{7.3}$$

This is an upper certificate, not an exact count. For diagnostics, the important question is whether the available tail exponent can pay Equation 7.3.

7.3. Fill the gaps deterministically

A cover controls distances. It does not, by itself, control the function being evaluated. The opening map works because linearity and the operator norm bound the change between nearby directions.

Theorem 7.2 (Operator norm from an epsilon-net)**.** *Let $\mathcal{N}_\epsilon$ be an epsilon-net of S^{d-1} with $0 < \epsilon < 1$. For every real matrix $\mathbf{A}$ with d input columns,*

$$\|\mathbf{A}\|_{\mathrm{op}} \le \frac{1}{1-\epsilon} \max_{\mathbf{y}\in\mathcal{N}_\epsilon} \|\mathbf{A}\mathbf{y}\|_2 . \tag{7.4}$$

7.3.1. Proof

Let $\mathbf{x} \in S^{d-1}$ attain the operator norm and choose an anchor $\mathbf{y}$ with $\|\mathbf{x}-\mathbf{y}\|_2 \le \epsilon$. Then

$$\begin{aligned} \|\mathbf{A}\|_{\mathrm{op}} = \|\mathbf{A}\mathbf{x}\|_2 &\le \|\mathbf{A}\mathbf{y}\|_2 + \|\mathbf{A}(\mathbf{x}-\mathbf{y})\|_2 \\ &\le \max_{\mathbf{z}\in\mathcal{N}_\epsilon} \|\mathbf{A}\mathbf{z}\|_2 + \epsilon \|\mathbf{A}\|_{\mathrm{op}} . \end{aligned}$$

Move the last term to the left and divide by $1-\epsilon$. □

> ⚠ Named wrong answer: 'Enough random probes make a net'
>
> A cloud of random directions may have good coverage, but sample count alone does not certify its covering radius. The diagnostic needs either a verified construction or a separate probability statement for the random cover. A large probe count is not a geometric certificate.

7.4. Tail budget plus interpolation

The same pattern applies beyond linear maps. Fix a random function $f_\omega : K \to \mathbb{R}$. The randomness is in ω; the index $x \in K$ identifies the query.

Theorem 7.3 (Finite-cover uniformization)**.** *Let $\mathcal{N}_\epsilon$ be a finite epsilon-net of (K, d). Suppose every sample path is L-Lipschitz:*

$$|f_\omega(x) - f_\omega(y)| \le L\, d(x,y) \quad \text{for all } x, y \in K,$$

and every fixed anchor satisfies the common upper-tail bound

$$\Pr(f_\omega(y) > a + t) \le 2\exp\left(-\frac{t^2}{2\sigma^2}\right).$$

Then, with probability at least $1-\delta$,

$$\sup_{x\in K} f_\omega(x) \le a + \sigma\sqrt{2\left[\log\left(\frac{2}{\delta}\right) + \log|\mathcal{N}_\epsilon|\right]} + L\epsilon. \tag{7.5}$$

7.4.1. Proof

Apply the fixed-anchor bound and a union bound:

$$\Pr\left(\max_{y\in\mathcal{N}_\epsilon} f_\omega(y) > a + t\right) \le 2|\mathcal{N}_\epsilon| \exp\left(-\frac{t^2}{2\sigma^2}\right).$$

Choose t so the right side equals δ. On the complementary event, each $x \in K$ has an anchor y within epsilon, and Lipschitz continuity gives

$$f_\omega(x) \le f_\omega(y) + L\epsilon \le a + t + L\epsilon.$$

Take the supremum over x. □

There are three independent obligations:

1. prove a tail bound for each fixed anchor;
2. price a cover in the same metric used by the function;
3. justify the interpolation constant.

If L is random, the theorem does not permit silently replacing it with its mean. One must first control an event on which L is bounded, then include that event's failure probability in the budget.

7.5. Which tails can afford the sphere?

Set $\epsilon = 0.25$ and $\delta = 0.01$. The volumetric upper bound has

$$\log|\mathcal{N}_\epsilon| \le d \log 9.$$

Under a unit-scale sub-Gaussian anchor tail, the union threshold is

$$t_{\mathrm{SG}} = \sqrt{2\left[\log(2/\delta) + d\log 9\right]},$$

which grows like $\sqrt{d}$. Under only the second-moment envelope $\Pr(|X| \ge t) \le 1/t^2$, the same union calculation requires $t \ge \sqrt{|\mathcal{N}_\epsilon|/\delta}$, exponential in d.

The latter calculation is valid. It is simply too weak to give a useful high-dimensional diagnostic.

1. Fix the covering radius and failure probability.
2. Compute the log covering upper bound without materializing the cover.
3. Solve the sub-Gaussian union budget for its threshold.
4. Solve the second-moment union budget on a log-ten scale.
5. Compare how both thresholds grow with dimension.

```
# [1]
dimensions = np.array([8, 32, 128, 512])
epsilon, delta = 0.25, 0.01

# [2]
log_covers = np.array(
    [sphere_cover_log_upper(int(dimension), epsilon) for dimension in dimensions]
)

# [3]
subgaussian_thresholds = np.array(
    [
        uniform_subgaussian_threshold(log_cover, delta)
        for log_cover in log_covers
    ]
)

# [4]
chebyshev_log10_thresholds = (
    0.5 * (log_covers + np.log(1.0 / delta)) / np.log(10.0)
)

# [5]
fig, axes = plt.subplots(1, 2, figsize=(8.2, 3.5))
axes[0].plot(dimensions, subgaussian_thresholds, "o-", linewidth=2.2)
axes[0].set(
    xlabel="dimension",
    ylabel="sub-Gaussian anchor threshold",
    xscale="log",
)
axes[1].plot(dimensions, chebyshev_log10_thresholds, "s-", linewidth=2.2)
axes[1].set(
    xlabel="dimension",
    ylabel=r"$\log_{10}$ second-moment threshold",
    xscale="log",
)
for axis in axes:
    axis.grid(alpha=0.25)
plt.tight_layout()
plt.show()

for dimension, log_cover, sg, cheb_log in zip(
    dimensions,
    log_covers,
    subgaussian_thresholds,
    chebyshev_log10_thresholds,
    strict=True,
):
    print(
```

```
        f"d={dimension}: log-cover<={log_cover:.3f}; "
        f"sub-Gaussian={sg:.3f}"
    )
    print(
        f"  log10 second-moment threshold={cheb_log:.3f}"
    )
```

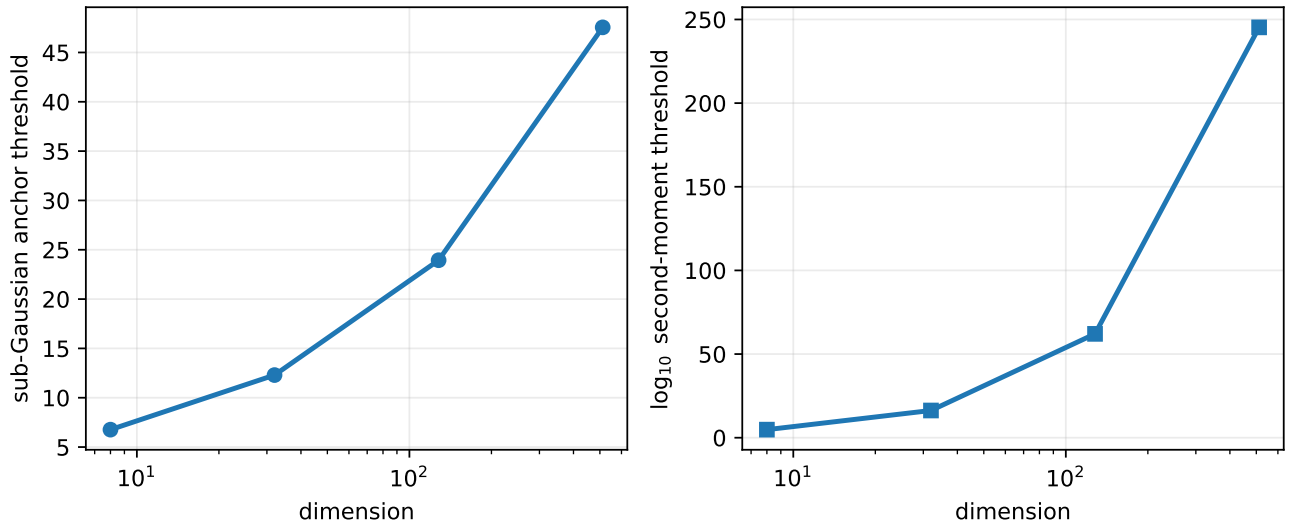


Figure 7.2.: Tail class determines whether an exponential cover is affordable. At left, the sub-Gaussian union threshold grows on a square-root-dimensional scale. At right, the base-ten logarithm of the second-moment threshold grows linearly with dimension; by dimension 512 the threshold is above ten to the 245. Both are valid union calculations under their stated envelopes, but only one is diagnostically useful.

```
d=8: log-cover<=17.578; sub-Gaussian=6.764
  log10 second-moment threshold=4.817
d=32: log-cover<=70.311; sub-Gaussian=12.297
  log10 second-moment threshold=16.268
d=128: log-cover<=281.245; sub-Gaussian=23.939
  log10 second-moment threshold=62.072
d=512: log-cover<=1124.979; sub-Gaussian=47.545
  log10 second-moment threshold=245.286
```

The comparison also explains why C06 had to distinguish tail regimes before this chapter. A linear Bernstein exponent can still pay a finite cover, but its threshold scales like the log covering number rather than its square root. The geometry has not changed; the available probabilistic currency has.

Field note: discretization is not quantization

A numerical format supplies representable scalar levels, rounding rules, range limits, and arithmetic semantics. An epsilon-net supplies metric coverage of a declared set. Counting 2^b scalar levels does not establish coverage of a d-dimensional parameter sphere, and the nearest representable vector need not lie on that sphere. Use C03's format contract for arithmetic and this chapter's covering

contract for geometry.

The same warning applies to data. A concentration theorem for a declared random model does not prove that a finite training set represents every population the system may encounter. The index set, metric, fixed-point tail, and interpolation rule must all be named.

Check yourself

An epsilon-net has N anchors. Each fixed anchor obeys $\Pr(f(y) > a + t) \leq 2e^{-t^2/(2\sigma^2)}$, and every sample path is L-Lipschitz. What is the smallest union-bound threshold that gives failure probability delta? Which term changes if the cover is refined from epsilon to epsilon over two, and which term does not?

7.6. Okay, so —

- **Inherited:** C05 supplied fixed-point sub-Gaussian control; C06 turned a tail exponent into a finite statistical budget.
- **Changed:** the target is now a continuum, so probability at anchors is insufficient without geometric coverage and deterministic interpolation.
- **Instrumented:** the harness constructs verified circle nets, measures their actual covering radii, certifies operator norms, and prices sphere covers without materializing them.
- **Established:** Euclidean covers have exponential size but linear log cost; an epsilon-net and a Lipschitz rule convert anchor control into a uniform bound.
- **Unresolved:** How can a finite-cover certificate control extremal singular directions selected only after a random matrix is observed?

7.7. Sources and further reading

Vershynin develops covering and packing numbers, the volumetric upper bound, and epsilon-net arguments for random operators (Vershynin 2026). Baraniuk, Davenport, DeVore, and Wakin combine fixed-vector concentration with finite-dimensional covering numbers in a restricted-isometry proof (Baraniuk et al. 2008).

For the general theory, use Vershynin (2026); this chapter's contribution is the sampled-maximum versus uniform-certificate diagnostic.

Reading order. Start with Vershynin (2026) for covering numbers and the interpolation lemma, then read Baraniuk et al. (2008) as a complete worked example of a finite tail budget paying for a continuous guarantee.

7.8. Exercises

1. **(Pencil.) A circle cover.** Derive the exact chord covering radius of k equally spaced points on S^1. Find the smallest k that certifies radius at most 0.1, and compare it with the volumetric upper bound.

2. **(Pencil.) Symmetric quadratic forms.** Let $\mathbf{B}$ be symmetric and $\mathcal{N}_\epsilon$ an epsilon-net of the unit sphere with $\epsilon < 1/2$. Prove
$$\|\mathbf{B}\|_{\text{op}} \leq \frac{1}{1-2\epsilon} \max_{\mathbf{x}\in\mathcal{N}_\epsilon} |\mathbf{x}^\mathsf{T}\mathbf{B}\mathbf{x}|.$$
Mark the two interpolation errors that create 2ϵ.

3. **(Code.) Random probes versus a verified cover.** In two dimensions, compare k uniformly random directions with k equally spaced directions over 10,000 independent probe sets. Report the distribution of the actual covering radius, not only the largest observed stretch.

4. **(Code.) Choose epsilon.** For Equation 7.5 with the volumetric sphere bound, minimize the sum of anchor threshold and $L\epsilon$ numerically over epsilon for several values of d, L, and delta. Explain why the finest cover need not give the smallest final certificate.

5. **(Audit.) A missing regularity contract.** A report evaluates a random function at one million directions, applies a union bound, and calls the result a bound on the sphere. Write the strongest claim the evidence actually supports. Then list two distinct regularity statements that could repair the continuum step.

6. **(Audit.) Paper audit: where the cover enters.** Complete the **Mathematical object**, **Evidence culture and interface**, **Assumption stress test**, and **Discriminating control** fields for Baraniuk, Davenport, DeVore, and Wakin (Baraniuk et al. 2008). Identify the fixed-vector concentration statement, the set being covered, the covering-number cost, and the deterministic step extending anchor control. Separate those ingredients from the paper's additional union over supports.

8. One Average, Two Edges: Random Operators and Singular Values

RS · Random-matrix spectra SG · The sub-Gaussian safe zone Geometric primary · Dynamic quiet · Algorithmic supporting

Draw a 64×64 matrix $\mathbf{A}$ with independent $\mathcal{N}(0, 1/64)$ entries. Before seeing it, choose the first coordinate vector $\mathbf{e}_1$. In the seeded realization below,

$$\|\mathbf{A}\mathbf{e}_1\|_2 \approx 0.984, \qquad \frac{\|\mathbf{A}\|_{\mathrm{F}}^2}{64} \approx 1.019.$$

Both checks say "near one." The same matrix has largest singular value about 2.036 and smallest singular value about 0.0204. Its most contracted direction loses nearly two orders of magnitude relative to its most stretched direction.

> **!** Prediction
>
> For every fixed unit vector $\mathbf{x}$, $\mathbb{E}\,\|\mathbf{A}\mathbf{x}\|_2^2 = 1$. Which word in that statement prevents it from proving that the realized matrix preserves every direction? What diagnostic replaces the expectation when the direction is chosen after the matrix is observed?

8.1. The direction can adapt to the matrix

The top and bottom singular vectors are functions of the realized matrix. They are not fixed anchors to which one may apply a single-direction concentration statement. C07 built the missing route from fixed anchors to adaptive extrema; this chapter gives those extrema names and instruments.

1. Load the chapter-pinned operator instruments.
2. Generate one scaled Gaussian matrix from a declared seed.
3. Compare a predeclared direction and the Frobenius average with exact SVD edges.
4. Verify the stretches of the top and bottom right singular vectors.
5. Plot the average checks beside the adaptive edges.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import (
    gaussian_singular_value_trials,
```

```
    matrix_spectrum_summary,
    power_iteration_spectral_norm,
)

# [2]
dimension, matrix_seed = 64, 6224
rng = np.random.default_rng(matrix_seed)
matrix = rng.standard_normal((dimension, dimension)) / np.sqrt(dimension)

# [3]
summary = matrix_spectrum_summary(matrix)
fixed_direction = np.zeros(dimension)
fixed_direction[0] = 1.0
fixed_stretch = np.linalg.norm(matrix @ fixed_direction)
root_mean_squared_stretch = np.sqrt(summary.mean_squared_stretch)

# [4]
_, singular_values, right_vectors_t = np.linalg.svd(
    matrix,
    full_matrices=False,
)
top_stretch = np.linalg.norm(matrix @ right_vectors_t[0])
bottom_stretch = np.linalg.norm(matrix @ right_vectors_t[-1])
assert np.isclose(top_stretch, summary.sigma_max)
assert np.isclose(bottom_stretch, summary.sigma_min)
assert np.allclose(singular_values, np.asarray(summary.singular_values))

# [5]
labels = ["fixed", "RMS", r"$\sigma_{\max}$", r"$\sigma_{\min}$"]
values = [
    fixed_stretch,
    root_mean_squared_stretch,
    summary.sigma_max,
    summary.sigma_min,
]
fig, axis = plt.subplots(figsize=(6.9, 3.7))
bars = axis.bar(labels, values, color=["#232D4B", "#2E7D32", "#E57200", "#9C2F2F"])
axis.axhline(1.0, color="black", linestyle=":", label="isotropic center")
axis.set(ylabel="directional stretch", yscale="log", ylim=(0.01, 3.0))
axis.grid(alpha=0.25, axis="y")
axis.legend(frameon=False)
axis.bar_label(bars, fmt="%.3f", padding=3)
plt.tight_layout()
plt.show()

print(f"harness={manifest['harness_ref']} wheel={manifest['wheel_sha256'][:12]}")
print(
    f"fixed={fixed_stretch:.6f}; "
    f"mean squared={summary.mean_squared_stretch:.6f}"
```

```
)
print(
    f"sigma_max={summary.sigma_max:.6f}; "
    f"sigma_min={summary.sigma_min:.6f}; "
    f"kappa={summary.condition_number:.3f}"
)
```

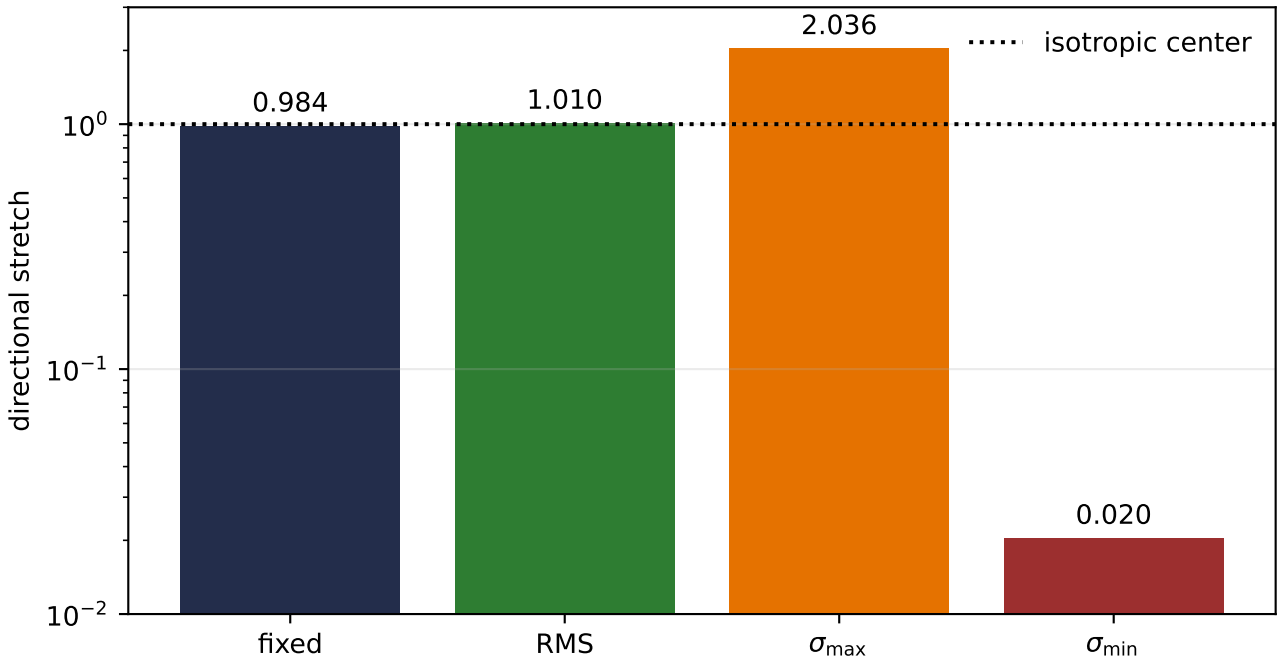


Figure 8.1.: One average conceals two adaptive edges. For the seeded square Gaussian matrix, the predeclared coordinate direction and root-mean-squared directional stretch sit near one. The top singular direction stretches by about two, while the bottom singular direction nearly collapses. Exact SVD is the control; this one matrix is a witness, not an asymptotic law.

```
harness=ch-08 wheel=9ec8176699a4
fixed=0.984012; mean squared=1.019185
sigma_max=2.036060; sigma_min=0.020407; kappa=99.773
```

This phenomenon does not contradict isotropy. For a fixed unit vector,

$$\mathbb{E}\,\|\mathbf{A}\mathbf{x}\|_2^2 = \sum_{i=1}^{m} \mathbb{E}\langle \mathbf{a}_i, \mathbf{x}\rangle^2 = 1 \tag{8.1}$$

under the $1/\sqrt{m}$ scaling. The quantifiers are:

$$\text{for each fixed } \mathbf{x}, \quad \mathbb{E}_{\mathbf{A}}[\cdots] = 1.$$

They are not:

$$\text{for each realized } \mathbf{A}, \quad \text{for every } \mathbf{x}, \ [\cdots] = 1.$$

Moving "every direction" through the random draw is exactly the uniformization problem.

8.2. Two edges define the stretch interval

For $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $m \geq n$, write the singular value decomposition

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}, \qquad \sigma_1 \geq \cdots \geq \sigma_n \geq 0. \tag{8.2}$$

We use

$$\sigma_{\max}(\mathbf{A}) = \sigma_1, \qquad \sigma_{\min}(\mathbf{A}) = \sigma_n, \qquad \|\mathbf{A}\|_{\mathrm{op}} = \sigma_{\max}(\mathbf{A}).$$

Theorem 8.1 (Singular-value stretch interval)**.** *For every* $\mathbf{x} \in \mathbb{R}^n$,

$$\sigma_{\min}(\mathbf{A})\,\|\mathbf{x}\|_2 \leq \|\mathbf{A}\mathbf{x}\|_2 \leq \sigma_{\max}(\mathbf{A})\,\|\mathbf{x}\|_2 . \tag{8.3}$$

Both inequalities are attained at the corresponding right singular vectors.

8.2.1. Proof

Put $\mathbf{z} = \mathbf{V}^{\mathsf{T}}\mathbf{x}$. Orthogonality preserves the Euclidean norm, so $\|\mathbf{z}\|_2 = \|\mathbf{x}\|_2$ and

$$\|\mathbf{A}\mathbf{x}\|_2^2 = \|\mathbf{\Sigma}\mathbf{z}\|_2^2 = \sum_{j=1}^{n} \sigma_j^2 z_j^2.$$

Bounding every σ_j^2 between σ_n^2 and σ_1^2 proves Equation 8.3. Choosing $\mathbf{x}$ as the first or last column of $\mathbf{V}$ gives equality. □

The Frobenius norm measures a different summary:

$$\|\mathbf{A}\|_{\mathrm{F}}^2 = \sum_{j=1}^{n} \sigma_j^2, \qquad \frac{\|\mathbf{A}\|_{\mathrm{F}}^2}{n} = \text{mean squared singular value.} \tag{8.4}$$

The average can equal one while the interval is wide. An exact isometry has every singular value equal to one. This is the first entry in the book's orthogonality/scale ledger.

> ⚠ Named wrong answer: 'Small singular value means underflow'
>
> A singular value is a relative geometric scale. Underflow is an absolute arithmetic event determined by input magnitude, subsequent operations, dtype, and storage/accumulation rules. A small lower edge predicts directional suppression; it does not, by itself, say that any represented number becomes subnormal or zero.

8.3. From fixed rows to all directions

Let the unscaled rows $\mathbf{a}_i \in \mathbb{R}^n$ be independent, centered, isotropic, and sub-Gaussian. For each fixed unit vector $\mathbf{x}$, $\langle \mathbf{a}_i, \mathbf{x} \rangle$ lies in C05's safe zone. The squared quantity

$$\|\mathbf{A}\mathbf{x}\|_2^2 = \sum_{i=1}^{m} \langle \mathbf{a}_i, \mathbf{x} \rangle^2$$

lies in C06's sub-exponential regime. Bernstein controls this sum at one fixed direction. A C07 net and a deterministic quadratic-form interpolation then extend the result over the sphere.

Theorem 8.2 (Two-sided sub-Gaussian matrix bound)**.** *Let* $\mathbf{A}$ *have independent, centered, isotropic, sub-Gaussian rows, and let*

$$K = \max_i \|\mathbf{a}_i\|_{\psi_2}.$$

There is a universal constant $C > 0$ *such that, for every* $t \geq 0$*, with probability at least* $1 - 2e^{-t^2}$*,*

$$\sqrt{m} - CK^2(\sqrt{n} + t) \leq \sigma_{\min}(\mathbf{A}) \leq \sigma_{\max}(\mathbf{A}) \leq \sqrt{m} + CK^2(\sqrt{n} + t). \tag{8.5}$$

8.3.1. Diagnostic proof sketch

For fixed $\mathbf{x}$, isotropy centers $m^{-1}\|\mathbf{A}\mathbf{x}\|_2^2$ at one. C06 turns each centered square into a sub-exponential variable, so Bernstein bounds its deviation. Choose a constant-radius net with exponentially many anchors, and let the exponent pay the log covering cost from C07. Apply the symmetric quadratic-form version of the net interpolation to

$$\frac{1}{m}\mathbf{A}^\mathsf{T}\mathbf{A} - \mathbf{I}.$$

The resulting operator-norm control bounds every eigenvalue of the Gram matrix, hence every squared singular value. Careful conversion from squared edges to singular edges gives Equation 8.5. Vershynin supplies the constant bookkeeping (Vershynin 2026).

This proof anatomy is more important diagnostically than the unnamed universal constant:

Ingredient	Contribution
isotropy	center at $\sqrt{m}$ before scaling
sub-Gaussian row projections	fixed-direction tail
squaring plus Bernstein	concentration of directional energy
sphere cover	$\sqrt{n}$ uniformization cost
interpolation	all directions between anchors

The lower statement becomes informative only when its displayed lower bound is positive. For a nearly square matrix, the generic theorem may correctly refuse to certify a margin.

8.4. Aspect ratio opens the lower edge

Gaussian matrices admit a sharper benchmark. The sharp constants require Gaussian comparison, not merely the elementary net calculation.

Theorem 8.3 (Gaussian singular-edge benchmark)**.** *Let* $\mathbf{G} \in \mathbb{R}^{m\times n}$*,* $m \geq n$*, have independent standard-normal entries. Then*

$$\mathbb{E}\sigma_{\max}(\mathbf{G}) \leq \sqrt{m} + \sqrt{n}, \qquad \mathbb{E}\sigma_{\min}(\mathbf{G}) \geq \sqrt{m} - \sqrt{n}. \tag{8.6}$$

Gaussian concentration gives sub-Gaussian upper and lower deviations around these edge locations. After scaling $\mathbf{A} = \mathbf{G}/\sqrt{m}$*, the benchmark interval is*

$$1 - \sqrt{\frac{n}{m}} \quad \textit{to} \quad 1 + \sqrt{\frac{n}{m}}. \tag{8.7}$$

8.4.1. Proof with pointer

The upper expectation is a supremum of the Gaussian process $\langle \mathbf{Gx}, \mathbf{y}\rangle$ over two unit spheres. The lower edge is the corresponding min–max process. Gaussian comparison bounds them by sums and differences of Gaussian-vector norms; Gaussian concentration then supplies deviations. This chapter uses the result as a diagnostic benchmark. The comparison proof is given in Vershynin (Vershynin 2026).

For $m = n$, the lower benchmark is zero. It does not say that every finite square Gaussian matrix is singular. It says that this theorem supplies no dimension-independent positive lower margin. The opening realization's condition number near 99.8 is therefore not evidence against isotropy.

The next study holds $n = 64$ fixed and increases m. Every point is computed from exact SVDs of 160 independently seeded finite matrices. The curves in Equation 8.7 are benchmarks, not fitted parameters and not an empirical spectral-density theorem.

1. Fix the input dimension and a grid of row counts.
2. Generate independent scaled Gaussian matrices for each shape.
3. Compute exact upper and lower singular edges for every trial.
4. Report predeclared 5th, 50th, and 95th percentiles.
5. Compare the finite ensembles with the aspect-ratio benchmark.

```
# [1]
columns = 64
rows_grid = np.array([64, 128, 256])
trials = 160

# [2]
ensembles = [
    gaussian_singular_value_trials(
        int(rows),
        columns,
        trials=trials,
```

```
        seed=6225 + index,
    )
    for index, rows in enumerate(rows_grid)
]

# [3]
lower = np.array([np.median(item["sigma_min"]) for item in ensembles])
upper = np.array([np.median(item["sigma_max"]) for item in ensembles])
lower_interval = np.array(
    [
        [np.quantile(item["sigma_min"], 0.05), np.quantile(item["sigma_min"], 0.95)]
        for item in ensembles
    ]
)
upper_interval = np.array(
    [
        [np.quantile(item["sigma_max"], 0.05), np.quantile(item["sigma_max"], 0.95)]
        for item in ensembles
    ]
)

# [4]
lower_error = np.vstack((lower - lower_interval[:, 0], lower_interval[:, 1] - lower))
upper_error = np.vstack((upper - upper_interval[:, 0], upper_interval[:, 1] - upper))

# [5]
ratios = rows_grid / columns
benchmark_lower = 1.0 - 1.0 / np.sqrt(ratios)
benchmark_upper = 1.0 + 1.0 / np.sqrt(ratios)
fig, axis = plt.subplots(figsize=(7.2, 3.8))
axis.errorbar(
    ratios,
    upper,
    yerr=upper_error,
    fmt="o-",
    capsize=4,
    linewidth=2.0,
    label=r"exact $\sigma_{\max}$ quantiles",
)
axis.errorbar(
    ratios,
    lower,
    yerr=lower_error,
    fmt="s-",
    capsize=4,
    linewidth=2.0,
    label=r"exact $\sigma_{\min}$ quantiles",
)
axis.plot(ratios, benchmark_upper, "--", color="gray", label="Gaussian benchmark")
```

```
axis.plot(ratios, benchmark_lower, "--", color="gray")
axis.axhline(1.0, color="black", linestyle=":")
axis.set(
    xlabel=r"aspect ratio $m/n$",
    ylabel="scaled singular value",
    xticks=ratios,
)
axis.grid(alpha=0.25)
axis.legend(frameon=False, ncol=2)
plt.tight_layout()
plt.show()

for rows, lo, hi in zip(rows_grid, lower, upper, strict=True):
    print(
        f"shape={rows}x{columns}: "
        f"median sigma_min={lo:.6f}, median sigma_max={hi:.6f}"
    )
```

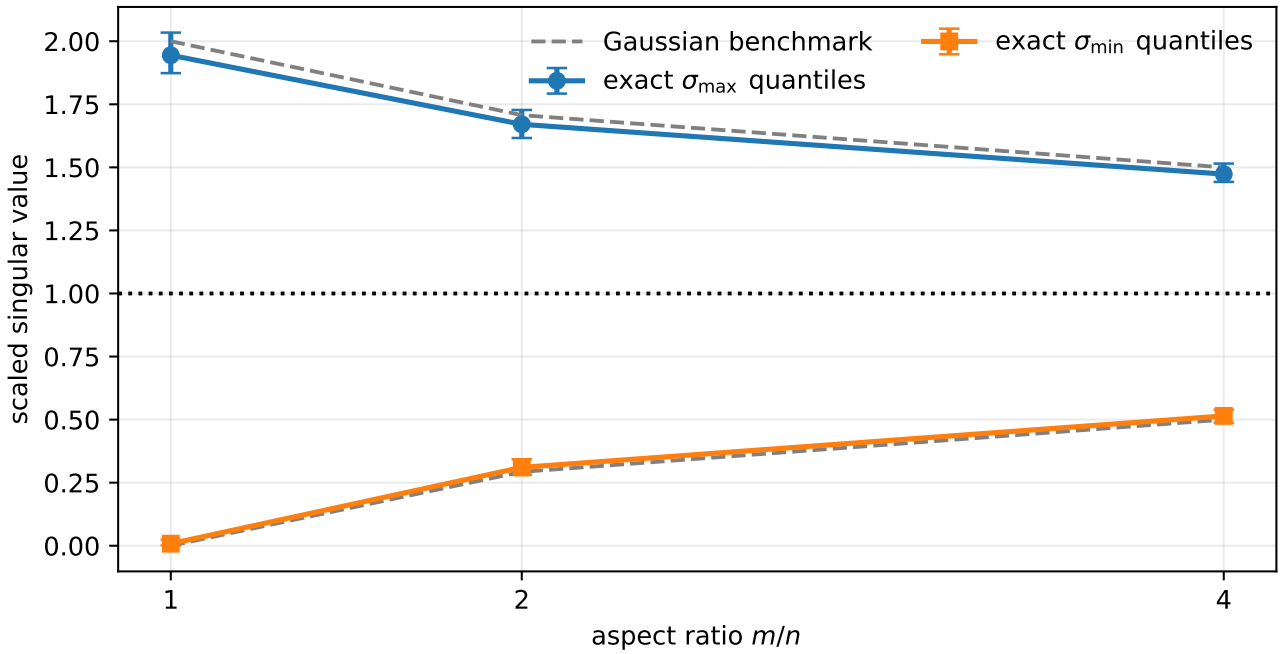


Figure 8.2.: Aspect ratio changes the margin between singular edges. Exact SVD quantiles from 160 finite Gaussian matrices per shape move toward the scaled Gaussian benchmark as the row count grows relative to the fixed input width. The square case has no positive lower benchmark; taller maps open one. Error bars show empirical fifth-to-ninety-fifth percentile ranges, not confidence intervals for a population edge.

```
shape=64x64: median sigma_min=0.008079, median sigma_max=1.944530
shape=128x64: median sigma_min=0.310570, median sigma_max=1.670645
shape=256x64: median sigma_min=0.514655, median sigma_max=1.473261
```

The median lower edge rises from about 0.0081 in the square ensemble to 0.515 at aspect ratio four. Meanwhile, the fixed-direction squared stretch and the mean squared singular value remain centered near one. “Preserves scale on average” and “has a positive uniform margin” are different claims.

8.5. One-sided instruments answer one-sided questions

Exact SVD is appropriate for these laptop controls, but expensive for a large matrix. **Power iteration** repeatedly applies $\mathbf{A}^\top\mathbf{A}$:

$$\mathbf{v}_{k+1} = \frac{\mathbf{A}^\top\mathbf{A}\mathbf{v}_k}{\|\mathbf{A}^\top\mathbf{A}\mathbf{v}_k\|_2}. \tag{8.8}$$

When the initialization has a component in the top right singular direction and the top eigenvalue is separated, the iteration amplifies that component. The estimate $\|\mathbf{A}\mathbf{v}_k\|_2$ approaches $\sigma_{\max}(\mathbf{A})$.

1. Reuse the seeded matrix with an exact SVD control.
2. Initialize one unit vector from a separate declared seed.
3. Apply power iteration to the Gram operator.
4. Record the upper-edge estimate at every iteration.
5. Plot relative error and state which edge the instrument never queried.

