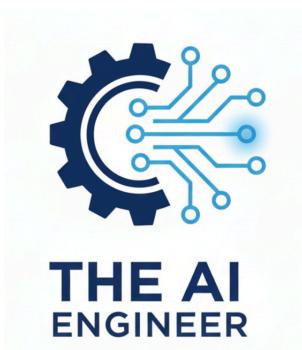


Python & Mathematics for Data Science and Machine Learning

Executable Math for Data Science and ML

Dr. Yves J. Hilpisch¹

February 3, 2026



¹Get in touch: <https://linktr.ee/dyjh>. Web page: <https://theaiengineer.dev>. Research, structuring, drafting, and visualizations were assisted by GPT 5.x as a co-writing tool under human direction. Comments and feedback are welcome.

Preface

Math you can run. That’s the spirit of this book: every definition, claim, or diagram can be checked with a few lines of readable Python. When ideas and numbers agree, intuition sticks.

Why this book and why now?

Practical ML and modern AI rest on a small core: linear algebra for shapes and projections; calculus for sensitivities; probability for uncertainty; optimization for learning; and a disciplined way to make claims falsifiable. Python lowers the activation energy. NumPy arrays make algebra concrete; Matplotlib turns signal into sight; tiny experiments make mistakes visible and progress tangible.

How to use this book

Follow the “Math \leftrightarrow Code” loop: learn the concept, run the smallest check you can write, and read the result. Repeat until it feels obvious. Keep runs reproducible: set seeds, print shapes and scalars, and save figures with captions that say what to observe. Treat exercises as experiments. They do not test memory — they teach reflexes: framing, checking, and communicating results.

What’s inside

Parts I–III build the language: vectors and matrices as actions; derivatives as local models; landscapes and the chain rule as engines for learning. Parts IV–V make uncertainty and optimization usable: distributions you’ll see often; calibration that makes probabilities honest; convexity and constraints that keep solutions meaningful; stochastic methods that scale. Parts VI–VII bridge to practice: end-to-end modeling; attention and embeddings; autodiff and PyTorch; small patterns that carry into transformers and LLMs.

If at any point you feel stuck, shrink the problem. Smaller shapes, fixed seeds, and one printed number will get you moving again. Keep your curiosity, and let the code keep you honest.

Technical & Legal Note

This book and its code are provided for educational purposes only. Examples are simplified and may omit edge cases; verify results and assumptions for your context, especially in safety-, medical-, or financial-critical settings. The author and publisher provide the material “as is” without warranties and are not liable for losses arising from use; external dependencies and links can change over time.

Contents

Preface	i
---------	---

I The Back-and-Forth Method: Python Meets Math	1
--	---

1 Why Math With Code (and Code With Math)	3
1.1 Concept & Intuition	3
1.2 Formalism	4
1.3 From Math → Code	4
1.4 From Code → Math	5
1.5 Visualization: How the Sample Mean Stabilizes	5
1.6 Common Pitfalls & Stability	5
1.7 Exercises	6
1.8 Where We're Heading Next	7

2 Python Essentials	8
2.1 Concept & Intuition	8
2.2 Formalism	8
2.3 From Math → Code	9
2.4 Scalar Math vs NumPy and Broadcasting	10
2.5 Floating-Point Associativity: When Order Matters	10
2.6 Visualization: Rounding Error in Summation	11
2.7 Plotting for Intuition	11
2.8 Common Pitfalls & Stability	12
2.9 Exercises	13
2.10 Where We're Heading Next	13

3 Numerical Thinking & Error	14
3.1 Concept & Intuition	14
3.2 Formalism	15
3.3 From Math → Code	15
3.4 From Code → Math	16
3.5 Visualization: Cancellation Error vs Magnitude	16
3.6 Common Pitfalls & Stability	16
3.7 Exercises	17
3.8 Where We're Heading Next	17

