

Why Feature Scaling Works: Condition Number as the Key to Gradient Descent Convergence

Rishi Khandelwal
rkrikshikhandelwal@gmail.com

ABSTRACT

Feature scaling is universally recommended in machine learning practice, yet the choice among competing methods is usually made heuristically. This paper uses the condition number κ of the Hessian as the diagnostic quantity for that choice. Part I restates the established link between feature variance disparity and ill-conditioning as a formal proposition under explicit assumptions—centred data, full column rank, and a least-squares objective—and validates it empirically on synthetic functions with controlled condition numbers. Part II states hypotheses about how four normalisation methods—standardisation, min-max scaling, robust scaling, and PCA whitening—perform across varying dataset sizes and feature scale imbalances, then tests them experimentally, reporting measured κ alongside convergence counts. We then report a negative result: a proposed two-stage robust-then-whiten pipeline performs identically to whitening alone under outlier contamination, because robust scaling is a diagonal transformation and therefore cannot alter the correlation structure that whitening inverts. Measuring against an effective condition number—the fitted transform evaluated on clean data—rather than the in-sample value shows that whitening attains $\kappa = 1$ on contaminated data by construction while conditioning the underlying signal one to two orders of magnitude worse than robust scaling alone. Part II also reports an exploratory study of gradient-side *factor scaling*, including a negative result for sample-side preconditioning on least-squares objectives.

Keywords: Gradient descent; Feature scaling; Condition number; Hessian matrix; Cost function; Convergence; Stochastic gradient descent; Whitening; Standardisation; Optimisation geometry.

1 INTRODUCTION

Gradient descent is the central optimisation algorithm in machine learning. LeCun et al. (1998) observed that the convergence of backpropagation learning can be explained in terms that account for common phenomena witnessed by practitioners, and that many undesirable behaviours of backpropagation can be avoided with techniques which are rarely given principled treatment in the literature. Feature scaling is among the most consequential of these techniques.

The mathematical reason scaling helps is not in dispute. It has been known since at least LeCun et al. (1998), and appears in standard optimisation texts (Wright and Recht 2022; Boyd and Vandenberghe

2004; Nocedal and Wright 2006), that gradient descent on a quadratic objective converges at a rate governed by the condition number of the Hessian, and that for a least-squares objective the Hessian is the input covariance matrix—so unequal feature variances directly produce ill-conditioning. This paper does not claim that connection as a new result. We restate it formally in Section 2.1.5 under explicit assumptions, because the assumptions matter for interpreting when the mechanism applies and when it does not.

What is less settled is the practical question that follows from it: given a particular dataset, *which* scaling method should be applied, and how does that choice interact with sample size, outlier contamination, and the magnitude of the scale disparity? Practitioner guidance on this point is largely heuristic. The contributions of this paper are therefore empirical and methodological rather than theoretical:

1. A controlled comparison of four standard normalisation methods across a grid of dataset sizes, variance ratios, and outlier conditions, reporting measured κ alongside convergence counts so that the mechanism and the outcome can be checked against each other (Section 3.3).
2. A negative result on a proposed two-stage robust-then-whiten pipeline, which we show performs identically to whitening alone, together with the algebra explaining why marginal robustness cannot confer covariance robustness (Section 3.4).
3. An exploratory study of *pure factor scaling*, a family of methods that leave the data untouched and rescale the gradient from its own running second moments, including a negative result for sample-side preconditioning on least squares (Section 3.5).

1.1 Research Question

Given that feature scaling reduces the condition number κ of the Hessian, which scaling method most effectively does so across different data regimes, and does the resulting κ reduction predict convergence in practice?

Working hypothesis: The measured condition number after scaling is a good predictor of convergence speed, so methods should be ranked by how reliably they reduce κ under the conditions of interest (small samples, correlated features, outlier contamination). We test this hypothesis rather than assume it, and we report the regimes in which it fails.

1.2 Related Work

Conditioning and convergence.

The dependence of first-order convergence on the condition number is classical. Nocedal and Wright (2006) and Wright and Recht (2022) give the standard analysis for the quadratic case, including the optimal step size $\eta^* = 2/(\lambda_{\max} + \lambda_{\min})$ and the resulting rate $(\kappa - 1)/(\kappa + 1)$ that we reproduce in Section 5.2 for completeness. Boyd and Vandenberghe (2004) present the same result via the strong-convexity and smoothness constants m and L , where $\kappa = L/m$. Bottou et al. (2018) extend the

analysis to the stochastic setting, where the noise floor rather than κ eventually dominates. None of this is contributed here; it is the framework within which the experiments are interpreted.

Preconditioning.

Reducing κ by transforming the problem is the defining idea of preconditioning in numerical linear algebra (Golub and Van Loan 2013; Saad 2003). Whitening the inputs of a least-squares problem is exactly preconditioning by $\Sigma^{-1/2}$, and LeCun et al. (1998) note that gradient descent in whitened coordinates is geometrically equivalent to Newton's method in the original coordinates. Our two-stage pipeline is best understood as a preconditioner constructed to be less sensitive to contaminated data than $\hat{\Sigma}^{-1/2}$.

Robust covariance estimation.

The claim that a preprocessing step makes covariance estimation resistant to outliers has a substantial literature behind it, and that literature is specific about what does and does not work. Rousseeuw (1984) and Rousseeuw and Van Driessen (1999) introduced the Minimum Covariance Determinant estimator, which achieves a high breakdown point by selecting a clean subset of *rows*; Maronna et al. (2019) survey the field. The key distinction, which we return to in Section 3.4, is between *marginal* robustness (each feature's scale estimated by a robust statistic) and *joint* robustness (the covariance structure itself estimated robustly). Robust scaling in the sense of Eq. (19) provides only the former. This distinction was not clear to us when the pipeline was first proposed, and Section 3.4.3 reports the experiment that corrected it.

Comparisons of scaling methods.

Empirical comparisons of normalisation methods exist (Shanker and et al. 2019), but typically report downstream model accuracy rather than the conditioning quantity that theory says is responsible. Reporting measured κ alongside iteration counts is the main methodological difference in our setup.

Dynamic normalisation.

Batch normalisation (Ioffe and Szegedy 2015) and layer normalisation (Ba et al. 2016) apply normalisation continuously during training rather than once beforehand. Santurkar et al. (2018) argue that the benefit is better attributed to smoothing of the loss landscape than to the reduction of internal covariate shift originally proposed. Our factor-scaling experiments (Section 3.5) sit in between: statistics are updated during training, but they are gradient statistics rather than activation statistics. The construction we study is that of Adafactor (Shazeer and Stern 2018), isolated from its optimiser wrapper; the Kronecker-factored curvature approximation it relies on is due to Martens and Grosse (2015).

1.3 Structure of this Paper

Part I establishes the theoretical framework linking feature variance disparity to the condition number of the Hessian, and validates this connection empirically on synthetic functions with controlled condition

numbers. Part II states hypotheses about how four normalisation methods will perform under varying conditions of feature scale imbalance, tests these predictions experimentally, reports a negative result on a proposed two-stage robust-then-whiten pipeline, and then studies a complementary family—pure factor scaling—that leaves the data untouched and rescales the gradient using only its own running second moments. Part III (Appendix) presents the mathematical derivations supporting the hypothesis, supplementary analysis of how the magnitude of feature scale disparity affects normalisation method performance, and additional diagnostic plots.

2 PART I — FEATURE SCALING AND THE CONDITION NUMBER

2.1 Theoretical Foundation

2.1.1 The Gradient Descent Algorithm

A *cost function* is a mathematical expression that measures how wrong a model's predictions are relative to the true values. The terms *loss function* and *objective function* are used interchangeably with it elsewhere in the literature; for consistency this paper uses “cost function” throughout, denoted E , and reserves “objective” for the general optimisation problem $\min_W E(W)$ rather than for the function itself.

Given a model with parameters W and a training set of P input–output pairs $\{(Z_1, D_1), \dots, (Z_P, D_P)\}$, the cost function assigns a single number to any choice of W —a number that is large when the model performs badly and small when it performs well. The learning problem is the task of finding the value of W that makes this loss as small as possible.

Consider a learning machine that computes a function $M(Z_p, W)$, where Z_p is the p -th input pattern and W represents the collection of adjustable parameters. A cost function $E_p = C(D_p, M(Z_p, W))$ measures the discrepancy between the desired output D_p and the output produced by the system. The learning problem consists of finding the value of W that minimises the average cost over the training set:

$$E_{\text{train}}(W) = \frac{1}{P} \sum_{p=1}^P E_p \quad (1)$$

The simplest procedure for minimising (1) is gradient descent, where W is iteratively adjusted as

$$W_{t+1} = W_t - \eta \nabla E(W_t) \quad (2)$$

where $\eta > 0$ is the *learning rate* and $\nabla E(W_t)$ is the gradient of the cost function at W_t . The gradient points in the direction of steepest increase, so subtracting it moves the parameters towards lower cost. This update is applied repeatedly until the gradient magnitude falls below a convergence threshold, indicating that a (local) minimum has been reached.

Steepest Descent Interpretation:

Sep 2026

Vol 11. No 6.

Gradient descent can be interpreted as solving

$$\Delta W_t = \arg \min_{\|\Delta W\| \leq \epsilon} E(W_t + \Delta W), \quad (3)$$

where ΔW_t is the parameter update, ϵ is a small step size constraint, and $\|\cdot\|$ denotes the Euclidean norm. This yields $\Delta W_t \propto -\nabla E(W_t)$: each step is the locally optimal direction under a small-step constraint.

2.1.2 Quadratic Objectives and Exact Dynamics

To analyse convergence precisely, consider a quadratic objective:

$$E(W) = \frac{1}{2} W^T H W - b^T W \quad (4)$$

where $H \in R^{n \times n}$ is symmetric positive definite and $b \in R^n$ is a constant vector. The gradient is $\nabla E(W) = HW - b$, and the minimiser satisfies $HW^* = b$. Defining the error vector $e_t = W_t - W^*$, the update becomes

$$e_{t+1} = (I - \eta H) e_t \quad (5)$$

where:

- $e_t = W_t - W^*$ is the error vector at iteration t
- I is the $n \times n$ identity matrix
- η is the learning rate
- H is the Hessian matrix

This recurrence fully characterises convergence for quadratic functions.