```
# [1]
exact_upper = summary.sigma_max
exact_lower = summary.sigma_min

# [2]
iteration_seed, iterations = 6228, 25

# [3]
trace = power_iteration_spectral_norm(
    matrix,
    iterations=iterations,
    seed=iteration_seed,
)

# [4]
estimates = np.asarray(trace.estimates)
relative_error = np.abs(estimates - exact_upper) / exact_upper
assert relative_error[-1] < relative_error[0]

# [5]
fig, axis = plt.subplots(figsize=(6.9, 3.6))
axis.semilogy(
    np.arange(1, iterations + 1),
    relative_error,
    "o-",
    markersize=3.5,
    linewidth=2.0,
```

```
)
axis.set(
    xlabel="power iteration",
    ylabel=r"relative error in $\sigma_{\max}$",
)
axis.grid(alpha=0.25, which="both")
axis.text(
    0.98,
    0.94,
    rf"$\sigma_{{\min}}={exact_lower:.4f}$ was not queried",
    transform=axis.transAxes,
    ha="right",
    va="top",
)
plt.tight_layout()
plt.show()

print(
    f"first estimate={estimates[0]:.6f}; "
    f"last estimate={estimates[-1]:.6f}; "
    f"exact sigma_max={exact_upper:.6f}"
)
print(f"exact sigma_min={exact_lower:.6f}; power-iteration estimate=not produced")
```

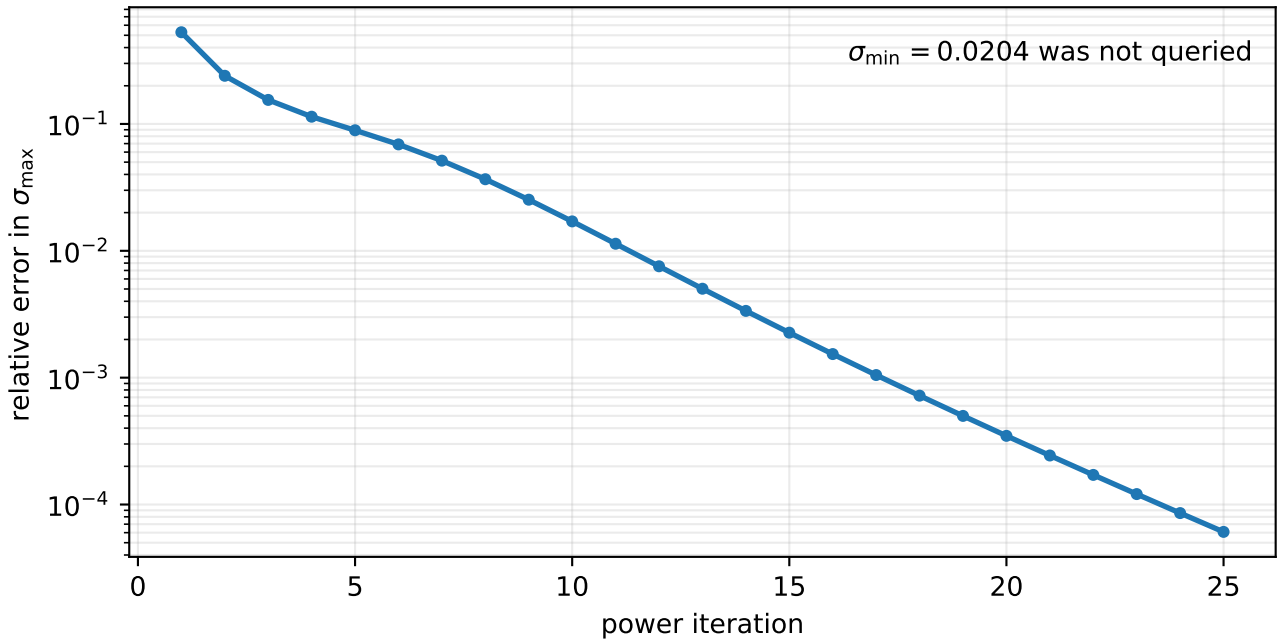


Figure 8.3.: Power iteration is an upper-edge instrument. On the seeded control matrix, the estimate approaches the exact largest singular value over 25 iterations. The exact smallest singular value is shown only as a separate SVD control; it is not estimated by this trajectory. Convergence here validates the implementation on one matrix, not a universal iteration count.

```
first estimate=0.960212; last estimate=2.035936; exact sigma_max=2.036060
exact sigma_min=0.020407; power-iteration estimate=not produced
```

Power iteration controls the largest singular value only. Applying the same routine longer does not turn it into a lower-edge diagnostic. The smallest edge requires a different method—often an inverse or shifted procedure—and that procedure has its own conditioning and linear-solve contract.

> **Field note: keep an exact small control**
>
> At production scale, randomized or iterative estimators may be necessary. Before trusting them, run the identical code path on matrices small enough for an exact SVD. Record the stopping rule, iteration count, initialization, and spectral-gap sensitivity. An approximate instrument without a control can confuse estimator failure with a changing matrix.

The boundary between lenses is now explicit:

Observation	What it diagnoses	What remains missing
small $\sigma_{\min}$	relative directional suppression	absolute activation/gradient scale and dtype
large $\sigma_{\max}$	relative directional amplification	nonlinearities and trajectory
wide singular interval	poor realized isometry	whether products across depth worsen or cancel it
power estimate	upper edge under an iteration contract	lower edge and full spectral mass

C03 decides when an absolute represented value is unsafe. C14 will decide how singular intervals compose across depth. C09 asks the question omitted here: how is the spectral mass distributed between the two edges?

> **Check yourself**
>
> A scaled random matrix satisfies $\|\mathbf{A}\|_{\mathrm{F}}^2/n = 1$ exactly, and a power iteration estimates $\sigma_{\max} = 1.8$. Can you bound $\sigma_{\min}$ away from zero from those two numbers alone? Construct a set of singular values with the same Frobenius average and upper edge but an arbitrarily small lower edge.

8.6. Okay, so —

- **Inherited:** C05 controlled fixed sub-Gaussian projections, C06 controlled their squares, and C07 supplied the finite-cover route to adaptive extrema.
- **Changed:** the diagnostic target is a realized random operator, so a fixed expectation or Frobenius average no longer stands in for all directions.
- **Instrumented:** the harness now reports exact singular summaries, seeded aspect-ratio ensembles, and a power-iteration trace validated against exact SVD.
- **Established:** singular values bound every directional stretch; isotropy sets a fixed-direction center; concentration plus covering creates a dimension-dependent edge interval; aspect ratio controls whether the Gaussian lower benchmark has positive margin.

- **Unresolved:** What does the spectral mass between two random-matrix edges say, and when does a finite outlier carry evidence rather than noise?

8.7. Sources and further reading

Vershynin develops the singular-value identities, two-sided sub-Gaussian matrix theorem, and Gaussian comparison bounds used here (Vershynin 2026). Miyato, Kataoka, Koyama, and Yoshida use power iteration inside an upper-edge intervention (Miyato et al. 2018); the paper supplies a useful audit of what such an instrument controls and what it leaves unmeasured.

For the general theory, use Vershynin (2026); this chapter's contribution is the matched two-edge witness and its finite certificate boundary.

Reading order. Start with Vershynin (2026) for the two-edge theorem and its covering proof, then use Miyato et al. (2018) to audit what an upper-edge intervention does not say about the lower edge.

8.8. Exercises

1. **(Pencil.) Same average, different interval.** Construct two diagonal 4×4 matrices with equal Frobenius norm and unequal operator norm. Then make the second matrix's smallest singular value arbitrarily close to zero while preserving its Frobenius norm.
2. **(Pencil.) Shape is part of the theorem.** For a wide matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m < n$, prove that its action on the full input space has lower edge zero. Explain why NumPy's list of $\min(m, n)$ returned singular values can obscure this nullspace if the shape contract is omitted.
3. **(Code.) Fixed versus adaptive directions.** Repeat the opening study for dimensions 16, 64, and 256. Predeclare ten unit directions before each matrix draw. Compare their largest stretch with exact singular edges, reporting seeds, trial denominators, and empirical quantiles.
4. **(Code.) Instrument sensitivity.** Build diagonal matrices with controlled ratios σ_2/σ_1. Measure the iterations required for power iteration to reach a fixed relative error over many initializations. Relate failures to the starting component in the top direction.
5. **(Audit.) Geometry is not arithmetic.** A report observes $\sigma_{\min} = 10^{-4}$ and declares that gradients underflow in FP16. List the missing absolute-scale, operation, storage, and accumulation facts. Design the smallest dtype-aware experiment that could support the narrower arithmetic claim.
6. **(Audit.) Paper audit: an upper-edge intervention.** Complete the **Mathematical object**, **Evidence culture and interface**, **Numerical and hardware contract**, **Discriminating control**, and **Transfer verdict** fields for Miyato, Kataoka, Koyama, and Yoshida (Miyato et al. 2018). Extract the target singular quantity, the power-iteration approximation, the persistence of its auxiliary vectors, and the empirical stability endpoint. State precisely what the method controls, then identify why that evidence does not imply a positive lower singular edge or a bounded condition number.

9. The Bulk Is Not a Verdict: Marchenko–Pastur and Finite-Null Calibration

RS · Random-matrix spectra SG · The sub-Gaussian safe zone Geometric primary · Dynamic quiet · Algorithmic supporting

Generate 128 observations with 64 independent standard-normal coordinates. The population covariance is exactly $\mathbf{I}_{64}$. For the seeded realization below, the largest eigenvalue of the sample covariance is

$$\lambda_{\max}(\widehat{\mathbf{\Sigma}}) \approx 2.921.$$

The Marchenko–Pastur upper edge at aspect ratio $\gamma = 64/128 = 1/2$ is

$$\lambda_+ = (1 + \sqrt{\gamma})^2 \approx 2.914.$$

The sample crosses the edge even though there is no population signal to find.

> **!** Prediction
>
> Does $\lambda_{\max} > \lambda_+$ reject the pure-noise model? If not, what must be calibrated before the word "signal" is justified?

9.1. The outlier is made of noise

The matrix convention matters. Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ have observations in rows and coordinates in columns. Here the population mean is known to be zero, so the estimator is the sample second moment

$$\widehat{\mathbf{\Sigma}} = \frac{1}{m}\mathbf{X}^{\mathsf{T}}\mathbf{X}. \tag{9.1}$$

Its eigenvalues are the squared singular values of $\mathbf{X}/\sqrt{m}$. C08 located those edges; this chapter instruments the mass between them. To prevent a hidden seed search, the witness seed is the first 32 bits of the SHA-256 digest of the registered phenomenon ID `c09-bulk-not-verdict`.

1. Load the chapter-pinned spectral instruments.
2. Generate the predeclared pure-noise witness.
3. Form the known-zero-mean second-moment estimator and compute its exact eigenvalues.
4. Evaluate the Marchenko–Pastur density under the matching aspect ratio.
5. Plot the finite empirical spectrum and mark the limiting support edges.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import (
    covariance_eigenvalues,
    empirical_upper_threshold,
    gaussian_covariance_trials,
    marchenko_pastur_density,
    marchenko_pastur_support,
)

# [2]
sample_count, feature_count = 128, 64
witness_seed = int(
    sha256(b"c09-bulk-not-verdict").hexdigest()[:8],
    16,
)
rng = np.random.default_rng(witness_seed)
data = rng.standard_normal((sample_count, feature_count))

# [3]
eigenvalues = covariance_eigenvalues(data)
aspect_ratio = feature_count / sample_count
support = marchenko_pastur_support(aspect_ratio)
assert eigenvalues[-1] > support.lambda_plus

# [4]
grid = np.linspace(support.lambda_minus, support.lambda_plus, 700)
density = marchenko_pastur_density(grid, aspect_ratio)

# [5]
fig, axis = plt.subplots(figsize=(7.2, 3.8))
axis.hist(
    eigenvalues,
    bins=np.linspace(0.0, 3.12, 17),
    density=True,
    alpha=0.55,
    color="#5379AA",
    label="finite ESD histogram",
)
axis.plot(grid, density, color="#232D4B", linewidth=2.2, label="MP density")
axis.axvline(
    support.lambda_minus,
    color="#6b6b6b",
    linestyle="--",
    label=r"limiting edges $\lambda_\pm$",
)
```

```
axis.axvline(support.lambda_plus, color="#6b6b6b", linestyle="--")
axis.axvline(
    eigenvalues[-1],
    color="#9C2F2F",
    linewidth=2.0,
    label="pure-noise top eigenvalue",
)
axis.set(xlabel=r"eigenvalue $\lambda$", ylabel="spectral density")
axis.grid(alpha=0.22)
axis.legend(frameon=False, fontsize=8)
plt.tight_layout()
plt.show()

print(f"harness={manifest['harness_ref']} wheel={manifest['wheel_sha256'][:12]}")
print(
    f"mean eigenvalue={np.mean(eigenvalues):.6f}; "
    f"lambda_max={eigenvalues[-1]:.6f}"
)
print(
    f"MP edges=[{support.lambda_minus:.6f}, "
    f"{support.lambda_plus:.6f}]"
)
```

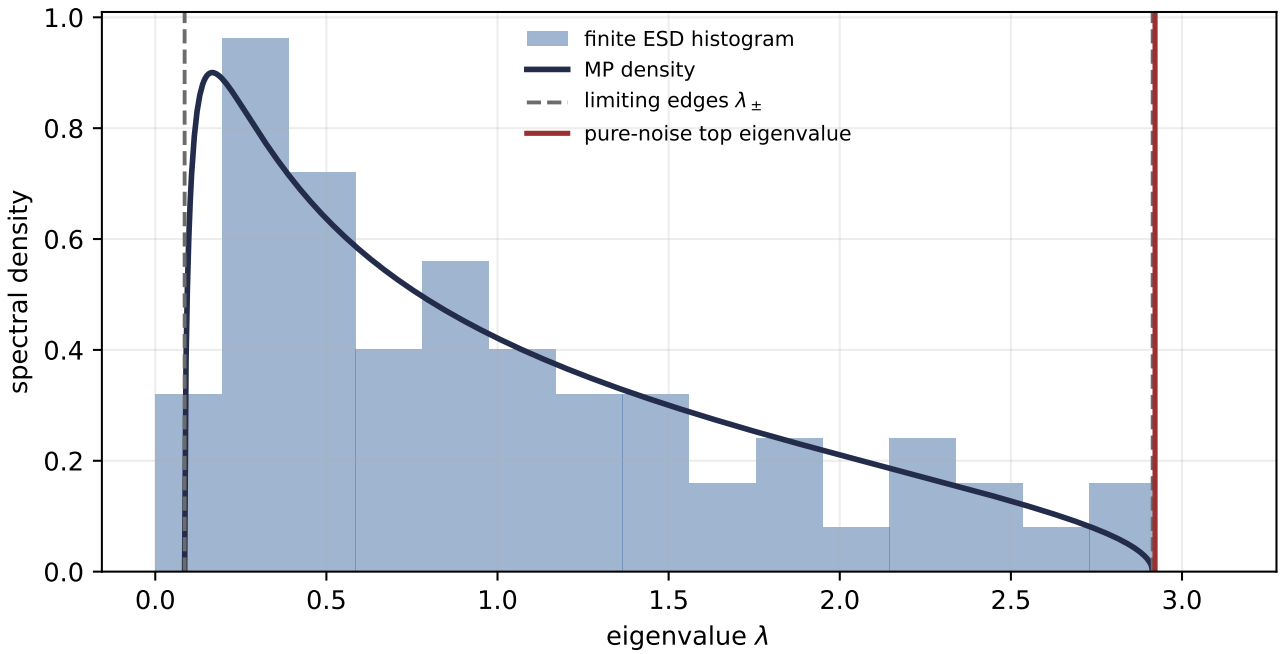


Figure 9.1.: A pure-noise finite spectrum can cross an asymptotic edge. The histogram is the exact eigenvalue spectrum of one seeded 64-coordinate sample covariance from 128 independent standard-normal observations. The curve is the Marchenko–Pastur limiting density for the matching aspect ratio. The top eigenvalue lies beyond the limiting upper edge; that crossing is the symptom, not yet a rejection rule.

```
harness=ch-09 wheel=0d4b6e953774
mean eigenvalue=1.021187; lambda_max=2.921244
MP edges=[0.085786, 2.914214]
```

The overlay is visually persuasive and logically insufficient. The Marchenko–Pastur curve is a limit under a declared generative model. The histogram is one finite random measure. Neither object says how often the largest finite eigenvalue crosses the limiting support.

9.2. Name the estimator before naming the law

Three similar-looking matrices answer different questions:

Mean contract	Computation	Denominator meaning
population mean known to be zero	$\frac{1}{m}\sum_i \mathbf{x}_i\mathbf{x}_i^\mathsf{T}$	unbiased second moment under the declared mean
population mean unknown; maximum-likelihood Gaussian convention	$\frac{1}{m}\sum_i (\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_i-\bar{\mathbf{x}})^\mathsf{T}$	centered average
population mean unknown; unbiased covariance convention	$\frac{1}{m-1}\sum_i (\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_i-\bar{\mathbf{x}})^\mathsf{T}$	degrees-of-freedom correction

Changing centering or denominator changes the finite null. The opening uses the first row of the table. A report that says only "we computed the covariance" has not specified its estimator.

For a symmetric matrix $\mathbf{M}$ with eigenvalues $\lambda_1, \dots, \lambda_n$, define its **empirical spectral distribution**

$$\mu_{\mathbf{M}} = \frac{1}{n}\sum_{j=1}^{n} \delta_{\lambda_j}. \tag{9.2}$$

This is not merely a histogram. It is the exact probability measure that places mass $1/n$ at each finite eigenvalue. A histogram adds a binning choice for visualization.

Theorem 9.1 (Trace moments of an empirical spectrum)**.** *For every nonnegative integer k,*

$$\int \lambda^k \, d\mu_{\mathbf{M}}(\lambda) = \frac{1}{n}\operatorname{tr}(\mathbf{M}^k). \tag{9.3}$$

9.2.1. Proof

Write $\mathbf{M} = \mathbf{Q}\operatorname{diag}(\lambda_1, \dots, \lambda_n)\mathbf{Q}^\mathsf{T}$. Then

$$\operatorname{tr}(\mathbf{M}^k) = \sum_{j=1}^{n} \lambda_j^k.$$

Integrating against the point masses in Equation 9.2 gives the same sum divided by n. For $k = 0$, the identity says that the ESD has total mass one. □

The first moment is normalized trace. For the sample covariance,

$$\int \lambda \, d\mu_{\widehat{\mathbf{\Sigma}}}(\lambda) = \frac{1}{mn} \|\mathbf{X}\|_{\mathrm{F}}^2 .$$

The largest eigenvalue is a different functional. A correct mean does not pin the upper edge, just as C08's Frobenius average did not pin both singular edges.

9.3. The bulk law is asymptotic

Theorem 9.2 (Marchenko–Pastur law)**.** *Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ have independent entries with mean zero, variance one, and finite fourth moment. Suppose $m, n \to \infty$ with*

$$\frac{n}{m} \longrightarrow \gamma \in (0, \infty).$$

Then the ESD of $\widehat{\mathbf{\Sigma}} = \mathbf{X}^{\mathsf{T}}\mathbf{X}/m$ converges almost surely, weakly, to the Marchenko–Pastur measure. Its continuous density is

$$\rho_\gamma(\lambda) = \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{2\pi\gamma\lambda} \mathbf{1}_{[\lambda_-, \lambda_+]}(\lambda), \tag{9.4}$$

where

$$\lambda_\pm = (1 \pm \sqrt{\gamma})^2. \tag{9.5}$$

When $\gamma > 1$, the measure also has a point mass $1 - 1/\gamma$ at zero.

9.3.1. Proof with pointer

One proof studies normalized traces of powers; another studies the Stieltjes transform

$$s_n(z) = \frac{1}{n} \operatorname{tr}\left(\widehat{\mathbf{\Sigma}} - z\mathbf{I}\right)^{-1} .$$

Independence and proportional growth produce a deterministic fixed-point equation for the limiting transform, whose inversion yields Equation 9.4. Establishing almost-sure convergence and controlling the resolvent error is the technical work. We use the theorem diagnostically and point to the original paper for the full argument (Marchenko and Pastur 1967).

The scaling is part of the theorem. The entries of $\mathbf{X}$ have variance one and the Gram matrix is divided by m. Equivalently, $\mathbf{A} = \mathbf{X}/\sqrt{m}$ has entry variance $1/m$ and $\widehat{\mathbf{\Sigma}} = \mathbf{A}^{\mathsf{T}}\mathbf{A}$. Mixing variance $1/n$ with the edges in Equation 9.5 changes the mean spectral scale by a factor of m/n.

C08's scaled Gaussian singular edges were $1 \pm \sqrt{\gamma}$. The eigenvalue edges here are their squares. The Marchenko–Pastur law adds the missing mass between them.

⚠ Named wrong answer: 'Above the MP edge means signal'

The support describes a limiting measure. It does not say that every eigenvalue of every finite null matrix lies inside the support. An edge crossing becomes evidence only after the finite statistic, calibration rule, false-positive budget, and alternative model are declared.

9.4. A deterministic bulk can still be a poor covariance estimate

The sample spectrum can converge to a beautiful deterministic shape while the sample covariance remains far from the population covariance in operator norm. For the isotropic case,

$$\left\|\widehat{\boldsymbol{\Sigma}} - \mathbf{I}\right\|_{\mathrm{op}} = \max\left\{\lambda_{\max}(\widehat{\boldsymbol{\Sigma}}) - 1,\, 1 - \lambda_{\min}(\widehat{\boldsymbol{\Sigma}})\right\}. \tag{9.6}$$

At fixed positive γ, the limiting edges do not collapse to one.

Theorem 9.3 (Sub-Gaussian covariance-estimation error)**.** *Let $\mathbf{X} \in \mathbb{R}^n$ be centered with covariance $\boldsymbol{\Sigma}$, and assume that for every $\mathbf{x}$,*

$$\|\langle \mathbf{X}, \mathbf{x}\rangle\|_{\psi_2} \le K \|\langle \mathbf{X}, \mathbf{x}\rangle\|_{L^2}.$$

For m independent copies and the estimator in Equation 9.1,

$$\mathbb{E}\left\|\widehat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}\right\|_{\mathrm{op}} \le CK^2\left(\sqrt{\frac{n}{m}} + \frac{n}{m}\right)\|\boldsymbol{\Sigma}\|_{\mathrm{op}}. \tag{9.7}$$

9.4.1. Diagnostic proof sketch

Whiten the data: $\mathbf{Z} = \boldsymbol{\Sigma}^{-1/2}\mathbf{X}$. The whitened rows are isotropic and sub-Gaussian, so C08 controls all singular values of their data matrix. Squaring those bounds controls $\left\|m^{-1}\mathbf{Z}^{\mathsf{T}}\mathbf{Z} - \mathbf{I}\right\|_{\mathrm{op}}$. Conjugating by $\boldsymbol{\Sigma}^{1/2}$ contributes $\|\boldsymbol{\Sigma}\|_{\mathrm{op}}$. Vershynin gives the full expectation and high-probability bookkeeping (Vershynin 2026).

The ambient-dimension rate says that m must scale like n/ε^2 for relative operator error ε. C10 will replace n with an effective dimension when the population spectrum is concentrated. That promise is now contractual, not rhetorical.

9.5. Calibrate the finite statistic

Return to the opening shape. The finite null must match:

- $m = 128$, $n = 64$;
- independent standard-normal entries;
- known population mean zero, with no sample centering;
- denominator m;

- statistic $\lambda_{\max}$;
- false-positive rate $\alpha = 0.05$;
- one predeclared look.

The next study uses 2,000 independent null matrices with a seed different from the opening witness. The 95th empirical percentile is a calibrated threshold for this contract—not a universal constant.

1. Reuse the declared shape and analytic Marchenko–Pastur edge.
2. Generate an independently seeded finite Gaussian null ensemble.
3. Compute the largest eigenvalue for every exact covariance matrix.
4. Predeclare the upper five-percent empirical threshold.
5. Compare the opening witness with both the asymptotic edge and finite threshold.

```
# [1]
asymptotic_edge = support.lambda_plus

# [2]
null_trials, null_seed = 2000, 6291
null = gaussian_covariance_trials(
    sample_count,
    feature_count,
    trials=null_trials,
    seed=null_seed,
)

# [3]
null_max = null["lambda_max"]
crossing_rate = np.mean(null_max > asymptotic_edge)

# [4]
false_positive_rate = 0.05
finite_threshold = empirical_upper_threshold(
    null_max,
    false_positive_rate=false_positive_rate,
)

# [5]
witness_max = eigenvalues[-1]
fig, axis = plt.subplots(figsize=(7.2, 3.7))
axis.hist(
    null_max,
    bins=34,
    density=True,
    color="#2E7D32",
    alpha=0.7,
    label=r"finite null for $\lambda_{\max}$",
)
axis.axvline(
    asymptotic_edge,
    color="#6b6b6b",
```

```
    linestyle="--",
    linewidth=2.0,
    label="asymptotic upper edge",
)
axis.axvline(
    finite_threshold,
    color="#232D4B",
    linewidth=2.0,
    label="finite 95th percentile",
)
axis.axvline(
    witness_max,
    color="#9C2F2F",
    linewidth=2.0,
    label="opening witness",
)
axis.set(xlabel=r"largest eigenvalue $\lambda_{\max}$", ylabel="density")
axis.grid(alpha=0.22)
axis.legend(frameon=False, fontsize=8)
plt.tight_layout()
plt.show()

print(
    f"null edge-crossing rate={crossing_rate:.4f}; "
    f"finite threshold={finite_threshold:.6f}"
)
print(
    f"witness={witness_max:.6f}; "
    f"reject at alpha={false_positive_rate:.2f}: "
    f"{witness_max > finite_threshold}"
)
```

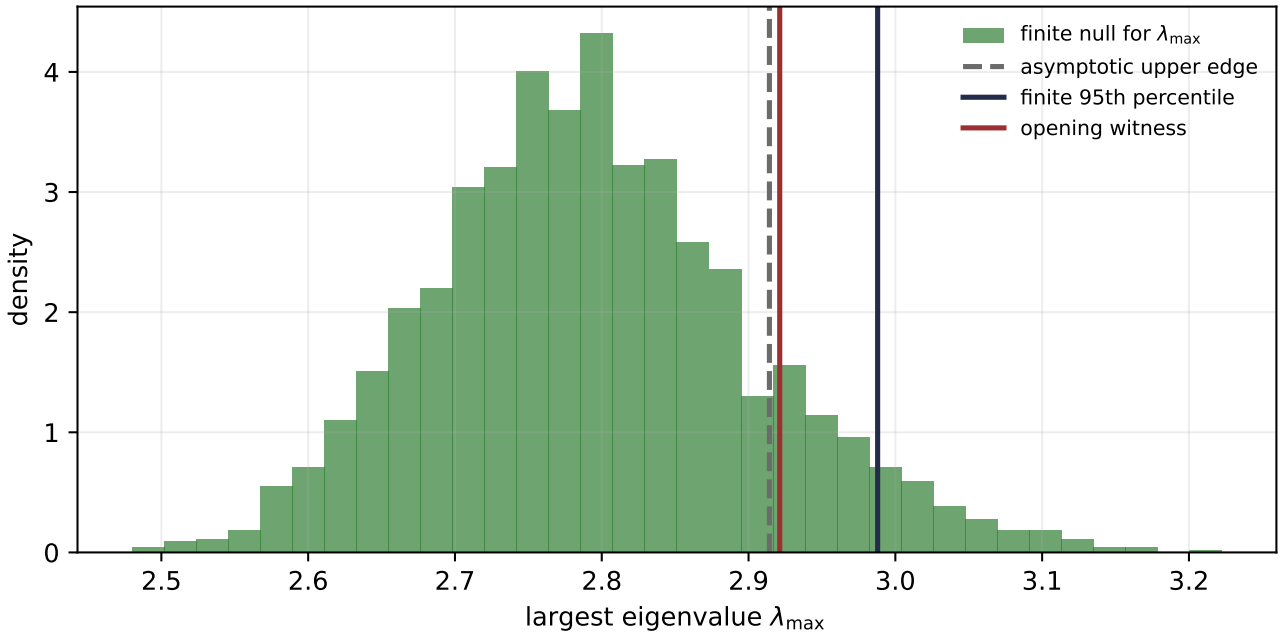


Figure 9.2.: The limiting upper edge is not a finite five-percent threshold. In 2,000 independently seeded pure-noise covariance matrices of the opening shape, 13.85 percent of largest eigenvalues exceed the asymptotic edge. The opening witness crosses that edge but remains below the empirical 95th-percentile null threshold. The histogram is a distribution of one predeclared statistic, not an ESD.

```
null edge-crossing rate=0.1385; finite threshold=2.987940
witness=2.921244; reject at alpha=0.05: False
```

The result resolves the symptom. In this finite ensemble, 13.85% of pure noise matrices cross the limiting edge. The opening value 2.921 is above the edge but below the finite 95th percentile 2.988. Under this predeclared test, it is not rejected.

The threshold is only as portable as its null contract. Correlated rows, heavy-tailed coordinates, sample centering, a different aspect ratio, or a search across many layers all require recalibration.

Theorem 9.4 (Repeated-look false alarms)**.** *If each of K inspected spectra has a null event with probability at most α, then*

$$\Pr\{\textit{at least one false alarm}\} \leq K\alpha.$$

If the looks are independent and each event has probability exactly α, then the probability is

$$1 - (1 - \alpha)^K. \tag{9.8}$$

9.5.1. Proof

The first statement is the union bound. Under independence, no false alarm occurs with probability $(1-\alpha)^K$; take the complement. □

At twenty independent looks, a nominal five-percent per-look rule has about a 64.2% chance of at least one false alarm. If the asymptotic edge were mistaken for the threshold, the measured 13.85% per-look crossing rate would produce about a 94.9% chance of at least one crossing. Monitoring creates a multiple-comparison problem even before training dynamics enter.

> **i** Field note: nulls are executable specifications
>
> Store the null generator beside the diagnostic. Record shape, orientation, centering, denominator, marginal distribution, dependence assumptions, statistic, seed, trial count, quantile method, false-positive level, and number of looks. "Compared with MP" is a picture caption; it is not a reproducible test.

9.6. Signal needs an alternative model

A finite null tells us which values are unusual under noise. It does not say which structured mechanism produced an unusual value. The smallest useful alternative is a rank-one population spike:

$$\mathbf{\Sigma}_\beta = \operatorname{diag}(\beta, 1, \dots, 1), \qquad \beta > 1. \tag{9.9}$$

Even this genuine population structure need not create a separated sample eigenvalue.

Theorem 9.5 (Rank-one spiked-covariance separation)**.** *Under the model in Equation 9.9, independent standardized entries with finite fourth moment, and $n/m \to \gamma \in (0,1)$, the largest sample eigenvalue converges almost surely to*

$$\lambda_{\max} \longrightarrow \begin{cases} (1+\sqrt{\gamma})^2, & \beta \le 1+\sqrt{\gamma}, \\ \beta\left(1+\dfrac{\gamma}{\beta-1}\right), & \beta > 1+\sqrt{\gamma}. \end{cases} \tag{9.10}$$

9.6.1. Proof with pointer

Baik and Silverstein characterize the almost-sure limits of sample eigenvalues for finite-rank perturbations of the identity covariance (Baik and Silverstein 2006). The displayed rank-one formula is their specialization. The important diagnostic fact is the phase boundary: below $1+\sqrt{\gamma}$, a real population spike is asymptotically absorbed at the null edge.

For $\gamma = 1/2$, the population threshold is about 1.707. The next finite study compares $\beta = 1$, a subcritical spike $\beta = 1.5$, and a supercritical spike $\beta = 2.5$, each over 2,000 independently seeded matrices. All three use the same five-percent finite null threshold from the previous study.

1. Declare null, subcritical, and supercritical population eigenvalues.
2. Generate independent finite ensembles under the rank-one Gaussian model.

3. Compute the exact largest sample eigenvalue in every trial.
4. Compare each ensemble with the previously calibrated finite null threshold.
5. Plot finite distributions and rejection fractions beside the asymptotic separation boundary.

```
# [1]
models = [
    ("null", 1.0, null_seed),
    ("subcritical", 1.5, 6292),
    ("supercritical", 2.5, 6293),
]

# [2]
ensembles = [
    null
    if beta == 1.0
    else gaussian_covariance_trials(
        sample_count,
        feature_count,
        trials=null_trials,
        seed=seed,
        population_spike=beta,
    )
    for _, beta, seed in models
]

# [3]
maxima = [ensemble["lambda_max"] for ensemble in ensembles]

# [4]
rejection_rates = np.array(
    [np.mean(values > finite_threshold) for values in maxima]
)
separation_threshold = 1.0 + np.sqrt(aspect_ratio)

# [5]
labels = [f"{label}\n" + rf"$\beta={beta:g}$" for label, beta, _ in models]
fig, axes = plt.subplots(1, 2, figsize=(8.0, 3.8))
axes[0].boxplot(
    maxima,
    tick_labels=labels,
    whis=(5, 95),
    showfliers=False,
    patch_artist=True,
    boxprops={"facecolor": "#5379AA"},
    medianprops={"color": "#232D4B", "linewidth": 2.0},
)
axes[0].axhline(
    finite_threshold,
    color="#9C2F2F",
```

```
    linestyle="--",
    label="finite null threshold",
)
axes[0].set(ylabel=r"largest sample eigenvalue $\lambda_{\max}$")
axes[0].grid(alpha=0.22, axis="y")
axes[0].legend(frameon=False, fontsize=8)

bars = axes[1].bar(
    labels,
    rejection_rates,
    color=["#2E7D32", "#E57200", "#9C2F2F"],
)
axes[1].axhline(
    false_positive_rate,
    color="#232D4B",
    linestyle=":",
    label="declared null rate",
)
axes[1].set(ylabel="fraction above finite threshold", ylim=(0.0, 1.0))
axes[1].bar_label(bars, fmt="%.3f", padding=3)
axes[1].grid(alpha=0.22, axis="y")
axes[1].legend(frameon=False, fontsize=8)
plt.tight_layout()
plt.show()

print(f"population separation threshold={separation_threshold:.6f}")
for (label, beta, _), rate in zip(models, rejection_rates, strict=True):
    print(f"{label}: beta={beta:.1f}; fraction above threshold={rate:.4f}")
```

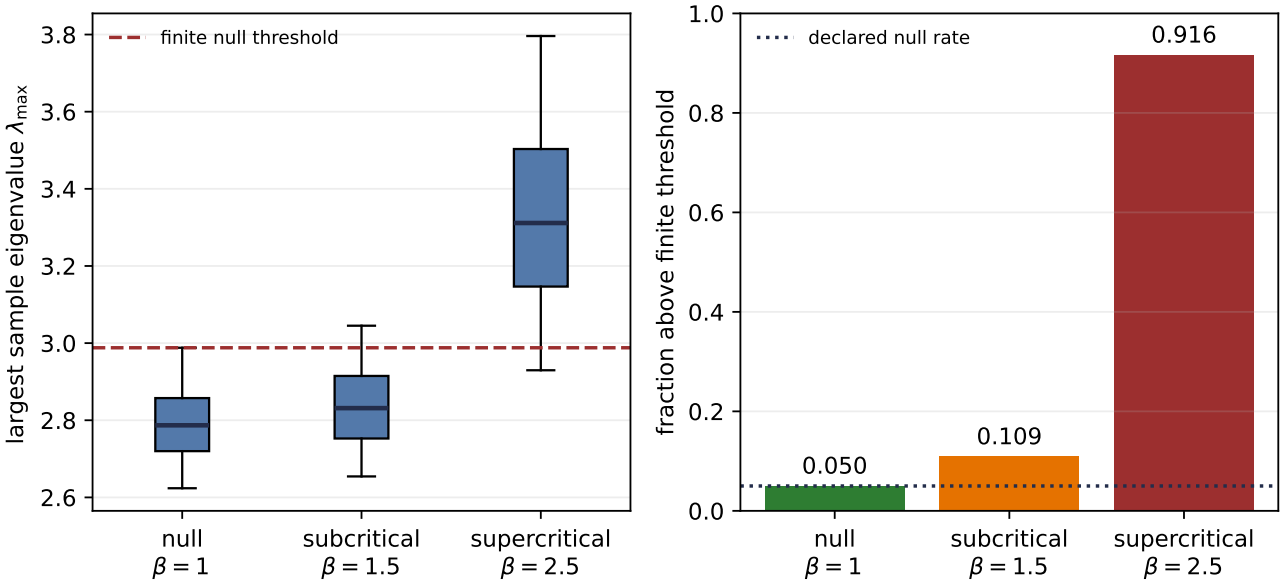


Figure 9.3.: A population spike is not the same as a separated sample eigenvalue. At aspect ratio one half, the subcritical population spike beta equals 1.5 remains close to the null largest-eigenvalue distribution and crosses the finite five-percent threshold in only 10.9 percent of trials. The supercritical spike beta equals 2.5 separates and crosses in 91.65 percent. Boxes span the interquartile range; whiskers show the fifth and ninety-fifth percentiles.

```
population separation threshold=1.707107
null: beta=1.0; fraction above threshold=0.0500
subcritical: beta=1.5; fraction above threshold=0.1090
supercritical: beta=2.5; fraction above threshold=0.9165
```

The subcritical model contains genuine population structure, yet most finite sample spectra do not separate from the null. Conversely, an arbitrary outlier in a learned matrix does not inherit this theorem unless the spiked covariance model is defensible.

The diagnostic hierarchy is now explicit:

Object	Question answered	Invalid shortcut
ESD	Where did this finite matrix place spectral mass?	a histogram proves the null
Marchenko–Pastur law	What bulk emerges under an iid proportional-growth null?	the limiting edge is a finite test
finite null	Is the predeclared statistic unusual at this exact shape and estimator?	unusual identifies the mechanism
spiked alternative	When does one declared population perturbation separate?	every learned outlier is a spike

A Hessian, a parameter update, and a sample covariance are all symmetric matrices after suitable products, but symmetry does not give them the same generative law. C12 will build a curvature-specific null. C14 will ask how one-layer spectra compose through depth.