II Linear Algebra That Actually Moves Code	19
--	----

4 Vectors, Norms, and Geometry	21
4.1 Concept & Intuition	21

4.2 Formalism	21
4.3 From Math → Code	22
4.4 From Code → Math	22
4.5 Visualizations	23
4.6 Common Pitfalls & Stability	25
4.7 Exercises	26
4.8 Where We're Heading Next	26
5 Matrices, Linear Maps, and Bases	28
5.1 Concept & Intuition	28
5.2 Formalism	28
5.3 From Math → Code	29
5.4 From Code → Math	30
5.5 Visualizations	30
5.6 Common Pitfalls & Stability	32
5.7 Exercises	32
5.8 Where We're Heading Next	32
6 Eigenvalues, SVD, and Low-Rank Structure	34
6.1 Concept & Intuition	34
6.2 Formalism	35
6.3 From Math → Code	35
6.4 From Code → Math	36
6.5 Visualizations	36
6.6 Common Pitfalls & Stability	38
6.7 Exercises	38
6.8 Where We're Heading Next	38
7 Tensors & Broadcasting for ML	40
7.1 Concept & Intuition	40
7.2 Formalism	41
7.3 From Math → Code	41
7.4 From Code → Math	42
7.5 Visualizations	42
7.6 Common Pitfalls & Stability	44
7.7 Exercises	44
7.8 Where We're Heading Next	44
III Calculus & Matrix Calculus for Learning	45
8 Differentiation & Taylor Intuition	47
8.1 Concept & Intuition	47
8.2 Formalism	47
8.3 From Math → Code	48
8.4 From Code → Math	49
8.5 Visualizations	50
8.6 Common Pitfalls & Stability	51
8.7 Exercises	51
8.8 Where We're Heading Next	52

9 Chain Rule at Scale & Matrix Calculus	53
9.1 Concept & Intuition	53
9.2 Formalism	53
9.3 From Math → Code	54
9.4 From Code → Math	55
9.5 Visualizations	55
9.6 Common Pitfalls & Stability	56
9.7 Exercises	57
9.8 Where We're Heading Next	57
10 Optimization Landscapes	58
10.1 Concept & Intuition	58
10.2 Formalism	59
10.3 From Math → Code	59
10.4 From Code → Math	60
10.5 Visualizations	60
10.6 Common Pitfalls & Stability	61
10.7 Exercises	62
10.8 Where We're Heading Next	62
IV Probability & Statistics: From RNG to Inference	63
11 Probability Foundations	65
11.1 Concept & Intuition	65
11.2 Formalism	66
11.3 From Math → Code	66
11.4 From Code → Math	68
11.5 Visualizations	68
11.6 Common Pitfalls & Stability	69
11.7 Exercises	69
11.8 Where We're Heading Next	70
12 Named Distributions & Transformations	71
12.1 Concept & Intuition	71
12.2 Formalism	72
12.3 From Math → Code	72
12.4 From Code → Math	74
12.5 Visualizations	74
12.6 Common Pitfalls & Stability	75
12.7 Exercises	75
12.8 Where We're Heading Next	76
13 Bayes & Information	77
13.1 Concept & Intuition	77
13.2 Formalism	78
13.3 From Math → Code	78
13.4 From Code → Math	79
13.5 Visualizations	79
13.6 Common Pitfalls & Stability	81
13.7 Exercises	81
13.8 Where We're Heading Next	82

14 Statistical Learning Basics	83
14.1 Concept & Intuition	83
14.2 Formalism	84
14.3 From Math → Code	84
14.4 From Code → Math	85
14.5 Visualizations	85
14.6 Common Pitfalls & Stability	87
14.7 Exercises	87
14.8 Where We're Heading Next	88
V Optimization for ML	89
15 Unconstrained Optimization	91
15.1 Concept & Intuition	91
15.2 Formalism	91
15.3 From Math → Code	92
15.4 From Code → Math	95
15.5 Visualizations	95
15.6 Common Pitfalls & Stability	97
15.7 Exercises	97
15.8 Where We're Heading Next	98
16 Constrained & Convex Optimization	99
16.1 Concept & Intuition	99
16.2 Formalism	100
16.3 From Math → Code	100
16.4 From Code → Math	101
16.5 Visualizations	102
16.6 Common Pitfalls & Stability	104
16.7 Exercises	104
16.8 Where We're Heading Next	104
17 Stochastic Optimization	106
17.1 Concept & Intuition	106
17.2 Formalism	107
17.3 From Math → Code	107
17.4 From Code → Math	109
17.5 Visualizations	110
17.6 Common Pitfalls & Stability	111
17.7 Exercises	111
17.8 Where We're Heading Next	112
VI Statistical Models & Classical ML Bridge	113
18 Linear Models Revisited (End-to-End)	115
18.1 Concept & Intuition	115
18.2 Formalism	116
18.3 From Math → Code	116
18.4 From Code → Math	117
18.5 Visualizations	118