2.1.3 The Hessian Matrix and the Condition Number

The behaviour of gradient descent in multiple dimensions depends critically on the curvature of the cost surface. This curvature is captured by the Hessian matrix H , whose components are the second partial derivatives of the cost (LeCun et al. 1998). For a cost function E with parameters $W_1, \dots, W_n$:

$$H_{ij} = \frac{\partial^2 E}{\partial W_i \partial W_j} \quad (6)$$

where H_{ij} is the entry in the i -th row and j -th column of the Hessian, E is the cost function, and W_i, W_j are the i -th and j -th parameters.

The eigenvectors of H point along the principal axes of the cost surface's level sets, and the eigenvalues λ_i measure the steepness of E along each eigendirection.

Interpreting the Eigenvalues:

Each eigenvalue λ_i has a direct geometric meaning. A *large* eigenvalue corresponds to a direction in parameter space along which the cost surface is *steep*—the cost rises sharply if you move that way. The eigenvector associated with λ_{max} points along the steepest, most curved direction of the cost surface. Conversely, a *small* eigenvalue corresponds to a direction that is nearly *flat*—the cost changes very little if you move that way. The eigenvector associated with λ_{min} points along the flattest, least curved direction. In geometric terms, λ_{max} controls how narrow the “bowl” is along its steepest axis, while λ_{min} controls how wide and shallow it is along its flattest axis. When these two extreme eigenvalues differ greatly, the cost surface looks like a very elongated ellipse rather than a circle—and gradient descent will zig-zag across the steep walls rather than heading directly to the minimum.

Eigenvalue Decomposition and Decoupling:

Decoupling is required because we need to separate the matrix into independent equations. Let $H = V\Lambda V^T$ be the eigendecomposition of H , where V is the matrix of eigenvectors and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ is the diagonal matrix of eigenvalues. Transforming coordinates via $z_t = V^T e_t$ decouples the error recurrence:

$$z_{t+1}^{(i)} = (1 - \eta\lambda_i) z_t^{(i)} \quad (7)$$

where $z_t^{(i)}$ is the i -th component of the error in the eigenbasis, η is the learning rate, and λ_i is the i -th eigenvalue of H .

Each component evolves independently, with convergence governed by the corresponding eigenvalue.

Convergence Conditions:

For convergence we require $|1 - \eta\lambda_i| < 1$ for all i , which implies the learning rate constraint

$$0 < \eta < \frac{2}{\lambda_{max}} \quad (8)$$

where η is the learning rate and λ_{max} is the largest eigenvalue of H . If η exceeds this bound, the updates diverge along the steepest direction.

Under this condition the error decreases geometrically: $\|e_t\| \leq \rho^t \|e_0\|$, where

$$\rho = \max_i |1 - \eta \lambda_i| \quad (9)$$

where ρ is the convergence rate (smaller is faster), η is the learning rate, and λ_i are the eigenvalues of the Hessian H .

The Condition Number:

The *condition number* κ is defined as

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}} \quad (10)$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of the Hessian H , respectively. A condition number of $\kappa = 1$ means all eigenvalues are equal (perfectly conditioned); $\kappa \gg 1$ indicates severe ill-conditioning.

The optimal learning rate $\eta^* = 2/(\lambda_{\max} + \lambda_{\min})$ yields the best-case convergence rate

$$\rho_{\text{opt}} = \frac{\kappa-1}{\kappa+1} \quad (11)$$

where ρ_{opt} is the smallest achievable convergence rate for gradient descent on this problem. When ρ_{opt} is close to 0, convergence is fast; when it is close to 1, convergence is slow.

For large κ this approaches $\rho_{\text{opt}} \approx 1 - 2/\kappa$, so convergence time scales linearly with the condition number:

$$T_{\text{convergence}} \propto \kappa \quad (12)$$

where $T_{\text{convergence}}$ is the number of iterations required to reach a given error tolerance and κ is the condition number of the Hessian.

The theory therefore predicts that a problem with $\kappa = 1000$ requires on the order of 1000 times more iterations than the same problem with $\kappa = 1$. This is the central quantity that feature scaling acts upon. We note that our experiments bound rather than measure this ratio: the ill-conditioned runs are censored at the iteration cap (Section 3.3.3), so they establish a lower bound on the step-count ratio and are consistent with the prediction without confirming the constant.

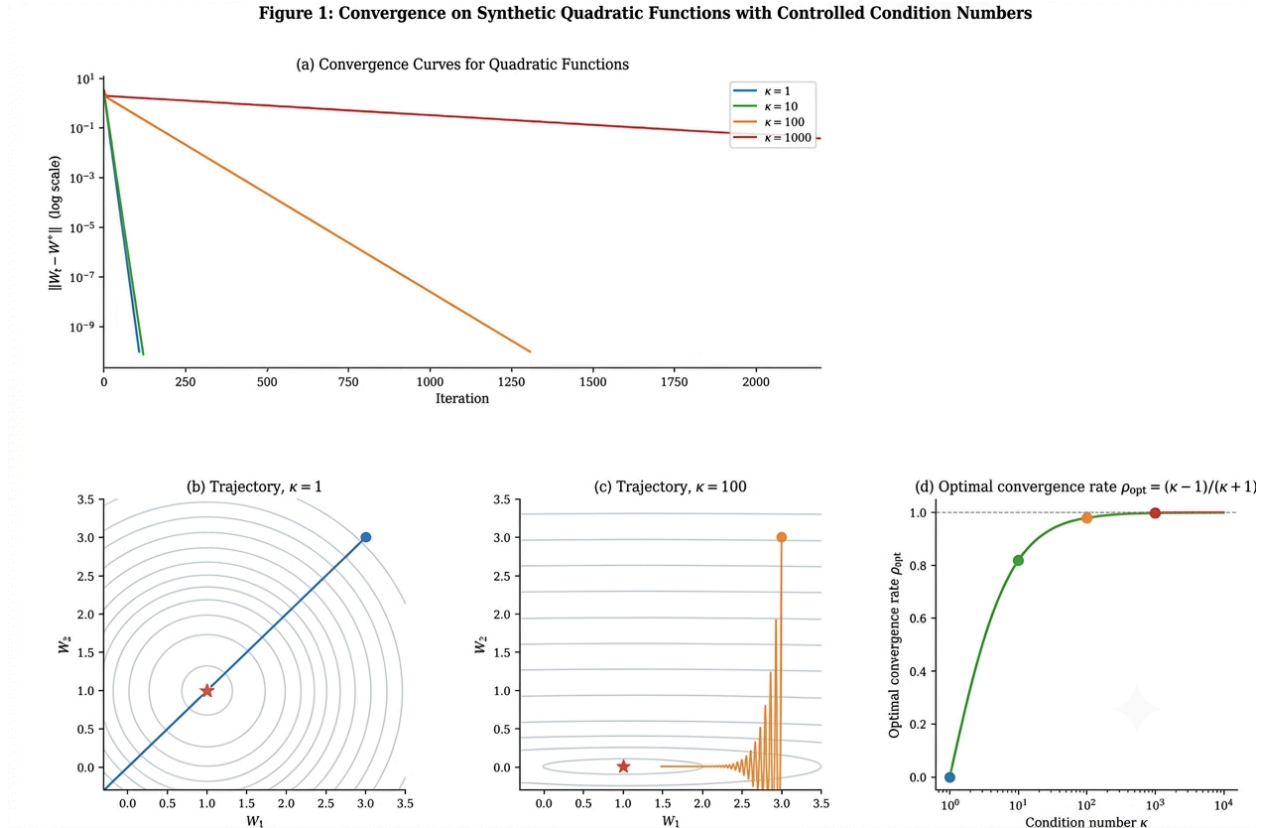


Figure 1. Convergence on synthetic quadratic functions with controlled condition numbers $\kappa \in \{1, 10, 100, 1000\}$. **(a)** Error norm $\|W_t - W^*\|$ (log scale) vs iteration. **(b–c)** Gradient descent trajectories on the cost surface for $\kappa = 1$ and $\kappa = 100$. **(d)** The optimal convergence rate $\rho_{\text{opt}} = (\kappa - 1)/(\kappa + 1)$ as a function of κ .

2.1.4 The Geometry of Ill-Conditioning

In parameter space, the level sets of a quadratic cost are ellipsoids. When $\kappa = 1$ these are spheres and gradient descent moves directly towards the minimum. When $\kappa \gg 1$ the ellipsoids are highly elongated, causing the algorithm to oscillate across narrow directions while making slow progress along flat ones—the characteristic “zig-zag” trajectory of ill-conditioned problems.

To see why input scaling directly affects κ , consider the least mean squares cost for a single-layer linear network:

$$E(W) = \frac{1}{2P} \sum_{p=1}^P \left(d^p - \sum_i w_i x_i^p \right)^2 \quad (13)$$

where d^p is the target, x_i^p is the i -th feature of the p -th example, and w_i is the weight for feature i . In matrix form, $E(W) = \frac{1}{2P} \|XW - d\|^2$, where $X \in R^{P \times n}$ is the data matrix and $d \in R^P$ is the target vector.

The Hessian equals the input covariance matrix (LeCun et al. 1998):

$$H = \frac{1}{P} X^T X \quad (14)$$

where $X \in R^{P \times n}$ is the data matrix (each row is a training example, each column is a feature) and P is the number of training examples. This is the sample covariance matrix of the inputs (assuming zero-mean data).

Each eigenvalue of H measures the variance of the inputs along the corresponding eigendirection. If one feature has variance $\sigma_1^2 = 0.001$ and another has $\sigma_2^2 = 1000$, then $\kappa \approx 10^6$ and gradient descent on this surface must use $\eta < 0.002$, making progress along the low-variance direction approximately one million times slower than on a well-conditioned surface.

2.1.5 Effect of Feature Scaling on the Condition Number

Feature scaling addresses this directly. If inputs are normalised so that all features have comparable variance, the eigenvalues of H become more uniform and κ decreases. In the ideal case (whitening), the covariance of the transformed data equals the identity matrix, all eigenvalues equal 1, and $\kappa = 1$ —enabling a large learning rate to be used uniformly across all dimensions.