Check yourself

Two teams inspect the same 50 independently initialized layers. Team A flags every eigenvalue above the asymptotic Marchenko–Pastur edge. Team B calibrates the maximum across all 50 layers under a matched joint null. Which team controls the experiment-level false-positive probability? List the metadata needed to reproduce Team B's threshold.

9.7. Okay, so —

- **Inherited:** C08 supplied singular edges and the aspect-ratio geometry; C05 supplied a sub-Gaussian null class, and C06 warned that covariance entries are squared quantities.
- **Changed:** two edges became a full empirical spectral measure, and the asymptotic bulk became a baseline rather than a finite verdict.
- **Instrumented:** the harness now declares covariance orientation, centering, denominator, Marchenko–Pastur support and density, finite extreme-eigenvalue trials, and an empirical upper threshold.
- **Established:** Marchenko–Pastur explains the limiting null mass under its assumptions; covariance estimation can still have order-one operator error; repeated looks spend a false-positive budget; and a genuine subcritical population spike can remain inside the sample bulk.
- **Unresolved:** Which dimension should replace ambient coordinate count when most spectral mass lies in only a few directions?

9.8. Sources and further reading

Marchenko and Pastur establish the limiting empirical spectral law (Marchenko and Pastur 1967). Vershynin supplies the sub-Gaussian covariance-estimation theorem and its whitening argument (Vershynin 2026). Baik and Silverstein derive the almost-sure sample-eigenvalue limits for spiked population models (Baik and Silverstein 2006).

For the general theory, use Vershynin (2026) and the primary random-matrix papers; this chapter's contribution is an executable finite-null calibration.

Reading order. Start with Marchenko and Pastur (1967) for the limiting bulk, use Vershynin (2026) for finite covariance error, and read Baik and Silverstein (2006) only after separating a population spike from a finite sample verdict.

9.9. Exercises

1. **(Pencil.) Mass, mean, and edge.** Construct two probability measures on four nonnegative points with the same first moment but different maxima. Realize them as ESDs of diagonal matrices. Explain why normalized trace cannot replace an upper-edge diagnostic.

2. **(Pencil.) The zero atom.** Let $n > m$. Prove directly from rank that $\mathbf{X}^\top\mathbf{X}$ has at least $n - m$ zero eigenvalues. Show that their ESD mass tends to $1 - 1/\gamma$ when $n/m \to \gamma > 1$.

3. **(Code.) Estimator contract.** Under one fixed Gaussian data generator, compare known-zero-mean division by m, sample centering with division by m, and sample centering with division by $m-1$. Calibrate a separate finite largest-eigenvalue threshold for each. Report seeds, denominators, trial counts, and quantile method.

4. **(Code.) Finite shape audit.** Hold $\gamma = 1/2$ approximately fixed while increasing m, n. Measure the probability that $\lambda_{\max}$ crosses the limiting edge and the distance between the edge and the empirical 95th percentile. Do not infer a convergence rate from a log–log line without a theorem.

5. **(Audit.) Null misspecification.** Replace independent Gaussian rows with temporally correlated rows, then with heavy-tailed coordinates. Keep the old threshold long enough to measure its false-positive rate. Identify which Marchenko–Pastur and covariance-estimation assumptions were broken.

6. **(Audit.) Paper audit: a model-specific outlier.** Complete the **Mathematical object**, **Resolution of the theory**, **Dynamic regime**, **Assumption stress test**, **Discriminating control**, and **Transfer verdict** fields for Baik and Silverstein (Baik and Silverstein 2006). Extract the population model, aspect-ratio limit, moment assumptions, separation condition, almost-sure conclusion, and the quantity that remains asymptotic. Then review a spectral-outlier claim in a current machine-learning paper: state whether its matrix has the same generative model and propose a matched finite null if it does not.

10. Full Rank, Few Directions: Effective Rank and Covariance Complexity

RS · Random-matrix spectra SG · The sub-Gaussian safe zone Geometric primary · Dynamic quiet · Algorithmic supporting

Consider two Gaussian populations in $\mathbb{R}^{96}$. Both covariance matrices have algebraic rank 96. Both have operator norm one. Yet with only 64 observations, the median relative operator-norm error of the sample covariance is about 3.71 for one population and 0.42 for the other.

Nothing about rank or largest eigenvalue alone predicts this nine-fold difference. The missing quantity measures how much spectral mass sits below the leading direction.

> **!** Prediction
>
> If two covariance matrices have the same dimension, rank, and largest eigenvalue, which remaining rival would you inspect first: condition number, determinant, a truncation rank, or the full decay profile? Name the scalar summary you would want that rival to justify.

10.1. Same rank, different sample demand

Let

$$\widehat{\mathbf{\Sigma}} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i \mathbf{x}_i^{\mathsf{T}} \tag{10.1}$$

for independent, mean-zero observations. We compare

$$\mathbf{\Sigma}_{\text{iso}} = \mathbf{I}_{96}, \qquad \mathbf{\Sigma}_{\text{dec}} = \operatorname{diag}(1, 0.82, 0.82^2, \dots, 0.82^{95}). \tag{10.2}$$

Every diagonal entry in the second matrix is positive, so both matrices are full rank. Their leading eigenvalues are both one. The experiment below uses the same standard-normal draws for the two populations at every sample count; only the spectral coloring changes.

1. Load the chapter-pinned covariance instruments.
2. Construct the two full-rank population spectra.
3. Run the predeclared matched Gaussian covariance trials.
4. Compute relative operator-norm error quantiles.
5. Compare the $m = 64$ error distributions.
6. Trace the median and 10–90 percent bands across sample counts.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import (
    effective_rank_psd,
    gaussian_covariance_error_trials,
)

# [2]
dimension = 96
spectra = {
    "isotropic": np.ones(dimension),
    "decaying": 0.82 ** np.arange(dimension),
}
effective_ranks = {
    name: effective_rank_psd(np.diag(eigenvalues)).effective_rank
    for name, eigenvalues in spectra.items()
}

# [3]
sample_counts = np.array([24, 48, 64, 96, 192, 384])
trial_count = 300
base_seed = int(
    sha256(b"c10-full-rank-few-directions").hexdigest()[:8],
    16,
)
errors = {name: {} for name in spectra}
for sample_count in sample_counts:
    seed = (base_seed + int(sample_count)) % (2**32)
    for name, eigenvalues in spectra.items():
        errors[name][sample_count] = gaussian_covariance_error_trials(
            eigenvalues,
            int(sample_count),
            trials=trial_count,
            seed=seed,
        )["relative_operator_error"]

# [4]
quantiles = {
    name: np.array([
        np.quantile(errors[name][sample_count], [0.1, 0.5, 0.9])
        for sample_count in sample_counts
    ])
    for name in spectra
}

# [5]
```

```
fig, axes = plt.subplots(1, 2, figsize=(9.4, 3.8))
axes[0].boxplot(
    [errors["isotropic"][64], errors["decaying"][64]],
    tick_labels=["isotropic", "decaying"],
    showfliers=False,
)
axes[0].set(
    title="$m=64$ matched trials",
    ylabel=r"$\|\widehat{\Sigma}-\Sigma\|_{\mathrm{op}}/\|\Sigma\|_{\mathrm{op}}$",
)
axes[0].grid(axis="y", alpha=0.22)

# [6]
colors = {"isotropic": "#9C2F2F", "decaying": "#232D4B"}
for name in spectra:
    q = quantiles[name]
    axes[1].plot(
        sample_counts, q[:, 1], marker="o", color=colors[name],
        label=rf"{name}: $r_{{\rm eff}}={effective_ranks[name]:.2f}$",
    )
    axes[1].fill_between(
        sample_counts, q[:, 0], q[:, 2],
        color=colors[name], alpha=0.16,
    )
axes[1].set(
    xscale="log", yscale="log",
    xlabel="sample count $m$", ylabel="relative operator error",
)
shown_counts = [24, 64, 192, 384]
axes[1].set_xticks(shown_counts, labels=[str(value) for value in shown_counts])
axes[1].minorticks_off()
axes[1].grid(alpha=0.22)
axes[1].legend(frameon=False, fontsize=8)
plt.tight_layout()
plt.show()

print(f"harness={manifest['harness_ref']} wheel={manifest['wheel_sha256'][:12]}")
for name in spectra:
    print(
        f"{name}: rank={np.linalg.matrix_rank(np.diag(spectra[name]))}, "
        f"operator_norm={spectra[name][0]:.1f}, "
        f"effective_rank={effective_ranks[name]:.6f}, "
        f"median_error_m64={np.median(errors[name][64]):.6f}"
    )
```

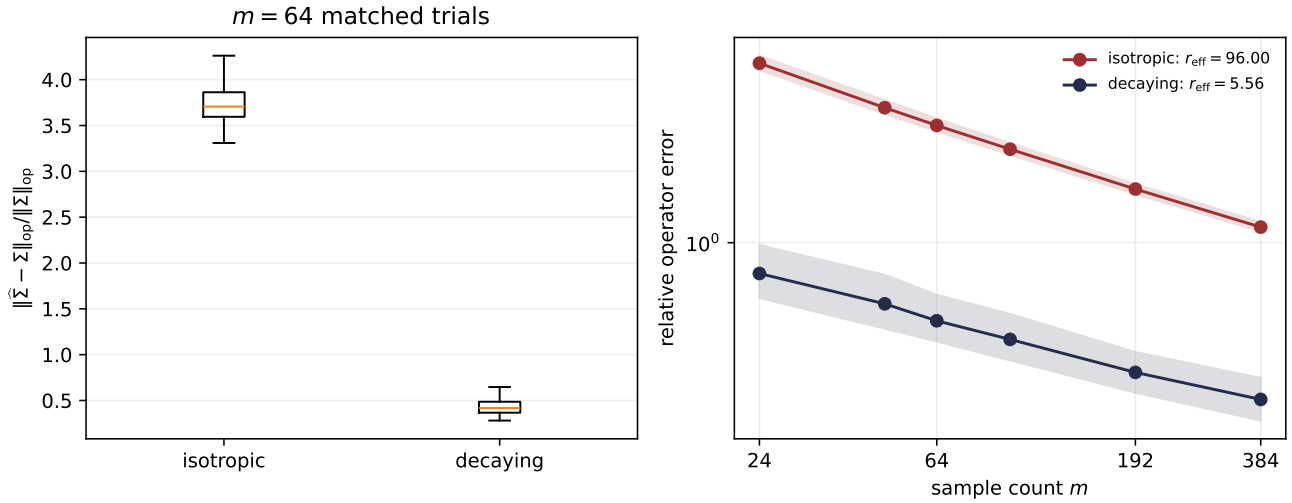


Figure 10.1.: Rank and top eigenvalue do not determine covariance-estimation difficulty. Both 96-dimensional populations are full rank with operator norm one. Their effective ranks are 96 and about 5.56. Across 300 matched Gaussian trials, the decaying-spectrum covariance has far smaller relative operator-norm error.

```
harness=ch-10 wheel=bc9dbad6f5e1
isotropic: rank=96, operator_norm=1.0, effective_rank=96.000000, median_error_m64=3.706284
decaying: rank=96, operator_norm=1.0, effective_rank=5.555556, median_error_m64=0.417820
```

The comparison does not say that decay makes every statistical problem easy. It isolates one claim: for operator-norm covariance estimation under this model, the spectral mass below the top eigenvalue changes the relevant dimension.

10.2. Replace the coordinate count by spectral mass

For a positive-semidefinite matrix $\mathbf{\Sigma} \neq \mathbf{0}$, define its **effective rank**

$$r_{\mathrm{eff}}(\mathbf{\Sigma}) = \frac{\operatorname{tr}(\mathbf{\Sigma})}{\|\mathbf{\Sigma}\|_{\mathrm{op}}}. \tag{10.3}$$

If the eigenvalues satisfy $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$, then

$$r_{\mathrm{eff}}(\mathbf{\Sigma}) = \frac{\sum_{j=1}^{n} \lambda_j}{\lambda_1}.$$

It is the total spectral mass measured in units of the largest eigenvalue. The isotropic covariance spends one full unit in every direction, giving $r_{\mathrm{eff}} = 96$. The decaying spectrum spends $\sum_{j=0}^{95} 0.82^j \approx 5.56$ such units.

Theorem 10.1 (Range and invariances of effective rank)**.** *For every nonzero positive-semidefinite* $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$*,*

$$1 \le r_{\text{eff}}(\mathbf{\Sigma}) \le \operatorname{rank}(\mathbf{\Sigma}) \le n. \tag{10.4}$$

Moreover, effective rank is unchanged by positive scalar rescaling and by orthogonal change of coordinates.

10.2.1. Proof

Let the positive eigenvalues be $\lambda_1 \ge \cdots \ge \lambda_r > 0$. Because the trace includes λ_1,

$$\frac{\sum_{j=1}^{r} \lambda_j}{\lambda_1} \ge 1.$$

Because every positive eigenvalue is at most λ_1,

$$\frac{\sum_{j=1}^{r} \lambda_j}{\lambda_1} \le r.$$

Multiplying $\mathbf{\Sigma}$ by $c > 0$ multiplies numerator and denominator by c. Orthogonal similarity preserves the eigenvalues, hence both trace and operator norm. □

Algebraic rank changes only when an eigenvalue becomes exactly zero. Effective rank changes continuously as spectral mass moves.

> ⚠ Named wrong answer: effective rank is stable rank
>
> The notions are related, but they apply to different objects. For a general matrix $\mathbf{A}$, stable rank is $\|\mathbf{A}\|_{\text{F}}^2 / \|\mathbf{A}\|_{\text{op}}^2$. For a covariance $\mathbf{\Sigma}$, that expression is $\sum_j \lambda_j^2/\lambda_1^2$, not $\sum_j \lambda_j/\lambda_1$.

Theorem 10.2 (Stable rank is effective rank after squaring)**.** *For every nonzero real matrix* $\mathbf{A}$*,*

$$\operatorname{sr}(\mathbf{A}) := \frac{\|\mathbf{A}\|_{\text{F}}^2}{\|\mathbf{A}\|_{\text{op}}^2} = r_{\text{eff}}(\mathbf{A}^\top \mathbf{A}). \tag{10.5}$$

10.2.2. Proof

The trace of $\mathbf{A}^\top\mathbf{A}$ is $\|\mathbf{A}\|_{\text{F}}^2$, while its operator norm is $\|\mathbf{A}\|_{\text{op}}^2$. Substitute these identities into Equation 10.3. □

This identity is the safe bridge between the terms. It does not license swapping them on the same matrix.

10.3. Why this quantity enters covariance error

Write the estimation error as an average of centered random matrices:

$$\widehat{\mathbf{\Sigma}} - \mathbf{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{x}_i\mathbf{x}_i^{\mathsf{T}} - \mathbf{\Sigma}\right). \tag{10.6}$$

C06 warned that squaring a sub-Gaussian quantity changes the concentration class. Covariance estimation is exactly that operation, simultaneously over all directions. C07 supplied the finite approximation of the sphere, and C08 identified the operator norm as the worst directional error. Effective rank records the matrix variance scale left after those steps.

Theorem 10.3 (Gaussian covariance summand)**.** *If* $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$*, then*

$$\mathbb{E}\left[\left(\mathbf{x}\mathbf{x}^{\mathsf{T}} - \mathbf{\Sigma}\right)^2\right] = \operatorname{tr}(\mathbf{\Sigma})\mathbf{\Sigma} + \mathbf{\Sigma}^2. \tag{10.7}$$

10.3.1. Proof

Work in an eigenbasis and write $x_j = \sqrt{\lambda_j}g_j$ with independent standard-normal g_j. The (j,k) entry of $\mathbb{E}[\mathbf{x}\mathbf{x}^{\mathsf{T}}\mathbf{x}\mathbf{x}^{\mathsf{T}}]$ is $\mathbb{E}[x_j x_k \sum_\ell x_\ell^2]$. For $j \neq k$, symmetry makes this zero. For $j = k$, Gaussian fourth moments give

$$\mathbb{E}\left[x_j^2\sum_\ell x_\ell^2\right] = 3\lambda_j^2 + \sum_{\ell\neq j}\lambda_j\lambda_\ell = \lambda_j\operatorname{tr}(\mathbf{\Sigma}) + 2\lambda_j^2.$$

Thus $\mathbb{E}[(\mathbf{x}\mathbf{x}^{\mathsf{T}})^2] = \operatorname{tr}(\mathbf{\Sigma})\mathbf{\Sigma} + 2\mathbf{\Sigma}^2$. Expanding the centered square subtracts one copy of $\mathbf{\Sigma}^2$. □

Consequently,

$$\left\|\mathbb{E}\left[(\mathbf{x}\mathbf{x}^{\mathsf{T}} - \mathbf{\Sigma})^2\right]\right\|_{\mathrm{op}} = \|\mathbf{\Sigma}\|_{\mathrm{op}}^2\left(r_{\mathrm{eff}}(\mathbf{\Sigma}) + 1\right).$$

The coordinate dimension has disappeared from this variance expression. It re-enters when the spectrum actually distributes mass across that many directions.

10.4. A usable covariance theorem

Theorem 10.4 (Effective-rank covariance bound)**.** *Let* $\mathbf{x}_1, \dots, \mathbf{x}_m$ *be independent copies of a mean-zero random vector with covariance* $\mathbf{\Sigma}$*. Assume that for every* $\mathbf{u}$*,*

$$\|\langle \mathbf{x}, \mathbf{u}\rangle\|_{\psi_2} \le K\sqrt{\mathbf{u}^{\mathsf{T}}\mathbf{\Sigma}\mathbf{u}}. \tag{10.8}$$

Then a universal constant C *exists such that*

$$\mathbb{E}\left\|\widehat{\mathbf{\Sigma}} - \mathbf{\Sigma}\right\|_{\mathrm{op}} \le CK^4\|\mathbf{\Sigma}\|_{\mathrm{op}}\left[\sqrt{\frac{r_{\mathrm{eff}}(\mathbf{\Sigma})}{m}} + \frac{r_{\mathrm{eff}}(\mathbf{\Sigma})}{m}\right]. \tag{10.9}$$

10.4.1. Proof with pointer

The proof symmetrizes the empirical process, controls a quadratic process indexed by the sphere, and converts its complexity to effective rank. That chain requires machinery beyond this chapter's diagnostic target. See Vershynin's second-edition Theorem 9.2.2 (Vershynin 2026) for a nonasymptotic derivation and Koltchinskii and Lounici (2017) for sharp effective-rank formulations.

10.4.2. Diagnostic proof sketch

The pointer hides technical machinery, not the causal route. First, symmetrization replaces the centered empirical error by a signed quadratic process:

$$\mathbb{E}\left\|\widehat{\boldsymbol{\Sigma}}-\boldsymbol{\Sigma}\right\|_{\mathrm{op}} \leq \frac{2}{m}\mathbb{E}\sup_{\|\mathbf{u}\|=1}\left|\sum_{i=1}^{m}\varepsilon_i\langle\mathbf{x}_i,\mathbf{u}\rangle^2\right|, \tag{10.10}$$

where the signs ε_i are independent of the observations. Second, C06's square warning identifies the fixed-$\mathbf{u}$ object as sub-exponential rather than sub-Gaussian. Third, C07's finite-cover move turns fixed directions into a uniform operator statement.

A crude Euclidean net pays ambient dimension and loses the point of this chapter. The refined argument measures the ellipsoid $\boldsymbol{\Sigma}^{1/2}S^{n-1}$ in its covariance metric. Its squared average radius is $\mathrm{tr}(\boldsymbol{\Sigma})$ and its largest squared radius is $\|\boldsymbol{\Sigma}\|_{\mathrm{op}}$; their ratio is exactly r_{eff}. Chaining or a matrix-deviation theorem carries that anisotropic geometry through the supremum and produces the $\sqrt{r_{\mathrm{eff}}/m}+r_{\mathrm{eff}}/m$ scale. The pointer supplies that final uniform-process step; the sketch explains why effective rank, and not algebraic rank alone, is the complexity that enters.

The usable scaling is $m \gg r_{\mathrm{eff}}(\boldsymbol{\Sigma})$. Dimension has not been abolished; it has been replaced by a spectrum-sensitive complexity under explicit tail assumptions.

> Field note: one scalar is not a compression budget
>
> Effective rank does not say how many coordinates may be deleted at a chosen error tolerance. It does not identify important eigenvectors, certify a low-rank approximation, or explain why a trained model generalizes. Those questions require information this scalar discards.

10.5. Five dimensions that must not be conflated

Quantity	Definition or question	What it diagnoses
parameter count	how many stored scalar coordinates?	storage and nominal model size
algebraic rank	how many singular values are nonzero?	exact image dimension
effective rank	$\mathrm{tr}(\Sigma)/\|\Sigma\|_{\mathrm{op}}$	covariance complexity relative to its top scale
stable rank	$\|A\|_{\mathrm{F}}^2/\|A\|_{\mathrm{op}}^2$	squared singular mass relative to the top singular value

Quantity	Definition or question	What it diagnoses
truncation rank	how many directions meet a declared tolerance?	task-dependent compression

The table prevents a rhetorical slide: observing a small effective rank and then speaking as though a compression guarantee or generalization theorem had been proved.

10.6. The Act I handoff

Act I began with a safe zone for one random projection. It followed squares into a two-regime tail, paid for a finite cover, measured both singular edges, calibrated an empirical spectral bulk, and replaced nominal dimension by a spectrum-sensitive complexity.

The next act changes the moving object. Instead of asking what a random matrix looks like, we ask how derivatives move backward through a computation and what state the machine must retain to make that motion possible.

> **Check yourself**
>
> A covariance has eigenvalues $(1, 1/2, 1/4, 1/8)$. Compute its algebraic rank, effective rank, and stable rank when the covariance itself is treated as the matrix. Then compute the stable rank of a square root $\mathbf{A}$ satisfying $\mathbf{A}^\top \mathbf{A} = \mathbf{\Sigma}$. Which two quantities agree?

10.7. Okay, so —

- **Inherited:** operator norm asks for the worst direction; empirical spectra require a finite null; squares leave the sub-Gaussian safe zone.
- **Changed:** nominal coordinate count became total spectral mass measured in top-eigenvalue units.
- **Instrumented:** matched Gaussian trials separated two full-rank covariances with the same operator norm.
- **Established:** effective rank controls the diagnostic sample scale for covariance estimation under a relative sub-Gaussian contract.
- **Unresolved:** How does derivative information traverse a computation when a scalar spectrum summary cannot say which paths matter?

10.8. Sources and further reading

The effective/stable-rank identity is recorded in Vershynin's second-edition Remark 5.6.4, and the covariance theorem is Theorem 9.2.2 (Vershynin 2026). Sharp expectation and high-probability bounds are developed by Koltchinskii and Lounici (2017). The warning that effective dimension alone does not imply generalization is consistent with the interpolation evidence of Zhang et al. (2017).

For the general theory, use Vershynin (2026) and Koltchinskii and Lounici (2017); this chapter's contribution is the full-rank/few-directions witness and the separation of five dimension summaries.

Reading order. Start with Vershynin (2026) for the book's exact notation and the covariance theorem, then read Koltchinskii and Lounici (2017) for sharper bounds and Zhang et al. (2017) for the boundary on generalization claims.

10.9. Exercises

1. **(Pencil.)** Prove that $r_{\text{eff}}(\mathbf{\Sigma}) = n$ for a nonzero $n \times n$ positive-semidefinite matrix if and only if all eigenvalues are equal.
2. **(Code.)** Replace 0.82 in Equation 10.2 by values from 0.5 to 0.99. Plot median relative operator error against effective rank at a fixed sample count. Keep the matched-randomness contract.
3. **(Audit.)** A paper reports "the representation is effectively ten-dimensional" from a scalar effective-rank estimate. List three claims that do not follow and name the missing evidence for each.
4. **(Audit.)** **Paper audit:** Complete the **Mathematical object**, **Evidence culture and interface**, **Assumption stress test**, **Discriminating control**, and **Transfer verdict** fields of the Paper Autopsy Protocol. Identify the matrix, centering and scaling convention, effective-rank definition, and target conclusion. Does the cited theorem support that conclusion?
5. **(Pencil.)** Construct two six-dimensional spectra with the same operator norm and effective rank but different truncation ranks at relative tolerance 0.1. State which covariance claim remains comparable and which compression claim changes.
6. **(Audit.)** **Act checkpoint — extend the Incident Card.** Complete the Act I assignment. Add a named matrix, finite-null or perturbation control, and a "detects only" option to the Act 0 card. **Route:** Act checkpoint. **Estimated time:** 75 minutes. **Deliverable:** the revised card and one spectrum plot whose matrix is named in the caption. **Hint:** plotting the wrong matrix can produce a true spectrum and a false diagnosis.

Act II — Dynamics of optimization

Act I built the arena's instruments: a tail class that survives a union bound, the price of a continuum, two singular edges, a calibrated bulk, and a dimension that counts spectral mass instead of coordinates. What it cannot do is move. Every object so far was measured at a point chosen in advance. Training chooses its points as it goes: the derivative traverses a graph, the estimator is drawn mid-flight, and the landscape is evaluated exactly where the trajectory drags it. Act II turns the dynamic lens forward. The geometry does not retire; it becomes the control group.

11. The Gradient Has a Memory: Reverse Accumulation and Checkpointing

NS · Numerical stability Geometric quiet · Dynamic supporting · Algorithmic primary

Take the scalar computation

$$a = xy, \qquad b = a + x, \qquad c = ay, \qquad L = bc \tag{11.1}$$

at $x = 2$ and $y = 3$. The intermediate a feeds two children. On the reverse sweep, one path contributes 18 to the cotangent of a and the other contributes 24. The correct value is their sum, 42.

An implementation that stores only the last contribution returns

$$\frac{\partial L}{\partial x} = 90, \qquad \frac{\partial L}{\partial y} = 96,$$

instead of the correct $(144, 132)$. Every local derivative can be correct while the total gradient is wrong.

> **!** Prediction
>
> What must a reverse derivative engine remember: graph structure, numerical values, path contributions, or all three?

11.1. One node, two debts

The sibling volume owns the mechanics of the chain rule and the construction of a tiny reverse derivative engine. Its backpropagation chapter is the recap if those mechanics are unfamiliar. This chapter owns a different question: once the computation is a directed acyclic graph, what does a correct reverse sweep accumulate, retain, and recompute?

1. Load the chapter-pinned reverse-accumulation instruments.
2. Encode the fan-out computation as a scalar tape.
3. Run a correct additive reverse accumulation.
4. Run the named wrong implementation that overwrites cotangents.
5. Inspect the two path contributions into the shared node.
6. Compare the resulting input derivatives.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import TapeNode, reverse_accumulate

# [2]
x, y = 2.0, 3.0
tape = [
    TapeNode(x),
    TapeNode(y),
    TapeNode(x * y, ((0, y), (1, x))),
    TapeNode(x * y + x, ((2, 1.0), (0, 1.0))),
    TapeNode(x * y * y, ((2, y), (1, x * y))),
    TapeNode(
        (x * y + x) * (x * y * y),
        ((3, x * y * y), (4, x * y + x)),
    ),
]

# [3]
correct = reverse_accumulate(tape)

# [4]
def overwrite_shared_node(nodes, shared_index, reverse_order):
    cotangents = np.zeros(len(nodes), dtype=np.float64)
    cotangents[-1] = 1.0
    for node_index in reverse_order:
        incoming = cotangents[node_index]
        for parent_index, local_derivative in nodes[node_index].parents:
            contribution = incoming * local_derivative
            if parent_index == shared_index:
                cotangents[parent_index] = contribution
            else:
                cotangents[parent_index] += contribution
    return cotangents

overwrite = overwrite_shared_node(
    tape,
    shared_index=2,
    reverse_order=(5, 3, 4, 2, 1, 0),
)

# [5]
contributions = [correct[3], correct[4] * y]
assert np.allclose(contributions, [18.0, 24.0])
assert np.allclose(correct[:2], [144.0, 132.0])
```

```
# [6]
labels = ["x", "y"]
positions = np.arange(len(labels))
width = 0.36
fig, axis = plt.subplots(figsize=(6.6, 3.6))
axis.bar(
    positions - width / 2,
    correct[:2],
    width,
    label="add contributions",
    color="#232D4B",
)
axis.bar(
    positions + width / 2,
    overwrite[:2],
    width,
    label="last writer wins",
    color="#9C2F2F",
)
axis.set(
    xticks=positions,
    xticklabels=labels,
    ylabel="computed input derivative",
)
axis.grid(axis="y", alpha=0.22)
axis.legend(frameon=False)
plt.tight_layout()
plt.show()

print(f"harness={manifest['harness_ref']} wheel={manifest['wheel_sha256'][:12]}")
print(f"contributions into a={contributions}; accumulated={correct[2]}")
print(f"correct: dx={correct[0]:.1f}, dy={correct[1]:.1f}")
print(f"overwrite: dx={overwrite[0]:.1f}, dy={overwrite[1]:.1f}")
```

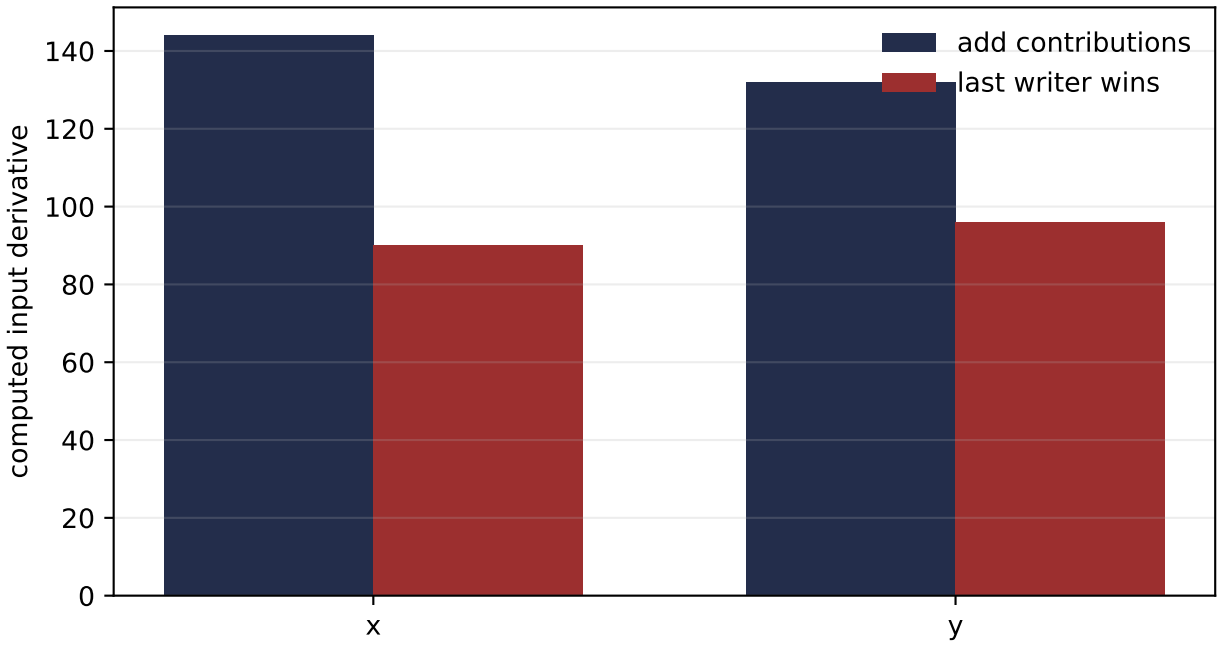


Figure 11.1.: A reverse sweep must add all downstream contributions at a fan-out node. Overwriting the cotangent of the shared intermediate a loses one path and produces incorrect derivatives even though every local derivative rule is correct.

```
harness=ch-11 wheel=c9e645efbb2e
contributions into a=[np.float64(18.0), np.float64(24.0)]; accumulated=42.0
correct: dx=144.0, dy=132.0
overwrite: dx=90.0, dy=96.0
```

The word **backward** can hide this obligation. A graph is not necessarily a chain. Reverse accumulation solves a conservation problem: every downstream use creates a derivative debt to its parent, and all debts must be paid before that parent propagates further.

The defective sweep visits the incomparable siblings b and c in that order, so the c path becomes the last writer at a. Reversing the sibling order produces a different wrong gradient. The bug is therefore sensitive to an implementation detail the derivative should not depend on.

11.2. One derivative, two orientations

Let $F : \mathbb{R}^n \to \mathbb{R}^m$ and let $\mathbf{J}_F(\mathbf{x}) \in \mathbb{R}^{m \times n}$ be its Jacobian. Two matrix-vector products expose two computational orientations:

$$\underbrace{\mathbf{J}_F(\mathbf{x})\mathbf{v}}_{\text{JVP}} \in \mathbb{R}^m, \qquad \underbrace{\mathbf{u}^{\mathsf{T}}\mathbf{J}_F(\mathbf{x})}_{\text{VJP}} \in \mathbb{R}^n. \tag{11.2}$$

A Jacobian–vector product propagates an input perturbation forward. A vector–Jacobian product propagates an output cotangent backward.

Theorem 11.1 (Sweep counts follow Jacobian orientation)**.** *Suppose one forward-mode sweep computes* $\mathbf{J}_F(\mathbf{x})\mathbf{v}$ *and one reverse-mode sweep computes* $\mathbf{u}^\mathsf{T}\mathbf{J}_F(\mathbf{x})$*. Reconstructing the full Jacobian from basis directions requires* n *forward sweeps or* m *reverse sweeps. For scalar output,* $m = 1$*, one reverse sweep returns the full gradient.*

11.2.1. Proof

Applying a JVP to the input basis vector $\mathbf{e}_j$ returns column j of the Jacobian, so all n columns require n sweeps. Applying a VJP to the output basis vector $\mathbf{e}_i$ returns row i, so all m rows require m sweeps. When $m = 1$, the single row is $\nabla F(\mathbf{x})^\mathsf{T}$. □

This is an orientation result, not the slogan "reverse is always better." When inputs are few and outputs are many, forward mode can be the cheaper choice. Scalar training objectives create the opposite geometry.

11.3. The reverse rule on a graph

Let node v store a value z_v. Each edge $v \to w$ carries the local derivative $\partial z_w/\partial z_v$. For scalar output L, define the cotangent

$$\bar{z}_v := \frac{\partial L}{\partial z_v}.$$

The graph chain rule is

$$\bar{z}_v = \sum_{w \in \text{children}(v)} \bar{z}_w \frac{\partial z_w}{\partial z_v}. \tag{11.3}$$

Theorem 11.2 (Reverse accumulation on a directed acyclic graph)**.** *Process the nodes of a scalar-output directed acyclic computation in reverse topological order. Initialize the output cotangent to one and every other cotangent to zero. For every edge* $v \to w$*, add*

$$\bar{z}_w \frac{\partial z_w}{\partial z_v}$$

to $\bar{z}_v$*. After the sweep,* $\bar{z}_v = \partial L/\partial z_v$ *at every node.*

11.3.1. Proof

Proceed by reverse topological induction. The output node is correct by initialization. Suppose every child w of v already holds $\bar{z}_w = \partial L/\partial z_w$. Every directed path from v to the output begins with exactly one edge $v \to w$. Grouping the ordinary multivariable chain rule by that first edge gives

$$\frac{\partial L}{\partial z_v} = \sum_w \frac{\partial L}{\partial z_w} \frac{\partial z_w}{\partial z_v}.$$

The additive updates compute precisely this sum. Reverse topological order ensures that all child cotangents are final before v is propagated. □

> ⚠ Named wrong answer: last writer wins
>
> Assignment is valid only when a node has one downstream path. At fan-out, the graph chain rule is a sum. An overwrite implementation can pass every chain-shaped test and fail on the first branch.

11.4. Instructions are cheap; saved values are not

A reverse rule often needs primal values from the forward execution. Multiplication needs the other operand. A nonlinear primitive may need its input, its output, or auxiliary statistics. The reverse program therefore has two distinct kinds of memory:

Retained object	Why it exists	Typical scale
graph or tape instructions	identify parents and local reverse rules	number of executed primitives
primal states	evaluate those reverse rules later	activation volume
cotangent buffers	accumulate downstream contributions	live differentiated values
stateful metadata	replay randomness, normalization state, or control flow	operation dependent

The tape can be compact while the saved numerical states dominate memory. This is why "reverse mode has constant-factor arithmetic overhead" does not mean "reverse mode is memory-free." The algebraic complexity result of Baur and Strassen (1983) and the implementation survey of Baydin et al. (2018) answer the former question; training systems must also answer the latter.