18.6 Common Pitfalls & Stability	118
18.7 Exercises	120
18.8 Where We're Heading Next	120
19 Classification & Calibration	121
19.1 Concept & Intuition	121
19.2 Formalism	122
19.3 From Math → Code	122
19.4 From Code → Math	123
19.5 Visualizations	124
19.6 Common Pitfalls & Stability	126
19.7 Exercises	126
19.8 Where We're Heading Next	126
20 Unsupervised Learning & Spectral Methods	127
20.1 Concept & Intuition	127
20.2 Formalism	128
20.3 From Math → Code	128
20.4 From Code → Math	129
20.5 Visualizations	130
20.6 Common Pitfalls & Stability	132
20.7 Exercises	132
20.8 Where We're Heading Next	132
VII Bridges to Deep Learning & LLMs	133
21 Autodiff, Backprop, and What PyTorch Automates	135
21.1 Concept & Intuition	135
21.2 Formalism	136
21.3 From Math → Code	136
21.4 From Code → Math	139
21.5 Visualizations	139
21.6 Common Pitfalls & Stability	141
21.7 Exercises	141
21.8 Where We're Heading Next	141
22 Embeddings & Attention as Linear Algebra	142
22.1 Concept & Intuition	142
22.2 Formalism	143
22.3 From Math → Code	143
22.4 From Code → Math	144
22.5 Visualizations	145
22.6 Common Pitfalls & Stability	146
22.7 Exercises	146
22.8 Where We're Heading Next	147
23 From Here to PyTorch & LLM Books	148
23.1 Concept & Intuition	148
23.2 Formalism	149
23.3 From Math → Code	149
23.4 From Code → Math	150

23.5 Visualizations	150
23.6 Common Pitfalls & Stability	151
23.7 Exercises	151
23.8 Where We're Heading Next	152
VIII Projects, Patterns, and Practices	153
24 Mini Projects & Executable Math	155
24.1 Concept & Intuition	155
24.2 Formalism	155
24.3 From Math → Code	156
24.4 From Code → Math	158
24.5 Visualizations	158
24.6 Common Pitfalls & Stability	158
24.7 Exercises	161
24.8 Where We're Heading Next	161
25 Reproducible Research Playbook	162
25.1 Concept & Intuition	162
25.2 Formalism	162
25.3 From Math → Code	163
25.4 From Code → Math	164
25.5 Visualizations	165
25.6 Common Pitfalls & Stability	165
25.7 Exercises	166
25.8 Where We're Heading Next	166
26 Where to Go Next	168
26.1 Concept & Intuition	168
26.2 Formalism	169
26.3 From Math → Code	169
26.4 From Code → Math	169
26.5 Visualization: A Mental Roadmap	170
26.6 Common Pitfalls & Stability	170
26.7 Exercises	170
26.8 Where We're Heading Next	170
Epilogue	171
Glossary, Notation, and Cheatsheets	172

List of Figures

1.1 LLN in action: running mean vs. sample size on a log x-axis (dashed zero line)	6
2.1 Rounding error when summing many small numbers: naive vs compensated (Kahan) summation on log-scaled axes	11
2.2 Standard normal: histogram of samples with analytic PDF overlay	12
2.3 Gaussian bump surface: $z = e^{-(x^2+y^2)}$ as a smooth radially symmetric surface	12
3.1 Relative error of naive vs stable formulas for $f(x) = \sqrt{x+1} - \sqrt{x}$ across large x on log-log axes	16
4.1 Unit balls in 2D for ℓ_1 , ℓ_2 , and ℓ_∞	24
4.2 Decision boundaries under ℓ_1 , ℓ_2 , and ℓ_∞	24
4.3 Cosine similarity between random vectors and the angle–cosine relationship	25
4.4 Projection of a vector onto a line and the orthogonal residual	26
5.1 Projection of b onto the column space of A with orthogonal residual $r = b - Pb$	30
5.2 Normal equations vs QR on an ill-conditioned design: residuals and solution quality can diverge when conditioning is poor	31
5.3 Change of basis in 2D: the same vector expressed in standard and new coordinates	31
5.4 Singular values of a nearly rank-deficient matrix; sharp drops suggest approximate rank	32
6.1 Effect of a matrix on the unit circle: right singular vectors and singular values determine how the circle becomes an ellipse	37
6.2 Power iteration on a symmetric matrix: Rayleigh quotient and angle converge to the leading eigenpair	37
6.3 Rank- k approximation: original vs reconstruction and error curve as a function of k	37
6.4 PCA variance explained: bar and cumulative line plot of variance captured by each principal component	38
7.1 Broadcasting a column and a row: $(m, 1) + (1, n) \rightarrow (m, n)$ grid of pairwise sums	43
7.2 Batched matrix multiply: shapes (b, m, k) and (b, k, n) producing (b, m, n) , with timing comparison to explicit loops	43
8.1 Function and tangent line at x_0 : locally, the graph looks linear and the line provides a first-order Taylor approximation	50
8.2 Function vs first- and second-order Taylor polynomials around x_0 : the quadratic model matches the function over a wider neighborhood	51
9.1 Chain rule as sensitivity flow on a computational graph: forward values in one lane, backward gradients in another	56