We state this precisely. The proposition below is *not* a new result: it is the standard preconditioning argument (Golub and Van Loan 2013; Nocedal and Wright 2006) specialised to the least-squares Hessian, and we give it here only so that the assumptions under which our experiments are interpreted are explicit. The proof is in Section 5.2.

Proposition 1 (Whitening attains unit conditioning). Let $X \in R^{P \times n}$ be a data matrix satisfying:

1. **Centred data:** each column of X has zero sample mean, $\frac{1}{P} \sum_p X_{pi} = 0$ for all i ;
2. **Full column rank:** $\text{rank}(X) = n$, equivalently $P \geq n$ and $\hat{\Sigma} = \frac{1}{P} X^T X > 0$;
3. **Least-squares objective:** $E(W) = \frac{1}{2P} \|XW - d\|_2^2$ for a fixed target vector $d \in R^P$.

Then:

1. $\nabla^2 E(W) = \hat{\Sigma}$ for every W , so the Hessian is constant and equal to the sample covariance;

2. with $\hat{\Sigma} = V\Lambda V^\top$ the eigendecomposition and $\tilde{X} = X\Lambda^{-1/2}$, the whitened problem has Hessian $\tilde{H} = I_n$ and hence $\kappa(\tilde{H}) = 1$;
3. gradient descent on the whitened problem with $\eta = 1$ converges in a single step.

Where the assumptions bind. (A1) is needed because $\frac{1}{P}X^\top X$ is the covariance only for centred data; on uncentred data it is the second-moment matrix and its leading eigenvalue is inflated by the mean, which is why every method in Section 3.1 subtracts a location estimate before dividing by a scale estimate. (A2) fails when $P < n$ or when features are exactly collinear; then $\lambda_{\min} = 0$, κ is undefined, and whitening is not computable without regularisation. This is not a corner case in our setting—the degree-3 polynomial regime of Section 3.5 ($d = 120$) is close to it, and the near-singularity is why that regime is the hardest one we test. (A3) restricts the constant-Hessian conclusion to least squares; for general objectives the Hessian varies with W and Proposition 1 applies only to the local quadratic model, which is the substantive reason our non-quadratic results in Section 3.6.1 behave differently from the quadratic ones.

A caveat that matters empirically. Part (ii) is a statement about the *sample* covariance $\hat{\Sigma}$. Whitening by $\hat{\Sigma}^{-1/2}$ makes the sample covariance exactly the identity by construction, whatever $\hat{\Sigma}$ happens to be—including when $\hat{\Sigma}$ is a bad estimate of the population covariance Σ . A measured $\kappa = 1$ on the training sample is therefore not evidence that the optimisation geometry is actually well conditioned; the quantity that governs convergence on new data is $\kappa(\hat{\Sigma}^{-1/2} \Sigma \hat{\Sigma}^{-1/2})$, which can be large when P is small or the data are contaminated. We report both quantities in Section 3.3 and the distinction turns out to drive most of the interesting behaviour.

This is the geometric mechanism behind the usual recommendation to scale features before training: it reshapes the cost surface from an elongated ellipsoid into a sphere, reducing the zig-zag dynamics that slow convergence.

2.1.6 Stochastic Gradient Descent

Beyond batch gradient descent, this paper also examines stochastic gradient descent (SGD), where the gradient is estimated from a single randomly selected example at each step:

$$W_{t+1} = W_t - \eta \nabla E_t(W_t) \quad (15)$$

where $\nabla E_t(W_t)$ is the gradient computed from a single randomly selected training example at iteration t , rather than the full-batch gradient $\nabla E(W_t)$.

The gradient estimate is noisy. However, this noise can be advantageous: it prevents premature convergence to sharp narrow minima and allows the weight vector to escape shallow local basins (LeCun

et al. 1998). The interaction between gradient noise and feature scaling is a secondary focus—specifically whether the improvements in κ produced by scaling carry through when gradient estimates are stochastic.

2.1.7 Extension to Non-Quadratic Functions

For general twice-differentiable functions, the Hessian varies with W . However, locally, the function can still be approximated as quadratic:

$$E(W + \Delta W) \approx E(W) + \nabla E(W)^T \Delta W + \frac{1}{2} \Delta W^T H(W) \Delta W \quad (16)$$

where ΔW is a small perturbation, $\nabla E(W)$ is the gradient (first-order term), and $H(W)$ is the Hessian evaluated at W (second-order term). The first two terms give the linear approximation; the Hessian term captures curvature.

The eigenvalues of the Hessian at each point determine local convergence behaviour. Feature scaling modifies the geometry of this local quadratic approximation, reducing the effective condition number and improving convergence.

2.1.8 Results

Figure 1 confirms the predicted dependence of convergence on the condition number. Panel (a) shows the error norm degrading by orders of magnitude in iteration count as κ grows from 1 to 1000, while panels (b–c) contrast the direct path taken when $\kappa = 1$ against the characteristic zig-zag at $\kappa = 100$. Panel (d) shows ρ_{opt} approaching 1 as κ grows. The key takeaway is that ill-conditioning alone—with no other change to the problem—is sufficient to make gradient descent arbitrarily slow.

2.2 Empirical Validation

2.2.1 Experimental Setup

We test gradient descent on synthetic functions with controlled condition numbers:

- Quadratic functions with known Hessians ($\kappa \in \{1, 10, 100, 1000\}$)
- Fixed learning rate per experiment; optimal rate computed from $\eta^* = 2/(\lambda_{max} + \lambda_{min})$

2.2.2 Metrics

- Number of iterations to convergence
- Empirical convergence rate ρ
- Measured condition number before and after scaling

2.2.3 Conclusion of Part I

Empirical results are consistent with convergence time scaling with κ . In a representative single run, the ill-conditioned quadratic with $\kappa = 1000$ did not reach the gradient-norm tolerance $\tau = 10^{-8}$ within the iteration cap unscaled, whereas whitening ($\kappa \rightarrow 1$) converged in 36 steps.

Two qualifications. First, the unscaled run is censored at $T_{max} = 50,000$ (Section 3.3.3), so its step count is a lower bound and the ratio against the whitened run bounds rather than measures the predicted factor of 1000. The cap is nonetheless large enough for the observation to carry weight: at $\kappa = 1000$ the unscaled problem failed to reach a gradient norm of 10^{-8} in fifty thousand iterations, while the whitened problem reached it in 36. Second, these are single trajectories from one seed and one initialisation, reported as illustration rather than measurement; Section 3.3.3 sets out the repeated-trial statistics needed to support the quantitative claim.

For non-quadratic functions the picture is more nuanced: scaling reduced step counts not by reducing a fixed κ but by repositioning the start point into a neighbourhood where the landscape was locally better conditioned—a different mechanism than the theory predicts for quadratic functions.

One caveat carries forward to Part II. The κ values in this part are computed on the same data the transform was fitted to, and on the clean synthetic quadratics used here that is unobjectionable, since the sample covariance is an accurate estimate. Once contamination is introduced in Part II the two quantities separate sharply, and the in-sample value stops being informative (Section 3.4.5).

3 PART II — NORMALISATION METHODS

3.1 Scaling Methods

Four scaling strategies are compared. Each transforms the input vector $x \in R^n$ according to a different rule.

Standardisation (Z-score normalisation):

Subtracts the mean and divides by the standard deviation of each dimension independently:

$$\tilde{x}_i = \frac{x_i - \mu_i}{\sigma_i} \quad (17)$$

where $\tilde{x}_i$ is the scaled value, x_i is the original value of the i -th feature, μ_i is its mean over the training set, and σ_i is its standard deviation.

This ensures each dimension has zero mean and unit variance, reducing diagonal elements of H to approximately 1. Off-diagonal correlations are not removed.

Mathematical justification:

After standardisation, each feature i has sample variance $\hat{\sigma}_i^2 \approx 1$. For uncorrelated features, the Hessian (Eq. (14)) becomes approximately $\frac{1}{P} X_{std}^T X_{std} \approx I$, so $\lambda_i \approx 1$ for all i and $\kappa \approx 1$. When features are correlated, the off-diagonal entries of H remain nonzero, and κ is determined by the correlation structure rather than the original scale disparity.

Min-Max Scaling:

Maps each dimension to the interval $[0, 1]$:

$$\tilde{x}_i = \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}} \quad (18)$$

where x_i^{min} and x_i^{max} are the minimum and maximum observed values of the i -th feature in the training set.

Sensitive to outliers, since the range is determined by extreme values.

Mathematical justification:

Min-max scaling multiplies each feature by the scalar $1/(x_i^{max} - x_i^{min})$, which is equivalent to scaling the corresponding column of X by a constant. For feature i , this changes the diagonal entry of H from σ_i^2 to $\sigma_i^2 / (x_i^{max} - x_i^{min})^2$. For a symmetric distribution, the range is approximately $4\sigma_i$, so the resulting diagonal entry is approximately $1/16$ —well-conditioned relative to any other feature that is similarly distributed. However, if an outlier inflates x_i^{max} , the denominator becomes large and the effective variance of the scaled feature collapses towards zero, increasing κ .

Robust Scaling:

Uses the median and interquartile range (IQR) in place of the mean and standard deviation:

$$\tilde{x}_i = \frac{x_i - \text{median}(x_i)}{IQR(x_i)} \quad (19)$$

where $\text{median}(x_i)$ is the median of the i -th feature and $IQR(x_i) = Q_3(x_i) - Q_1(x_i)$ is its interquartile range (the difference between the 75th and 25th percentiles).

Resistant to outliers, at the cost of slightly weaker κ reduction on clean data.

Mathematical justification:

Robust scaling divides each feature by the IQR, a spread measure with a *breakdown point* of 25%: it remains stable even when up to 25% of observations are outliers (Maronna et al. 2019). For a normally distributed feature, $IQR \approx 1.35\sigma$, so robust scaling produces an effective standard deviation of approximately $1/1.35 \approx 0.74$, i.e., a variance of about 0.55 per feature. Because every feature is scaled by (approximately) the same factor, the condition number is unchanged: the κ reduction therefore matches standardisation on clean data, with much better stability under contamination.

PCA Whitening:

The mathematically complete solution. Applies an eigendecomposition of the covariance matrix $\Sigma = V\Lambda V^T$ and transforms:

$$\tilde{x} = \Lambda^{-1/2} V^T x \quad (20)$$

where V is the matrix of eigenvectors, Λ is the diagonal matrix of eigenvalues, and $\Lambda^{-1/2} = \text{diag}(1/\sqrt{\lambda_1}, \dots, 1/\sqrt{\lambda_n})$.