11.5. Trade retained states for replay

Consider a depth-L chain. Storing every intermediate requires $L + 1$ primal states. A simple uniform checkpoint schedule chooses a block length k: retain the input to every block during the forward pass, then replay each block during the reverse pass.

Theorem 11.3 (Uniform checkpoint count)**.** *If k divides L, a uniform block schedule can execute reverse accumulation with peak primal-state count*

$$M(k) = \frac{L}{k} + k \tag{11.4}$$

and exactly

$$R(k) = L - \frac{L}{k} \tag{11.5}$$

additional forward primitive evaluations. The continuous minimizer of $M(k)$ is $k = \sqrt{L}$, giving peak state $2\sqrt{L}$.

11.5.1. Proof

The forward pass retains one state at each of the L/k block boundaries. During reverse execution of one block, at most k within-block states are materialized, so the peak is their sum. Within each block, the boundary state already exists and the remaining $k-1$ forward transitions must be replayed. Across L/k blocks, this costs $(L/k)(k-1) = L - L/k$. Finally,

$$\frac{L}{k} + k \geq 2\sqrt{L}$$

by the arithmetic–geometric mean inequality, with equality at $k = \sqrt{L}$. □

The theorem is a count model. It does not claim that every state has equal size, every primitive has equal cost, or the measured wall-clock optimum is exactly $\sqrt{L}$.

1. Reuse the verified C11 harness.
2. Enumerate all integer block lengths for a depth-64 chain.
3. Compute peak retained states and additional forward work.
4. Mark the square-root schedule.
5. Compare it with the store-all baseline.

```
from trainable_harness import uniform_checkpoint_cost

# [1]
assert manifest["harness_ref"] == "ch-11"

# [2]
depth = 64
block_lengths = np.arange(1, depth + 1)

# [3]
costs = [uniform_checkpoint_cost(depth, int(k)) for k in block_lengths]
peak_states = np.array([cost.peak_state_units for cost in costs])
extra_forwards = np.array([
    cost.recomputed_forward_evaluations for cost in costs
])

# [4]
chosen = uniform_checkpoint_cost(depth, 8)
assert chosen.peak_state_units == 16
assert chosen.recomputed_forward_evaluations == 56

# [5]
fig, axis = plt.subplots(figsize=(7.2, 3.8))
axis.plot(block_lengths, peak_states, color="#232D4B", label="peak states")
axis.plot(
```

```
    block_lengths, extra_forwards,
    color="#E57200", label="extra forward evaluations",
)
axis.scatter([8], [16], color="#9C2F2F", zorder=3, label="$k=\sqrt{64}$")
axis.axhline(65, color="#6b6b6b", linestyle="--", label="store-all states")
axis.set(xlabel="block length $k$", ylabel="count")
axis.grid(alpha=0.22)
axis.legend(frameon=False, fontsize=8)
plt.tight_layout()
plt.show()

print(
    f"depth={depth}; block={chosen.block_size}; "
    f"peak_states={chosen.peak_state_units}; "
    f"extra_forwards={chosen.recomputed_forward_evaluations}; "
    f"store_all={depth + 1}"
)
```

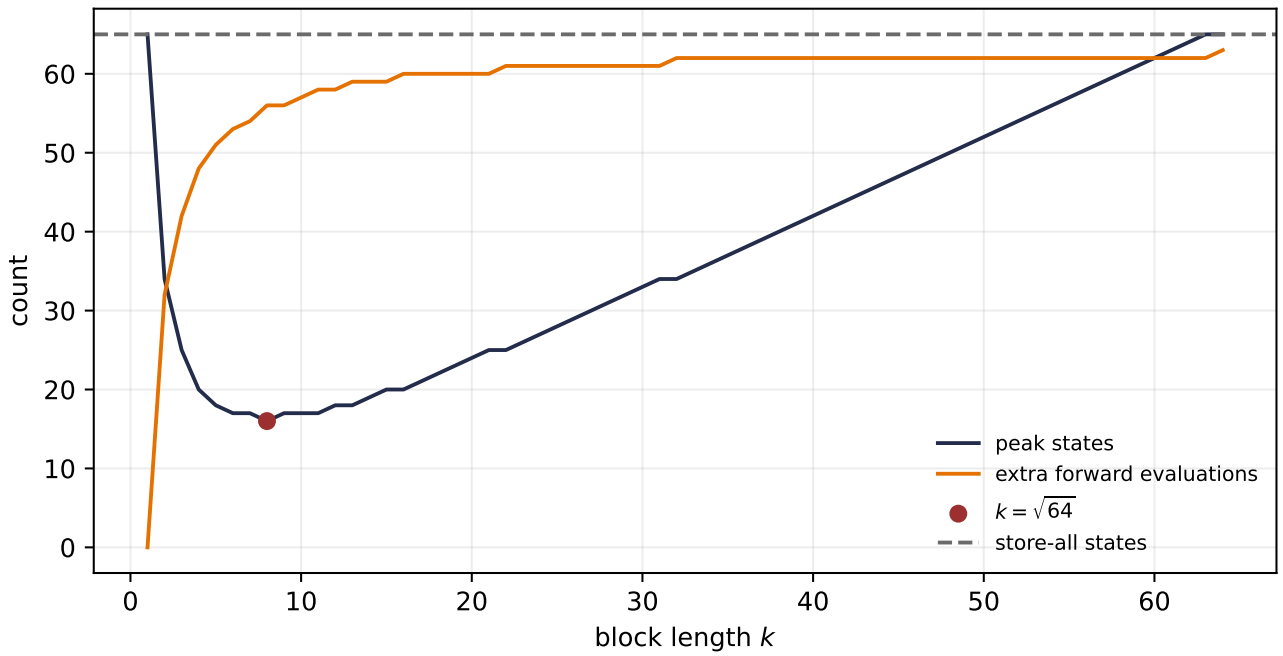


Figure 11.2.: Uniform checkpointing trades retained primal states for forward replay in a depth-64 chain. The square-root block length k=8 reduces the count-model peak from 65 store-all states to 16 states while adding 56 forward primitive evaluations.

```
depth=64; block=8; peak_states=16; extra_forwards=56; store_all=65
```

The square-root checkpoint schedule originates in reverse-AD memory analysis by Griewank (1992) and is developed systematically by Griewank and Walther (2008). Chen et al. (2016) reintroduced the trade in the deep-learning systems setting. More elaborate schedules improve it for particular graphs and budgets. The invariant remains: discarded primal state must be reconstructed somehow.

11.6. Replay is part of the numerical contract

Recomputation sounds exact only in real arithmetic and for pure functions. On a machine, a checkpoint boundary must declare:

- the floating-point format and accumulation mode used on replay;
- random-number state for stochastic primitives;
- mutable running statistics and other stateful updates;
- control-flow decisions whose predicates depend on rounded values.

Reverse accumulation avoids the perturb-and-subtract cancellation of finite differences, but it does not escape floating-point reduction. Cotangents from many children are still summed, and their order can change the result. The reduction-order lesson from C02 has returned inside the gradient engine.

> Field note: a gradient-memory claim
>
> Ask four questions. What values are retained? What values are replayed? What state makes replay deterministic? Is the reported saving a count model, allocated bytes, or measured peak device memory?

Three tempting escape routes do not solve the problem:

- Numerical differentiation estimates derivatives by perturbation; it is not exact and pays cancellation plus repeated evaluations.
- Symbolic differentiation can preserve graph reuse; expression explosion is not inevitable if common subexpressions remain shared.
- Reverse accumulation is not always the cheapest orientation; its advantage follows from the input/output dimensions of the derivative query.

11.7. The next diagnostic object

The reverse sweep applies transposed local Jacobians repeatedly. That tells us how a gradient is assembled, but not how the gradient changes when parameters move. C12 will differentiate the gradient action itself, using Hessian–vector products to probe curvature without materializing a dense Hessian. Later, the same product-of-local-Jacobians view will diagnose why depth can preserve, shrink, or amplify signals.

> Check yourself
>
> Suppose $F : \mathbb{R}^{1000} \to \mathbb{R}^4$. How many basis sweeps are needed to materialize its Jacobian using forward mode and reverse mode? If the objective is the sum of the four outputs, how many reverse sweeps are needed for its gradient with respect to the input?

11.8. Okay, so —

- **Inherited:** the local chain rule and primitive reverse rules belong to the sibling volume.
- **Changed:** a chain-rule story became a graph accumulation and liveness problem.

- **Instrumented:** a one-line overwrite defect lost one of two cotangent paths; a depth-64 schedule exposed the state–replay trade.
- **Established:** reverse mode is efficient for scalar outputs because of Jacobian orientation, and correct DAG execution requires additive cotangents plus retained or recomputable primal state.
- **Unresolved:** Which curvature object should be probed before a spectrum is interpreted, and can its action be computed without materializing it?

11.9. Sources and further reading

The first published reverse-accumulation account is Linnainmaa (1970); the modern taxonomy and implementation vocabulary follow Baydin et al. (2018). The algebraic complexity boundary for derivative evaluation is due to Baur and Strassen (1983). The square-root checkpoint trade originates with Griewank (1992) and is treated in book form by Griewank and Walther (2008); Chen et al. (2016) is the deep-learning-era rediscovery and systems translation.

Reading order. Start with Baydin et al. (2018) for the JVP/VJP taxonomy, use Griewank and Walther (2008) for checkpointing and priority, then separate Baur and Strassen (1983)'s arithmetic result from Chen et al. (2016)'s systems setting.

11.10. Exercises

1. **(Pencil.)** Add a third child $d = a^2$ to Equation 11.1 and replace the objective by $L = bc + d$. Compute all three contributions into $\bar{a}$ before computing the input gradients.
2. **(Code.)** Implement a finite-difference check of the opening gradient in binary32. Sweep the perturbation over powers of two and separate truncation error from cancellation.
3. **(Code.)** For depths from 16 to 4096, find the best integer uniform block length under Equation 11.4. Compare it with $\sqrt{L}$ and report both peak states and replay counts.
4. **(Audit.)** **Paper audit:** Complete the **Mathematical object**, **Evidence culture and interface**, **Numerical and hardware contract**, **Discriminating control**, and **Transfer verdict** fields of the Paper Autopsy Protocol for one memory-efficient training paper. Mark which graph class it assumes, whether memory means saved activations or measured device allocation, whether stochastic replay is specified, and whether arithmetic count is confused with wall-clock speed.
5. **(Code.)** Implement replay for a chain containing one stateful random operation. Compare saved-state and recomputed derivatives under matched and unmatched random streams. Report arithmetic work, peak state, and the first derivative mismatch separately.

12. Curvature Is Not One Matrix: Hessian, Generalized Gauss–Newton, Fisher, and Matrix-Free Products

RS · Random-matrix spectra · LG · Landscape and update geometry Geometric primary · Dynamic supporting · Algorithmic supporting

Take one Bernoulli observation with label $y = 1$. Let its logit be the composed map

$$z(w) = w^2, \qquad \ell(w) = \log(1 + e^{z(w)}) - yz(w). \tag{12.1}$$

At $w = 1/2$, the predicted probability is $p = \sigma(1/4) = 0.5621765009$. Four quantities that are routinely called "curvature" give

$$\begin{aligned} H &= -0.6295129155, & G &= 0.2461340827, \\ F &= 0.2461340827, & \widetilde{F} &= 0.1916894164. \end{aligned} \tag{12.2}$$

The first object says the realized loss bends downward. The other three are nonnegative. Two agree exactly; the third does not. No floating-point format, batch, parameter, or label changed. Only the mathematical question changed.

> **!** Prediction
>
> Which term makes H negative? Why do G and F agree here? What distribution would have to change before $\widetilde{F}$ became an empirical estimate of F?

12.1. Hold the problem fixed; change the object

The one-observation arithmetic is short enough to expose. For Bernoulli negative log likelihood,

$$\frac{\partial \ell}{\partial z} = p - y, \qquad \frac{\partial^2 \ell}{\partial z^2} = p(1-p). \tag{12.3}$$

Because $z'(w) = 2w$ and $z''(w) = 2$,

$$\frac{d^2\ell}{dw^2} = \underbrace{p(1-p)[z'(w)]^2}_{G} + \underbrace{(p-y)z''(w)}_{\text{model-curvature term}}. \tag{12.4}$$

The first term is 0.2461340827. The second is −0.8756469982. Their sum is the negative Hessian in Equation 12.2. The Fisher and empirical-Fisher values require different averaging contracts; they arrive in Section 12.3.

The next witness adds an intercept and four observations. It compares an affine logit $z = ax + b$ with a composed logit $z = a^2x + b$ while holding the parameter, labels, and Bernoulli loss fixed.

1. Load the chapter-pinned curvature instruments.
2. Fix four inputs, four labels, and one parameter vector.
3. Compute the realized Hessian, GGN, model Fisher, and empirical Fisher.
4. Repeat for affine and composed logits.
5. Compare eigenvalues rather than collapsing each matrix to one norm.
6. Check the committed claim artifact before interpreting the plot.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]

from trainable_harness import bernoulli_curvature_matrices

# [2]
inputs = np.array([-2.0, -1.0, 1.0, 2.0])
labels = np.array([0.0, 0.0, 1.0, 1.0])
parameters = np.array([0.5, 0.0])

# [3]
def evaluate(nonlinear):
    return bernoulli_curvature_matrices(
        parameters,
        inputs,
        labels,
        nonlinear_first_parameter=nonlinear,
    )

affine = evaluate(False)

# [4]
composed = evaluate(True)

# [5]
objects = ("Hessian", "GGN = model Fisher", "Empirical Fisher")
fig, axes = plt.subplots(1, 2, figsize=(9.0, 3.5), sharey=True)
for axis, title, result in zip(
    axes,
    ("Affine logit", "Composed logit"),
    (affine, composed),
    strict=True,
):
    spectra = (
```

```
        np.linalg.eigvalsh(result.hessian),
        np.linalg.eigvalsh(result.generalized_gauss_newton),
        np.linalg.eigvalsh(result.empirical_fisher),
    )
    positions = np.arange(2)
    for offset, (name, spectrum) in enumerate(zip(objects, spectra, strict=True)):
        axis.bar(positions + 0.24 * (offset - 1), spectrum, 0.22, label=name)
    axis.axhline(0.0, color="black", linewidth=0.8)
    axis.set_xticks(positions, (r"$\lambda_1$", r"$\lambda_2$"))
    axis.set_title(title)
    axis.set_xlabel("ordered eigenvalue")
axes[0].set_ylabel("eigenvalue")
axes[1].legend(frameon=False, fontsize=8)
fig.tight_layout()

# [6]
claim = verify_claim("c12-curvature-choice-001", expected_harness=pin)
assert np.allclose(
    claim["result"]["composed_logit"]["hessian_eigenvalues"],
    np.linalg.eigvalsh(composed.hessian),
    atol=1e-12,
    rtol=0.0,
)
print("claim c12-curvature-choice-001: verified")
```

```
claim c12-curvature-choice-001: verified
```

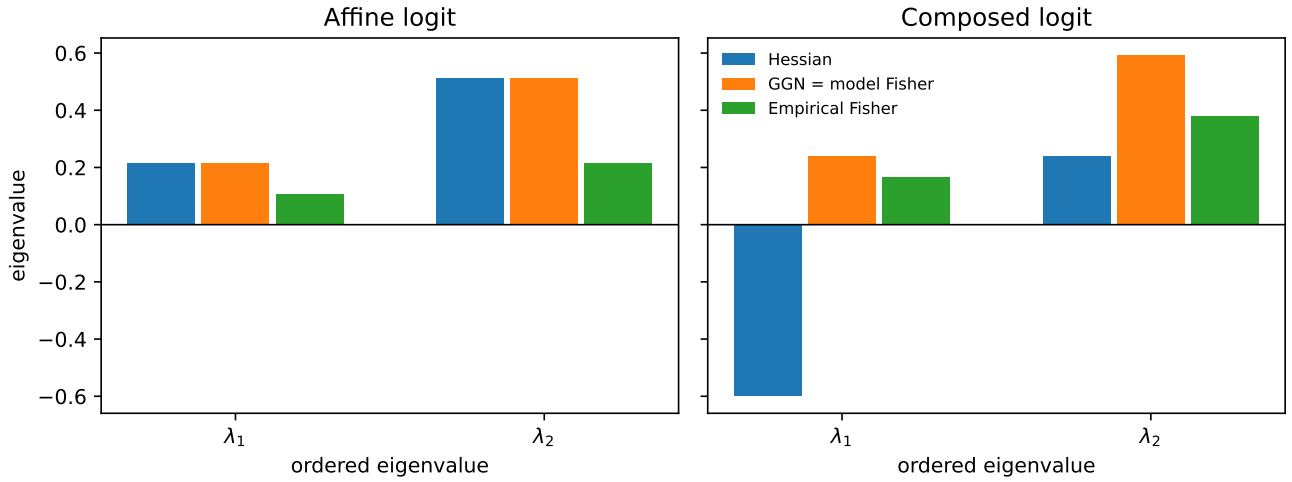


Figure 12.1.: The same Bernoulli loss produces different curvature spectra. With an affine logit, the realized Hessian, GGN, and model Fisher coincide. Curving the parameter-to-logit map adds an indefinite model-curvature term; the empirical Fisher remains a distinct observed-label outer product in both controls.

The affine control removes every Hessian of the parameter-to-logit map. The realized Hessian then

equals GGN. The composed control restores one omitted term and one Hessian eigenvalue crosses zero. This is not a contest in which one matrix is universally correct. Each answers a different question.

12.2. The term a positive-semidefinite surrogate drops

Let $f_{\boldsymbol{\theta}} \in \mathbb{R}^q$ be the model output, let $\ell(y, \mathbf{z})$ be a local loss, and define $\mathbf{J} = \partial f_{\boldsymbol{\theta}} / \partial \boldsymbol{\theta}$. Dimensions are $\mathbf{J} \in \mathbb{R}^{q \times p}$ and $\mathbf{H}_{\mathbf{z}}\ell \in \mathbb{R}^{q \times q}$.

Theorem 12.1 (Composite Hessian decomposition)**.** *If $f_{\boldsymbol{\theta}}$ and ℓ are twice differentiable, then*

$$\nabla^2_{\boldsymbol{\theta}} \ell = \underbrace{\mathbf{J}^{\mathsf{T}} \mathbf{H}_{\mathbf{z}} \ell \, \mathbf{J}}_{\mathbf{G}:\ \textit{generalized Gauss–Newton}} + \underbrace{\sum_{j=1}^{q} \frac{\partial \ell}{\partial z_j} \nabla^2_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}, j}}_{\mathbf{R}:\ \textit{model curvature}} . \tag{12.5}$$

If ℓ is convex in $\mathbf{z}$, then $\mathbf{G}$ is positive semidefinite. The residual term $\mathbf{R}$ can have either sign.

12.2.1. Proof

The gradient is $\nabla_{\boldsymbol{\theta}} \ell = \mathbf{J}^{\mathsf{T}} \nabla_{\mathbf{z}} \ell$. Differentiate its r-th coordinate with respect to θ_s. Differentiating $\nabla_{\mathbf{z}} \ell$ produces $[\mathbf{J}^{\mathsf{T}} \mathbf{H}_{\mathbf{z}} \ell \mathbf{J}]_{rs}$; differentiating the entries of $\mathbf{J}$ produces $\sum_j (\partial \ell / \partial z_j) \partial^2 f_j / (\partial \theta_r \partial \theta_s)$. That is Equation 12.5. If $\mathbf{H}_{\mathbf{z}} \ell \succeq 0$, then $\mathbf{v}^{\mathsf{T}} \mathbf{G} \mathbf{v} = (\mathbf{J}\mathbf{v})^{\mathsf{T}} \mathbf{H}_{\mathbf{z}} \ell (\mathbf{J}\mathbf{v}) \geq 0$.

The theorem explains both panels. In the affine model, $\nabla^2_{\boldsymbol{\theta}} f_j = \mathbf{0}$ and $\mathbf{R} = \mathbf{0}$. In the composed model, the observed residual multiplies the curvature of $a^2 x + b$.

> ⚠ Named wrong answer: GGN is the Hessian made safe
>
> GGN is not the Hessian with negative eigenvalues clipped. It is the Hessian of a locally linearized model composed with the declared output loss. It can omit positive as well as negative model curvature, and it changes if the boundary between model and output loss changes.

12.3. Three expectations hiding behind two words

For a conditional statistical model $p_{\boldsymbol{\theta}}(y \mid x)$, define the score $\mathbf{s}_{\boldsymbol{\theta}}(y, x) = \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(y \mid x)$. Fixing the observed inputs, the **model Fisher** is

$$\mathbf{F}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{Y \sim p_{\boldsymbol{\theta}}(\cdot \mid x_i)} \left[\mathbf{s}_{\boldsymbol{\theta}}(Y, x_i) \mathbf{s}_{\boldsymbol{\theta}}(Y, x_i)^{\mathsf{T}} \right] . \tag{12.6}$$

The expectation resamples the label from the current model. The empirical Fisher used in the opening instead keeps the observed label:

$$\widetilde{\mathbf{F}}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{s}_{\boldsymbol{\theta}}(y_i, x_i) \mathbf{s}_{\boldsymbol{\theta}}(y_i, x_i)^{\mathsf{T}} . \tag{12.7}$$

Both are positive semidefinite sums of outer products. That shared algebra does not make their averaging distributions the same.

Theorem 12.2 (Score identity and qualified GGN equality)**.** *Suppose the support of $p_{\boldsymbol{\theta}}(y \mid x)$ does not depend on $\boldsymbol{\theta}$ and differentiation may pass through its integral or sum. Then*

$$\mathbb{E}[\mathbf{s}_{\boldsymbol{\theta}}] = \mathbf{0}, \qquad \mathbb{E}[\mathbf{s}_{\boldsymbol{\theta}}\mathbf{s}_{\boldsymbol{\theta}}^{\mathsf{T}}] = -\mathbb{E}[\nabla^2_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}]. \tag{12.8}$$

When the output variable in the GGN is a compatible natural parameter for the negative log likelihood, the model Fisher equals that GGN.

12.3.1. Proof of the score identity

Differentiate normalization: $\int p_{\boldsymbol{\theta}}(y \mid x)\, dy = 1$. The first derivative gives $\mathbb{E}[\mathbf{s}_{\boldsymbol{\theta}}] = \mathbf{0}$. Differentiating the score mean once more gives

$$\mathbf{0} = \mathbb{E}[\nabla^2_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}] + \mathbb{E}[\mathbf{s}_{\boldsymbol{\theta}}\mathbf{s}_{\boldsymbol{\theta}}^{\mathsf{T}}],$$

which proves Equation 12.8. For an exponential-family likelihood in its natural output parameter, the conditional score covariance is the output negative-log-likelihood Hessian. Pulling it through $\mathbf{J}$ gives $\mathbf{J}^{\mathsf{T}}\mathbf{H}_{\mathbf{z}}\ell\mathbf{J}$; see Martens (2020) for the precise qualified equivalence.

For the Bernoulli logit, the model averages $(Y - p)^2$ under $Y \sim \text{Bernoulli}(p)$, producing $p(1-p)$. The observed-label outer product uses $(y - p)^2$. At $p = 0.5622$ and $y = 1$, those are different numbers. The adjective "empirical" does not repair the change of measure (Kunstner et al. 2019).

Object	What is averaged?	Definite?	What it can diagnose
Realized Hessian $\mathbf{H}$	derivatives of the observed objective	indefinite	local Taylor curvature of that objective
GGN $\mathbf{G}$	observed examples; linearized model, curved output loss	PSD under output convexity	curvature retained by the declared split
Model Fisher $\mathbf{F}$	model-generated labels at observed inputs	PSD	local statistical-model geometry
Empirical Fisher $\widetilde{\mathbf{F}}$	observed-label score outer products	PSD	gradient outer-product geometry; not generally $\mathbf{F}$

Field note: name the expectation

A paper that writes "the Fisher" without naming what is random has left the operator underspecified. Ask whether the expectation is over model-generated labels, observed examples, inputs from a population, or some combination. Then ask which of those expectations the code actually estimates.

12.4. Ask for an action, not a dense matrix

A dense Hessian of a model with p parameters contains p^2 entries. Most curvature algorithms do not need those entries. They need the action $\mathbf{Hv}$ on a chosen direction.

Theorem 12.3 (Hessian–vector products)**.** *Let $L : \mathbb{R}^p \to \mathbb{R}$ be twice differentiable and let $\mathbf{v} \in \mathbb{R}^p$ be independent of $\boldsymbol{\theta}$. Then*

$$\mathbf{H}(\boldsymbol{\theta})\mathbf{v} = \nabla_{\boldsymbol{\theta}} \left[\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})^{\mathsf{T}} \mathbf{v} \right]. \tag{12.9}$$

The right-hand side can be evaluated by composing derivative programs without forming the dense Hessian.

12.4.1. Proof

The scalar inside the gradient is $\sum_j (\partial L/\partial \theta_j) v_j$. Its ith derivative is $\sum_j (\partial^2 L/\partial \theta_i \partial \theta_j) v_j$, the ith entry of $\mathbf{Hv}$. The identity is exact. Its implementation cost depends on the derivative primitives, graph, and retained-state policy; it is not a universal timing constant (Pearlmutter 1994).

The opening model is only two-dimensional, so we can use its dense Hessian as a control while the iteration sees only an action function.

1. Reuse the verified composed-logit witness.
2. Define a function that accepts a vector and returns $\mathbf{Hv}$.
3. Check one action against dense multiplication.
4. Run seeded power iteration through the action interface.
5. Compare the signed estimate with the dense eigendecomposition.
6. Verify the second committed claim artifact.

```
from trainable_harness import (
    bernoulli_hessian_vector_product,
    dominant_symmetric_eigenpair,
)

# [1]
assert manifest["harness_ref"] == "ch-12"

# [2]
def hessian_action(vector):
    return bernoulli_hessian_vector_product(
        parameters,
        inputs,
        labels,
        vector,
        nonlinear_first_parameter=True,
    )

# [3]
probe = np.array([0.3, -0.7])
```

```
dense_product = composed.hessian @ probe
operator_product = hessian_action(probe)
assert np.allclose(dense_product, operator_product, atol=1e-12, rtol=0.0)

# [4]
estimate = dominant_symmetric_eigenpair(
    hessian_action,
    dimension=2,
    iterations=24,
    seed=6212,
)

# [5]
dense_eigenvalues = np.linalg.eigvalsh(composed.hessian)
dominant = dense_eigenvalues[np.argmax(np.abs(dense_eigenvalues))]
assert np.isclose(estimate.eigenvalue, dominant, atol=1e-8, rtol=0.0)
iterations = np.arange(1, len(estimate.rayleigh_trace) + 1)
fig, axis = plt.subplots(figsize=(7.2, 3.6))
axis.plot(iterations, estimate.rayleigh_trace, color="#232D4B")
axis.axhline(
    dominant,
    color="#9C2F2F",
    linestyle="--",
    label=f"dense control: {dominant:.4f}",
)
axis.set(
    xlabel="power iteration",
    ylabel="Rayleigh quotient",
    title="The dominant-magnitude curvature direction is negative",
)
axis.grid(alpha=0.22)
axis.legend(frameon=False)
fig.tight_layout()

# [6]
hvp_claim = verify_claim("c12-hvp-001", expected_harness=pin)
assert np.allclose(
    hvp_claim["result"]["operator_product"],
    operator_product,
    atol=1e-12,
    rtol=0.0,
)
assert np.isclose(
    hvp_claim["result"]["dominant_magnitude_eigenvalue_estimate"],
    estimate.eigenvalue,
    atol=1e-12,
    rtol=0.0,
)
print(
```

```
    "claim c12-hvp-001: verified; "
    f"eigenvalue={estimate.eigenvalue:.10f}; "
    f"residual={estimate.residual_norm:.3e}"
)
```

```
claim c12-hvp-001: verified; eigenvalue=-0.5998303709; residual=2.807e-10
```

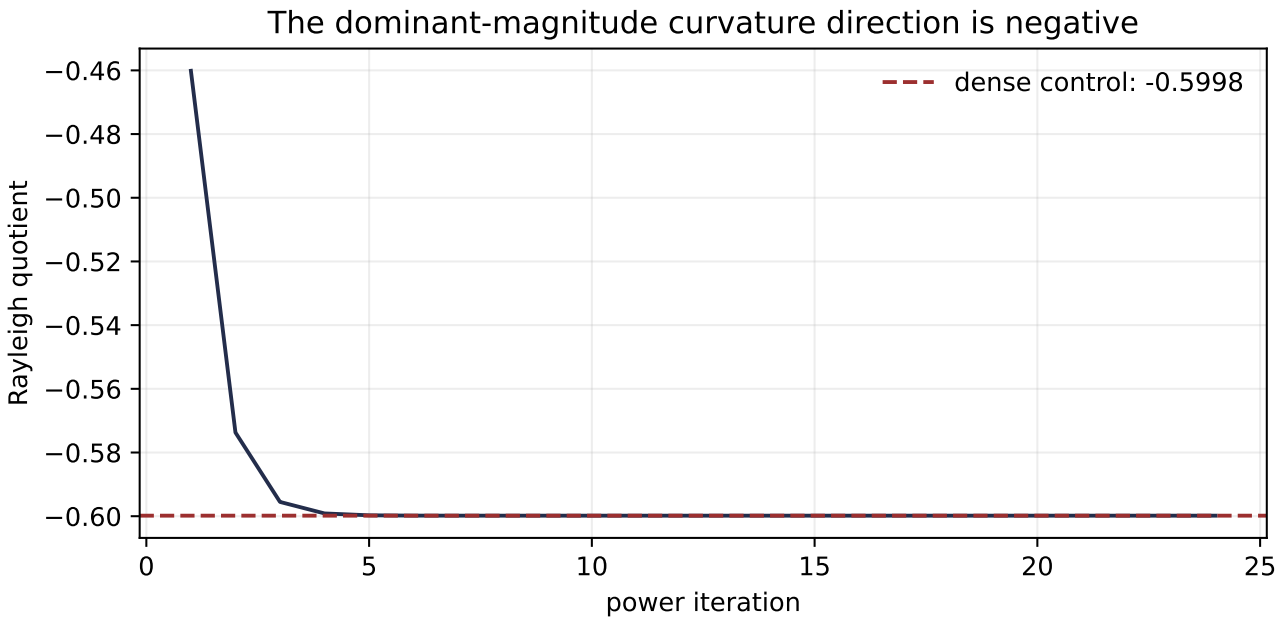


Figure 12.2.: Matrix-free power iteration approaches the Hessian eigenvalue of largest magnitude. The estimate is negative because magnitude, not algebraic order, governs ordinary power iteration.

> ⚠ Named wrong answer: power iteration finds the largest eigenvalue
>
> Ordinary power iteration targets the eigenvalue of largest **magnitude** when one is uniquely dominant. It does not automatically target the largest algebraic eigenvalue. Here that distinction is the difference between reporting negative curvature and erasing it.

The point of the operator interface is larger than this tiny witness. Lanczos iteration, conjugate gradients, trace estimators, and several second-order diagnostics can be written in terms of matrix–vector products. The matrix is then defined by what action the program computes. That makes the choice among $\mathbf{H}$, $\mathbf{G}$, $\mathbf{F}$, and $\widetilde{\mathbf{F}}$ an executable API decision.

12.5. Object, representation, and solver are separate choices

Three decisions are often compressed into the phrase “use second order”:

Decision	Examples	Question it answers
curvature object	$\mathbf{H}$, $\mathbf{G}$, $\mathbf{F}$, $\widetilde{\mathbf{F}}$	What local geometry is being represented?
representation	exact action, diagonal, blocks, low rank, structured factorization	What information is retained and affordable?
solver or use	power iteration, conjugate gradients, damping, preconditioning	What action is taken with that representation?

A diagonal Hessian and a diagonal Fisher differ before either becomes diagonal. A low-rank approximation can be applied to an indefinite Hessian or a positive-semidefinite GGN. Damping changes the system being solved; it does not retroactively make an empirical Fisher into a model Fisher.

This separation is also a hardware boundary. A matrix-free action avoids storing p^2 entries, but it can still reread activations, replay derivative graphs, and perform reductions. "Matrix free" is a storage statement until a measurement establishes more. CPU timing here would not establish accelerator performance.

Representation can also change the numerical problem before a solver takes a step. Let $\mathbf{J} \in \mathbb{R}^{m \times p}$ have full column rank, with singular values $\sigma_1 \geq \cdots \geq \sigma_p > 0$. The eigenvalues of $\mathbf{J}^\mathsf{T}\mathbf{J}$ are σ_i^2, so

$$\kappa_2(\mathbf{J}^\mathsf{T}\mathbf{J}) = \frac{\sigma_1^2}{\sigma_p^2} = \kappa_2(\mathbf{J})^2. \tag{12.10}$$

Forming normal equations therefore squares the spectral condition number and can spend roughly twice the digits that the original least-squares operator made vulnerable. This is the C03 precision contract applied to a curvature representation, not a reason to declare one solver universally superior. QR avoids explicitly squaring the operator; LSQR uses products with $\mathbf{J}$ and $\mathbf{J}^\mathsf{T}$; matrix-free curvature products avoid forming the dense Gram matrix. Each alternative still owes a cost, stopping, precision, and residual contract. Damping replaces the system by a different one rather than undoing the squaring (Higham 2002).

12.6. One linear solver, two lenses

Suppose a positive-definite curvature operator $\mathbf{A}$ defines the system $\mathbf{Ax} = \mathbf{b}$. Conjugate gradients can be read dynamically as a sequence of mutually $\mathbf{A}$-conjugate search directions. It can also be read inferentially: under a Gaussian prior and exact linear observations, the same algebra updates a distribution over the unknown solution (Hennig 2015). These are two interpretations of one computation, not evidence that every iterative solver is Bayesian.

The seam matters because curvature estimation is itself an information problem. Secant pairs reveal actions of an unknown Hessian. Probabilistic quasi-Newton work asks what posterior over a matrix is compatible with those observations (Hennig and Kiefel 2013). This chapter keeps that as a branch from the trunk: students need the operator/observation distinction to read such work, but not the full probabilistic-numerics apparatus to diagnose the opening failure.

Field note: a curvature card

Before accepting a curvature claim, record five fields: the mathematical object; every averaging distribution; the representation or approximation; the matrix action actually implemented; and the solver, damping, or preconditioning policy that consumes it.

Check yourself

For an affine logit with Bernoulli negative log likelihood, which term in Equation 12.5 vanishes? Does that equality make the empirical Fisher equal to the model Fisher for an arbitrary observed label? If a power iteration returns a negative Rayleigh quotient, what ordering of eigenvalues must you check before declaring the implementation wrong?

12.7. Okay, so —

- **Inherited:** reverse accumulation supplies derivative programs, while Act I supplies the language for spectra and operator probes.
- **Changed:** "the curvature" became a declared object with an averaging distribution, representation, and consumer.
- **Instrumented:** the same loss and data produced an indefinite Hessian, a positive-semidefinite GGN/model Fisher, and a different empirical Fisher; an action-only power iteration recovered the signed dominant-magnitude mode.
- **Established:** the Hessian splits into GGN plus model curvature; the Fisher identity requires a model expectation and regularity; dense Hessians are unnecessary when the diagnostic needs only $\mathbf{Hv}$; forming $\mathbf{J}^\mathsf{T}\mathbf{J}$ squares $\mathbf{J}$'s spectral condition number when $\mathbf{J}$ has full column rank.
- **Unresolved:** Even with the curvature object named, when does a stochastic gradient estimator predict descent rather than noise-dominated motion?

12.8. Sources and further reading

The matrix-free Hessian–vector identity and its implementation boundaries follow Pearlmutter (1994). The distinctions among the Hessian, generalized Gauss–Newton matrix, Fisher information, and empirical Fisher are checked against Martens (2020) and Kunstner et al. (2019). The inferential interpretation of conjugate gradients follows Hennig (2015); the quasi-Newton branch follows Hennig and Kiefel (2013). The normal-equations precision boundary follows the singular-value calculation on the page and the numerical linear algebra contract in Higham (2002). The ICML probabilistic-numerics tutorial supplied an intellectual seam, not the technical authority for these claims.

Reading order. Start with Martens (2020) and Kunstner et al. (2019) to name the curvature object, use Pearlmutter (1994) for action-only computation, and then read Hennig (2015) or Hennig and Kiefel (2013) for the bounded inferential branches.

12.9. Exercises

1. **(Pencil.)** Starting from Equation 12.1, derive all four numbers in Equation 12.2. Then replace $y = 1$ with $y = 0$ without changing w. Which quantities change, and which remain fixed?
2. **(Pencil.)** Give a scalar example in which the model-curvature term $\mathbf{R}$ in Equation 12.5 is positive. Explain why this disproves the claim that GGN merely deletes negative Hessian curvature.
3. **(Code.)** Replace the two-parameter witness by a small nonlinear model. Implement Hessian and GGN actions, then use the same Lanczos or power iteration driver on both. Report the signed extremal Ritz values and an operator-residual check.
4. **(Code.)** For Bernoulli logits, simulate labels from the current model and average observed-label score outer products. Show convergence toward the model Fisher, then repeat with labels from a misspecified distribution.
5. **(Audit.)** **Paper audit:** Select a paper that reports a Fisher, GGN, or Hessian spectrum. Complete the **Mathematical object**, **Estimator and comparison contract**, **Evidence culture and interface**, **Assumption stress test**, and **Transfer verdict** fields. Identify the expectation implemented by the code, whether the reported matrix is exact or approximated, and which conclusion would fail if the object were replaced by the empirical Fisher.

13. The Mean Gradient Is Not the Regime: Conditional Noise and Stochastic Dynamics

SG · Sub-Gaussian safe zone · LG · Landscape and update geometry Geometric supporting · Dynamic primary · Algorithmic supporting

At $w = 1$, the scalar loss $f(w) = w^2/2$ has gradient one. Take the update $w^+ = w - 0.5g$, where $g = 1 + \xi$ is conditionally unbiased. If $\mathbb{E}[\xi^2] = 0.25$, then $\mathbb{E}[f(w^+)] = 0.15625$: expected descent from the initial loss 0.5. If $\mathbb{E}[\xi^2] = 4$, then $\mathbb{E}[f(w^+)] = 0.625$: expected ascent. The current state, mean gradient, curvature, and step size are identical. Only the conditional dispersion changed.

> **!** Prediction
>
> An optimizer plot reports a mean gradient norm but not the estimator's conditional variance or tail behavior. Can it identify whether step decay or batch growth is the relevant intervention? Name one observation that would make those two interventions distinguishable.

13.1. Declare the estimator before its variance

Let $\mathcal{F}_k$ contain everything known before sampling at step k. A usable stochastic-gradient contract is

$$\mathbb{E}[\mathbf{g}_k \mid \mathcal{F}_k] = \nabla f(\mathbf{w}_k), \qquad \mathbb{E}[\|\mathbf{g}_k - \nabla f(\mathbf{w}_k)\|^2 \mid \mathcal{F}_k] \le \sigma_k^2. \tag{13.1}$$

The conditioning matters. Sampling without replacement, correlated examples, adaptive data selection, clipping, and quantization can all alter the mean or dispersion. "Gradient noise" without a filtration, averaging rule, and target is not yet a mathematical object.

Theorem 13.1 (Conditional expected descent)**.** *Suppose f is L-smooth and Equation 13.1 holds with finite conditional second moment. For $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \mathbf{g}_k$,*

$$\mathbb{E}[f(\mathbf{w}_{k+1}) \mid \mathcal{F}_k] \le f(\mathbf{w}_k) - \alpha \left(1 - \frac{L\alpha}{2}\right) \|\nabla f(\mathbf{w}_k)\|^2 + \frac{L\alpha^2}{2}\sigma_k^2. \tag{13.2}$$

13.1.1. Proof

Apply the smoothness upper bound to $-\alpha \mathbf{g}_k$, condition on $\mathcal{F}_k$, and use $\mathbb{E}\|\mathbf{g}_k\|^2 = \|\nabla f(\mathbf{w}_k)\|^2 + \mathbb{E}\|\mathbf{g}_k - \nabla f(\mathbf{w}_k)\|^2$.

When the variance bound is tight, the right side predicts descent whenever

$$\alpha < \frac{2/L}{1 + \sigma_k^2 / \|\nabla f(\mathbf{w}_k)\|^2}. \tag{13.3}$$

The ratio $\chi_k = \sigma_k^2 / \|\nabla f(\mathbf{w}_k)\|^2$ is the diagnostic axis. Small χ_k is gradient-dominated; large χ_k is noise-dominated. This is an internal regime ledger, not a fifth narrative sigil.

1. Load the chapter-pinned stochastic-regime instruments.
2. Evaluate the exact scalar quadratic expectation over a variance grid.
3. Mark the two opening estimators.
4. Compute the conditional safe-step threshold.
5. Verify the committed claim before interpreting the curve.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]
from trainable_harness import (
    centered_pareto_clip_bias,
    conditional_safe_step,
    scalar_quadratic_regime,
)

# [2]
ratios = np.linspace(0.0, 5.0, 200)
expected = [
    scalar_quadratic_regime(1.0, 0.5, ratio).expected_next_loss
    for ratio in ratios
]
# [3]
fig, axis = plt.subplots(figsize=(7.2, 3.7))
axis.plot(ratios, expected, color="#232D4B")
axis.axhline(
    0.5, color="#9C2F2F", linestyle="--", label="starting loss"
)
axis.scatter(
    [0.25, 4.0], [0.15625, 0.625], color="#E57200", zorder=3
)
axis.set(
    xlabel=r"conditional noise-to-signal ratio $\chi_k$",
    ylabel="expected next loss",
)
axis.legend(frameon=False)
fig.tight_layout()
# [4]
assert np.isclose(conditional_safe_step(1.0, 4.0, 1.0), 0.4)
# [5]
claim = verify_claim("c13-regime-001", expected_harness=pin)
assert claim["result"]["high_noise"]["expected_next_loss"] == 0.625
print("claim c13-regime-001: verified")
```

```
claim c13-regime-001: verified
```