9.2 VJP vs JVP on a tanh layer: reverse mode propagates sensitivities backward via $u^\top J$; forward mode pushes perturbations forward via Jv	56
9.3 Schematic cost comparison of Jacobian–vector (forward mode) vs vector–Jacobian (reverse mode) products in two scenarios: many inputs / few outputs vs few inputs / many outputs	57
10.1 Convex bowl vs saddle: contour plots with gradient arrows showing attraction to a minimizer vs repulsion along unstable axes	61
10.2 Gradient descent trajectories on an ill-conditioned quadratic: different step sizes produce very different paths and convergence rates	61
11.1 Law of Large Numbers and Central Limit Theorem for Bernoulli samples: running means stabilise around the expectation and properly normalised errors look approximately Gaussian	68
11.2 Covariance structure for bivariate Gaussians: correlated samples form an elongated cloud aligned with the principal directions, whereas independent samples remain roughly circular	69
12.1 Overlayed PDFs/PMFs for Normal, Gamma, Poisson, and Beta families, with markers for means and typical spreads	74
12.2 Log-likelihood accumulation using naive exponentiation versus the log-sum-exp trick. The stable computation maintains accuracy far into the small-probability regime	75
13.1 Beta–Bernoulli prior and posteriors for increasing sample sizes. As more data arrives, the posterior concentrates around the true probability and the influence of the prior diminishes	80
13.2 Bernoulli entropy and KL divergence. Entropy peaks at $p = 0.5$ and vanishes near 0 or 1; KL grows large when beliefs disagree, especially when confident models are wrong	80
13.3 Log-loss as a function of predicted probability. Correct, confident predictions have small loss, while confident but wrong predictions incur very large penalties	81
14.1 Bias–variance trade-off for polynomial regression: training MSE decreases with degree while test MSE forms a U-shaped curve, with underfitting at low degrees and overfitting at high degrees	86
14.2 Ridge paths for polynomial regression: the L2 norm and individual coefficients shrink smoothly as the regularisation strength λ increases	86
14.3 K-fold cross-validation splits: each fold holds out a different set of samples while the others form the training set, and averaging validation losses across folds estimates generalisation	87
15.1 Gradient-descent paths on an anisotropic quadratic. Fixed steps zig-zag across the narrow direction; backtracking adapts step sizes and follows a smoother path toward the minimiser	96
15.2 Rosenbrock function with fixed-step vs backtracking GD paths. The curved valley makes naive steps zig-zag and stall, while backtracking adapts step sizes to follow the valley toward the minimum	96
15.3 Backtracking line search on a one-dimensional slice: rejected candidate steps violate the Armijo bound, while the final accepted step sits just inside the sufficient-decrease region	96
15.4 Armijo condition on a one-dimensional slice: loss along the search direction and the linear bound $f(x) - c\alpha\ \nabla f(x)\ ^2$ for different choices of c and β	97