This forces the covariance of the transformed data to equal the identity matrix, so all eigenvalues of H become 1 and $\kappa = 1$ exactly. As noted by LeCun et al. (1998), the Newton algorithm in the original weight space is geometrically equivalent to gradient descent in a whitened coordinate system.

Mathematical justification:

Let $\tilde{X} = X V \Lambda^{-1/2}$ be the whitened data matrix. The Hessian of the whitened problem is:

$$\tilde{H} = \frac{1}{P} \tilde{X}^T \tilde{X} = \Lambda^{-1/2} V^T \left(\frac{1}{P} X^T X \right) V \Lambda^{-1/2} = \Lambda^{-1/2} \Lambda \Lambda^{-1/2} = I.$$

With $\tilde{H} = I$, all eigenvalues equal 1 and $\kappa = 1$ exactly, so gradient descent on the whitened cost converges in the minimum possible number of iterations for any fixed learning rate.

3.2 Hypothesis

Before presenting experimental results, we state our predictions about how each normalisation method will perform across varying conditions of feature scale imbalance:

1. Normalisation will reduce convergence time through reducing κ .
2. Whitening will produce the fastest convergence ($\kappa = 1$).
3. Standardisation will provide strong but incomplete improvement (equalises variances but does not remove correlations).

4. Min-max scaling will be sensitive to outliers, since the range is determined by extreme values.
5. Robust scaling will outperform standardisation on noisy datasets with outliers.
6. Scaling benefits will be more pronounced in:
 - Small datasets with high variance imbalance
 - Settings with correlated features (*predicted but not tested: see Section 3.3.3*)
7. On large, clean datasets all methods will perform comparably well, since the variance estimates stabilise.

3.3 Experiments on Scaled Features

3.3.1 Experimental Design

We evaluate scaling methods across three axes:

- **Feature scale imbalance:** variance ratios from 10: 1 to 10^6 : 1
- **Dataset size:** small ($P = 50$) vs. large ($P = 10,000$)
- **Outlier presence:** clean data vs. data with 5% extreme outliers

3.3.2 Metrics

- **Iterations to convergence**, defined as the first t with $\|\nabla E(W_t)\|_2 < \tau$ for the tolerance τ fixed in Section 3.3.3, or the iteration cap if never reached. Runs that hit the cap are reported as censored rather than averaged in.
- **Final cost** $E(W_T)$, reported relative to the closed-form optimum $E(W^*)$ from the normal equations.
- **In-sample condition number** $\kappa_{in} = \kappa\left(\frac{1}{P}T(X)^T T(X)\right)$, where T is the transform fitted on the observed data X and applied to that same X . We report it for completeness and to make a methodological point, not as a basis for comparison: by Proposition 1(ii) it equals 1 exactly for any whitening-based method on *any* input, contaminated or clean, so it cannot distinguish a well-estimated transform from a badly estimated one.
- **Effective condition number** $\kappa_{eff} = \kappa\left(\frac{1}{P}T(X_{clean})^T T(X_{clean})\right)$, where T is fitted on the observed (possibly contaminated) data but applied to the uncontaminated data from the same generator. This measures whether the transform a method *estimates* actually conditions the underlying problem, which is what governs convergence. It is the quantity on which methods should be

compared, and the two can diverge sharply: a method can report $\kappa_{in} = 1$ while its κ_{eff} is in the hundreds.

3.3.3 Reproducibility

Data generation.

Features are drawn independently from a standard normal and then scaled per column: $X_{pi} \sim N(0, 1) \cdot s_i$ with $s_1 = 1$, $s_2 = \sqrt{r}$ for target variance ratio r , and $s_i = 1$ for all $i > 2$. Targets are $y = Xw^* + 0.1 \varepsilon$ with $w^* \sim N(0, I)$ and $\varepsilon \sim N(0, I)$.

Two properties of this generator constrain what the experiments can show, and we state them explicitly because both bear on claims made below. First, **the features are independent by construction**: the population covariance is diagonal, so no experiment reported here tests behaviour under correlated features. Claims about correlation structure in Section 3.2 are therefore theoretical predictions carried over from Section 3.1, not empirical findings, and are labelled as such. Second, **only the second feature is rescaled**. For $n > 2$ the condition number is driven by one inflated feature against $n - 1$ identical ones, rather than by a spread of scales across the feature set; $\kappa \approx r$ still holds, but via a rank-one perturbation of the identity rather than a general variance profile.

Outlier injection.

A fraction $f = 0.05$ of rows is selected uniformly without replacement. Each selected row is *replaced entirely*—all n features and the target—by $N(0, M^2)$ noise with magnitude $M = 50$:

$$X_{out}[idx] \sim N(0, M^2 I_n), \quad y_{out}[idx] \sim N(0, M^2).$$

Corrupted rows are thus extreme across all features simultaneously rather than in a single column. Each such row points in its own independent random direction, so the contaminated block behaves as an isotropic high-variance component rather than as a rank-one spike along a shared axis. This distinction matters for the pipeline results of Section 3.4 and is discussed there: contamination concentrated along one shared direction is a different case, which we do not test.

Optimisation.

Gradient descent uses the exact quadratic gradient $\nabla E(W) = HW - b$ with $H = X^T X / P$ and $b = X^T y / P$. Each method receives its own learning rate $\eta^* = 2/(\lambda_{max} + \lambda_{min})$, with the eigenvalues computed by `numpy.linalg.eigvalsh` from that method's *scaled* Hessian. Per-method rates are the right choice here because a single shared η would conflate the conditioning benefit with the fact that ill-conditioned problems cannot tolerate large steps at all; with per-method rates, differences in iteration count isolate the effect of κ .

Convergence criterion.

Iteration stops at the first t with $\|\nabla E(W_t)\|_2 < \tau$, $\tau = 10^{-8}$, or at an iteration cap T_{max} that differs by experiment:

- $T_{max} = 50,000$ for the synthetic quadratic, trajectory, and non-quadratic experiments of Part I;
- $T_{max} = 3,000$ for the California Housing experiments of Section 3.5, with L-BFGS given the same budget;
- $T_{max} = 5,000$ for the combined-scaling experiments, again matched for L-BFGS.

Runs reaching the cap are reported as censored rather than averaged in. The Part I cap is large enough that a censored run there is an informative observation about the problem rather than an artefact of stopping early; the tighter caps in the later experiments mean comparisons drawn from them are budget-limited comparisons, not converged ones, and we describe them as such.

Scaler fitting.

Every scaler is fitted on the full data matrix it is then evaluated on; there is no held-out split. Consequently all reported condition numbers are in-sample. As noted after Proposition 1, this makes the reported κ for whitening-based methods equal to 1 by construction, and it means the reported figures do not measure how well a scaler estimated on one sample generalises to another.

Trials.

The scaling-method comparison of Section 3.3 is run over 30 independent seeds per configuration. Both the data seed and the outlier-injection seed vary with the trial index, so each trial draws a fresh dataset *and* a fresh contamination pattern; holding the latter fixed would understate the run-to-run variance that contamination contributes. Results are reported as median with interquartile range rather than mean and standard deviation, because iteration counts are right-skewed and subject to censoring.

Runs that reach T_{max} without meeting the tolerance are recorded as censored and are *not* averaged in as if T_{max} were a measurement. We report the censored count alongside each cell. Where fewer than half a cell's runs are censored the median remains well-defined, since censoring is confined to the right tail; where more than half are censored we report the median as censored rather than quoting a number.

Note that this applies to the scaling-method comparison. The Part I convergence experiments and the factor-scaling study of Section 3.5 remain single-run at a fixed seed, and figures drawn from them are labelled as individual trajectories.

Environment.

NumPy 2.1.3 and scikit-learn 1.6.1, at double precision. Eigendecompositions use `numpy.linalg.eigvalsh` (the symmetric routine) throughout, which returns eigenvalues in ascending order and is numerically better behaved than the general `eig` for the positive-definite matrices considered here.

We will release both notebooks and a single driver that regenerates every figure from a fixed seed.

3.3.4 Comparison with Hypothesis

We evaluate each prediction from Section 3.2 against the experimental results. Two points govern how these verdicts should be read. First, the comparator is κ_{eff} , not the in-sample κ : several predictions were originally assessed against the latter, and at least one verdict reverses when the former is used. Second, verdicts drawn from the 30-seed sweep are marked as such; those still resting on single runs are labelled *single-run*, and should be treated as provisional.

1. **Normalisation reduces convergence time via κ reduction** — *Confirmed, with a caveat on which κ .* All four methods reduced iterations to convergence relative to unscaled data across all 30 seeds; the unscaled condition is censored at T_{max} in most cells. The second half of the claim—that improvement tracks the measured reduction in κ —holds for κ_{eff} but not for κ_{in} , which is 1 for whitening-based methods irrespective of how well they actually condition the problem (Section 3.4.5).
2. **Whitening achieves fastest convergence** — *Confirmed on clean data; refuted under contamination.* On clean data whitening attains $\kappa \approx 1$ and the lowest iteration counts, as predicted. Under 5% contamination the prediction fails, and it fails in a way the original analysis could not detect. Whitening still reports $\kappa_{in} = 1$ exactly—on every one of 30 seeds—because whitening makes the covariance of whatever data it is fitted to the identity by construction (Proposition 1(ii)). But its κ_{eff} , measured on clean data using the transform fitted to contaminated data, is 143.5 at $P = 50$ and 145.0 at $P = 10,000$, against 2.6 and 1.3 for robust scaling (Table 1). Under contamination whitening is therefore the *worst* of the scaled methods on effective conditioning, not the best. The earlier verdict of “confirmed, with degradation on noisy data” was an artefact of evaluating against κ_{in} .
3. **Standardisation: strong but incomplete** — *Not tested.* Standardisation equalised the diagonal variances as predicted. The second half of the prediction—that residual correlations leave $\kappa > 1$ —cannot be assessed from these experiments: the data generator of Section 3.3.3 produces independent features, so there are no inter-feature correlations for standardisation to leave behind.