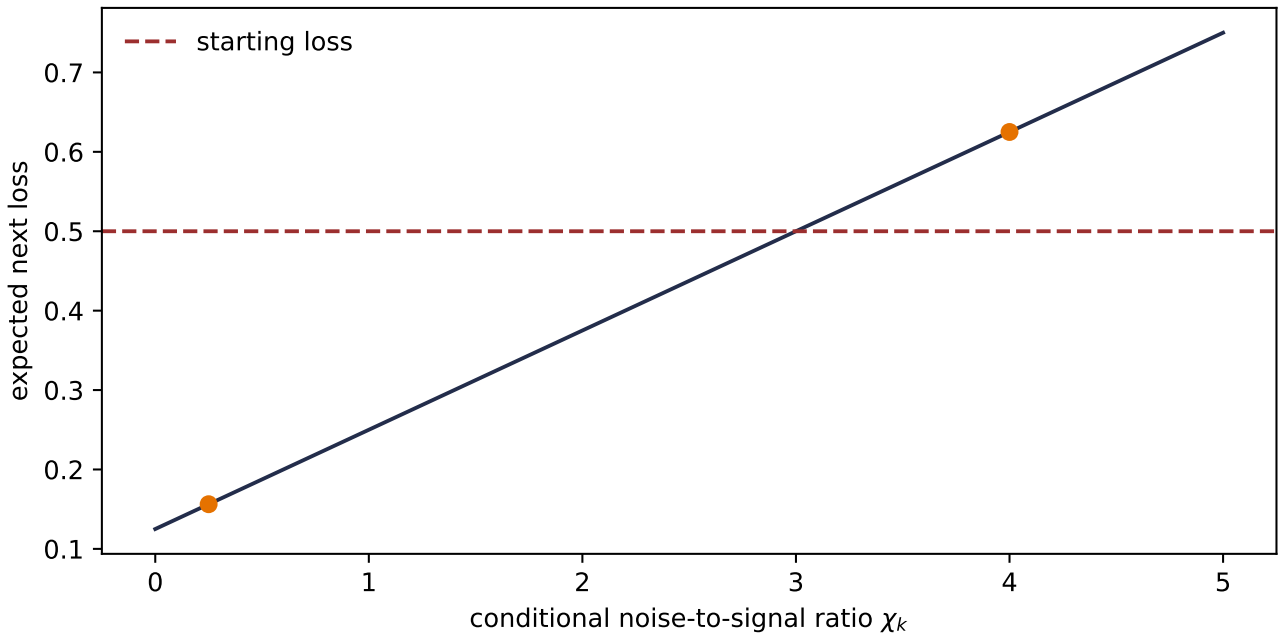


Figure 13.1.: The same mean gradient crosses from expected descent to expected ascent as conditional noise grows. The dashed line is the starting loss, not a fitted threshold.

Claim `c13-regime-001` · seed: none (exact expectation) · dtype: FP64 · device: CPU · estimator: conditional second moment · artifact

13.2. Watch the regime change while the objective stays fixed

The opening calculation compares two estimators at one state. A training run changes the denominator of χ_k: even if the noise scale stays constant, the gradient norm can shrink. To see that transition, keep the objective $f(w) = w^2/2$, nominal step 0.12, per-example noise standard deviation 0.8, and batch size 16 fixed for 80 steps. At each step, estimate the variance of the batch mean from the 16 per-example gradients and divide by the exact full-gradient norm squared.

The same executable control then asks three adjacent questions. Does batch averaging follow the promised $1/b$ law? Does momentum help the exact ill-conditioned quadratic while amplifying stochastic forcing? What remains of a variance-only diagnosis when the noise has no finite second moment?

1. Derive independent seeds from the claim identities.
2. Run one online quadratic and estimate its conditional ratio each step.
3. Verify batch-mean variance across four batch sizes.
4. Compare plain and momentum recurrences with and without shared noise.
5. Reuse C06's exact clipping bias for an infinite-variance Pareto model.
6. Plot the controls and verify the committed claim.

```
# [1]
def derived_seed(label):
    return int(sha256(label.encode()).hexdigest()[:8], 16)

# [2]
online_rng = np.random.default_rng(derived_seed("c13-online-regime"))
state, online_step, noise_sd, batch_size = 1.0, 0.12, 0.8, 16
online_losses, online_chi = [], []
for _ in range(80):
    gradients = state + online_rng.normal(scale=noise_sd, size=batch_size)
    online_losses.append(0.5 * state**2)
    online_chi.append(
        gradients.var(ddof=1) / batch_size / max(state**2, 1e-30)
    )
    state -= online_step * gradients.mean()
first_crossing = int(np.flatnonzero(np.asarray(online_chi) >= 1)[0])

# [3]
batch_sizes, trial_count = np.array([1, 4, 16, 64]), 20_000
batch_streams = np.random.SeedSequence(
    derived_seed("c13-batch-scaling")
).spawn(len(batch_sizes))
observed_batch_variances = []
for size, stream in zip(batch_sizes, batch_streams, strict=True):
    means = np.random.default_rng(stream).normal(
        scale=noise_sd, size=(trial_count, int(size))
    ).mean(axis=1)
    observed_batch_variances.append(means.var(ddof=1))
observed_batch_variances = np.asarray(observed_batch_variances)
theory_batch_variances = noise_sd**2 / batch_sizes

# [4]
curvatures = np.array([0.01, 1.0])
momentum_step, beta = 0.5, 0.9
def deterministic_trace(memory):
    point, velocity, trace = np.ones(2), np.zeros(2), []
    for _ in range(80):
        trace.append(0.5 * np.sum(curvatures * point**2))
        velocity = memory * velocity + curvatures * point
        point -= momentum_step * velocity
    return np.asarray(trace)

plain_trace = deterministic_trace(0.0)
momentum_trace = deterministic_trace(beta)
forcing_rng = np.random.default_rng(derived_seed("c13-momentum-noise"))
plain = np.ones((20_000, 2))
momentum = plain.copy()
velocity = np.zeros_like(momentum)
for _ in range(500):
```

```
    forcing = forcing_rng.normal(scale=0.2, size=plain.shape)
    plain -= momentum_step * (curvatures * plain + forcing)
    velocity = beta * velocity + curvatures * momentum + forcing
    momentum -= momentum_step * velocity
stochastic_losses = np.array([
    np.mean(0.5 * np.sum(curvatures * plain**2, axis=1)),
    np.mean(0.5 * np.sum(curvatures * momentum**2, axis=1)),
])

# [5]
pareto_bias = centered_pareto_clip_bias(1.5, [2.0, 4.0, 16.0, 64.0])

# [6]
fig, axes = plt.subplots(2, 2, figsize=(9.2, 6.2))
axes[0, 0].semilogy(online_chi, color="#232D4B")
axes[0, 0].axhline(1, color="#9C2F2F", linestyle="--")
axes[0, 0].set(xlabel="step", ylabel=r"estimated $\chi_k$")
axes[0, 1].loglog(
    batch_sizes, observed_batch_variances, "o-", color="#232D4B",
    label="observed",
)
axes[0, 1].loglog(
    batch_sizes, theory_batch_variances, "--", color="#9C2F2F",
    label=r"$\sigma^2/b$",
)
axes[0, 1].set(xlabel="batch size", ylabel="variance of batch mean")
axes[0, 1].legend(frameon=False)
axes[1, 0].semilogy(plain_trace, color="#9C2F2F", label="plain")
axes[1, 0].semilogy(momentum_trace, color="#232D4B", label="momentum")
axes[1, 0].set(xlabel="deterministic step", ylabel="quadratic loss")
axes[1, 0].legend(frameon=False)
axes[1, 1].bar(
    ["plain", "momentum"], stochastic_losses,
    color=["#9C2F2F", "#232D4B"],
)
axes[1, 1].set(ylabel="mean loss after 500 noisy steps")
for axis in axes.flat:
    axis.grid(alpha=0.2)
fig.tight_layout()

controls_claim = verify_claim("c13-regime-controls-001", expected_harness=pin)
assert first_crossing == controls_claim["result"]["online_crossing"][
    "first_chi_at_least_one"
]
assert np.allclose(
    observed_batch_variances,
    [
        controls_claim["result"]["batch_scaling"]["by_batch_size"][str(size)][
            "observed_variance"
```

```
        ]
        for size in batch_sizes
    ],
)
print(f"first chi crossing: step {first_crossing}")
print(
    "noisy loss, plain/momentum: "
    f"{stochastic_losses[0]:.4f}/{stochastic_losses[1]:.4f}"
)
print("Pareto clip bias, C=2/4/16/64: " + "/".join(
    f"{value:.3f}" for value in pareto_bias.values()
))
```

```
first chi crossing: step 11
noisy loss, plain/momentum: 0.0117/0.1077
Pareto clip bias, C=2/4/16/64: -0.894/-0.756/-0.459/-0.244
```

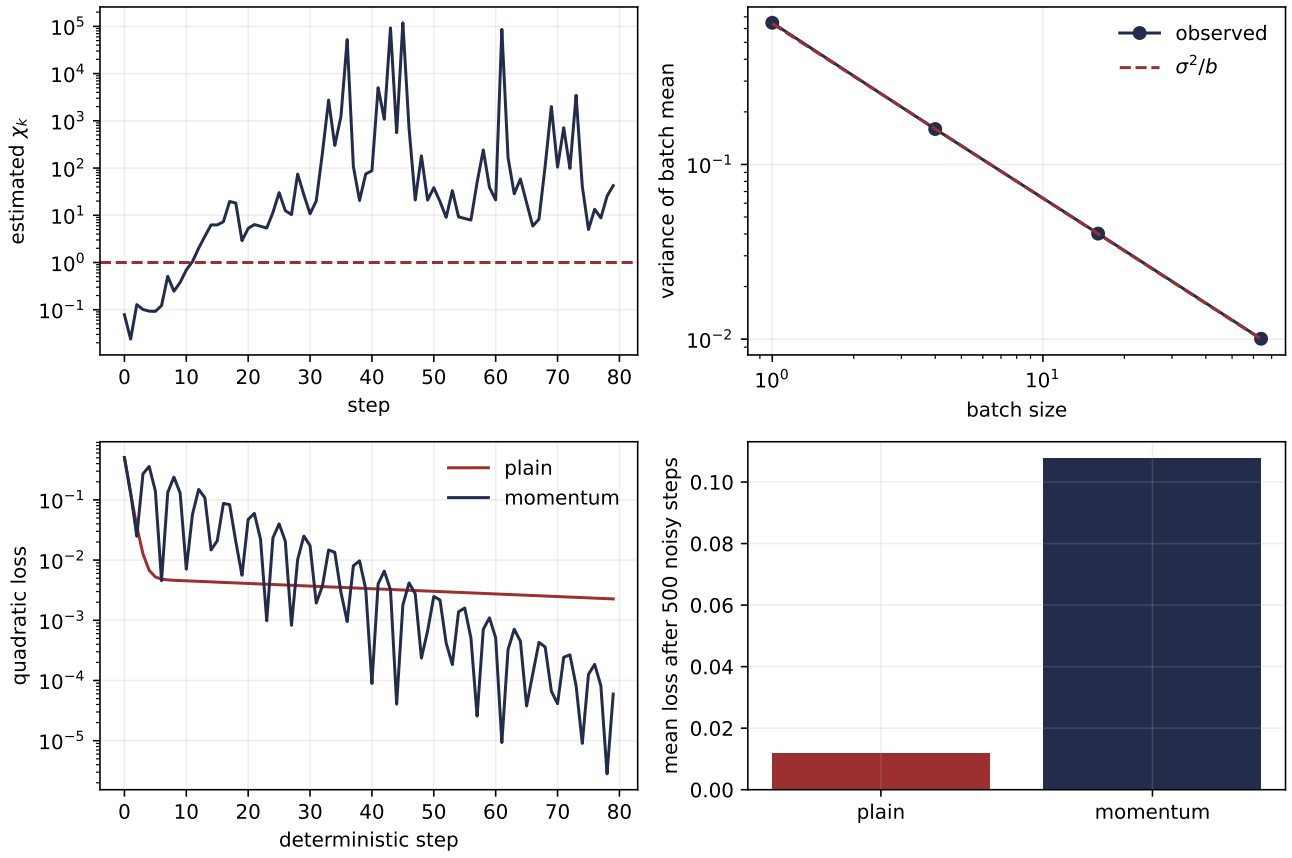


Figure 13.2.: Four controls separate four mechanisms. The online run crosses chi equals one with objective, step, noise scale, and batch fixed; independent batch means follow the one-over-b variance law; momentum improves the deterministic ill-conditioned control but amplifies shared stochastic forcing. The Pareto clipping calculation is reported separately because its variance is infinite.

The measured ratio first exceeds one at step 11. This is a transition in the diagnostic regime, not a change in the objective or nominal schedule. The batch panel follows σ^2/b because the code deliberately uses independent, identically distributed finite-variance samples. It is a control for that assumption, not evidence that real training batches obey it.

For the two-mode quadratic with curvatures 0.01 and 1, momentum uses

$$r^2 - (1 + \beta - \alpha\lambda)r + \beta = 0. \tag{13.4}$$

When $0 \le \beta < 1$, this exact scalar control is stable for $0 < \alpha\lambda < 2(1 + \beta)$. With $\alpha = 0.5$ and $\beta = 0.9$, the deterministic loss after 80 steps is about 1.20×10^{-4}, versus 2.24×10^{-3} without memory. Under shared additive forcing, however, the mean losses after 500 steps are about 0.108 and 0.0117. Memory changed the noise filter as well as the deterministic roots.

Finally, the centered Pareto model with shape 3/2 has a finite mean and infinite variance. Its χ_k is undefined, so no larger batch can be justified by a σ^2/b calculation. C06's exact clipping audit still applies: thresholds $2, 4, 16, 64$ induce mean biases about $-0.894, -0.756, -0.459, -0.244$. Clipping makes the update bounded while changing its target, and the bias disappears only as the threshold recedes.

13.3. Batch size is an assumption-bearing lever

For a mean of b conditionally independent, identically distributed per-example estimators with finite variance, the variance becomes σ_k^2/b. Then Equation 13.3 explains the linear-scaling heuristic only in the noise-dominated range: multiplying batch and step by the same factor approximately preserves the variance penalty while that factor is small enough that the smoothness term remains benign. Correlation, changing data distributions, or infinite variance break the derivation.

The inequality also explains why "decrease the step" and "increase the batch" can address the same term while having different compute and communication costs. C01's traffic contract must therefore accompany the statistical one.

13.4. Clipping repairs a tail by changing the target

Let $g = -9$ with probability 0.1 and $g = 19/9$ otherwise. Its mean is one and variance is $100/9$. Clipping to $[-2, 2]$ lowers variance to 1.44 but moves the mean to 1.6. The estimator is no longer unbiased for the original gradient. This is the C06 lesson inside an update: variance reduction and target preservation are separate audit columns.

> ⚠ Named wrong answer: clipping makes the theorem apply
>
> Clipping supplies bounded tails for the clipped estimator. It does not make that estimator unbiased for the unclipped gradient. A convergence argument must carry the bias term or redefine the objective it targets.

13.5. Acceleration is a branch, not a survey

The paired witness supplies the promised boundary: the exact quadratic recurrence can improve the deterministic slow mode while the same β amplifies stochastic forcing. Chebyshev schedules, conjugate-gradient polynomials, and other acceleration variants remain branches. The trunk question is which resolution the theory uses—global, local, direction-aware, or finite-horizon—and which regime the evidence probes.

13.6. Check yourself

With $L = 1$, gradient norm two, and conditional variance 12, what is the noise-to-signal ratio and the threshold in Equation 13.3? If four independent copies are averaged, what changes? Which two steps of that answer fail for the shape-3/2 Pareto model?

13.7. Okay, so —

- **Inherited:** C04's curvature modes and C06's tail boundary remain the controls.
- **Changed:** an update claim now begins with a conditional estimator, not the word "stochastic."
- **Instrumented:** an exact one-step claim and an online control separate regime crossing, batch scaling, momentum filtering, and clipping bias.
- **Established:** expected descent depends on both smoothness and a noise-to-signal ratio.
- **Unresolved:** Why can preserving one scalar variance through depth still destroy individual derivative directions?

13.8. Sources and further reading

The conditional descent calculation is standard in large-scale stochastic optimization; see Bottou et al. (2018) and the foundational approximation scheme of Robbins and Monro (1951). The heavy-ball control originates with Polyak (1964). The named diagonal-moment method and its decoupled-decay variant are Kingma and Ba (2015) and Loshchilov and Hutter (2019); this chapter uses them as literature bridges, not as a substitute for the estimator contract. Heavy-tailed gradient evidence is discussed by Simsekli et al. (2019). The 2026 ICML tutorial by Mark Schmidt supplied the intellectual map from theory resolution to regime diagnosis; the equations and qualifications here are derived on the page rather than attributed to the tutorial.

Reading order. Start with Bottou et al. (2018) for conditional descent and batch effects, use Polyak (1964) for the exact momentum control and Simsekli et al. (2019) to stress the finite-variance assumption, then read Schmidt (2026) last as a map of research resolutions and regimes.

13.9. Exercises

1. **(Pencil.)** Derive the exact stationary variance of scalar quadratic descent with additive zero-mean noise and compare it with Equation 13.2.

2. **(Pencil.)** Apply the Jury criterion to Equation 13.4 and derive $0 < \alpha\lambda < 2(1+\beta)$ for $0 \le \beta < 1$. State why this is an exact-quadratic result rather than a global nonlinear guarantee.
3. **(Code.)** Replace the two-point noise by both a finite-variance heavy tail and the shape-3/2 Pareto model. Across seeds and batch sizes, compare mean, median, clipped estimate, and failure of the sample-variance diagnostic.
4. **(Code.)** Reproduce the online crossing with shared random streams while varying only batch size. Report first-crossing time, estimator variance, and total examples processed; do not rank batches by step count alone.
5. **(Audit.) Paper audit:** For an optimizer paper, complete the **Mathematical object**, **Dynamic regime**, **Estimator and comparison contract**, **Assumption stress test**, **Discriminating control**, and **Transfer verdict** fields of the Paper Autopsy Protocol. Name the conditional estimator and decide whether clipping changes its target.
6. **(Audit.)** A warmup–stable–decay schedule is explained as three successive regimes. Identify the observable needed to label each transition, then give one rival explanation based on changing curvature and one based on changing estimator noise.

14. Critical on Average, Broken by Direction: Initialization and Dynamical Isometry

RS · Random-matrix spectra · SG · Sub-Gaussian safe zone · LG · Landscape and update geometry

Geometric primary · Dynamic supporting · Algorithmic supporting

Twelve layers each preserve squared norm **in expectation**. Their product destroys one direction to 9.6×10^{-13}.

The registered witness uses independent 24×24 Gaussian matrices with entry variance $1/24$. Their product has mean squared singular value 0.795 and largest singular value 3.62. An equally deep product of orthogonal factors has every singular value equal to one up to rounding. The average-moment story cannot distinguish them.

> **!** Prediction
>
> Two initializations preserve activation variance layer by layer. Must they propagate every gradient direction comparably? Choose the control you would inspect next: another activation moment, the smallest and largest singular values, or only the parameter norm. What would falsify your choice?

14.1. Criticality is a recursion with assumptions

Consider a wide coordinate

$$h_i^{(\ell+1)} = \phi\left(\sum_{j=1}^{n} W_{ij}^{(\ell+1)} h_j^{(\ell)} + b_i^{(\ell+1)}\right), \tag{14.1}$$

with independent mean-zero weights of variance σ_w^2/n. Under the usual wide, exchangeable, self-averaging approximation, a typical preactivation is approximately Gaussian and the second moment follows a scalar map.

Theorem 14.1 (Wide-model second-moment recursion)**.** *Under the independence and Gaussian-limit assumptions above,*

$$q_{\ell+1} = \sigma_w^2\, \mathbb{E}\big[\phi(\sqrt{q_\ell} Z)^2\big] + \sigma_b^2, \qquad Z \sim \mathcal{N}(0,1). \tag{14.2}$$

For zero-bias ReLU coordinates, $q_{\ell+1} = \sigma_w^2 q_\ell/2$; the variance-critical scale is $\sigma_w^2 = 2$.

14.1.1. Diagnostic proof

Condition on the previous coordinates. Independence and the $1/n$ scaling make the preactivation variance the empirical second moment times σ_w^2, plus bias variance. The wide limit replaces that empirical moment by q_ℓ. Symmetry gives $\mathbb{E}[\max(0, \sqrt{q}Z)^2] = q/2$.

The theorem identifies a boundary: scale one decays by $2^{-\ell}$, scale two is fixed, and scale three grows by $(3/2)^\ell$. It does **not** say that a finite network exactly follows the recursion, nor that scale two has a well-conditioned input-output Jacobian.

14.2. The moment survived; the distinction did not

A one-input moment cannot say whether two inputs remain distinguishable. Take equal-moment inputs and write their normalized inner product as c_ℓ. In the same wide Gaussian model, zero-bias ReLU propagation closes on a second scalar map.

Theorem 14.2 (Wide-model two-input correlation recursion)**.** *For two equal-moment inputs under the assumptions of Theorem 14.1,*

$$c_{\ell+1} = \frac{\sqrt{1 - c_\ell^2} + (\pi - \arccos c_\ell)c_\ell}{\pi}. \tag{14.3}$$

For zero bias, ReLU homogeneity cancels the weight scale from this normalized map. At the variance-critical scale $\sigma_w^2 = 2$, each input's moment stays fixed while every finite initial angle tends toward alignment. If $c_\ell = \cos\psi_\ell$ and ψ_ℓ is small, then $\psi_{\ell+1} = \psi_\ell - \psi_\ell^2/(3\pi) + O(\psi_\ell^3)$, so $\psi_\ell \asymp 1/\ell$.

14.2.1. Diagnostic derivation

The two preactivations are correlated Gaussians. Integrating the product of their positive parts over the angle between them gives Equation 14.3. Expanding $\sin\psi - \psi\cos\psi = \psi^3/3 + O(\psi^5)$ near alignment gives the stated angle recurrence. Thus local pair separation is marginal, yet finite-angle separation still decays polynomially rather than remaining fixed.

The closed form in Equation 14.3 is the degree-one arc-cosine kernel introduced by Cho and Saul (2009); Williams (1997) is the Gaussian-process covariance antecedent. Here the standard name arrives only after the normalized correlation object and its wide-model assumptions are fixed.

This is the missing contract behind the scalar critical point. The structure kept every norm and slowly lost the difference between its inputs. Changing only σ_w^2 changes the magnitude traces but not this normalized ReLU correlation trace; three weight scales do not create three pair-geometry regimes.

The forward and backward gains should also be named separately. Let $\chi_\parallel$ be the derivative of the one-input moment map at its fixed point, and let $\chi_\perp$ be the infinitesimal pair-separation gain. The second factor also appears in the mean-square backward recursion. They coincide for the scale-invariant piecewise-linear family used here, but that coincidence is a property of the family, not a general law connecting forward criticality to gradient preservation.

14.3. Width is relative to depth

The wide map now needs its own trust instrument. In the finite-width critical ReLU model, let $q_\ell = n^{-1}\sum_i (h_i^{(\ell)})^2$. Conditional on the previous layer,

$$\frac{q_{\ell+1}}{q_\ell} = R_\ell = \frac{2}{n}\sum_{i=1}^{n}(Z_i)_+^2, \qquad Z_i \overset{\text{iid}}{\sim} \mathcal{N}(0,1). \tag{14.4}$$

Theorem 14.3 (Exact finite-width moment-fluctuation control)**.** *With independent Gaussian weights of variance $2/n$ across layers, zero bias, and ReLU coordinates,*

$$\mathbb{E}\left[\frac{q_L}{q_0}\right] = 1, \qquad \operatorname{Var}\left(\frac{q_L}{q_0}\right) = \left(1+\frac{5}{n}\right)^L - 1. \tag{14.5}$$

Therefore the variance is $5L/n + O((L/n)^2)$ only while L/n is small. If $L, n \to \infty$ together with $L/n \to r$, the same control approaches $e^{5r} - 1$, not a line.

14.3.1. Proof

For $Y = (Z)_+^2$, symmetry gives $\mathbb{E}Y = 1/2$ and $\mathbb{E}Y^2 = \mathbb{E}[Z^4 \mathbf{1}_{Z>0}] = 3/2$, hence $\operatorname{Var}(Y) = 5/4$. Thus $\mathbb{E}R_\ell = 1$ and $\operatorname{Var}(R_\ell) = 5/n$. Independent layer weights make the multipliers independent, so $\mathbb{E}[(q_L/q_0)^2] = (1+5/n)^L$. Subtracting the squared mean proves the result.

1. Evaluate the wide two-input correlation control.
2. Load its seeded finite-width check.
3. Evaluate the exact depth-to-width fluctuation control.
4. Load the seeded moment-ratio checks.
5. Plot both contracts and test the declared perturbative boundary.

```
from book_support.criticality_figure import plot_pair_width_controls
from trainable_harness import (
    relu_correlation_trace,
    relu_moment_fluctuation_variance,
)

# [1]
pair_trace = relu_correlation_trace(0.5, 64)
# [2]
pair_claim = verify_claim("c14-pair-geometry-001", expected_harness=pin)
pair_final_mean = pair_claim["result"]["finite_width_correlation"]["64"]["mean"]
# [3]
exact_at_check = relu_moment_fluctuation_variance(512, 64)
# [4]
width_claim = verify_claim("c14-depth-width-001", expected_harness=pin)
points = width_claim["result"]["points"]
max_relative_deviation = max(
    abs(row["sample_variance"] / row["exact_variance"] - 1)
    for row in points
```

```
)
# [5]
assert abs(pair_final_mean - pair_trace[-1]) < 2e-3
assert exact_at_check > 5 * (64 / 512)
assert max_relative_deviation < 0.025
fig = plot_pair_width_controls(
    pair_trace, pair_claim, width_claim,
    relu_moment_fluctuation_variance,
)
print("pair correlation at depth 64: " f"{pair_trace[-1]:.6f}")
print("largest Monte Carlo relative deviation: " f"{max_relative_deviation:.3%}")
```

```
pair correlation at depth 64: 0.992491
largest Monte Carlo relative deviation: 1.756%
```

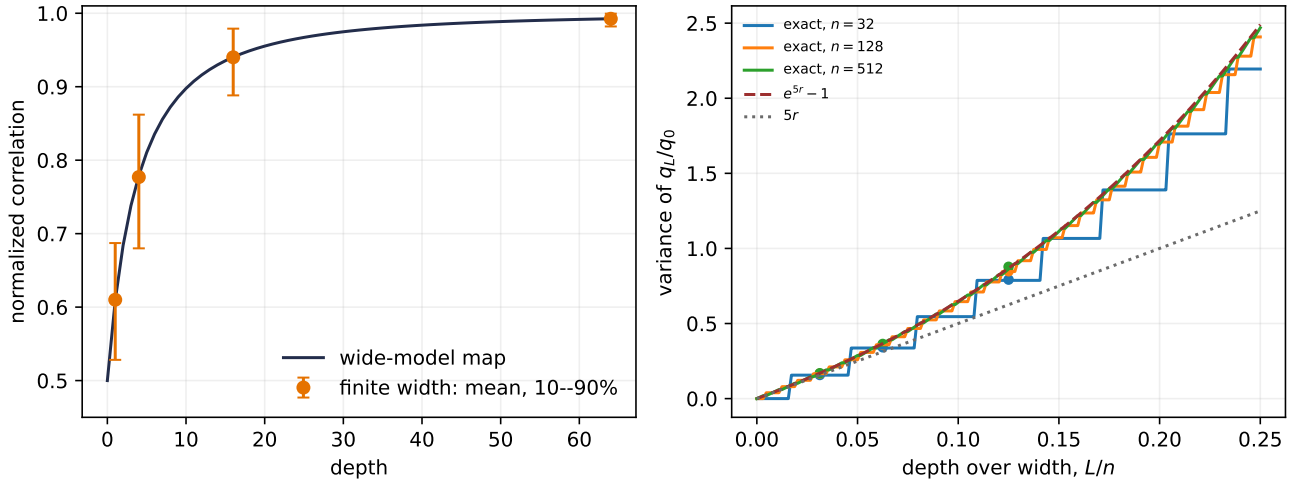


Figure 14.1.: Left: at critical ReLU scale, each input keeps its moment while two inputs align; finite-width means follow the wide correlation map. Right: the exact moment-fluctuation variance bends away from its 5L/n tangent as depth over width grows. The Monte Carlo points check the finite-width identity rather than replacing it.

Claims `c14-pair-geometry-001`, `c14-depth-width-001` · seeds: SHA-256-derived 3972769481 and 3834729383 · dtype: FP64 · device: CPU · estimators: 2,048-trial pair-correlation summaries and 200,000-trial population variances · pair artifact · depth/width artifact

The final wide-model correlation is 0.992491; the finite-width mean differs by about 2×10^{-6}. Across the nine fluctuation checks, the largest Monte Carlo deviation from the exact variance is under 1.8%. The seeded points validate the implementation. The exact curve carries the claim.

> **Field note: an approximation names its small parameter**
>
> Every usable limit in this book buys solvability with a declared small quantity: a step–curvature product, a noise-to-signal ratio, a covering radius, and now a depth-to-width ratio. The claim, retained order, and order of limits travel together; the claim dies where that contract stops being

small. The first audit question for a clean approximation is therefore: which quantity did it assume was small, and how large is that quantity in the run on your screen?

14.4. Average control does not survive multiplication

The scalar recursion is easiest to trust after carrying it twice. Set $q_0 = 1$, zero bias, and $\sigma_w^2 = 2(1+\varepsilon)$. For rectified coordinates,

$$q_1 = 1+\varepsilon, \qquad q_2 = (1+\varepsilon)^2, \qquad q_L = (1+\varepsilon)^L. \tag{14.6}$$

At $\varepsilon = 0$, the average second moment is fixed at every depth. For a small nonzero mismatch, depth compounds the local error: the relevant quantity is $L\log(1+\varepsilon)$, not ε alone.

Now compare the directional statement. The two matrices

$$\mathbf{J}_1 = \operatorname{diag}(\sqrt{2}, 0), \qquad \mathbf{J}_2 = \operatorname{diag}(0, \sqrt{2})$$

each have mean squared singular value one, yet $\mathbf{J}_2\mathbf{J}_1 = \mathbf{0}$. Two layers can therefore be mean-square critical individually while their derivative product destroys every direction. Repeating the scalar recursion to depth L controls an average under its independence model; controlling the product requires directional or spectral information at the same depth.