16.1 Projected gradient descent on a quadratic with box constraints. Left: unconstrained updates leave the box; right: projected steps snap to the box and follow its edges while still decreasing the objective.	103
16.2 KKT tangency for a quadratic subject to $x_1 + x_2 = 1$: the active constraint line intersects a quadratic level set exactly at the optimum, where the gradient is parallel to the constraint normal.	103
17.1 Loss versus epoch for full-batch GD, SGD, and SGD with momentum. Full-batch is smooth but slower per epoch; SGD is faster early but noisy; momentum smooths the curve and accelerates convergence.	110
17.2 Summed gradient variance versus batch size. The approximate slope near -1 indicates the expected $1/B$ scaling.	110
17.3 Loss versus epoch for constant and step-decay learning-rate schedules. Step-decay reduces plateaus and drives the loss lower in later epochs.	111
18.1 Ridge (ℓ_2) versus Lasso (ℓ_1) coefficient paths on a correlated design. Ridge shrinks all coefficients smoothly; Lasso exhibits piecewise-linear paths with some coefficients driven exactly to zero.	118
18.2 Train versus validation MSE as a function of Ridge regularisation strength. Validation error exhibits a U-shape; good λ choices lie near the bottom rather than at the extremes.	119
18.3 Residual diagnostics for OLS versus Ridge on a correlated design. Ridge often yields tighter residuals and reduces the influence of leverage points relative to OLS.	119
19.1 Reliability diagrams before and after temperature scaling. The calibrated curve tracks the diagonal more closely, and the legend reports ECE values for quick comparison.	125
19.2 ROC and precision–recall curves for an imbalanced dataset. ROC can look strong even when precision remains modest; PR surfaces the impact of imbalance more directly.	125
19.3 Expected cost versus decision threshold for different cost ratios. The empirical minima align with the theoretical Bayes thresholds once probabilities are calibrated.	125
20.1 K-means elbow plot and example assignments. WCSS decreases as k grows, with diminishing returns past the elbow; scatter plots illustrate how centroids partition the space.	130
20.2 PCA on an elongated cloud: the first principal axis explains most variance and its scalar projection provides a 1D summary with a high explained-variance ratio.	131
20.3 Two moons: k-means vs spectral clustering. In the original space k-means cuts across the crescent; after spectral embedding, clusters become nearly blob-like and k-means recovers them.	131
21.1 Tanh and its derivative: derivatives are near zero outside a moderate band around the origin, explaining vanishing gradients in saturated regimes.	139
21.2 Finite-difference versus autodiff gradients for selected parameters. Points lie close to the $y = x$ line and residuals cluster near zero, indicating a correct implementation.	140
21.3 Conceptual cost of forward and reverse mode. Reverse mode requires roughly one backward pass, while forward mode needs d sweeps for ∇f , which becomes expensive as d grows.	140

22.1 Attention heatmaps without and with a causal mask. The causal version is strictly lower triangular with a strong diagonal, while the non-causal variant can attend anywhere.	145
22.2 Attention as convex mixing: value vectors live at the vertices, while attention outputs fall inside their convex hull. Sharper weight distributions push outputs closer to a single value.	145
22.3 Scaling and sharpness in attention. Without $1/\sqrt{d}$, scores grow with dimension, softmax saturates, and entropy collapses; scaling keeps entropy in a healthy range.	146
23.1 Sinusoidal positional encodings: a heatmap over positions and dimensions, with selected dimensions plotted versus position to show the frequency ladder.	151
23.2 Self-attention costs as a function of sequence length: compute and memory both grow like $O(n^2)$ for the score matrix, quickly dominating for long contexts.	151
24.1 PCA denoising: reconstruction MSE decreases with rank k , with a modest k already capturing most of the structure.	159
24.2 PCA denoising gallery: rows show clean, noisy, and reconstructed images for a small rank k ; reconstructions are visibly cleaner than the noisy inputs.	159
24.3 Calibration mini-project: reliability diagrams before and after temperature scaling, with ECE annotated in the legend.	160
24.4 Seed sensitivity: histogram of validation log-loss across seeds, with mean and standard deviation indicated.	160
25.1 Validation log-loss across many runs: left, random seeds yield a distribution; right, a fixed seed collapses runs to a single value, making drift easy to spot.	165
25.2 PRNG streams: identical seeds reproduce the same sequence of values, while different seeds diverge; re-seeding restores the original path.	166

Contact

Python Primer for Data Science and Deep Learning
The AI Engineer

Get in touch:

<https://linktr.ee/dyjh>

<https://theaiengineer.dev>

© 2026 Dr. Yves J. Hilpisch — All rights reserved.