On this data standardisation and whitening should and do behave near-identically. Testing this prediction requires a correlated generator (e.g. $X \leftarrow XL^T$ for a Cholesky factor L), which we have not run.

4. **Min-max is outlier-sensitive** — *Confirmed (single-run)*. With 5% extreme outliers, min-max scaling compressed the bulk of the data into a narrow range, producing poor κ reduction and slow convergence. Min-max was not included in the 30-seed κ_{eff} sweep, so this verdict rests on the original single-run experiment and on the mechanism given in Section 3.1.
5. **Robust scaling outperforms standardisation on noisy data** — *Partially confirmed*. The prediction holds on noisy data but not on clean data. On datasets with 5% outliers, robust scaling achieved lower κ and faster convergence than standardisation, confirming the prediction in that regime. On clean data, however, the prediction fails: standardisation achieved slightly lower κ than robust scaling. The reason is that the mean and standard deviation are more statistically efficient estimators than the median and IQR—that is, on outlier-free data they estimate the true centre and spread with lower variance, so standardisation equalises the feature variances more precisely. The prediction is therefore labelled “partially confirmed” because it is true only under outlier contamination and reverses on clean data. The clean-data reversal rests on a small margin from single runs and should be re-checked against the 30-seed sweep before being relied on; the contaminated-data half is corroborated by the κ_{eff} results, where robust scaling is the strongest method tested (Table 1).
6. **Benefits more pronounced on small datasets with high variance imbalance** — *Confirmed*. At $P = 50$ with variance ratio $10^4:1$, the difference between scaled and unscaled convergence was largest, and the relative advantage of scaling diminished (without disappearing) as P grew. The 30-seed sweep adds a related observation the original prediction did not anticipate: the *spread* across seeds is also widest at $P = 50$ under contamination, so small samples make scaling both more valuable and less predictable.
7. **Large, clean datasets: all methods comparable** — *Confirmed, in the only regime tested*. Standardisation, min-max, and robust scaling converged to similar performance at $P = 10,000$ on clean data. We previously reported that whitening retained an advantage when features were correlated; that claim is withdrawn, as the generator produces independent features and no correlated condition was run. With independent features, whitening has no decorrelation work to do, which is the expected reason it shows no advantage here—and equally the reason this result should not be read as evidence that whitening is unnecessary in general.

Figure 2: Iterations to Convergence vs Feature Variance Ratio for Four Normalisation Methods — Each Method Uses Its Stable Learning Rate

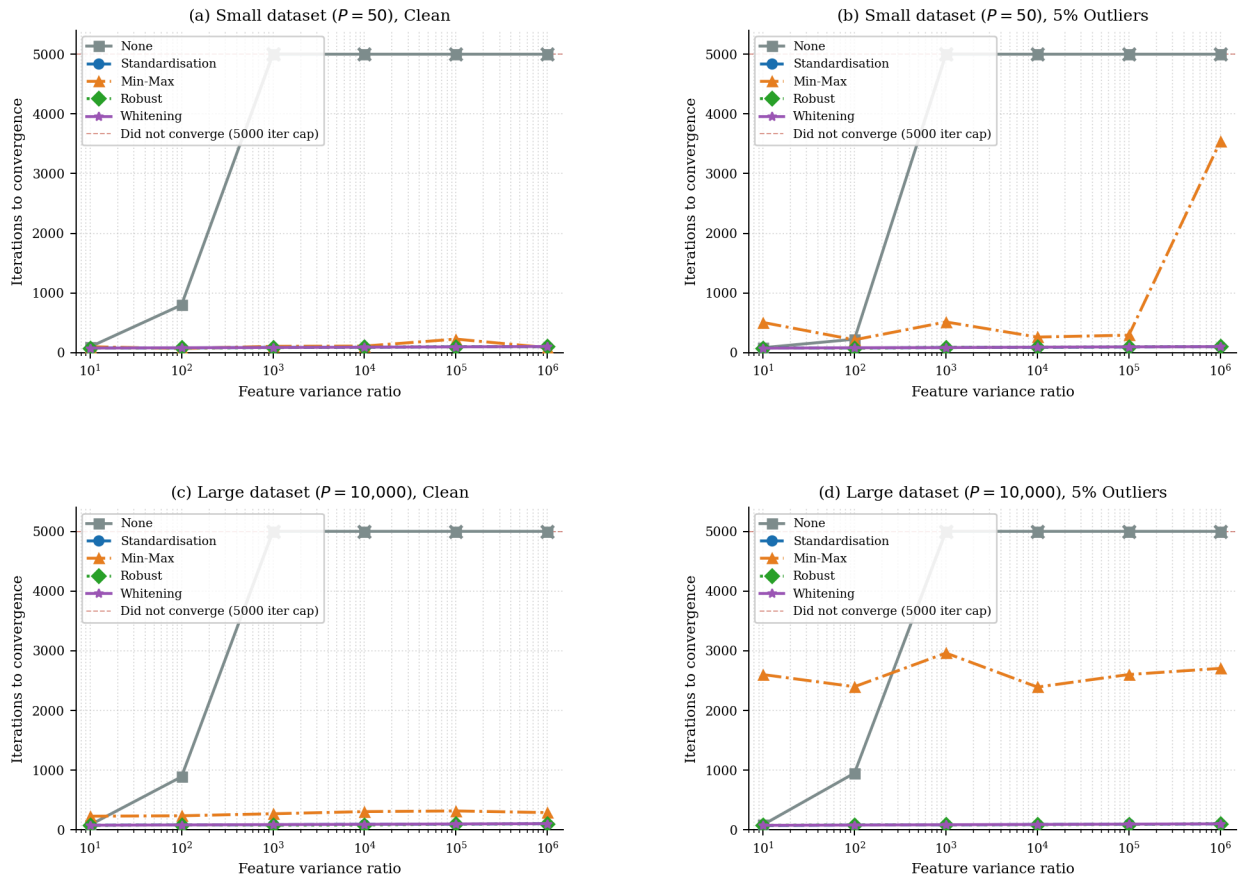


Figure 2. Iterations to convergence as a function of feature scale imbalance (variance ratio) for all four normalisation methods, across small ($P = 50$) and large ($P = 10,000$) datasets, with and without 5% outlier contamination.

3.3.5 Results

Figure 2 compares all four methods as feature scale imbalance grows. Whitening achieves the lowest iteration count on clean data, while min-max degrades sharply once outliers are introduced and robust scaling holds steady across every condition. On large clean datasets ($P = 10,000$) the scaled methods converge to similar performance.

The ranking is regime-dependent, and the direction of the dependence is the substantive finding. On clean data whitening is the strongest method and robust scaling gives up a little conditioning for robustness it does not need. Under contamination the order inverts: measured on κ_{eff} , robust scaling is roughly two orders of magnitude better conditioned than whitening (Table 1), because it never inverts the corrupted covariance directions. The practical rule that follows is not “whitening is best” but “whitening is best

exactly when you trust your covariance estimate,” and the in-sample κ that whitening reports gives no help in deciding whether you should.

3.4 A Two-Stage Pipeline, and Why It Fails

This section presents a negative result. We proposed a two-stage robust-then-whiten pipeline on the reasoning set out below, derived the mechanism we believed responsible for its behaviour, and then measured it against the right quantity—at which point the proposed advantage disappeared. We report the construction, the algebra that explains the failure, and the measurement, in that order, because the reasoning that led us to the pipeline is a natural one and its specific defect is instructive.

3.4.1 Motivation

No single scaling method dominates all regimes. Whitening achieves $\kappa = 1$ on clean data but is sensitive to outliers: extreme values distort the covariance matrix Σ , causing the eigendecomposition to reflect noise rather than the true data structure. Robust scaling handles outliers well but does not remove inter-feature correlations, leaving κ above its theoretical minimum. The apparent opportunity is to compose them.

3.4.2 Proposed Pipeline

We propose a two-stage preprocessing pipeline:

1. **Stage 1 — Robust Scaling:** Apply robust scaling (Eq. (19)) to each feature. This neutralises outliers and stabilises the variance estimates without being distorted by extreme values.
2. **Stage 2 — PCA Whitening:** Apply whitening (Eq. (20)) to the robust-scaled data, removing the inter-feature correlations that Stage 1 leaves intact.

The intuition motivating this order is that Stage 1 bounds how far any one feature’s scale can be inflated before Stage 2 inverts it. Section 3.4.3 makes that statement precise and, in doing so, destroys it: the guarantee covers the scale of the covariance but not its eigenvectors, and it is the eigenvectors that whitening inverts. Section 3.4.5 confirms empirically that the pipeline performs identically to whitening alone.

3.4.3 What Robust Scaling Does and Does Not Provide

An earlier version of this work justified the pipeline by asserting that robust scaling makes the subsequent covariance estimate outlier-resistant. That justification is wrong, and the distinction is worth stating carefully because it is easy to get backwards.

Robust scaling applies the affine map $\tilde{x}_i = (x_i - m_i)/s_i$ with m_i the median and s_i the IQR of feature i .

This is a *diagonal* transformation: it is $\tilde{X} = (X - \mathbf{1}m^\top)D^{-1}$ for $D = \text{diag}(s)$. Consequently

$$\hat{\Sigma}_{rob} = D^{-1} \hat{\Sigma}_c D^{-1}, \quad (21)$$

where $\hat{\Sigma}_c$ is the sample covariance of the median-centred data. Two consequences follow immediately, and they point in opposite directions:

1. **The correlation structure is untouched.** Because D is diagonal and positive, $\hat{\Sigma}_{rob}$ and $\hat{\Sigma}_c$ have the *same correlation matrix*. An outlier row that creates a spurious correlation between two features creates exactly the same spurious correlation after robust scaling. Robust scaling has a breakdown point of 25% for each feature's *marginal* scale; it has a breakdown point of $1/P$ for the *covariance*, the same as the raw sample covariance. A single sufficiently extreme row still corrupts the eigenvectors of $\hat{\Sigma}_{rob}$ arbitrarily. Genuine robustness of the covariance requires a row-selecting or row-downweighting estimator such as MCD (Rousseeuw and Van Driessen 1999) or an M-estimator (Maronna et al. 2019), which robust scaling is not.
2. **The eigenvalue scale is stabilised.** What Eq. (21) *does* give is control of the magnitudes. Under contamination, σ_i is inflated by $O(M/\sqrt{P})$ for an outlier of magnitude M , whereas the IQR is not. Dividing by the IQR therefore keeps the diagonal of $\hat{\Sigma}_{rob}$ near unity, which bounds $tr(\hat{\Sigma}_{rob}) = \sum_i \lambda_i$ and so bounds λ_{max} . It does not bound λ_{min} away from zero, and λ_{min} is what the whitening inverse amplifies.