1. Load the chapter-pinned signal-propagation instruments.
2. Plot the ReLU moment recursion at three scales.
3. Recreate the registered Gaussian and orthogonal matrix products.
4. Insert ReLU gates from one finite-width forward pass at critical scale.
5. Compare all three spectra and verify both committed claims.

```
import matplotlib.pyplot as plt
import numpy as np
from hashlib import sha256

# [1]
from trainable_harness import deep_linear_product, relu_variance_trace

# [2]
fig, axes = plt.subplots(1, 2, figsize=(9.2, 3.6))
for scale in (1.0, 2.0, 3.0):
    axes[0].semilogy(relu_variance_trace(1.0, scale, 20), label=fr"$\sigma_w^2={scale:g}$")
axes[0].set(xlabel="depth", ylabel="second moment")
axes[0].legend(frameon=False)
# [3]
rng = np.random.default_rng(6214); n = 24; depth = 12
gaussian = [
    rng.normal(size=(n, n)) / np.sqrt(n) for _ in range(depth)
]
orthogonal = [
```

```
    np.linalg.qr(rng.normal(size=(n, n)))[0] for _ in range(depth)
]
# [4]
sg = np.linalg.svd(deep_linear_product(gaussian), compute_uv=False)
so = np.linalg.svd(deep_linear_product(orthogonal), compute_uv=False)
gated_seed = int(sha256(b"c14-gated-product").hexdigest()[:8], 16)
gated_rng = np.random.default_rng(gated_seed)
gated_state = gated_rng.normal(size=n)
gated_factors, active_counts = [], []
for _ in range(depth):
    weights = gated_rng.normal(size=(n, n)) * np.sqrt(2 / n)
    preactivation = weights @ gated_state
    gate = (preactivation > 0).astype(float)
    active_counts.append(int(gate.sum()))
    gated_factors.append(gate[:, None] * weights)
    gated_state = np.maximum(preactivation, 0)
sgated = np.linalg.svd(
    deep_linear_product(gated_factors), compute_uv=False
)
axes[1].semilogy(sg, "o-", label="Gaussian factors")
axes[1].semilogy(so, "o-", label="orthogonal factors")
axes[1].semilogy(sgated, "o-", label="ReLU-gated factors")
axes[1].set(xlabel="ordered direction", ylabel="singular value")
axes[1].legend(frameon=False)
fig.tight_layout()
# [5]
claim = verify_claim("c14-criticality-001", expected_harness=legacy_pin)
gated_claim = verify_claim("c14-gated-product-001", expected_harness=legacy_pin)
assert np.isclose(
    sg[0],
    claim["result"]["gaussian_product"]["maximum"],
    atol=1e-12,
)
assert active_counts == gated_claim["result"]["active_coordinates_by_layer"]
assert int(np.sum(sgated > 1e-10)) == gated_claim["result"][
    "jacobian_rank_tolerance_1e-10"
]
print("claim c14-criticality-001: verified")
print("claim c14-gated-product-001: verified")
```

```
claim c14-criticality-001: verified
claim c14-gated-product-001: verified
```

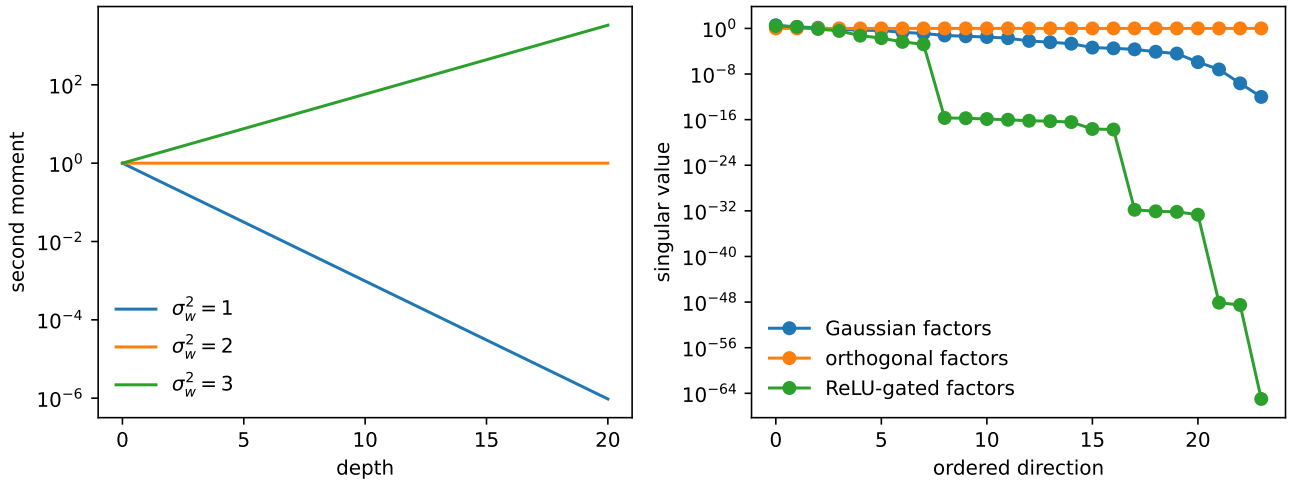


Figure 14.2.: Left: a scalar critical point separates decay from growth. Right: average preservation does not imply directional preservation; the orthogonal linear control stays isometric, while Gaussian products spread and a finite ReLU-gated product also loses rank.

Claims `c14-criticality-001`, `c14-gated-product-001` · seed: 6214 · dtype: FP64 · device: CPU · estimator: finite-product singular spectrum · artifact

The gated product is not an independent random-matrix toy. Its diagonal gates come from one forward pass through the same finite-width compositional structure whose Jacobian is being measured. At the variance-critical scale $2/n$, the twelve layers retain between 8 and 20 active coordinates each, but the final Jacobian has only eight singular values above 10^{-10}. Its largest singular value is about 2.63 and its smallest is numerically zero. This one realization does not define a law for nonlinear Jacobians; it shows exactly where the deep-linear isometry control stops being the network itself.

14.5. Trace control is not edge control

For an input-output Jacobian $\mathbf{J} \in \mathbb{R}^{m \times n}$,

$$\frac{1}{n} \operatorname{tr}(\mathbf{J}^\top \mathbf{J}) = \frac{1}{n} \sum_i \sigma_i(\mathbf{J})^2. \tag{14.7}$$

Theorem 14.4 (Mean-square preservation does not imply isometry)**.** *The quantity in Equation 14.7 is the mean squared stretch over an orthonormal input basis. It can equal one while the smallest singular value is zero and the largest is* $\sqrt{n}$.

14.5.1. Proof

The trace identity follows from the SVD. The diagonal matrix $\operatorname{diag}(\sqrt{n}, 0, \ldots, 0)$ has mean squared singular value one and the stated edges.

Dynamical isometry asks for substantially more: the singular values of the relevant Jacobian should cluster near one (Pennington et al. 2017). Orthogonal factors solve the deep-linear control exactly.

Nonlinear gates, finite width, correlations, biases, normalization, and trained weights change the product; orthogonality at initialization is an instrument, not a universal cure.

The word *critical* is therefore incomplete until it names a rung:

Contract	Observable	What it can certify	What it cannot exclude
one-input magnitude	q_ℓ	average scale survives	two inputs become indistinguishable
pair geometry	c_ℓ or ψ_ℓ	separation contracts, expands, or is marginal	derivative directions are ill-conditioned
derivative geometry	singular values of $\mathbf{J}$	directional stretch and rank	the infinite-width model is inaccurate
approximation trust	L/n with retained order	the declared expansion is controlled	training leaves the declared scaling regime

The first three rows diagnose the compositional structure. The fourth audits the theory used to describe it. In particular, an infinite-width calculation can make training linearized by suppressing feature movement under its declared parameterization and time horizon. The feature-learning branch is mapped in Beyond This Volume; no width-only diagnosis is licensed here (Jacot et al. 2018; Lee et al. 2019; Chizat et al. 2019; Roberts and Yaida 2022).

14.6. The de-branded object first

We have derived **variance-preserving initialization** from a moment recursion. The Rosetta appendix carries the conventional aliases for search and ties each to its fan-in/fan-out and nonlinearity assumptions. The mathematics decides which moment a name actually covers.

> ⚠ Named wrong answer: every activation has one critical initialization
>
> The moment map may produce a scale-invariant critical curve, a tuned point at a zero fixed point, a half-stable boundary, or no usable critical point in the declared family. Even when a scalar critical point exists, it controls one average statistic, not finite-width fluctuations, pair geometry, spectral edges, or numerical range.

14.7. Check yourself

At zero bias with ReLU and weight scale $2(1+\varepsilon)$, what is the moment after L layers? At the exact scalar critical point, why can two inputs still align? For $L/n = 1/8$, which is the claim: $5L/n$, the exact finite-width formula, or its simultaneous-limit curve? Finally, what do the mean squared singular value and lower spectral edge say for $\mathrm{diag}(\sqrt{2}, 0)$, and which statement governs a gradient component aligned with the second coordinate?

14.8. Okay, so —

- **Inherited:** C11's reverse product and C08's two spectral edges now describe signal propagation.
- **Changed:** "critical" is separated into magnitude, pair, derivative, and approximation-trust contracts.
- **Instrumented:** four claims compare moment recursion, pair correlation, depth-to-width fluctuations, and complete product spectra.
- **Established:** preserving a scalar moment neither preserves finite-angle distinctions nor controls Jacobian edges; L/n marks where the wide approximation accumulates finite-width corrections.
- **Unresolved:** Which invariances and null directions does a coordinate-wise normalization choose when it resets scale?

14.9. Sources and further reading

The Gaussian covariance antecedent is Williams (1997), and the ReLU closed form is the degree-one arc-cosine kernel of Cho and Saul (2009). Two-input signal propagation follows Poole et al. (2016) and Schoenholz et al. (2017); dynamical-isometry diagnostics follow Pennington et al. (2017). Orthogonal initialization as a controlled dynamical system appears in Saxe et al. (2014). Variance-preserving initialization was developed for saturating and rectified compositional structures by Glorot and Bengio (2010) and He et al. (2015). For a derivational companion to the moment and pair-correlation recursions, and for a systematic leading finite-width expansion whose trust parameter is depth over width, see Roberts and Yaida (2022); begin with Chapter 5, Sections 5.1, 5.4, and 5.5.

Reading order. Begin with Cho and Saul (2009) for the closed form, continue with Schoenholz et al. (2017) for the one- and two-input maps and Pennington et al. (2017) for the Jacobian spectrum, then use Roberts and Yaida (2022) to audit the depth-to-width expansion and order of limits.

14.10. Exercises

1. **(Pencil.)** Derive Equation 14.2 for an odd saturating nonlinearity and identify its small-variance linearization. Decide whether the fixed point is stable from one side, both sides, or neither. **Route:** Core. **Estimated time:** 25 minutes. **Prerequisite:** C04 fixed points. **Deliverable:** derivation plus stability classification. **Hint:** differentiate the moment map at the fixed point.
2. **(Code.)** Repeat the product experiment across 100 seeds. Plot distributions of trace, lower edge, and upper edge separately. **Route:** Core. **Estimated time:** 45 minutes. **Prerequisite:** C08 spectrum summaries. **Deliverable:** three matched-seed distributions and a one-paragraph verdict. **Hint:** do not average the lower and upper edges together.
3. **(Pencil.)** For $\mathbf{J} = \mathbf{D}_L\mathbf{W}_L \cdots \mathbf{D}_1\mathbf{W}_1$, show that $\operatorname{rank}(\mathbf{J})$ cannot exceed the smallest gate rank. Explain why this bound can be loose. **Route:** Core. **Estimated time:** 20 minutes. **Prerequisite:** rank inequalities. **Deliverable:** proof and one strict example. **Hint:** apply the rank inequality one factor at a time.
4. **(Audit.)** Build the implication graph among moment preservation, infinitesimal pair preservation, finite-angle preservation, mean-square Jacobian preservation, dynamical isometry, and trainability. Break every invalid arrow with a counterexample or a missing assumption. In particular, locate the invalid transfer from $\chi_\perp = 1$ to finite-angle preservation and state the needed smoothness or uniformity condition. **Route:** Extension. **Estimated time:** 50 minutes. **Prerequisite:** this

chapter. **Deliverable:** annotated implication graph. **Hint:** Equation 14.3 is marginal locally but not constant at finite angle.

5. **(Code.)** Reproduce Figure 14.1 for a smooth saturating activation. Estimate $\chi_{\parallel}$ and $\chi_{\perp}$ separately, and classify the critical point without assuming they coincide. **Route:** Extension. **Estimated time:** 75 minutes. **Prerequisite:** Gaussian quadrature or Monte Carlo. **Deliverable:** two gain estimates, a pair trace, and a boundary statement. **Hint:** perturb the diagonal and off-diagonal kernel coordinates independently.
6. **(Audit.) Paper audit:** Locate a paper that uses "critical" and complete the **Mathematical object**, **Resolution of the theory**, **Dynamic regime**, **Assumption stress test**, **Discriminating control**, and **Transfer verdict** fields of the Paper Autopsy Protocol. Name its small parameter, retained order, and order of limits. As an agent-generated-proof control, remove one independence or width assumption silently, identify the first invalid step, and state the weakest repair. **Route:** Research. **Estimated time:** 90 minutes. **Prerequisite:** C11–C14. **Deliverable:** six-field autopsy plus repaired statement. **Hint:** "infinite width" is not an order of limits until depth is named.

15. Normalization Chooses an Invariance: Coordinate Statistics and Derivative Paths

NS · Numerical stability · LG · Landscape and update geometry

Geometric primary · Dynamic supporting · Algorithmic supporting

Take $\mathbf{x} = (-2, -1, 1, 2)$ and shift every coordinate by ten. Centering and scaling sends both vectors to the same output. Scaling by the uncentered root mean square does not: the maximum coordinate-wise change is 2.055. The two maps are often offered as nearly interchangeable stabilizers, but they erase different information.

The arithmetic also matters. For FP32 values $(9998, 9999, 10001, 10002)$, the raw-moment expression $\text{mean}(x^2) - \text{mean}(x)^2$ evaluates to zero in the registered runtime, while centering first gives 2.5. C02's cancellation has returned inside a standard training primitive.

> **!** Prediction
>
> If a normalization rule is removed or replaced, what behavior should change? Answer by naming its invariance, Jacobian nullspace, statistic, placement, and precision contract—not by invoking the method name.

15.1. Two maps, two quotients

Ignoring learned affine parameters and an implementation epsilon, define

$$\mathcal{N}_C(\mathbf{x}) = \sqrt{d}\,\frac{\mathbf{P}\mathbf{x}}{\|\mathbf{P}\mathbf{x}\|_2}, \quad \mathbf{P} = \mathbf{I} - \frac{1}{d}\mathbf{1}\mathbf{1}^\top, \tag{15.1}$$

and

$$\mathcal{N}_R(\mathbf{x}) = \sqrt{d}\,\frac{\mathbf{x}}{\|\mathbf{x}\|_2}. \tag{15.2}$$

The centered map is invariant to positive scaling and constant-coordinate translation. The RMS map is invariant to positive scaling but retains the constant component. Neither contract is universally superior; the choice is architectural information.

Theorem 15.1 (Jacobians expose the erased directions)**.** *Away from degenerate states and with zero epsilon, let* $\mathbf{u} = \mathbf{x}/\|\mathbf{x}\|$ *and* $\mathbf{v} = \mathbf{P}\mathbf{x}/\|\mathbf{P}\mathbf{x}\|$*. Then*

$$\mathbf{J}_R = \frac{\sqrt{d}}{\|\mathbf{x}\|}(\mathbf{I} - \mathbf{u}\mathbf{u}^\top), \qquad \mathbf{J}_C = \frac{\sqrt{d}}{\|\mathbf{P}\mathbf{x}\|}(\mathbf{P} - \mathbf{v}\mathbf{v}^\top). \tag{15.3}$$

Generically, $\mathbf{J}_R$ *has rank* $d-1$ *and* $\mathbf{J}_C$ *rank* $d-2$.

15.1.1. Proof

Differentiate $\mathbf{z}/\|\mathbf{z}\|$ to obtain $(\mathbf{I}-\hat{\mathbf{z}}\hat{\mathbf{z}}^\top)/\|\mathbf{z}\|$. For Equation 15.1, compose with the fixed projector $\mathbf{P}$. The RMS map annihilates its radial direction. The centered map additionally annihilates $\mathbf{1}$ because $\mathbf{P1}=0$.

1. Load the chapter-pinned normalization instruments.
2. Normalize one vector before and after a constant shift.
3. Compute both exact zero-epsilon Jacobians.
4. Compare raw-moment and centered FP32 variance.
5. Verify the registered invariance and rank claims.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]
from trainable_harness import (
    centered_normalize,
    merge_moments,
    normalization_jacobian,
    raw_and_centered_variance,
    rms_normalize,
    stable_online_moments,
)

# [2]
x = np.array([-2.0, -1.0, 1.0, 2.0])
shifted = x + 10.0
outputs = {
    "centered x": centered_normalize(x),
    "centered x+10": centered_normalize(shifted),
    "RMS x": rms_normalize(x),
    "RMS x+10": rms_normalize(shifted),
}
fig, axis = plt.subplots(figsize=(7.4, 3.7))
for label, values in outputs.items():
    axis.plot(values, "o-", label=label)
axis.set(xlabel="coordinate", ylabel="normalized value")
axis.legend(frameon=False, ncol=2)
fig.tight_layout()
# [3]
assert np.linalg.matrix_rank(
    normalization_jacobian(x, centered=True), tol=1e-10
) == 2
assert np.linalg.matrix_rank(
    normalization_jacobian(x, centered=False), tol=1e-10
) == 3
```

```
# [4]
assert raw_and_centered_variance(
    np.array([9998, 9999, 10001, 10002])
) == (0.0, 2.5)
# [5]
claim = verify_claim("c15-invariance-001", expected_harness=pin)
assert claim["result"]["centered_shift_deviation"] == 0.0
print("claim c15-invariance-001: verified")
```

```
claim c15-invariance-001: verified
```

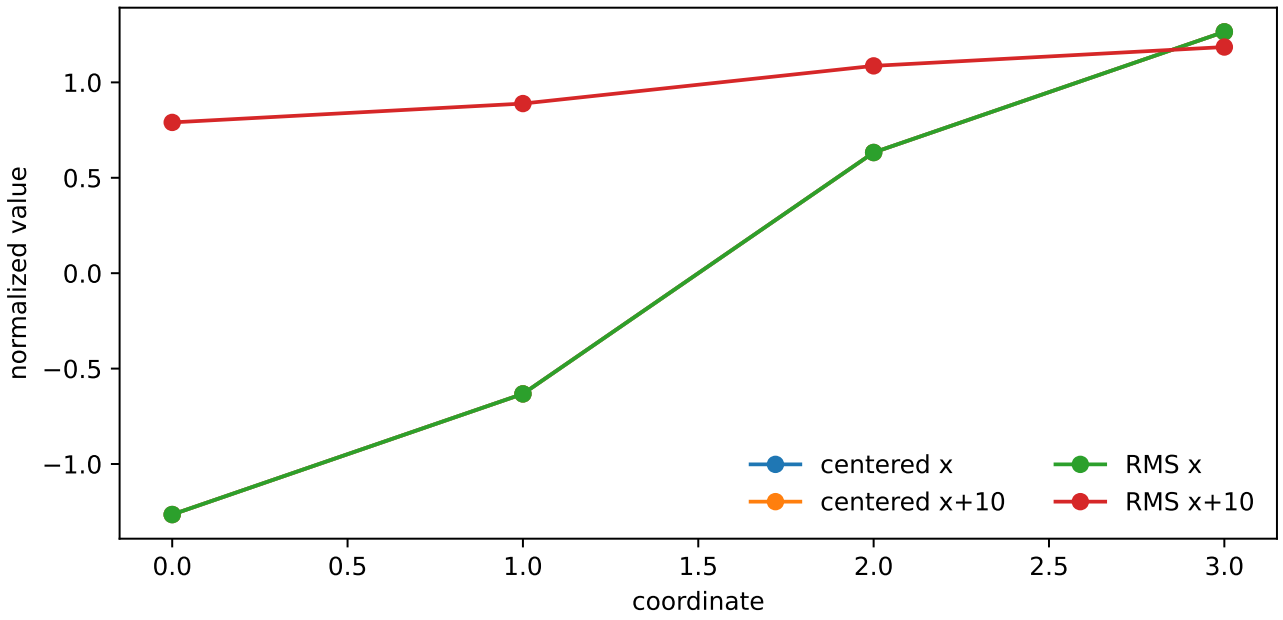


Figure 15.1.: Centering and RMS-only scaling choose different equivalence classes. Centered normalization removes a constant shift; RMS normalization preserves evidence of it.

Claim `c15-invariance-001` · seed: none · dtype: FP64 with paired FP32 cancellation check · device: CPU · estimator: deterministic map and Jacobian · artifact

15.2. Epsilon changes the map

Production rules use a denominator such as $\sqrt{m_2 + \varepsilon}$. Epsilon prevents division by zero and caps local amplification, but it breaks exact scale invariance when m_2 is comparable to epsilon. Its value and placement are therefore part of the algorithm. Accumulating m_2 in a wider format can matter even when stored activations are narrow. Raw moments should not be used when a centered statistic is intended.

Batch-coordinate normalization adds another state: running estimates used at evaluation. Those are streaming centered statistics of the kind built in C02, with the same estimator-denominator and reduction-order obligations.

15.3. Train state and evaluation state are different estimators

The oldest forward promise in the book now becomes executable. Draw eight training batches and one evaluation batch from the same scalar Gaussian population. There is no distribution shift. Train mode normalizes the evaluation batch using its own 32 observations; evaluation mode uses the running state merged from 256 earlier observations. The two outputs can still differ because the estimators saw different finite samples.

1. Derive a claim-specific seed and draw train and evaluation batches.
2. Build one centered state per training batch and merge the states.
3. Build the evaluation batch state with the same denominator contract.
4. Normalize the evaluation values under the two estimator states.
5. Plot convergence and output discrepancy, then verify the claim.

```
# [1]
running_seed = int(sha256(b"c15-running-state").hexdigest()[:8], 16)
running_rng = np.random.default_rng(running_seed)
training_batches = running_rng.normal(4, 2, size=(8, 32))
evaluation_batch = running_rng.normal(4, 2, size=32)

# [2]
batch_states = [
    stable_online_moments(batch, dtype=np.float64)
    for batch in training_batches
]
running_state = batch_states[0]
running_trace = [running_state]
for batch_state in batch_states[1:]:
    running_state = merge_moments(running_state, batch_state)
    running_trace.append(running_state)
direct_state = stable_online_moments(
    training_batches.ravel(), dtype=np.float64
)

# [3]
evaluation_state = stable_online_moments(
    evaluation_batch, dtype=np.float64
)
epsilon = 1e-5

# [4]
train_mode_output = (
    (evaluation_batch - evaluation_state.mean)
    / np.sqrt(evaluation_state.population_variance + epsilon)
)
evaluation_mode_output = (
    (evaluation_batch - running_state.mean)
    / np.sqrt(running_state.population_variance + epsilon)
)
```

```
# [5]
fig, axes = plt.subplots(1, 2, figsize=(9.2, 3.6))
batch_axis = np.arange(1, len(running_trace) + 1)
axes[0].plot(
    batch_axis, [state.mean for state in running_trace],
    "o-", color="#232D4B", label="running mean",
)
axes[0].plot(
    batch_axis, [state.population_variance for state in running_trace],
    "s-", color="#9C2F2F", label="running variance",
)
axes[0].axhline(4, color="#6B6B6B", linestyle=":", label="population value")
axes[0].set(xlabel="merged batch count", ylabel="estimated statistic")
axes[0].legend(frameon=False)
axes[1].plot(train_mode_output, "o-", color="#232D4B", label="batch state")
axes[1].plot(
    evaluation_mode_output, "s--", color="#9C2F2F",
    label="running state",
)
axes[1].set(xlabel="evaluation coordinate", ylabel="normalized value")
axes[1].legend(frameon=False)
for axis in axes:
    axis.grid(alpha=0.2)
fig.tight_layout()

running_claim = verify_claim("c15-running-state-001", expected_harness=pin)
assert np.isclose(running_state.mean, direct_state.mean, atol=3e-15)
assert running_state.m2 == direct_state.m2
assert np.isclose(
    np.max(np.abs(train_mode_output - evaluation_mode_output)),
    running_claim["result"]["normalized_discrepancy"]["maximum_absolute"],
)
print(
    "running/evaluation means: "
    f"{running_state.mean:.3f}/{evaluation_state.mean:.3f}"
)
print(
    "maximum normalized discrepancy: "
    f"{np.max(np.abs(train_mode_output - evaluation_mode_output)):.3f}"
)
```

```
running/evaluation means: 4.096/3.766
maximum normalized discrepancy: 0.448
```

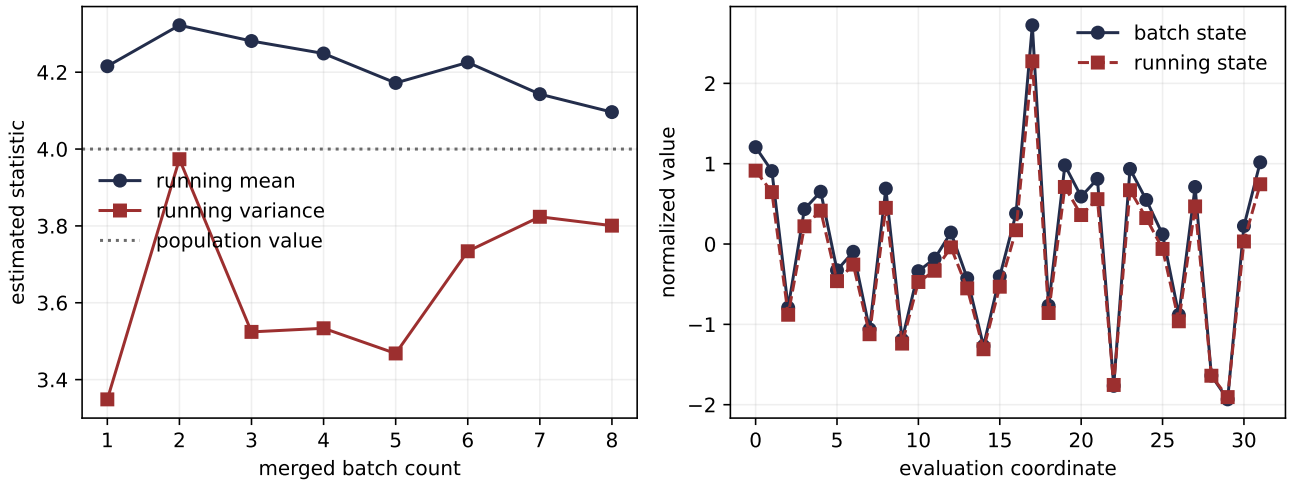


Figure 15.2.: Training and evaluation modes can disagree without distribution shift. The running state merges eight centered batch states exactly up to rounding, while the held-out batch has different finite estimates; normalizing the same evaluation values under those two states changes the outputs.

The merged state agrees with a single pass over all 256 training values to rounding precision. The evaluation batch mean is about 3.766, while the running mean is about 4.096; the corresponding population-variance estimates are 3.063 and 3.801. Normalizing the same evaluation values under those two states changes one coordinate by as much as 0.448. The discrepancy is neither a precision failure nor evidence of distribution shift. It is the finite-estimator gap that a train/evaluation policy must own.

15.4. Placement changes the derivative path

For a residual map, pre-normalization has Jacobian

$$\mathbf{J}_{\text{pre}} = \mathbf{I} + \mathbf{J}_F \mathbf{J}_N,$$

whereas post-normalization has

$$\mathbf{J}_{\text{post}} = \mathbf{J}_N(\mathbf{I} + \mathbf{J}_F).$$

The first contains an explicit identity summand; the second filters every path through $\mathbf{J}_N$. The identity summand changes the criticality budget; it does not abolish the need to control the residual branch (Roberts and Yaida 2022). This algebra explains a diagnostic difference. It does not prove that every pre-normalized architecture trains better, because the learned function, scale, finite epsilon, and update dynamics also change.

15.5. Brand bridge and claim boundary

The centered coordinate-wise rule is conventionally called LayerNorm (Ba et al. 2016); the uncentered rule is RMSNorm (Zhang and Sennrich 2019). These aliases appear here only after the mathematical objects are separated. The batch-and-spatial rule with running statistics is conventionally called BatchNorm (Ioffe and Szegedy 2015). Claims that normalization "smooths the landscape" require a named slice, metric, and experimental setting; Santurkar et al. (2018) provides influential evidence, not a universal theorem about all normalized models.

> ⚠ Named wrong answer: normalization preserves information
>
> Every invariant map deliberately discards directions. The right question is whether those directions should be irrelevant to the downstream computation.

15.6. Check yourself

For $d = 4$, which directions lie in the generic nullspace of centered normalization, and which lie in the RMS-only nullspace? What ranks follow? When epsilon is added, which invariances remain exact and which become only approximate?

15.7. Okay, so —

- **Inherited:** C02's centered state and C14's Jacobian spectrum now diagnose a training primitive.
- **Changed:** normalization is specified by invariance, statistic, epsilon, and placement.
- **Instrumented:** one claim checks invariance, rank, and FP32 arithmetic; another merges running state and exposes a train/evaluation estimator gap.
- **Established:** centered and RMS-only rules erase different directions.
- **Unresolved:** What happens when a locally controlled derivative map approaches a curvature boundary that moves with the trajectory?

15.8. Sources and further reading

See Ioffe and Szegedy (2015) for the batch-statistic method, Ba et al. (2016) for the centered coordinate-wise method, Zhang and Sennrich (2019) for its RMS-only variant, and Santurkar et al. (2018) for a later loss-geometry investigation.

Reading order. Start with Ioffe and Szegedy (2015) to fix the batch/evaluation-state contract, read Ba et al. (2016) and Zhang and Sennrich (2019) side by side to compare their chosen invariances, then use Santurkar et al. (2018) to audit which geometric mechanism its experiments actually support.

15.9. Exercises

1. **(Pencil.)** Differentiate both maps with nonzero epsilon and identify which nullspaces remain exact.
2. **(Code.)** Sweep offset magnitude and dtype for the four-coordinate variance witness. Report the first raw-moment failure.
3. **(Pencil.)** Derive the pairwise merge formula for two centered states and prove that the correction term is nonnegative. State the population and sample denominators separately.
4. **(Code.)** Repeat the running-state witness across batch counts and distribution shifts. Keep the no-shift control, and decompose total output discrepancy into finite-estimator and shift components.
5. **(Audit.)** **Paper audit:** For a normalization ablation, complete the **Mathematical object**, **Estimator and comparison contract**, **Numerical and hardware contract**, **Discriminating control**, and **Transfer verdict** fields of the Paper Autopsy Protocol. Record statistic, axes, epsilon, affine parameters, placement, accumulation format, and evaluation-state policy.

16. The Boundary Moves: Progressive Sharpening and Edge-of-Stability Diagnostics

LG · Landscape and update geometry · NS · Numerical stability

Geometric supporting · Dynamic primary · Algorithmic supporting

Run full-batch gradient descent with step size 0.2 on

$$f(x, y) = \frac{1}{2}(y - x^2)^2 + 0.05(x - 3)^2, \tag{16.1}$$

starting from the origin. The top Hessian eigenvalue begins at one, far below the fixed-quadratic boundary $2/\alpha = 10$. At step 155 it first crosses ten. Across 350 steps the loss increases 52 times, while the trajectory remains in a bounded, structured oscillatory regime. A stability test performed only at initialization gives the correct answer to the wrong time-indexed question.

> **!** Prediction
>
> If the top curvature crosses $2/\alpha$, must a nonlinear training run diverge immediately? Choose between **immediate divergence**, **a diagnostic event whose consequence must be measured**, and **no information at all**. Which control would distinguish your choice from the other two?

16.1. The fixed control

Theorem 16.1 (Exact stability of one quadratic mode)**.** *For $f(z) = \lambda z^2/2$ with $\lambda > 0$, gradient descent gives $z_{k+1} = (1 - \alpha\lambda)z_k$. The iterates converge to zero exactly when*

$$0 < \alpha\lambda < 2. \tag{16.2}$$

16.1.1. Proof

The recurrence converges precisely when its scalar multiplier has magnitude less than one: $|1 - \alpha\lambda| < 1$. At $\alpha\lambda > 1$, signs alternate; at two, the mode has a perfect two-cycle; beyond two, its magnitude grows.

For a positive-definite quadratic, apply the result to every eigenmode. This is C04's fixed spectral boundary. It is the indispensable control—not a license to substitute a local Hessian eigenvalue into a global conclusion.

16.2. Instrument the path, not one endpoint

For Equation 16.1,

$$\nabla^2 f(x, y) = \begin{bmatrix} 6x^2 - 2y + 0.1 & -2x \\ -2x & 1 \end{bmatrix}. \tag{16.3}$$

We record both $\lambda_{\max}(\nabla^2 f)$ and gradient-direction curvature $\mathbf{g}^\mathsf{T}\nabla^2 f\mathbf{g}/\|\mathbf{g}\|^2$. The first asks for the worst local direction; the second asks what curvature the current update actually sees. C12 already warned that the matrix itself must be named.

1. Load the chapter-pinned stability instruments.
2. Run matched deterministic and additive-noise trajectories.
3. Record loss, top curvature, and gradient-direction curvature each step.
4. Mark the fixed-quadratic threshold and first crossing.
5. Verify both committed crossing and nonmonotonicity claims.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]
from trainable_harness import (
    moving_curvature_trace,
    valley_loss_gradient_hessian,
)

# [2]
step_size = 0.2
trace = moving_curvature_trace(np.zeros(2), step_size, 350)
stochastic_seed = int(
    sha256(b"c16-stochastic-contrast").hexdigest()[:8], 16
)
stochastic_rng = np.random.default_rng(stochastic_seed)
stochastic_state = np.zeros(2)
stochastic_losses, stochastic_top = [], []
for _ in range(351):
    stochastic_loss, stochastic_gradient, stochastic_hessian = (
        valley_loss_gradient_hessian(stochastic_state)
    )
    stochastic_losses.append(stochastic_loss)
    stochastic_top.append(np.linalg.eigvalsh(stochastic_hessian)[-1])
    stochastic_state -= step_size * (
        stochastic_gradient + stochastic_rng.normal(scale=0.01, size=2)
    )
stochastic_losses = np.asarray(stochastic_losses)
stochastic_top = np.asarray(stochastic_top)
# [3]
fig, axes = plt.subplots(2, 1, figsize=(8.0, 5.6), sharex=True)
axes[0].plot(trace.losses, color="#232D4B", label="deterministic")
```

```
axes[0].plot(
    stochastic_losses, color="#E57200", linestyle="--",
    label="additive-noise control",
)
axes[0].set(ylabel="loss")
axes[0].legend(frameon=False)
axes[1].plot(trace.top_curvatures, label="top Hessian curvature")
axes[1].plot(
    trace.directional_curvatures,
    label="gradient-direction curvature",
    alpha=0.8,
)
axes[1].plot(
    stochastic_top, color="#E57200", linestyle="--",
    label="noisy top curvature",
)
# [4]
threshold = 2 / step_size
axes[1].axhline(
    threshold, color="#9C2F2F", linestyle=":", label=r"$2/\alpha$"
)
axes[1].set(xlabel="step", ylabel="curvature")
axes[1].legend(frameon=False, ncol=3, fontsize=8)
fig.tight_layout()
# [5]
claim = verify_claim("c16-moving-boundary-001", expected_harness=pin)
stochastic_claim = verify_claim(
    "c16-stochastic-contrast-001", expected_harness=pin
)
assert int(np.argmax(trace.top_curvatures > threshold)) == claim["result"][
    "first_crossing_step"
]
assert int(np.sum(np.diff(trace.losses) > 0)) == claim["result"][
    "number_loss_increases"
]
assert int(np.flatnonzero(stochastic_top > threshold)[0]) == stochastic_claim[
    "result"
]["first_crossing_step"]
assert int(np.sum(np.diff(stochastic_losses) > 0)) == stochastic_claim[
    "result"
]["number_loss_increases"]
print("claim c16-moving-boundary-001: verified")
print("claim c16-stochastic-contrast-001: verified")
```

```
claim c16-moving-boundary-001: verified
claim c16-stochastic-contrast-001: verified
```

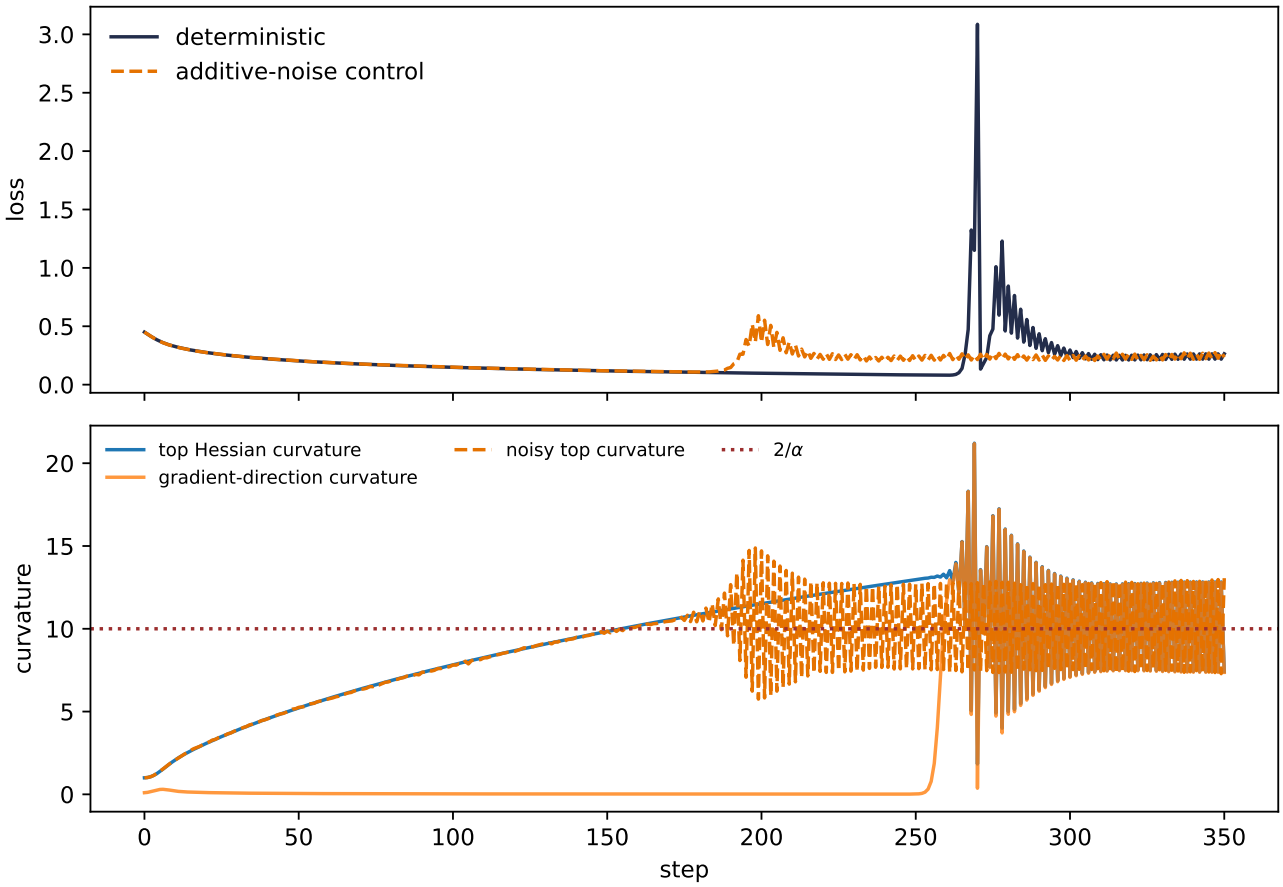


Figure 16.1.: A trajectory can begin below the quadratic stability edge and later move the local top curvature across it. A small additive-noise contrast preserves the crossing but shifts its timing and loss increments; neither trace turns a local quadratic threshold into a global divergence theorem.

Claims `c16-moving-boundary-001`, `c16-stochastic-contrast-001` · seed: deterministic primary trajectory; registered stochastic control · dtype: FP64 · device: CPU · estimator: top and directional curvature along the realized path · artifact

This is a laptop-exact demonstration: minutes are not needed, no GPU is required, and every point is regenerated on a CPU. It establishes the logical possibility of a moving boundary. The additive-noise trace crosses at step 156 rather than 155 and contains 91 loss increases rather than 52. It shows that the crossing story survives this small perturbation while its timing and nonmonotonicity do not. Neither trace claims that a two-parameter valley reproduces mini-batch training or every feature of a large run.

16.3. One boundary, two diagnostic objects

The exact quadratic and the nonlinear trajectory should be read in one panel, not as consecutive stories:

Diagnostic field	Fixed quadratic control	Moving nonlinear trajectory	What transfers
state	one eigenmode z_k	parameter vector $\mathbf{w}_k$	compare the update with curvature at the same time index
curvature	constant scalar λ	$\lambda_{\max}(\mathbf{H}_k)$ and gradient-direction curvature	$2/\alpha$ remains a reference scale
prediction	exact convergence iff $0 < \alpha\lambda < 2$	crossing is an event to investigate	sign alternation and local aggressiveness remain useful clues
history	one fixed multiplier	rotating directions and changing Hessian	the trajectory must be recorded
conclusion	necessary and sufficient for this mode	no universal divergence verdict	the quadratic is a control, not a theorem about the full run

The matched panel slows the transfer at the point where it is easiest to overreach. "Above two" is an exact statement only in the left column.

16.4. Local Taylor information is historical

For one update $\mathbf{w}^+ = \mathbf{w} - \alpha\mathbf{g}$, Taylor's theorem along the segment gives

$$f(\mathbf{w}^+) - f(\mathbf{w}) = -\alpha\|\mathbf{g}\|^2 + \frac{\alpha^2}{2}\mathbf{g}^\mathsf{T}\nabla^2 f(\mathbf{w} - \tau\alpha\mathbf{g})\mathbf{g} \tag{16.4}$$

for some $\tau \in (0, 1)$. The relevant curvature can lie between endpoints, and a top eigenvector can be nearly orthogonal to the step. Consequently a credible edge-of-stability panel reports at least step size, top curvature, directional curvature, loss increments, estimator/batch regime, and the curvature operator used.

16.5. From witness to empirical phenomenon

Large full-batch training runs have been observed to approach and operate near the classical threshold, with progressive sharpening and nonmonotone loss (Cohen et al. 2021). A useful one-line contrast is: classical optimization adapts the step to a curvature bound; deep compositional structures can move their local curvature toward the chosen step. This is an empirical organizing statement, not a universal convergence theorem.

The "staircase" behavior analyzed in later edge-of-stability work belongs in the paper-audit branch: students should identify its model, timescale, and measured curvature before treating it as the same phenomenon. The trunk keeps the fixed quadratic, the moving witness, and the measurement contract.

> ⚠ Named wrong answer: above two means divergent
>
> Above two means divergent for a fixed positive quadratic eigenmode. A local nonlinear Hessian can rotate and change after the step; crossing is a diagnostic event whose consequences must be measured.

16.6. Numerical stability returns

Near an oscillatory boundary, small arithmetic changes can alter which side of a narrow basin an update reaches. That does not make every loss spike a precision failure. It means curvature, reduction order, storage/accumulation format, and replay determinism belong in one incident report. The four threads have converged on a common diagnostic discipline.

16.7. Check yourself

For a fixed mode with $\lambda = 12$ and $\alpha = 0.2$, what is the multiplier and what does it imply? For a nonlinear objective whose local top eigenvalue is 12 at one point, which additional measurements would be needed before predicting the next several iterates?

16.8. Okay, so —

- **Inherited:** C04 supplies the exact fixed-quadratic boundary and C12 supplies operator-specific curvature probes.
- **Changed:** stability is treated as a trajectory diagnosis, not an initialization certificate.
- **Instrumented:** matched deterministic and additive-noise traces track top and directional curvature beside loss.
- **Established:** a reproducible nonlinear path can cross $2/\alpha$ after beginning below it.
- **Unresolved:** Which norm should shape a matrix-valued update when scalar curvature says only where one step struggles?

16.9. Sources and further reading

The edge-of-stability phenomenon and progressive sharpening measurements are documented by Cohen et al. (2021). The fixed control is elementary spectral gradient descent.

Reading order. Start with Cohen et al. (2021) for the measurements, then return to the fixed quadratic theorem in C04 before transferring the observed threshold to a moving nonlinear trajectory.

16.10. Exercises

1. **(Pencil.)** Derive the momentum stability polynomial for one positive quadratic mode, explicitly limiting the conclusion to that control.
2. **(Code.)** Sweep step size in Equation 16.1 and produce a phase diagram of first threshold crossing, loss increases, and boundedness.
3. **(Pencil.)** Use Equation 16.4 to give a sufficient one-step descent condition in terms of the maximum directional curvature along the update segment. Explain why endpoint curvature alone is weaker.
4. **(Code.)** Repeat the paired contrast over noise scales and seeds. Report crossing probability, crossing time, loss-increase count, and boundedness; keep the deterministic trace as a registered control.
5. **(Audit.)** **Paper audit:** Complete the **Mathematical object**, **Resolution of the theory**, **Dynamic regime**, **Evidence culture and interface**, **Discriminating control**, and **Transfer verdict** fields of the Paper Autopsy Protocol for an edge-of-stability result. Record curvature operator, eigenvalue estimator, batch regime, sampling frequency, threshold convention, and whether crossing predicts or merely accompanies the reported behavior.

17. The Norm Chooses the Update: Matrix Geometry and Orthogonalized Optimization

RS · Random-matrix spectra · LG · Landscape and update geometry · NS · Numerical stability

Geometric primary · Dynamic supporting · Algorithmic supporting

Let a 6×4 gradient matrix have singular values $(9, 3, 1, 0.2)$. Three unit-size update problems return three different answers. A Frobenius-unit step follows the normalized gradient. A nuclear-unit step spends its entire budget on the leading singular pair. An operator-unit step uses the polar factor and gives all four singular directions unit magnitude. Their inner products with the gradient are respectively -9.54, -9, and -13.2.

"Steepest" was never a property of the gradient alone; it was a property of a normed geometry.

> **!** Prediction
>
> A paper replaces a coordinate-wise update by a matrix transformation. Before asking whether it wins, ask: steepest under which norm, for which parameter blocks, with what approximation error, scaling rule, state traffic, and evidence regime?

17.1. One linearization, three unit balls

For a small update $\mathbf{D}$, the first-order change is $\langle \mathbf{G}, \mathbf{D} \rangle_F$. Choosing a step means minimizing this linear functional over a declared unit ball.

Theorem 17.1 (Matrix-norm steepest directions)**.** *Let* $\mathbf{G} = \mathbf{U}\,\mathrm{diag}(\boldsymbol{\sigma})\mathbf{V}^\mathsf{T}$. *Then*

constraint	*one minimizer*	*minimum*
$\|\mathbf{D}\|_F \le 1$	$-\mathbf{G}/\|\mathbf{G}\|_F$	$-\|\mathbf{G}\|_F$
$\|\mathbf{D}\|_{\mathrm{op}} \le 1$	$-\mathbf{U}\mathbf{V}^\mathsf{T}$	$-\|\mathbf{G}\|_*$
$\|\mathbf{D}\|_* \le 1$	$-\mathbf{u}_1\mathbf{v}_1^\mathsf{T}$	$-\|\mathbf{G}\|_{\mathrm{op}}$.

(17.1)

17.1.1. Proof

The Frobenius row is Cauchy–Schwarz. The other rows are duality between the operator and nuclear norms, with equality at the displayed SVD-aligned matrices. Notice the reversal: an operator-norm constraint yields the polar factor; a nuclear-norm constraint yields a rank-one direction.

This is the de-branded trunk beneath recent matrix-aware optimizers. It also returns to C08: singular values of the **update** are not singular values of the weights, Hessian, or data matrix. Every spectral plot must name its matrix.

1. Load the chapter-pinned matrix-update instruments.
2. Construct the registered rectangular gradient.
3. Compute the three norm-steepest directions.
4. Approximate the polar factor by Newton–Schulz iteration.
5. Verify the singular values, inner products, and residual trace.

```
import matplotlib.pyplot as plt
import numpy as np

# [1]
from trainable_harness import (
    newton_schulz_polar,
    norm_steepest_directions,
    polar_factor,
)

# [2]
gradient = np.zeros((6, 4))
gradient[:4, :] = np.diag([9.0, 3.0, 1.0, 0.2])
# [3]
directions = norm_steepest_directions(gradient)
fig, axes = plt.subplots(1, 2, figsize=(9.2, 3.6))
axes[0].plot(
    np.linalg.svd(gradient, compute_uv=False),
    "o-",
    label="gradient",
)
for name, direction in directions.items():
    axes[0].plot(
        np.linalg.svd(direction, compute_uv=False), "o-", label=name
    )
axes[0].set(xlabel="singular direction", ylabel="singular value")
axes[0].legend(frameon=False, fontsize=8)
# [4]
approximation, residuals = newton_schulz_polar(gradient, 12)
axes[1].semilogy(residuals, "o-", color="#232D4B")
axes[1].set(xlabel="Newton-Schulz step", ylabel=r"$\|X^T X-I\|_F$")
fig.tight_layout()
# [5]
claim = verify_claim("c17-update-norm-001", expected_harness=pin)
assert np.allclose(
    np.linalg.svd(polar_factor(gradient), compute_uv=False),
    claim["result"]["polar_singular_values"],
)
assert np.isclose(
    residuals[-1],
    claim["result"]["newton_schulz_residuals"][-1],
    atol=1e-12,
)
```

```
print("claim c17-update-norm-001: verified")
```

```
claim c17-update-norm-001: verified
```

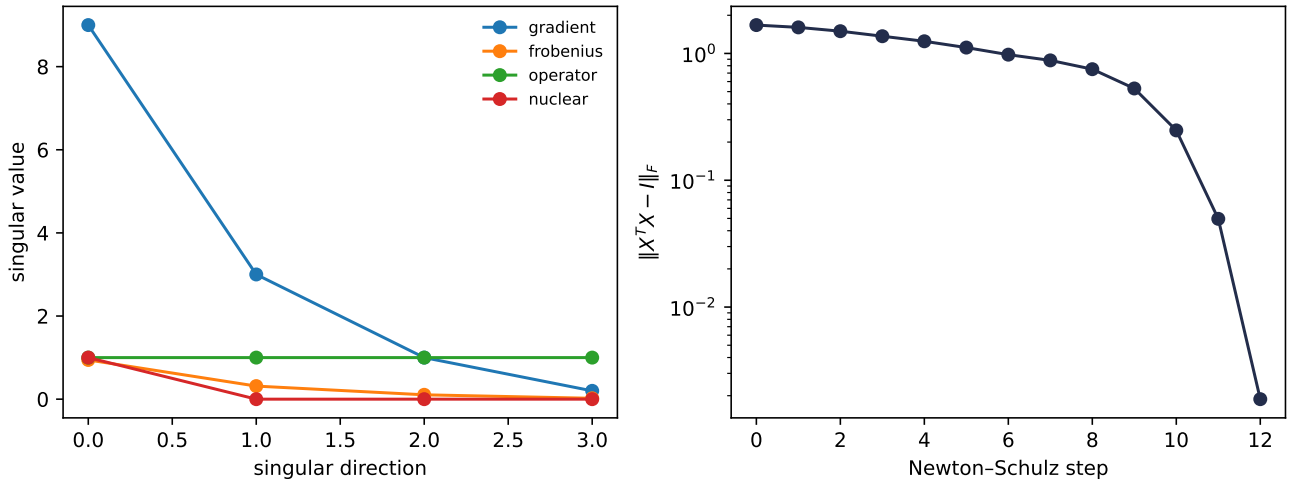


Figure 17.1.: The norm changes the update spectrum. The operator-ball solution replaces every nonzero gradient singular value by one; a short polynomial iteration approaches that polar factor without an SVD.

Claim `c17-update-norm-001` · seed: none · dtype: FP64 · device: CPU · estimator: exact singular values plus polynomial residual · artifact

17.2. A polynomial is an approximation contract

After scaling $\mathbf{X}_0 = \mathbf{G}/\|\mathbf{G}\|_F$, the iteration

$$\mathbf{X}_{t+1} = \frac{3}{2}\mathbf{X}_t - \frac{1}{2}\mathbf{X}_t(\mathbf{X}_t^\top \mathbf{X}_t) \tag{17.2}$$

acts on each singular value as $s \mapsto 1.5s - 0.5s^3$. For $0 < s < \sqrt{3}$, repeated steps move toward one; zero singular values remain zero. A finite iteration therefore needs a scaling rule, step count, residual metric, dtype, and rank-deficiency policy. In narrow arithmetic, forming the Gram factor and repeated matrix products also reopens C03's accumulation contract.

The exact polar factor is equivariant: $\mathrm{polar}(\mathbf{QGR}) = \mathbf{Q}\,\mathrm{polar}(\mathbf{G})\mathbf{R}$ for orthogonal $\mathbf{Q}, \mathbf{R}$, and it is invariant to positive scalar rescaling. Those properties explain what coordinate transformations the update respects.

17.3. The update spectrum is not the weight spectrum

The opening compares candidate updates at one gradient. The promise from C08 is stronger: show the update spectrum and the weight spectrum side by side while a model changes. Use a factorized least-squares control

$$\min_{\mathbf{A}\in\mathbb{R}^{6\times 4},\,\mathbf{B}\in\mathbb{R}^{4\times 4}} \frac{1}{2\cdot 24}\,\|\mathbf{AB}-\mathbf{T}\|_{\mathrm{F}}^{2}\,, \tag{17.3}$$

with matched initialization and twenty steps. One run normalizes each factor gradient in Frobenius norm. The other replaces it by the rectangular polar factor. Both then restore an external step scale $alpha = 0.08$.

1. Derive the claim seed and construct one target and matched factorization.
2. Run twenty Frobenius-normalized and polar steps.
3. Record the singular values of each update to factor A.
4. Record the singular values of factor A before each update.
5. Plot both spectra over time and verify the committed endpoints.

```
# [1]
trace_seed = int(sha256(b"c17-weight-update-spectrum").hexdigest()[:8], 16)
trace_rng = np.random.default_rng(trace_seed)
target = trace_rng.normal(size=(6, 4))
initial_a = 0.2 * trace_rng.normal(size=(6, 4))
initial_b = 0.2 * trace_rng.normal(size=(4, 4))
trace_step, trace_steps = 0.08, 20

# [2]
spectral_traces = {}
for rule in ("frobenius", "polar"):
    factor_a, factor_b = initial_a.copy(), initial_b.copy()
    update_spectra, weight_spectra = [], []
    for _ in range(trace_steps):
        residual = factor_a @ factor_b - target
        gradient_a = residual @ factor_b.T / residual.size
        gradient_b = factor_a.T @ residual / residual.size
        if rule == "frobenius":
            direction_a = -gradient_a / np.linalg.norm(gradient_a, "fro")
            direction_b = -gradient_b / np.linalg.norm(gradient_b, "fro")
        else:
            direction_a = -polar_factor(gradient_a)
            direction_b = -polar_factor(gradient_b)
        # [3]
        update_spectra.append(
            np.linalg.svd(trace_step * direction_a, compute_uv=False)
        )
        # [4]
        weight_spectra.append(
            np.linalg.svd(factor_a, compute_uv=False)
```

```
        )
        factor_a += trace_step * direction_a
        factor_b += trace_step * direction_b
    spectral_traces[rule] = {
        "update": np.asarray(update_spectra),
        "weight": np.asarray(weight_spectra),
    }

# [5]
fig, axes = plt.subplots(2, 2, figsize=(9.2, 6.0), sharex=True)
for row, rule in enumerate(("frobenius", "polar")):
    for singular_index in range(4):
        axes[row, 0].plot(
            spectral_traces[rule]["update"][:, singular_index],
            label=fr"$\sigma_{singular_index + 1}$",
        )
        axes[row, 1].plot(
            spectral_traces[rule]["weight"][:, singular_index]
        )
    axes[row, 0].set(ylabel=f"{rule}\nupdate singular value")
    axes[row, 1].set(ylabel=f"{rule}\nweight singular value")
for axis in axes[-1, :]:
    axis.set(xlabel="step")
axes[0, 0].legend(frameon=False, ncol=2, fontsize=8)
for axis in axes.flat:
    axis.grid(alpha=0.2)
fig.tight_layout()

trace_claim = verify_claim(
    "c17-weight-update-spectrum-001", expected_harness=pin
)
assert np.allclose(
    spectral_traces["polar"]["update"][-1],
    trace_claim["result"]["by_rule"]["polar"][
        "update_singular_values_step_19"
    ],
)
assert np.allclose(
    spectral_traces["frobenius"]["weight"][-1],
    trace_claim["result"]["by_rule"]["frobenius"][
        "weight_singular_values_step_19"
    ],
)
print("polar update spread: "
      f"{np.ptp(spectral_traces['polar']['update']).max():.2e}")
print("polar final weight spectrum: " + "/".join(
    f"{value:.3f}" for value in spectral_traces["polar"]["weight"][-1]
))
```

```
polar update spread: 1.39e-16
polar final weight spectrum: 1.706/1.584/0.969/0.686
```

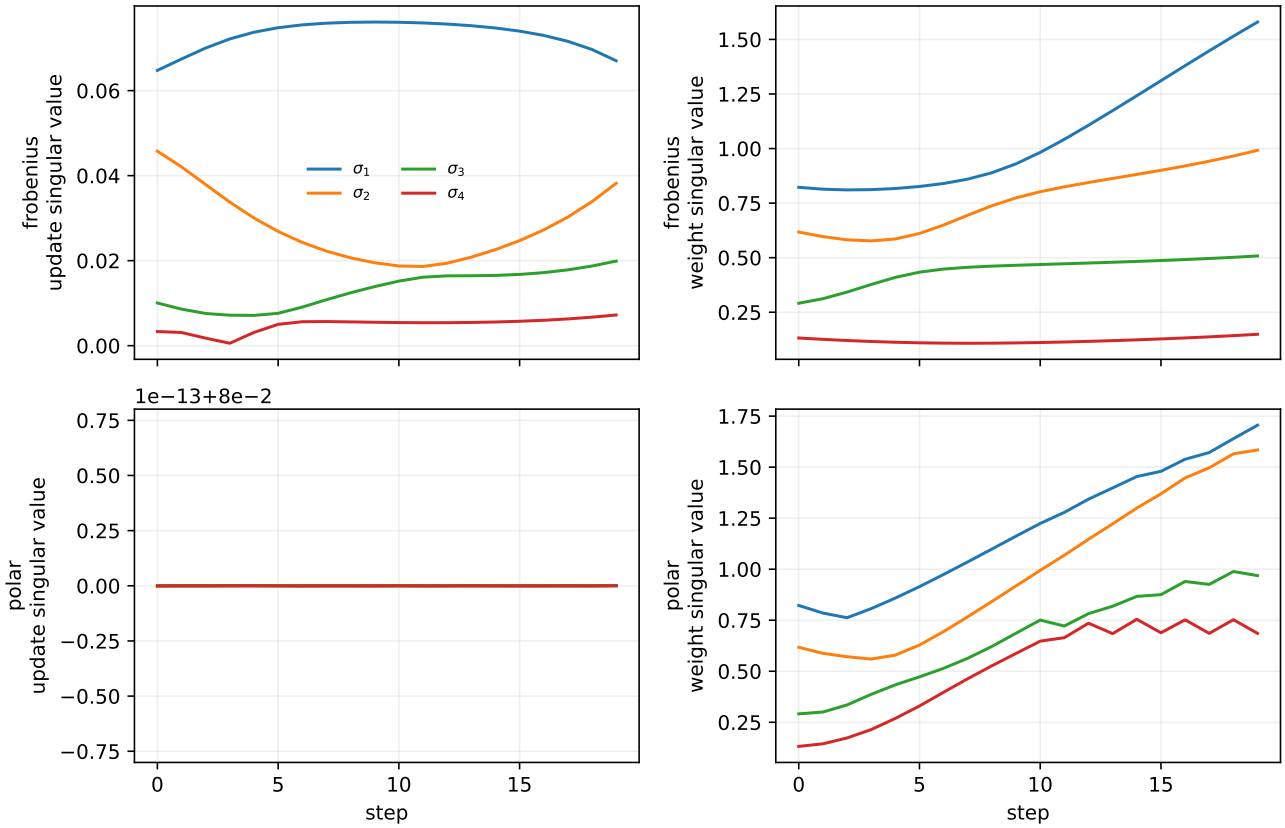


Figure 17.2.: Update and weight spectra are different objects. Polar steps give all four update singular directions the external scale 0.08 at every iteration, while the factor being updated remains anisotropic and evolves. Frobenius-normalized updates retain unequal singular values.

The polar update is an isometry on the four-dimensional input space at every step: its four singular values are all 0.08 up to rounding. Factor **A** is not. Its final singular values are approximately $(1.706, 1.584, 0.969, 0.686)$. "Orthogonalized update" therefore describes the step matrix, not the parameter matrix after the step and not the represented function **AB**.

The control also makes routing and scale explicit:

Parameter block	Routed transformation	Where magnitude re-enters	Audit boundary
matrix factor **A**	Frobenius or polar direction	external $alpha = 0.08$	width and layer scaling omitted
matrix factor **B**	same declared rule	external $alpha = 0.08$	square and rectangular blocks share no automatic scale law
a hypothetical bias vector	excluded	requires a separate vector rule	cannot be silently reshaped into a matrix

This is not an implementation footnote. A method that transforms selected matrices but routes embeddings, biases, scales, or output heads differently is a family of update rules. The paper autopsy must identify all of them.

17.4. One complete matrix-block step

For one matrix block $\mathbf{W} \in \mathbb{R}^{m\times n}$ with $m \geq n$, a complete routed step needs more than the polar idealization. Let

$$\mathbf{M}^{+} = \beta\mathbf{M} + (1-\beta)\mathbf{G}, \qquad \mathbf{X}_0 = \frac{\mathbf{M}^{+}}{\|\mathbf{M}^{+}\|_{\mathrm{F}}}, \qquad \mathbf{W}^{+} = \mathbf{W} - \eta s_{m,n}\mathbf{X}_T. \tag{17.4}$$

Here T polynomial iterations approximate the polar factor and $s_{m,n}$ restores a declared external scale. The following kernel exposes the whole block transition:

1. Update the momentum state.
2. Normalize the matrix entering the polynomial.
3. Apply a fixed number of Newton–Schulz steps.
4. Measure approximation error.
5. Restore the external scale and update the block.

```
# [1]
def matrix_block_step(weight, gradient, momentum, beta, step, scale, iterations):
    momentum = beta * momentum + (1 - beta) * gradient
    # [2]
    update = momentum / np.linalg.norm(momentum, "fro")
    # [3]
    for _ in range(iterations):
        update = 1.5 * update - 0.5 * update @ (update.T @ update)
    # [4]
    residual = np.linalg.norm(update.T @ update - np.eye(update.shape[1]), "fro")
    # [5]
    return weight - step * scale * update, momentum, residual

trial_weight = np.zeros_like(gradient)
trial_momentum = np.zeros_like(gradient)
trial_weight, trial_momentum, trial_residual = matrix_block_step(
    trial_weight, gradient, trial_momentum, 0.9, 0.08, 1.0, 12
)
assert trial_weight.shape == gradient.shape
assert np.isfinite(trial_residual)
```

For FP32 storage, $\mathbf{W}$ and persistent momentum already require $8mn$ bytes together. One Newton–Schulz step for a tall block forms an $n\times n$ Gram matrix and performs matrix products costing $O(mn^2)$ arithmetic. A materialized implementation also moves the block, momentum, iterates, and Gram temporary; fusion or tiling can change that traffic without changing Equation 17.4.

Contract field	This block step declares	Still requires measurement
normalization	Frobenius scale before iteration	overflow/underflow and low-precision accumulation
approximation	T steps and $\|X_T^{\mathsf{T}} X_T - I\|_{\mathrm{F}}$	error relative to the exact polar factor
persistent state	one $m \times n$ momentum matrix	optimizer state for excluded parameters
arithmetic	$O(Tmn^2)$ for $m \geq n$	achieved kernel rate
traffic	at least weights, gradients, momentum, and outputs	cache reuse, fusion, and communication
routing	this matrix block only	biases, scales, embeddings, and output blocks

The table is the algorithmic lens on the same update whose singular values supplied the geometric lens. A paper must own both.

17.5. Brand bridge

Muon applies momentum and then an approximate polar transformation to selected two-dimensional hidden-weight blocks (Jordan 2024; Liu et al. 2025). A useful shorthand is "sign descent on singular values," but it is a geometric surrogate, not a complete anatomy theorem: parameter routing, external scale, momentum, polynomial coefficients, dtype, and excluded parameters remain part of the method. Published compute-efficiency numbers are paper evidence tied to their hardware and baselines; this chapter does not reproduce them.

QJL (Zandieh et al. 2024) and TurboQuant (Zandieh et al. 2025) belong beside this discussion as a paper-audit branch about communication, quantization, and random projections—not as a second method tutorial. Their claims should be routed through C01's traffic boundary, C03's quantizer contract, and C07's uniformization question.

> ⚠ Named wrong answer: orthogonalized updates are scale free
>
> The polar map removes positive scalar magnitude from a full-rank matrix. A practical optimizer restores scale elsewhere and carries momentum and state. Audit the entire update, not one transformation.

17.6. The terminal capability: autopsy before adoption

The numbered core closes with a procedure rather than a winner. The Coda then uses the same discipline on a training incident. For a current paper, write a one-page autopsy:

1. **Claim:** What is proved, measured, interpreted, or merely proposed?
2. **Mathematical object:** Which matrix, estimator, norm, and population or sample quantity is named?
3. **Resolution of the theory:** Is the statement global, local, direction-aware, or finite-horizon?

4. **Dynamic regime:** Are the argument and experiments gradient- or noise-dominated, light- or heavy-tailed, fixed- or moving-curvature?
5. **Evidence culture and interface:** Which measured quantity corresponds to which assumption or conclusion?
6. **Estimator and comparison contract:** What are the target, reduction, denominator, sampling scheme, seeds, and baseline budget?
7. **Numerical and hardware contract:** Which dtype, scaling, state, bytes, device, and wall-clock protocol carry the endpoint?
8. **Assumption stress test:** Which conclusion survives when the most fragile assumption is changed?
9. **Discriminating control:** Which smallest control separates the proposed mechanism from its strongest rival?
10. **Transfer verdict:** State what is supported, what is not, and the cheapest next test.

This combines the course's Spring 2026 optimizer card with the theory-resolution and two-cultures maps. Credit the tutorial for its intellectual map and the primary paper for technical claims. Do not attribute the combined checklist to either source.

17.7. Check yourself

For singular values $(9, 3, 1, 0.2)$, what are the operator-ball and nuclear-ball minimum values, and which dual norms produce them? If a Newton–Schulz approximation has residual 10^{-3}, which claim does that support, and which two tempting optimizer claims remain unsupported?

17.8. Okay, so —

- **Inherited:** C01 supplies traffic provenance, C03 precision, C08 spectra, C12 curvature objects, and C13 regimes.
- **Changed:** an optimizer is read as a normed update contract plus approximation, routing, state, and evidence.
- **Instrumented:** one claim compares three dual geometries and a polynomial approximation; another separates update spectra from evolving weight spectra.
- **Established:** the norm chooses the steepest direction; the method name does not.
- **Unresolved:** When one loss spike activates several instruments, which control identifies the cause rather than merely detecting the event?

17.9. Sources and further reading

Matrix-norm steepest descent and recent optimizer connections are developed by Bernstein and Newhouse (2024). The original Muon technical account is Jordan (2024), with large-scale implementation evidence in Liu et al. (2025). The neighboring matrix-preconditioning artifacts are Shampoo (Gupta et al. 2018) and SOAP (Vyas et al. 2024). For the diagonal-moment and decoupled-decay baselines, see Kingma and Ba (2015) and Loshchilov and Hutter (2019). The quantized-projection audit branch starts with Zandieh et al. (2024) and continues with Zandieh et al. (2025).

Reading order. Start with Bernstein and Newhouse (2024) for dual-norm geometry, then audit Jordan (2024) and Liu et al. (2025) in that order: mathematical update contract first, scaling evidence second. If quantization is the target, read Zandieh et al. (2024) before Zandieh et al. (2025).

17.10. Exercises

1. **(Pencil.)** Prove the operator/nuclear duality rows of Equation 17.1 using von Neumann's trace inequality.
2. **(Code.)** Test Newton–Schulz across condition numbers, ranks, and dtypes. Separate polar error, orthogonality residual, and elapsed time.
3. **(Pencil.)** For the factorized control, derive both factor gradients and show why an isometric update to each factor does not imply an isometric update to their product.
4. **(Code.)** Add vector biases and per-layer scale factors to the trace. Implement an explicit routing table and, under equal nominal learning rates, compare update spectra, weight spectra, state bytes, final loss, and each block's induced output change or Jacobian-Gram contribution. Explain why equal parameter-step norms need not produce equal function-space steps.
5. **(Audit.)** **Paper audit:** Apply all ten fields of the Paper Autopsy Protocol to one routed matrix-update, quantization, or edge-of-stability paper. For the quantization branch, begin with Zandieh et al. (2024) or Zandieh et al. (2025) rather than a method name alone. End by naming the single assumption whose failure would most weaken its central claim.
6. **(Audit.)** **Act checkpoint — complete Incident Card.** For a run rather than a paper, keep this fixed order: **Symptom → Prediction → Contract → Precision control → Curvature control → Estimator control → Spectrum locator → State control → Verdict → Corrective control.** Complete the Act II assignment. **Route:** Act checkpoint. **Estimated time:** 90 minutes. **Deliverable:** one complete card plus the cheapest discriminating rerun. **Hint:** an activated instrument may detect the event without identifying its cause.

One Spike, Four Suspects

NS · Numerical stability · RS · Random-matrix spectra · LG · Landscape and update geometry · SG · The sub-Gaussian safe zone

The first page opened with an incident: one loss spike, at least four plausible causes. The intervening chapters built instruments one at a time. Now they must work together.

A seeded 6×4 linear map is trained for 120 mini-batch steps on 256 observations. The batch size is 32, the step is 0.12, and the same eight input batches repeat in a fixed order. At step 60, the full-data loss jumps from about 0.00514 to 0.310, a factor of 60.2. It then recovers toward its earlier level.

Nothing about that scalar trace identifies the cause.

> **!** Prediction
>
> Rank these four suspects before opening the panel: **precision**, **curvature**, **estimator/data**, and **implementation state**. The last suspect will be tested through both update geometry and normalization state. For your top choice, name one control that could acquit it.

Open the panel

The experiment is laptop-scale and paired. Its primary trajectory uses float64; a float32 execution uses the same data, batch order, and incident. The clean objective is matrix least squares,

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2m}\,\|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_{\mathrm{F}}^{2}\,, \qquad \nabla^{2}\mathcal{L} = \frac{1}{m}\mathbf{X}^{\mathsf{T}}\mathbf{X} \otimes \mathbf{I}_{4}.$$

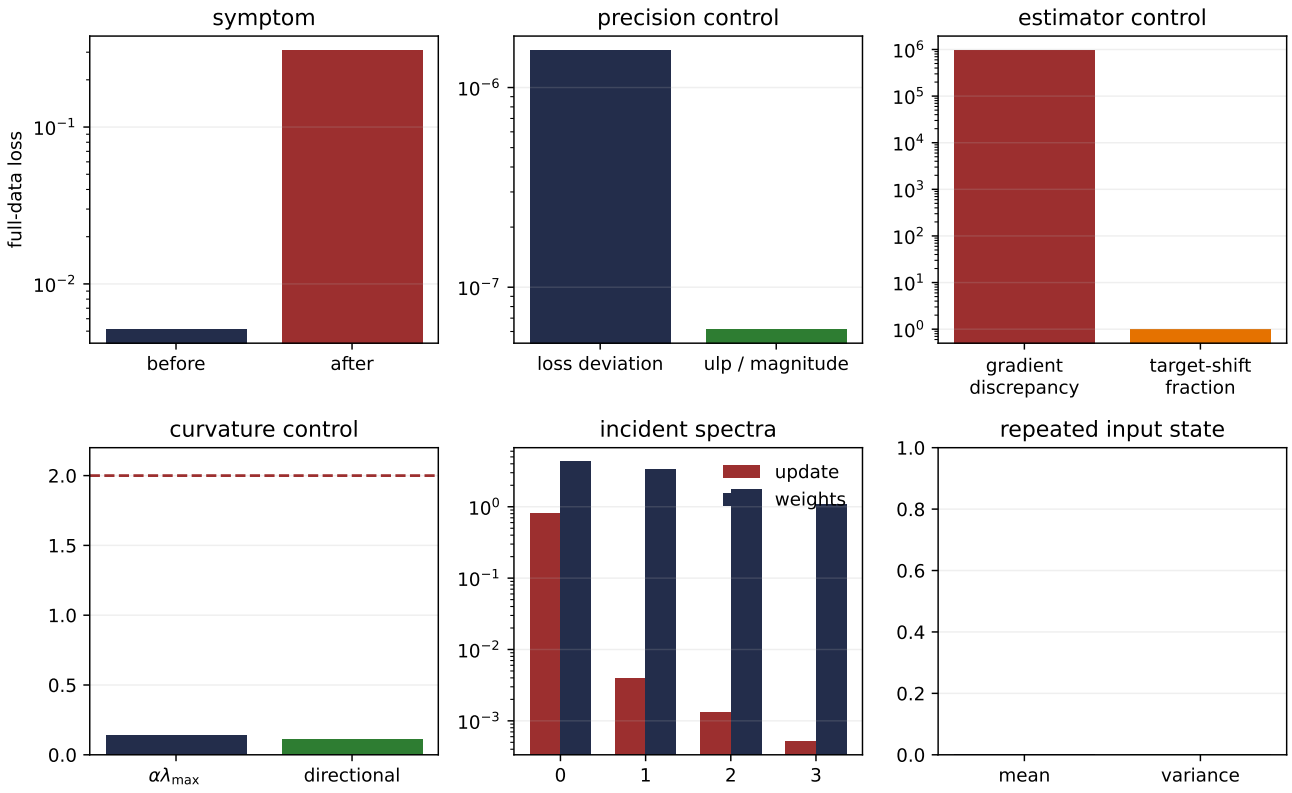
Every diagnostic uses the same step index. The claim record stores the estimator target, reduction, denominator, dtype pair, seed, and wheel digest.