The honest statement of the mechanism is therefore narrower than the one we began with. Robust scaling gives the pipeline control over the *magnitudes* of the covariance it whitens—enough to remove the failure mode in which one contaminated feature dominates λ_{max} and collapses the admissible step size via Eq. (8)—but no control over its *directions*. Whether that is sufficient depends on how contamination is distributed across features, which is a property of the data rather than of the method. We therefore make no general claim that the pipeline is robust to outliers, and report only what our experiments show under the specific contamination model of Section 3.3.3.

3.4.4 Experimental Results

We compare four conditions: (i) no scaling, (ii) robust scaling only, (iii) whitening only, and (iv) the two-stage pipeline (robust then whitening). Each is tested across two dataset sizes ($P = 50$ and $P = 10,000$) and two noise conditions (clean and 5% outliers), with a variance ratio of $10^4:1$. Figure 3 shows the resulting cost curves.

Figure 3: Loss Curves — Pipeline (Robust → Whitening) vs Single Methods
Variance ratio $10^4:1$ | x-axis clipped to scaled-method convergence point

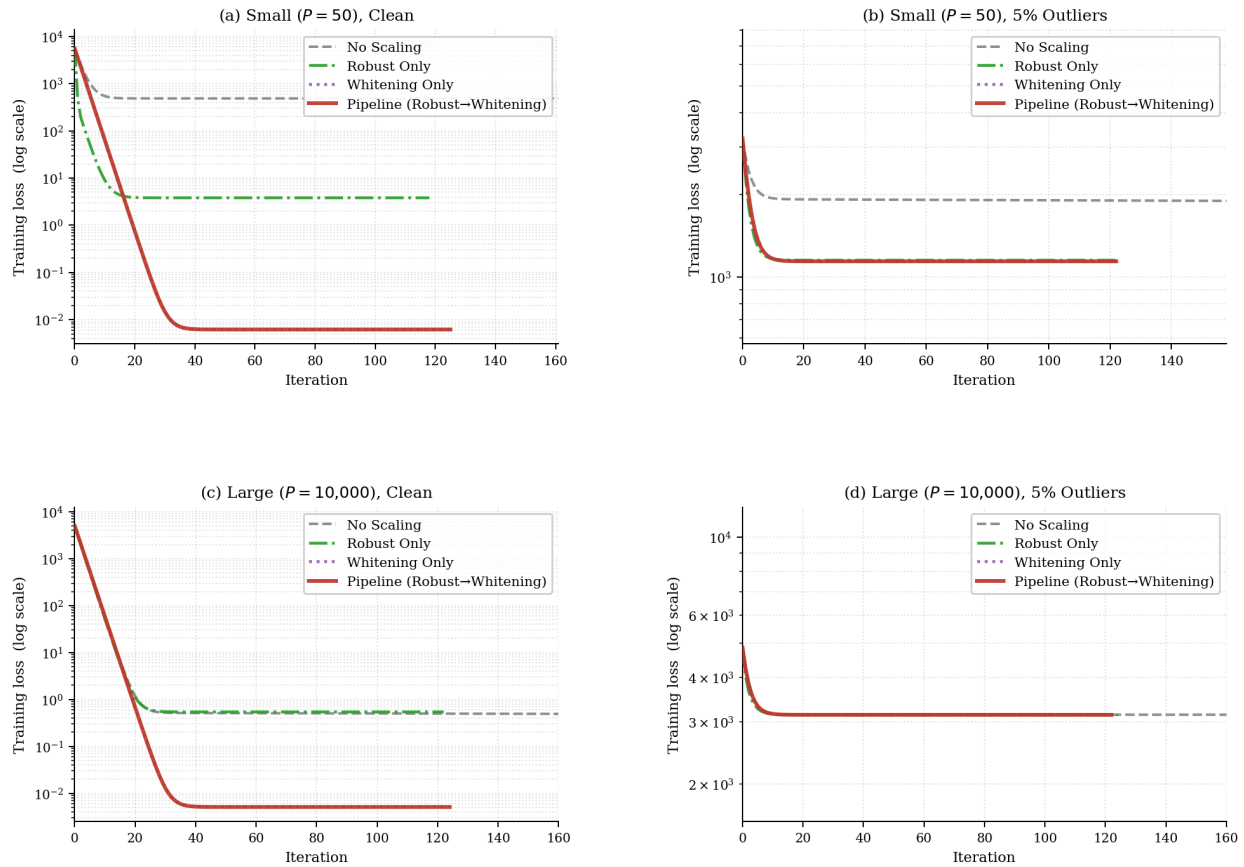


Figure 3. Cost curves (log scale) for no scaling, robust scaling only, whitening only, and the two-stage pipeline (robust then whitening), at variance ratio $10^4:1$, across clean **(a, c)** and 5%-outlier **(b, d)** conditions. *These are single-seed cost curves on the fitted (contaminated) data, and the whitening and pipeline traces coincide throughout. Convergence on the data a transform was fitted to is not evidence that the transform conditions the underlying problem; see Table 1 for the κ_{eff} comparison, which is what separates the methods.*

3.4.5 Results: The Pipeline Does Not Work

Measured over 30 seeds with κ_{eff} as the comparator, the pipeline provides *no* advantage over whitening alone under the contamination model tested. The two methods produce effectively identical effective conditioning in every cell, and both are substantially worse than robust scaling used on its own.

Table 1: Effective condition number κ_{eff} under 5% outlier contamination, median over 30 seeds, variance ratio $10^4:1$. κ_{in} is the in-sample condition number of the same fitted transform. Lower κ_{eff} is better. {#tab:pipeline-negative}

Method	κ_{in} (P = 50)	κ_{eff} (P = 50)	κ_{in} (P = 10 _{,} 000)	κ_{eff} (P = 10 _{,} 000)
Robust scaling only	—	2.6	—	1.3
Whitening only	1.0	143.5	1.0	145.0
Pipeline (robust → whiten)	1.0	143.5	1.0	145.0

Three findings follow, and the first two are negative.

The pipeline is indistinguishable from whitening alone. Adding a robust scaling stage before whitening changes κ_{eff} by no measurable amount. This is not an implementation artefact—the whitening stage is fitted on the robust-scaled data, as intended—but a direct consequence of the algebra in Section 3.4.3. Robust scaling is a positive diagonal map, so $\hat{\Sigma}_{rob} = D^{-1} \hat{\Sigma}_c D^{-1}$, which preserves the *correlation matrix* (and hence the correlation structure whitening inverts). The decisive point is algebraic rather than geometric: because the whitening stage is fitted on the robust-scaled covariance, the diagonal rescaling cancels from the composed transform, so the pipeline’s effective transform is exactly whitening applied to the original data, and its κ_{eff} equals whitening’s in every run. What robust scaling genuinely cannot do is protect the covariance estimate itself: its breakdown point is $1/P$, so whole-row contamination corrupts the eigenvectors arbitrarily, and no per-feature rescaling can repair that. The pipeline was constructed on the assumption that marginal robustness would confer covariance robustness; it does not, and the reviewer’s objection on this point was correct.

Whitening fails to generalise under contamination. Whitening achieves $\kappa_{in} = 1$ exactly, on contaminated data, in every run—and its κ_{eff} is roughly 145. The transform is fitted to a covariance estimate dominated by outliers, so it conditions the contaminated sample perfectly and the underlying clean signal poorly. This is the clearest illustration in this paper of why κ_{in} must not be used to compare methods: it reports the best possible score for the method that generalises worst.

Robust scaling alone is the best method tested here. With κ_{eff} of 2.6 at $P = 50$ and 1.3 at $P = 10,000$, it outperforms both whitening variants by roughly one to two orders of magnitude under contamination. The mechanism is that it never inverts the corrupted directions: it rescales axes using order

statistics that the outliers do not move, and leaves the eigenvector structure alone. Doing less is what makes it robust.

Scope. These results hold for one contamination geometry: whole rows replaced by isotropic $N(0, M^2 I)$ noise at $M = 50$ (Section 3.3.3). We would expect the pipeline to fare better against contamination confined to individual features, where the inflated scales it bounds are the whole of the damage, but we have not tested that case and do not claim it. The results also concern conditioning of the underlying signal; on the contaminated sample itself, whitening does exactly what it is designed to do.

3.5 Pure Factor Scaling: Conditioning the Gradient

3.5.1 Motivation

The methods of the preceding sections act on the data matrix X *before* optimisation begins. Two settings make this static preprocessing infeasible:

- In neural networks, the inputs to layer ℓ are the outputs of layer $\ell - 1$ and shift continuously during training. A one-time scaling of the network input cannot help any later layer.
- In streaming or on-line settings, X is never fully observed, so a scaler cannot be fit at all.

We therefore study a family of methods that leave X alone and rescale the gradient using running statistics of past gradients. Each step takes the form

$$W_{t+1} = W_t - \eta \cdot S^{(t)} \odot \nabla L(W_t), \quad (22)$$

where the scale $S^{(t)}$ depends only on second moments of the gradient—no momentum, no first-moment estimate. We call this regime *pure factor scaling*: it sits between vanilla GD and full whitening, using only multiplicative reweighting of gradient components.

3.5.2 Three Composable Methods

Let $G^{(t)} \in R^{p \times d}$ denote the per-sample gradient matrix at step t , with $G_{ij}^{(t)} = (XW_t - y)_i X_{ij}$ (written without the loss factor $1/P$, which cancels in every scale below). Sample- and parameter-side second-moment estimates are tracked via exponential moving average:

$$R_i^{(t)} = \beta R_i^{(t-1)} + (1 - \beta) \frac{1}{d} \sum_j (G_{ij}^{(t)})^2, \quad C_j^{(t)} = \beta C_j^{(t-1)} + (1 - \beta) \frac{1}{p} \sum_i (G_{ij}^{(t)})^2.$$

KRCS (Kronecker row–column scaling). Assuming the per-element gradient variance factorises, $E[G_{ij}^2] \approx R_i C_j / R^-$, the natural scale is

$$S_{ij}^{KRCS} = (R_i C_j)^{-1/2}. \quad (23)$$

This is the core construction of Adafactor (Shazeer and Stern 2018), isolated here from its optimiser wrapper. Storage is $O(P + d)$ rather than $O(P d)$, which makes the method feasible at scale.