1. Open the content-addressed incident claim through the verifier.
2. Extract the registered result.
3. Extract one control for each suspect.
4. Plot the spike and the complete diagnostic panel.
5. Check the incident step and the discriminating target-shift statistic.

```
# [1]
incident = verify_claim("coda-incident-001", expected_harness=pin)
# [2]
result = incident["result"]

# [3]
regime = result["regime_control"]
```

```
controls = tuple(
    result[name]
    for name in (
        "precision_control",
        "regime_control",
        "curvature_control",
        "spectrum_control",
        "normalization_state_control",
    )
)

# [4]
incident_figure(result)

# [5]
assert result["incident_step"] == 60
assert result["loss_spike_factor"] > 60
assert len(controls) == 5
assert regime["fraction_of_discrepancy_from_engineered_target_shift"] > 0.99
print(f"step 60 loss spike: {result['loss_spike_factor']:.1f}x")
print(
    "target-shift share of gradient discrepancy: "
    f"{100 * regime['fraction_of_discrepancy_from_engineered_target_shift']:.1f}%"
)
```

```
step 60 loss spike: 60.2x
target-shift share of gradient discrepancy: 99.8%
```



Claim `coda-incident-001` · seed: 3287492823 · dtype: FP64 with paired FP32 control · device: laptop CPU · estimator: same-step six-instrument incident panel · artifact

Figure Coda.1 — One engineered loss spike activates the complete diagnostic panel. Precision, curvature, and repeated-state controls acquit their named mechanisms; the update spectrum detects the event without explaining it; a same-input target control attributes 99.8% of the gradient discrepancy to the intervention. The panel diagnoses this pinned incident, not every loss spike.

The panels are not five votes. They are a sequence of attempts to falsify a cause.

Suspect 1: precision

The largest state or update magnitude is 1.935. The FP32 spacing at that scale is about 1.19×10^{-7}. Across the complete paired trajectories, the maximum relative difference between float32 and float64 losses is only 1.54×10^{-6}. Both runs exhibit the spike.

That does not prove the computation is numerically exact. It acquits range and local-grid resolution as explanations for a sixty-fold event in this run. A precision diagnosis has to predict the observed scale of the failure; here it does not.

Suspect 2: curvature

Because the displayed objective is linear least squares, its Hessian is constant. The top curvature is 1.155, so $\alpha\lambda_{\max} = 0.139$. The incident update's directional value is $\alpha\lambda_{\text{dir}} = 0.110$. Both remain far below the exact fixed-quadratic boundary at two.

C16 taught us not to use one local eigenvalue as a global theorem. This case is stronger: the landscape really is quadratic and its curvature does not move. The control predicts no curvature crossing, and none occurred.

Suspect 3: implementation state

Normalization state

The same eight input batches repeat every eight steps. The incident batch has the same coordinate means and variances as it had one cycle earlier: both maximum differences are exactly zero in the represented data. Any input normalizer using those declared statistics would receive the same state.

This does not acquit every stateful implementation. It eliminates the named rival: a changed input-statistic state at the incident step.

Update geometry

The leading update singular value is 176 times its value at the preceding step, and the incident update is nearly rank one. The spectrum therefore detects the event.

But detection is not diagnosis.

The current weight singular values remain broad, and nothing in the spectrum alone says whether the large direction came from curvature, data, arithmetic, or a routing bug.

C17's instrument is a locator. It identifies which matrix and directions changed; it must be paired with a mechanism-specific control.

Suspect 4: estimator and data

At the incident step, the realized batch gradient differs from the clean full gradient by about 9.74×10^5 times the squared full-gradient norm. More decisively, 99.8% of that discrepancy is reproduced by comparing the actual incident target with the clean target on the **same input batch**.

The sealed intervention can now be opened: at step 60, the four target coordinates in one batch were shifted by a fixed signed amount of magnitude eight. Inputs, batch order, step size, curvature, dtype, and normalization statistics were unchanged. This is not merely a high-variance draw under the declared clean estimator. Its conditional mean targets a different objective for one step.

Clipping the update could reduce the visible spike, but C13 already tells us what that repair would and would not do. It would bound the consequence while leaving the target violation unexplained. The correct incident response is to trace the batch target and its provenance.

Completed incident report

Field	Finding	Verdict
1. Symptom	full loss increases by 60.2× after step 60	established
2. Prediction	precision, curvature, estimator/data, and implementation state are predeclared rivals	each retains an acquitting control before the panel opens
3. Contract	seeded 6×4 linear map; 256 observations; batch 32; step 0.12; fixed batch order	target, axes, denominator, dtype pair, seed, device, and wheel are declared
4. Precision control	paired FP32/FP64 loss deviation at most 1.54×10^{-6}	acquitted for this event
5. Curvature control	$\alpha\lambda_{\max} = 0.139$; directional value 0.110	acquitted
6. Estimator control	99.8% of gradient discrepancy traced to target shift	cause identified
7. Spectrum locator	leading update singular value jumps 176×	detects only
8. State control	repeated input-batch mean and variance unchanged	named normalization-state rival acquitted
9. Verdict	the step-60 batch targets a different objective	cause identified; precision, curvature, and named state rival acquitted
10. Corrective control	restore clean target; rerun same seed and batch order	spike removed by construction

A blank copy and three cumulative act assignments are in the final appendix. Its second incident is intentionally unrevealed and may remain unidentifiable under the supplied instruments.

The first page's question now has an answer and a method. A loss trace proposes an incident; it does not name one. Geometry supplies candidate directions, dynamics supplies time-indexed boundaries, and the machine supplies arithmetic and state contracts. The estimator discipline binds them to the quantity the code actually computed.

That is what it means to make a training run diagnosable — and therefore trainable.

Beyond This Volume

The Coda closes this book's promise: a training failure becomes diagnosable when geometry, dynamics, arithmetic, state, and estimator controls are read together. DS 6210 continues beyond that endpoint. Those later topics are not missing chapters from this volume; they are branches that reuse its trunk.

Similarity-weighted aggregation and kernel geometry

An all-pairs similarity matrix raises three questions already owned here: which high-dimensional alignments are typical, which normalization keeps weights in a usable numerical range, and which Jacobian paths preserve signal. C05–C10 supply concentration and spectral baselines; C14–C15 supply criticality and normalization diagnostics.

The standard literature calls this primitive **attention**; the canonical self-attention architecture is Vaswani et al. (2017). The sibling volume supplies its assembly mechanics; the continuation of DS 6210 asks which kernel geometry, estimator, and derivative-path claims survive at finite width and precision.

IO-aware exact algorithms

An exact all-pairs result need not materialize the full intermediate matrix. C01 returns through arithmetic intensity and traffic lower bounds. C02 returns through bounded mergeable state. C03 and C15 return through stable online normalizers. The central audit is whether tiling and recomputation preserve the mathematical output while changing the bytes moved.

FlashAttention (Dao et al. 2022) is the standard named case. It belongs after the traffic model and online-normalizer invariant, not before them.

Parallel recurrence and associativity

A recurrence can sometimes be represented by an associative summary and evaluated by a parallel prefix scan. C02 supplies the state-and-merge question; C11 supplies retained-state and recomputation accounting; C03 supplies the warning that algebraic associativity does not imply floating-point order independence. The paper-level question is therefore both algebraic and physical: what summary composes, and what numerical result does a different reduction tree produce?

Lazy and feature-learning regimes

C04's fixed quadratic, C08's adaptive directions, C12's curvature objects, and C13's stochastic regimes form the control for asking whether training mostly changes coefficients in a nearly fixed feature map or changes the features themselves. C16 warns that the relevant operator moves with the trajectory. C17 asks whether the update geometry encourages that movement or suppresses it.

Under a declared tangent-kernel scaling and time horizon, the limit that makes training linearized also suppresses feature movement; finite-width corrections restore it at the order carried by the expansion (Jacot et al. 2018; Lee et al. 2019; Chizat et al. 2019; Roberts and Yaida 2022). Different parameterizations and simultaneous depth–width limits can retain movement, so "infinite width" is not itself a regime diagnosis.

These topics remain course synthesis and research-facing Paper audits. Keeping them outside the numbered core preserves the book's contract: build the instruments once, then recognize their return quickly in a new method.

Provenance and Acknowledgements

This book grows from DS 6210 materials developed and taught at the University of Virginia in Spring 2026: lecture derivations, assignments, optimizer cards, journal-club prompts, numerical studies, and the patterns revealed by student work. Those materials supplied many opening phenomena and much of the reveal order. Book prose was rewritten to stand without the classroom, and every load-bearing mathematical or empirical claim was independently derived, executed, or routed to a primary source.

External tutorials are credited as intellectual maps rather than silent technical authorities. In particular, the 2026 ICML optimization-theory and probabilistic-numerics tutorials helped locate useful seams across chapters. The book's combined phenomenon-first sequence, diagnostic ledgers, and Paper Autopsy Protocol are editorial syntheses; the technical claims inside them remain attributed to their primary literature.

Executable observations carry claim IDs, provenance classes, pinned harness wheels, and content digests. Negative results and rejected explanations are retained when they delimit a claim. Student work remains private editorial evidence unless permission and attribution are recorded explicitly.

The project also owes its production discipline to defects discovered while building the sibling volume, *Deep Learning: Making It Learnable*. Stable anchors, content-equivalent editions, estimator declarations, source audits, and mechanical promise checks are not ornamental infrastructure; they are the conditions under which a technical book can remain inspectable as it evolves.

References

Austin, Jacob, Sholto Douglas, Roy Frostig, et al. 2025. *How to Scale Your Model: A Systems View of LLMs on TPUs*. Online book. https://jax-ml.github.io/scaling-book/.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. *Layer Normalization*. https://doi.org/10.48550/arXiv.1607.06450.

Baik, Jinho, and Jack W. Silverstein. 2006. "Eigenvalues of Large Sample Covariance Matrices of Spiked Population Models." *Journal of Multivariate Analysis* 97 (6): 1382–408. https://doi.org/10.1016/j.jmva.2005.08.003.

Baraniuk, Richard, Mark Davenport, Ronald DeVore, and Michael Wakin. 2008. "A Simple Proof of the Restricted Isometry Property for Random Matrices." *Constructive Approximation* 28 (3): 253–63. https://doi.org/10.1007/s00365-007-9003-x.

Baur, Walter, and Volker Strassen. 1983. "The Complexity of Partial Derivatives." *Theoretical Computer Science* 22 (3): 317–30. https://doi.org/10.1016/0304-3975(83)90110-X.

Baydin, Atilim Gunes, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. 2018. "Automatic Differentiation in Machine Learning: A Survey." *Journal of Machine Learning Research* 18 (153): 1–43. https://www.jmlr.org/papers/v18/17-468.html.

Bernstein, Jeremy, and Laker Newhouse. 2024. *Old Optimizer, New Norm: An Anthology*. https://doi.org/10.48550/arXiv.2409.20325.

Bottou, Léon, Frank E. Curtis, and Jorge Nocedal. 2018. "Optimization Methods for Large-Scale Machine Learning." *SIAM Review* 60 (2): 223–311. https://doi.org/10.1137/16M1080173.

Chan, Tony F., Gene H. Golub, and Randall J. LeVeque. 1983. "Algorithms for Computing the Sample Variance: Analysis and Recommendations." *The American Statistician* 37 (3): 242–47. https://doi.org/10.1080/00031305.1983.10483115.

Chen, Tianqi, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. *Training Deep Nets with Sublinear Memory Cost*. https://doi.org/10.48550/arXiv.1604.06174.

Chernoff, Herman. 1952. "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations." *The Annals of Mathematical Statistics* 23 (4): 493–507. https://doi.org/10.1214/aoms/1177729330.

Chizat, Lénaïc, Edouard Oyallon, and Francis Bach. 2019. "On Lazy Training in Differentiable Programming." *Advances in Neural Information Processing Systems 32*. https://papers.nips.cc/paper/2019/hash/ae614c557843b1df326cb29c57225459-Abstract.html.

Cho, Youngmin, and Lawrence K. Saul. 2009. "Kernel Methods for Deep Learning." *Advances in Neural Information Processing Systems 22*, 342–50. https://proceedings.neurips.cc/paper/2009/hash/5751ec3e9a4feab575962e78e006250d-Abstract.html.

Cohen, Jeremy M., Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. 2021. "Gradient Descent on Neural Networks Typically Occurs at the Edge of Stability." *International Conference on Learning Representations*. https://openreview.net/forum?id=jh-rTtvkGeM.

Connolly, Michael P., Nicholas J. Higham, and Theo Mary. 2021. "Stochastic Rounding and Its Probabilistic Backward Error Analysis." *SIAM Journal on Scientific Computing* 43 (1): A566–85. https://doi.org/10.1137/20M1334796.

Dao, Tri, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness." *Advances in Neural Information Processing Systems 35*, 16344–59. https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Glorot, Xavier, and Yoshua Bengio. 2010. "Understanding the Difficulty of Training Deep Feedforward Neural Networks." *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–56. https://proceedings.mlr.press/v9/glorot10a.html.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. https://www.deeplearningbook.org/.

Griewank, Andreas. 1992. "Achieving Logarithmic Growth of Temporal and Spatial Complexity in Reverse Automatic Differentiation." *Optimization Methods and Software* 1 (1): 35–54. https://doi.org/10.1080/10556789208805505.

Griewank, Andreas, and Andrea Walther. 2008. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. 2nd ed. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9780898717761.

Gupta, Vineet, Tomer Koren, and Yoram Singer. 2018. "Shampoo: Preconditioned Stochastic Tensor Optimization." *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of machine learning research, vol. 80: 1842–50. https://proceedings.mlr.press/v80/gupta18a.html.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." *Proceedings of the IEEE International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2015.123.

Hennig, Philipp. 2015. "Probabilistic Interpretation of Linear Solvers." *SIAM Journal on Optimization* 25 (1): 234–60. https://doi.org/10.1137/140955501.

Hennig, Philipp, and Martin Kiefel. 2013. "Quasi-Newton Methods: A New Direction." *Journal of Machine Learning Research* 14: 843–65. https://www.jmlr.org/papers/v14/hennig13a.html.

Hennig, Philipp, Marvin Pförtner, and Tim Weiland. 2026. *Probabilistic Numerics: Computation Is Machine Learning*. Tutorial at the 43rd International Conference on Machine Learning. https://icml.cc/Downloads/2026.

Higham, Nicholas J. 2002. *Accuracy and Stability of Numerical Algorithms*. 2nd ed. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9780898718027.

Hoeffding, Wassily. 1963. "Probability Inequalities for Sums of Bounded Random Variables." *Journal of the American Statistical Association* 58 (301): 13–30. https://doi.org/10.1080/01621459.1963.10500830.

IEEE Standard for Floating-Point Arithmetic, Pub. L. Nos. IEEE Std 754-2019 (2019). https://doi.org/10.1109/IEEESTD.2019.8766229.

Ioffe, Sergey, and Christian Szegedy. 2015. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." *Proceedings of the 32nd International Conference on Machine Learning*, Proceedings of machine learning research, vol. 37: 448–56. https://proceedings.mlr.press/v37/ioffe15.html.

Jacot, Arthur, Franck Gabriel, and Clément Hongler. 2018. "Neural Tangent Kernel: Convergence and Generalization in Neural Networks." *Advances in Neural Information Processing Systems 31*. https://papers.nips.cc/paper/2018/hash/5a4be1fa34e62bb8a6ec6b7ae4572128-Abstract.html.

Jordan, Keller. 2024. *Muon: An Optimizer for Hidden Layers in Neural Networks*. https://kellerjordan.github.io/posts/muon/.

Kingma, Diederik P., and Jimmy Ba. 2015. "Adam: A Method for Stochastic Optimization." *International Conference on Learning Representations*. https://doi.org/10.48550/arXiv.1412.6980.

Koloskova, Anastasia, Hadrien Hendrikx, and Sebastian U. Stich. 2023. "Revisiting Gradient Clipping: Stochastic Bias and Tight Convergence Guarantees." *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of machine learning research, vol. 202: 17343–63. https://proceedings.mlr.press/v202/koloskova23a.html.

Koltchinskii, Vladimir, and Karim Lounici. 2017. "Concentration Inequalities and Moment Bounds for Sample Covariance Operators." *Bernoulli* 23 (1): 110–33. https://doi.org/10.3150/15-BEJ730.

Kunstner, Frederik, Lukas Balles, and Philipp Hennig. 2019. "Limitations of the Empirical Fisher Approximation for Natural Gradient Descent." *Advances in Neural Information Processing Systems 32*. https://proceedings.neurips.cc/paper/2019/hash/46a558d97954d0692411c861cf78ef79-Abstract.html.

Lee, Jaehoon, Lechao Xiao, Samuel S. Schoenholz, et al. 2019. "Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent." *Advances in Neural Information Processing Systems 32*. https://papers.nips.cc/paper/2019/hash/0d1a9651497a38d8b1c3871c84528bd4-Abstract.html.

Linnainmaa, Seppo. 1970. "The Representation of the Cumulative Rounding Error of an Algorithm as a Taylor Expansion of the Local Rounding Errors." Master's thesis, University of Helsinki. https://kansalliskirjasto.finna.fi/Record/helka.9933382303506253.

Liu, Jingyuan et al. 2025. *Muon Is Scalable for LLM Training*. https://doi.org/10.48550/arXiv.2502.16982.

Loshchilov, Ilya, and Frank Hutter. 2019. "Decoupled Weight Decay Regularization." *International Conference on Learning Representations*. https://doi.org/10.48550/arXiv.1711.05101.

Marchenko, V. A., and L. A. Pastur. 1967. "Distribution of Eigenvalues for Some Sets of Random Matrices." *Mathematics of the USSR-Sbornik* 1 (4): 457–83. https://doi.org/10.1070/SM1967v001n04ABEH001994.

Martens, James. 2020. "New Insights and Perspectives on the Natural Gradient Method." *Journal of Machine Learning Research* 21 (146): 1–76. https://www.jmlr.org/papers/v21/17-678.html.

Micikevicius, Paulius, Dusan Stosic, Neil Burgess, et al. 2022. "FP8 Formats for Deep Learning." *arXiv Preprint arXiv:2209.05433*, ahead of print. https://doi.org/10.48550/arXiv.2209.05433.

Miyato, Takeru, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. "Spectral Normalization for Generative Adversarial Networks." *International Conference on Learning Representations*. https://openreview.net/forum?id=B1QRgziT-.

Nesterov, Yurii. 2004. *Introductory Lectures on Convex Optimization: A Basic Course*. Vol. 87. Applied Optimization. Kluwer Academic Publishers. https://doi.org/10.1007/978-1-4419-8853-9.

Pearlmutter, Barak A. 1994. "Fast Exact Multiplication by the Hessian." *Neural Computation* 6 (1): 147–60. https://doi.org/10.1162/neco.1994.6.1.147.

Pennington, Jeffrey, Samuel Schoenholz, and Surya Ganguli. 2017. "Resurrecting the Sigmoid in Deep Learning Through Dynamical Isometry: Theory and Practice." *Advances in Neural Information Processing Systems 30*. https://papers.nips.cc/paper/2017/hash/d9fc0cdb67638d50f411432d0d41d0ba-Abstract.html.

Polyak, Boris T. 1964. "Some Methods of Speeding up the Convergence of Iteration Methods." *USSR Computational Mathematics and Mathematical Physics* 4 (5): 1–17. https://doi.org/10.1016/0041-5553(64)90137-5.

Poole, Ben, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. 2016. "Exponential Expressivity in Deep Neural Networks Through Transient Chaos." *Advances in Neural Information Processing Systems 29*. https://papers.nips.cc/paper/2016/hash/148510031349642de5ca0c544f31b2ef-Abstract.html.

Robbins, Herbert, and Sutton Monro. 1951. "A Stochastic Approximation Method." *The Annals of Mathematical Statistics* 22 (3): 400–407. https://doi.org/10.1214/aoms/1177729586.

Roberts, Daniel A., and Sho Yaida. 2022. *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks*. Cambridge University Press. https://doi.org/10.1017/9781009023405.

Santurkar, Shibani, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. 2018. "How Does Batch Normalization Help Optimization?" *Advances in Neural Information Processing Systems 31*. https://papers.neurips.cc/paper/2018/hash/905056c1ac1dad141560467e0a99e1cf-Abstract.html.

Saxe, Andrew M., James L. McClelland, and Surya Ganguli. 2014. "Exact Solutions to the Nonlinear

Dynamics of Learning in Deep Linear Neural Networks." *International Conference on Learning Representations*. https://doi.org/10.48550/arXiv.1312.6120.

Schmidt, Mark. 2026. *Is Numerical Optimization Theory Irrelevant to Machine Learning Practice in 2026?* Tutorial at the 43rd International Conference on Machine Learning. https://www.cs.ubc.ca/~schmidtm/Documents/2026_ICML_Tutorial.pdf.

Schoenholz, Samuel S., Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. 2017. "Deep Information Propagation." *International Conference on Learning Representations*. https://openreview.net/forum?id=H1W1UN9gg.

Simsekli, Umut, Levent Sagun, and Mert Gurbuzbalaban. 2019. "A Tail-Index Analysis of Stochastic Gradient Noise in Deep Neural Networks." *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of machine learning research, vol. 97: 5827–37. https://proceedings.mlr.press/v97/simsekli19a.html.

Tazi, Nouamane, Ferdinand Mom, Haojun Zhao, et al. 2025. *The Ultra-Scale Playbook: Training LLMs on GPU Clusters*. Online book. https://huggingface.co/spaces/nanotron/ultrascale-playbook.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, et al. 2017. "Attention Is All You Need." *Advances in Neural Information Processing Systems 30*. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Vershynin, Roman. 2026. *High-Dimensional Probability: An Introduction with Applications in Data Science*. 2nd ed. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press. https://www.math.uci.edu/~rvershyn/papers/HDP-book/HDP-book.html.

Vyas, Nikhil, Depen Morwani, Rosie Zhao, et al. 2024. *SOAP: Improving and Stabilizing Shampoo Using Adam*. https://doi.org/10.48550/arXiv.2409.11321.

Welford, B. P. 1962. "Note on a Method for Calculating Corrected Sums of Squares and Products." *Technometrics* 4 (3): 419–20. https://doi.org/10.1080/00401706.1962.10490022.

Williams, Christopher K. I. 1997. "Computing with Infinite Networks." *Advances in Neural Information Processing Systems 9*, 295–301. https://papers.nips.cc/paper/1996/hash/ae5e3ce40e0404a45ecacaaf05e5f735-Abstract.html.

Williams, Samuel, Andrew Waterman, and David Patterson. 2009. *Roofline: An Insightful Visual Performance Model for Multicore Architectures*. UCB/EECS-2008-134. University of California, Berkeley. https://digicoll.lib.berkeley.edu/record/136692.

Zandieh, Amir, Majid Daliri, Majid Hadian, and Vahab Mirrokni. 2025. *TurboQuant: Online Vector Quantization with Near-Optimal Distortion Rate*. https://doi.org/10.48550/arXiv.2504.19874.

Zandieh, Amir, Majid Daliri, and Insu Han. 2024. *QJL: 1-Bit Quantized JL Transform for KV Cache Quantization with Zero Overhead*. https://doi.org/10.48550/arXiv.2406.03482.

Zhang, Biao, and Rico Sennrich. 2019. "Root Mean Square Layer Normalization." *Advances in Neural Information Processing Systems 32*. https://papers.nips.cc/paper/2019/hash/

1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. "Understanding Deep Learning Requires Rethinking Generalization." *International Conference on Learning Representations*. https://openreview.net/forum?id=Sy8gdB9xx.

A. Notation Covenant

Notation is a public interface between this book and its sibling. Core symbols retain their meanings; this volume adds symbols when the new diagnostic objects require them.

A.1. Inherited core

Object	Notation	Contract
Scalar	x	Italic lowercase unless a field convention requires otherwise
Vector	$\mathbf{x}$	Bold; a column on paper
Matrix or batched array	$\mathbf{A}$	Bold; shape completes the type
Aggregate objective	$\mathcal{L}$	A local loss is ℓ_i
Expectation	$\mathbb{E}$	Population and conditioning are stated
Variance	Var	Denominator convention is stated
Covariance	Cov	Sample and population objects are distinguished
Norm	$\|\cdot\|$	The subscript names the norm when ambiguity matters
Definition	$:=$	Equality remains $=$
Approximation	$\approx$	Never used as silent equality
Algorithmic update	$\leftarrow$	Distinct from a mathematical identity

Code stores a batch by rows unless a chapter explicitly declares otherwise. Every non-obvious matrix product states dimensions first, and every reduction names its axes.

A.2. New in this volume

Object	Notation	First use
Centered sum of squares	$M_{2,n}$	Streaming state
Operator norm	$\|\mathbf{A}\|_{\mathrm{op}}$	Random operators
Largest singular value	$\sigma_{\max}(\mathbf{A})$	Random operators
Smallest singular value	$\sigma_{\min}(\mathbf{A})$	Random operators

Object	Notation	First use
Frobenius norm	$\|\mathbf{A}\|_{\mathrm{F}}$	Random operators
Epsilon-net	$\mathcal{N}_\epsilon$	Continuous extrema
Covering number	$N(K, d, \epsilon)$	Continuous extrema
Sub-Gaussian Orlicz norm	$\|X\|_{\psi_2}$	Safe-zone concentration
Sub-exponential Orlicz norm	$\|X\|_{\psi_1}$	Products and squares
Unit roundoff	u	Floating-point contract
Unit in the last place	$\mathrm{ulp}(x)$	Floating-point contract
Spectral condition number	κ	Quadratic modes
Spectral radius	$\rho(\mathbf{A})$	Quadratic modes
Sub-Gaussian MGF proxy scale	σ	Safe-zone concentration
Sample covariance	$\widehat{\mathbf{\Sigma}}$	Spectral nulls
Empirical spectral distribution	$\mu_{\mathbf{M}}$	Spectral nulls
Aspect ratio	$\gamma = n/m$	Spectral nulls
Marchenko–Pastur edges	λ_-, λ_+	Spectral nulls
Population spike	β	Spectral nulls
Effective rank	$r_{\mathrm{eff}}(\mathbf{\Sigma})$	Effective dimension
Stable rank	$\mathrm{sr}(\mathbf{A})$	Effective dimension
Jacobian	$\mathbf{J}_F(\mathbf{x})$	Reverse accumulation
Jacobian–vector product	$\mathbf{J}_F(\mathbf{x})\mathbf{v}$	Reverse accumulation
Vector–Jacobian product	$\mathbf{u}^\mathsf{T}\mathbf{J}_F(\mathbf{x})$	Reverse accumulation
Node cotangent	$\bar{z}_v$	Reverse accumulation
Realized Hessian	$\mathbf{H}$	Curvature objects
Generalized Gauss–Newton matrix	$\mathbf{G}$	Curvature objects
Model Fisher information	$\mathbf{F}$	Curvature objects
Empirical Fisher	$\widetilde{\mathbf{F}}$	Curvature objects
Model-curvature residual	$\mathbf{R}$	Curvature objects
Arithmetic intensity	$I = F/Q$	Runtime model
Machine balance	$I_* = P_{\max}/B$	Runtime model
Conditional noise-to-signal ratio	χ_k	Stochastic regimes
Centering projector	$\mathbf{P}$	Coordinate-wise normalization
Nuclear norm	$\|\mathbf{A}\|_*$	Matrix update geometry
Polar factor	$\mathrm{polar}(\mathbf{A})$	Matrix update geometry

The machine-readable source is `contracts/notation-covenant.yml`. A notation change is incomplete until that covenant, the HTML macros, the PDF macros, and this appendix agree.

B. Rosetta: Primitive to Literature Name

This appendix is the sanctioned bridge between the book's mechanism-first language and the names used in papers, libraries, and search queries. A name identifies a family; it does not remove the need to state axes, precision, state, or assumptions.

Primitive in this book	Standard name or aliases	Notation and diagnostic contract	Unit
Coordinate-wise or token-wise normalization	LayerNorm; RMSNorm	Axes; centering; scale; ε placement; Ba et al. (2016); Zhang and Sennrich (2019)	Section 15.1
Batch-and-spatial coordinate normalization	BatchNorm	Running and batch statistics are distinct estimators; Ioffe and Szegedy (2015)	Section 15.2
Variance-preserving initialization	Xavier/Glorot; He/Kaiming	Variance convention; nonlinearity assumptions; Glorot and Bengio (2010); He et al. (2015)	Section 14.1
Diagonal moment preconditioning	Adam; AdamW	Moments; bias correction; decay; ε placement; Kingma and Ba (2015); Loshchilov and Hutter (2019)	C13/C17
Gradient-dominated or noise-dominated regime	signal-to-noise regime; critical batch regime	$\chi_k = \sigma_k^2/\|\nabla f(w_k)\|^2$; conditional estimator, target, and finite-moment contract	Section 13.2
Similarity-weighted aggregation	attention; self-attention	Scale; normalized axis; mask; materialization; Vaswani et al. (2017)	Beyond this volume
IO-aware exact all-pairs aggregation	FlashAttention	Exact map; finite-precision order; Dao et al. (2022)	Beyond this volume
Matrix-aware or orthogonalized update	Muon; Shampoo; SOAP	Routing; iteration count; scale; state; Jordan (2024); Gupta et al. (2018); Vyas et al. (2024)	Section 17.1
Quantized random projection or update sketch	QJL; TurboQuant	Quantizer; estimator; sketch size; hardware endpoint; Zandieh et al. (2024); Zandieh et al. (2025)	Section 17.5
Reverse accumulation on a computation graph	reverse-mode automatic differentiation; backpropagation	JVP/VJP orientation; fan-out sums; retained primal state	C11

The unit column records where the full entry lives, not where the literature name is owned. A specific published artifact carries its primary citation in the row; a generic technique may instead point to the unit that owns its contract. Search aliases and those citations are tested in CI.

C. Executable Claim Contract

The book separates learner-visible code into an equation, a mechanism kernel, and build orchestration. Hashing paths, activating a vendored wheel, loading a JSON record, and checking its content digest are essential to publication but are not the semantic change a new chapter teaches.

The printed contract is therefore:

Surface	Kept on the page
equation	the mathematical object, with shapes and reductions
mechanism kernel	normally 10–25 lines exposing the chapter's new operation
critical assertions	two or three checks that protect the interpretation
provenance strip	claim ID, provenance class, seed, dtype, device, estimator, and artifact
full harness	linked source and collapsible HTML where useful; not repeatedly printed

Chapter source still executes the full verification. It calls the centralized interfaces below from a hidden setup cell:

```
pin = activate_harness("ch-13", expected_wheel_sha256)
claim = verify_claim("c13-regime-001", expected_harness=pin)
```

`activate_harness` verifies the manifest identity and wheel SHA-256 before placing the exact vendored wheel on the import path. `verify_claim` checks the claim registry, artifact identity, canonical payload digest, and wheel agreement. Removing those lines from repeated printed panels changes no assertion and no executed output.

Historical rolling-draft wheels are identified by their committed SHA-256; the book does not invent source tags after the fact. Beginning with `harness-ch-14-r2-v1`, each new wheel also records an annotated source tag and commit. The two cases are explicit in the manifest and checked by CI.

The complete implementation is available in `book_support/claims.py`. Claim records are browsable under `artifacts/claims/`, and historical harness pins under `artifacts/harness/`.

C.1. Reading a provenance strip

The strip is a locator, not evidence by itself:

- **claim** names the proposition being checked;

- **class** distinguishes laptop reproduction, CPU simulation, and a pinned external endpoint;
- **seed** names a stream but never substitutes for a seed panel;
- **dtype/device** scope the execution;
- **estimator** names the target, reduction, and denominator;
- **artifact** links the complete record and its environment sidecar.

The chapter remains responsible for the mathematical and empirical interpretation. A valid digest can prove that the intended artifact was read; it cannot make the artifact's assumptions true.

D. The Incident Card

When a run fails, complete one card. Fill fields 1–3 before opening any diagnostic panel. A strong diagnosis does not collect votes; it names a rival cause and the control that could acquit it.

D.1. Three cumulative act assignments

The card is built across the course rather than introduced after the diagnosis is complete.

D.1.1. Act 0 — execution contract

Choose one numerical or performance symptom from C01–C04. Complete Fields 1–4 and 10 before seeing the reference outcome. Your paired control must keep the mathematical target fixed while changing exactly one of traffic, reduction order, dtype, or step size.

The audit packet includes two intentionally plausible defects:

- a speedup claim with no traffic or device boundary;
- code that computes the right statistic over the wrong axis.

Deliverable: the partial card, the corrected claim or code, and one sentence explaining why the other defect would survive your control.

D.1.2. Act I — geometric locator

Extend the same card with Field 7 and revise the verdict vocabulary to include **detects only**. The packet supplies a spectrum plot whose values are correct but whose matrix is not named. Identify at least two candidate matrices, then state which conclusion changes when the matrix changes.

Deliverable: a caption that names matrix, scaling, centering, finite-null or perturbation control, and the conclusion the spectrum does not establish.

D.1.3. Act II — causal incident

Complete all ten fields. The packet now contains:

- a proof with one silently dropped independence or smoothness assumption;
- clipping described as variance reduction without its target shift;
- a matrix-update trace whose update spectrum is mislabeled as a weight spectrum.

Correct each item before assigning cause.

Incident B is deliberately unrevealed. Its available panel activates more than one mechanism-specific instrument, and the supplied observations are compatible with at least two causal explanations. There is no answer key in the book. A valid submission may conclude **not identifiable** if it proves the observational equivalence and requests the cheapest additional control. The card is not a deterministic troubleshooting flowchart.

Deliverable: a complete card, one rejected causal story, one surviving rival, and the next measurement that would distinguish them.

D.2. Blank incident report

Field	Record before interpretation
1. Symptom	Quantity, step or interval, magnitude, baseline: _____
2. Prediction	Rank suspects; one acquitting control each: _____
3. Contract	Target; axes; denominator; dtype; seed; device; wheel/commit: _____
4. Precision control	Paired-precision deviation versus local spacing and range (Section 3.2): _____
5. Curvature control	$\alpha\lambda_{\max}$; directional value; reference boundary (Section 16.2): _____
6. Estimator control	Conditional target; noise-to-signal ratio; discrepancy attribution (Section 13.1): _____
7. Spectrum locator	Matrix name; update and weight singular values (Section 17.3): _____
8. State control	Repeated-input statistics; retained state; replay determinism (Section 15.3): _____
9. Verdict	Acquitted / detects only / cause identified: _____
10. Corrective control	Rerun that removes the symptom while declared controls remain fixed: _____

D.3. Compact instrument panel

Thread	Symptom	Instrument and control	What the control cannot decide
Numerical stability	represented values stall, overflow, or disagree across precision	local spacing, range, and a paired-precision rerun (Section 3.7)	whether curvature, data, or retained state caused an event that survives the precision control
Random-matrix spectra	one matrix direction expands, collapses, or separates	name the matrix; compare edges, bulk, and update-versus-weight spectra (Section 17.3)	why the direction changed
Landscape and update geometry	loss oscillates or a scalar step becomes locally aggressive	top and directional curvature beside the exact fixed-quadratic boundary (Section 16.1)	a global nonlinear divergence claim from one local value
Sub-Gaussian safe zone	a mean estimate hides dispersion or rare updates dominate	conditional estimator, tail diagnostic, matched batch control, and clipping-bias audit (Section 13.4)	a population tail class from one finite trace

The completed card in the Coda is a worked example, not a universal ordering of suspects. When a control detects an event without identifying its cause, record **detects only** and keep the rival explanations alive.