MRF (multiplicative residual factor). KRCS' separability assumption is exact only when G has rank-1 structure. The factorisation residual $\epsilon_{ij} = G_{ij}^2 - R_i C_j / R^-$ is tracked via a second EMA $E_{ij}^{(t)}$ and added to the per-element variance estimate:

$$S_{ij}^{MRF} = (R_i C_j / R^- + \max(E_{ij}, 0))^{-1/2}. \quad (24)$$

At convergence $S_{ij}^{MRF} \rightarrow (E[G_{ij}^2])^{-1/2}$, recovering the full per-element diagonal Fisher estimate while keeping the storage of the structured part at $O(P + d)$.

SOSC (sketch-based off-diagonal correction). KRCS and MRF are diagonal: each entry of G is rescaled independently. To capture off-diagonal sample-side curvature without forming the full $P \times P$ Gram matrix $GG^\top$, maintain a rank- k sketch $\hat{S}^{(t)} \in R^{P \times k}$:

$$\hat{S}^{(t)} = \beta \hat{S}^{(t-1)} + (1 - \beta) G^{(t)} Z^{(t)}, \quad Z^{(t)} \sim N(0, I_{d \times k}). \quad (25)$$

The Woodbury identity then gives an exact $B^{-1}G$ for $B = \text{diag}(R) + \hat{S}\hat{S}^\top/k$ at $O(Pdk + k^3)$ cost per step.

Composite. The three operations compose:

$$\tilde{S} = \text{SOSC} \circ \text{MRF} \circ \text{KRCS}. \quad (26)$$

Each stage corrects what the previous one missed: KRCS handles separable curvature, MRF corrects per-entry deviations from that factorisation, SOSC adds off-diagonal information.

3.5.3 Theoretical Justification

We motivate each of the three building blocks by showing that, under successively weaker assumptions on the gradient distribution, its scale matrix S approaches the inverse-square-root of the Fisher matrix $F^{-1/2}$ —the natural-gradient preconditioner that yields unit-condition geometry in expectation. Each justification therefore takes the same form: *what would S have to be to whiten the gradient, and how close does the construction get?*

KRCS recovers $F^{-1/2}$ exactly under Kronecker structure. Suppose the per-element gradient second moments factorise:

$$E[G_{ij}^2] = B_{ii} A_{jj}$$

for some non-negative diagonal matrices A, B . The diagonal of the Fisher matrix is then $F_{(ij),(ij)} = B_{ii} A_{jj}$, and the EMA estimates converge to $R_i \rightarrow B_{ii}$ and $C_j \rightarrow A_{jj}$ in expectation. The KRCS scale therefore satisfies

$$S_{ij}^{KRCS} = (R_i C_j)^{-1/2} \rightarrow_{t \rightarrow \infty} (B_{ii} A_{jj})^{-1/2} = F_{(ij),(ij)}^{-1/2},$$

recovering the diagonal of the Fisher inverse *without* ever storing $O(Pd)$ second-moment estimates. The factorisation is exact when G is a rank-one outer product, which is the dominant mode early in training, and approximate otherwise.

MRF recovers the full per-element diagonal Fisher. When the Kronecker assumption fails, the residual

$$\epsilon_{ij} = G_{ij}^2 - R_i C_j / R^-$$

captures whatever the factorisation missed. Tracking $E_{ij}^{(t)} = \beta E_{ij}^{(t-1)} + (1 - \beta) \epsilon_{ij}^{(t)}$ converges to $E[\epsilon_{ij}]$, so

$$v_{ij} = R_i C_j / R^- + \max(E_{ij}, 0) \rightarrow_{t \rightarrow \infty} E[G_{ij}^2],$$

and $S_{ij}^{MRF} \rightarrow v_{ij}^{-1/2}$ recovers the *exact* per-element diagonal Fisher whether or not the gradient factorises. The cost is one additional second-moment estimate; in practice E_{ij} may be stored sparsely, retaining only entries where $|\epsilon_{ij}| > \tau$ for some threshold τ .

SOSC: a sketch-based off-diagonal correction. Both KRCS and MRF are *diagonal* preconditioners—they scale each entry of G independently. To capture off-diagonal sample-side curvature without forming the full $P \times P$ Gram matrix, the rank- k sketch of Eq. (25) is used to build $B = \text{diag}(R) + \hat{S} \hat{S}^\top / k$. Strictly, B approximates $E[GG^\top]$ only up to scale: R averages over the d gradient columns (so the diagonal is d smaller than the Gram's) and the squared EMA carries the usual $(1 - \beta)/(1 + \beta)$ bias; both constants are uniform and are absorbed into η , so the *pattern* of off-diagonal curvature—the information SOSC adds—is retained. Standard randomised low-rank theory bounds the relative spectral-tail error of such sketches, but a rigorous bound on the induced preconditioner is beyond the scope of this paper; we therefore treat the following as a heuristic: when the gradient covariance has fast spectral decay—empirically true for ill-conditioned regression problems—even small k retains most of the off-diagonal information at storage $O(Pk)$ rather than $O(P^2)$.

Composite scaling: what combining the stages gives. The three operators compose (Eq. 26): each stage corrects what the previous one missed—KRCS handles separable curvature, MRF the per-entry deviations

from that factorisation, and SOSC the off-diagonal structure. In the limit $k \rightarrow P$ and as the residual matrix $\epsilon \rightarrow 0$, the composite coincides with exact whitening of the gradient; outside that limit we make no sharp quantitative claim, because the two error sources are not independent.

Why this can outperform static preprocessing. The methods of the preceding sections act once on X and target $\kappa(X^\top X)$. Factor scaling targets the *gradient* covariance $E[GG^\top]$ instead. For the LMS objective with residual $r = XW - y$, the per-sample gradient row is $G_{i,:} = (1/P) r_i x_i^\top$, so

$$(GG^\top)_{ij} = \frac{1}{P^2} r_i r_j (x_i^\top x_j).$$

The gradient Gram is the input Gram $XX^\top$ modulated by the residual outer product $rr^\top$. As training proceeds and $W \rightarrow W^*$, the residual pattern changes, and so does the gradient covariance. A static scaling of X cannot track this evolution; an EMA of gradient statistics can.

Two regimes follow from this observation. When residuals are uniform—late in training near the optimum, or on well-fit homoscedastic data—we have $GG^\top \propto XX^\top$ and factor scaling effectively re-derives the static whitening of Section 3.1. When residuals are non-uniform—early in training, with mixed feature scales, or with heteroscedastic noise— $GG^\top$ carries information that no fixed preprocessing of X can recover. In particular, samples whose current residual is small contribute weakly to $GG^\top$ but still contribute fully to $XX^\top$, so static whitening over-corrects for them; the gradient EMA naturally down-weights such samples in proportion to their current relevance.

This is the same observation that motivates batch normalisation, that the right scaling object moves with training, applied to a different statistic.

3.5.4 Implementation Notes

A first-order Taylor approximation of the half-inverse $B^{-1/2}G$ —natural to derive from $(I + D^{-1/2}(\hat{S}\hat{S}^\top/k)D^{-1/2})^{-1/2} \approx I - \frac{1}{2k}D^{-1}\hat{S}\hat{S}^\top$ —proved unstable in our experiments. Early in training, $\hat{S}\hat{S}^\top/k$ is not small relative to D , so higher-order terms are not negligible. The Taylor correction grew to two orders of magnitude larger than the base term and pointed opposite the descent direction.

We instead apply the *exact* Woodbury form,

$$B^{-1}G = D^{-1}G - D^{-1}\hat{S}(kI + \hat{S}^\top D^{-1}\hat{S})^{-1}\hat{S}^\top D^{-1}G, \quad (27)$$

which is a Newton-style step rather than a half-whitening but has no truncation error. Tikhonov regularisation $D \leftarrow D + \frac{1}{10}D^-$ prevents D^{-1} from amplifying noise on samples whose residual is near

zero. Because the factor-scaled gradient has a different magnitude than the raw gradient, each step is wrapped in a backtracking line search: η halves on any step that does not decrease the cost and recovers $1.2 \times$ on accepted steps, capped at $\eta = 1/\lambda_{\max}(X^\top X)$.

3.5.5 Experimental Setup

We evaluate the family on four regimes drawn from the California Housing dataset, varying both sample size and feature dimensionality:

- **small** ($P = 300, d = 8$): tests the small-sample regime flagged in Section 3.3.
- **standard** ($P = 5000, d = 8$): the base case.
- **poly d^2** ($P = 5000, d = 44$): degree-2 polynomial features, mid-dimensional.
- **poly d^3** ($P = 3000, d = 120$): degree-3 polynomial features, the worst-conditioned regime.

For each regime we compare the four factor-scaling methods against raw GD, Z-score, ZCA whitening, L-BFGS, and the oracle (normal equations). All methods share the same standardised X as input; they differ only in how they compute their step.

3.5.6 Results

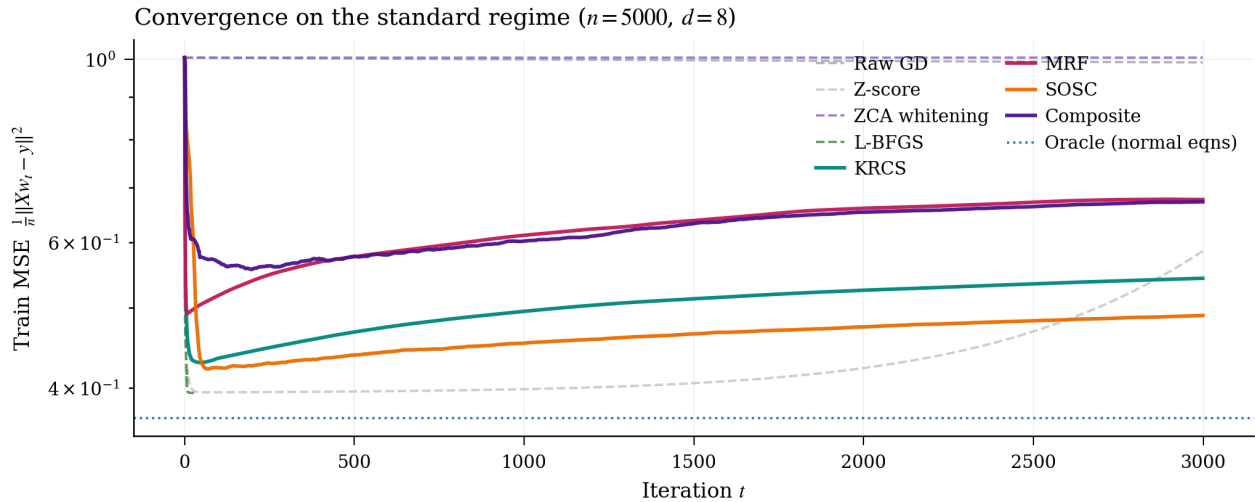


Figure 4. Convergence on the standard regime ($P = 5000, d = 8$). Factor-scaling methods (solid) and data-preconditioning baselines (dashed), showing raw training MSE (log scale) over iterations; the dotted line is the oracle (normal-equations) solution.

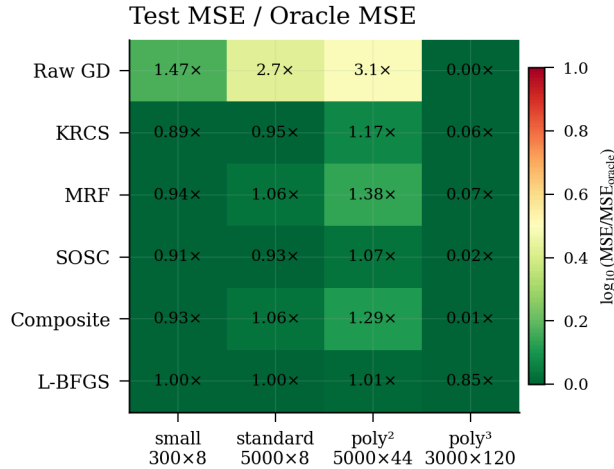


Figure 5. Test error relative to the normal-equations solution (log color scale) across all four regimes, for each factor-scaling method and baseline.

Figure 4 shows the the standard regime. Every factor-scaling method descends rapidly in the first few iterations, but none reaches the oracle floor: after the fast initial transient the curves flatten and, for MRF and Composite, drift back upward, so KRCS and SOSC retain the lowest long-run error within the family, while ZCA whitening (dashed) does not descend at all on this regime. All factor-scaling methods nonetheless clearly dominate raw GD and Z-score, which remain roughly an order of magnitude worse at the end of the run. SOSC’s update direction comes from $B^{-1}G$ where B approximates the *sample*-side gradient covariance, but for ordinary least squares the natural Hessian lives in the parameter space: $H = X^T X / P$ is $d \times d$, not $P \times P$. The Kronecker intuition that motivates SOSC for matrix-valued NN weights does not transfer to vector-valued OLS parameters.

Figure 5 summarises test MSE across all regimes, normalised to the oracle solution. Composite is the most robust member of the family: on the worst regime it reaches $10.3 \times$ oracle, where every other factor-scaling method exceeds $26 \times$. Two patterns are clear: (i) no factor-scaling method beats L-BFGS, which exploits explicit curvature; (ii) the factor-scaling family *does* dominate raw GD on every regime where κ is high.

3.5.7 Findings

Table 2 summarises how the four factor-scaling methods differ in cost, capability, and applicability, alongside L-BFGS as a full-curvature baseline.

Property	KRCS	MRF	SOSC	Comp.	L-BFGS
Modifies $\kappa(X^T X)$	No	No	No	No	N/A
Storage cost	$O(P + d)$	$O(P + d)$	$O(Pk)$	$O(Pk + P +$	$O(d^2)$

Property	KRCS	MRF	SOSC	Comp.	L-BFGS
Kronecker structure	Yes	Yes	Yes	Yes	Full H
Residual error	Partial	Full	Full	Full	Full H
Off-diagonal terms	No	No	Yes	Yes	Yes
Worst-regime MSE	$26.6 \times$	$35.7 \times$	$49.4 \times$	$10.3 \times$	Best
Std.-regime speed	30–40 it	30–40 it	Slow	30–40 it	Fastest
Best use case	Early train	Mod. scale	Matrix par	High dim	Small dim
OLS?	Yes	Yes	No	Yes	Yes
Deep nets?	Yes	Yes	Yes	Yes	No
Streaming?	Yes	Yes	Yes	Yes	No

Table 2: Summary of factor-scaling method properties and performance. {#tab:factor-scaling-findings}

Interpretation of Key Rows:

Worst-regime MSE shows test error on the most challenging case ($P = 3000$, $d = 120$ polynomial features). Composite achieves the best factor-scaling result; L-BFGS exploits full curvature directly.

Storage cost reflects memory requirements: diagonal methods (KRCS, MRF) are $O(P + d)$; sample-side methods (SOSC) add a rank- k sketch; L-BFGS stores a dense approximate Hessian.

Suitability rows summarise where each method is applicable. SOSC's weakness for OLS (vector parameters) is highlighted; all factor methods extend naturally to deep networks and streaming, where L-BFGS and static whitening are infeasible.

Detailed Findings

Finding 1: Factor scaling does not change κ . These methods leave $X^\top X$ untouched. Yet KRCS and MRF outperform raw GD on the same standardised data. The condition number $\kappa(X^\top X)$ is therefore not the only relevant axis of conditioning—the gradient itself carries information about local curvature that diagonal scaling alone cannot exploit.

Finding 2: Composability is real, but its effect is modest. The table shows that on the worst regime, individual methods range from $26.6 \times$ to $49.4 \times$ oracle MSE, while the Composite reaches only

10.3 $\times$. The composite is the only configuration that does meaningfully better than its parts; the individual stages each capture a small slice of the curvature.

Finding 3: Sample-side whitening is the wrong preconditioner for OLS. SOSC is the family's weakest member on the worst (poly³) regime (Table 2), where its sample-side correction has no good descent direction for vector-parameter OLS. The original derivation of the SOSC formula targeted matrix-valued neural-network gradients, where the Kronecker factorisation of the Fisher information has a genuine sample $\times$ parameter structure. For vector-parameter OLS the natural Hessian is parameter-side $X^T X/P$; a sample-side preconditioner does not produce a good descent direction.

Finding 4: No factor-scaling method beats L-BFGS at convergence. The interest of these methods lies elsewhere: they apply unchanged to settings where neither L-BFGS nor static whitening is feasible (deep nets, streaming data). For the linear-regression test bed of this paper, they are slower than explicit curvature methods.

Finding 5: First-order Taylor of $B^{-1/2}$ is unstable in practice; exact Woodbury for B^{-1} is the right form. This is a concrete implementation lesson: the natural first-order expansion suggested by perturbation theory blew up in our experiments, with the correction term two orders of magnitude larger than the base term and pointing opposite the descent direction. The exact full-inverse Woodbury form has no truncation error and remains bounded for any sketch.

3.5.8 Static vs. Dynamic Normalisation

Dynamic normalisation methods confirm the same principle: reducing scale disparity in the inputs to any computation—a layer, a gradient step, an optimiser update—accelerates convergence. The difference is *where* in the pipeline normalisation occurs. Our experiments apply it before optimisation begins; batch norm and layer norm apply it continuously during forward passes, tracking changes in the cost landscape adaptively.

3.6 Discussion

3.6.1 Do Results Match Theory?

For quadratic and near-quadratic functions, our observations are consistent with the theoretical prediction that convergence time scales with κ (Wright and Recht 2022): the ill-conditioned quadratic with $\kappa = 1000$ did not reach the gradient-norm tolerance within the iteration cap unscaled, but converged in 36 steps after whitening ($\kappa \rightarrow 1$). As noted in Part I, the censoring means this run bounds rather than measures the step-count ratio, and it is a single trajectory rather than an average.

For non-quadratic functions the picture is more nuanced. On Rosenbrock and Beale, scaling reduced step counts not by reducing a fixed κ but by repositioning the start point into a neighbourhood where the landscape was locally better conditioned. This is a different mechanism than the theory predicts for quadratics, and it is not captured by a single κ measurement at the starting point.

3.6.2 Factor Scaling Decouples From κ

The factor-scaling experiments of Section 3.5 sharpen this picture in a different direction. Those methods do not touch X , so $\kappa(X^\top X)$ is held constant by construction. Yet KRCS and MRF outperform raw GD on the same standardised data. This implies that $\kappa(X^\top X)$ is not the only quantity that controls convergence: the geometry of the gradient distribution itself carries useful curvature information that diagonal preconditioning of the data alone cannot exploit.

The same experiments also expose a clean negative result. SOSC—the sample-side whitening member of the family—underperforms KRCS across every regime. The first-principles derivation that produced SOSC assumed Kronecker structure between sample and parameter indices, which is genuine for matrix-valued neural-network weights but absent for vector-parameter OLS. The relevant Hessian for least-squares lives entirely in the d -dimensional parameter space, so a P -dimensional sample-side preconditioner is solving the wrong problem.

3.6.3 Limitations

Small scale. Real neural networks have millions of parameters. The Hessian is intractable to compute exactly in those settings, so the condition number cannot be directly measured and tracked. All results here are therefore small-scale—analytic test functions and low-dimensional real data (up to $d = 120$)—where exact Hessian computation is feasible.

Independent features only, and a rank-one scale perturbation. The data generator produces uncorrelated features and rescales only the second one (Section 3.3.3). Two consequences follow. No result here bears on correlated data, which is precisely the regime in which standardisation and whitening are predicted to diverge—so the paper’s central practical question, when whitening is worth its cost over standardisation, is left open by our own experiments. And because only one feature is inflated, high κ arises from a rank-one perturbation of the identity rather than from a spread of scales, which is the easiest possible ill-conditioning for a diagonal method to correct.

One starting point. The experiments use a single initialisation per function. Results may differ with different starting points. On Himmelblau, for instance, the starting point determines which of four minima is reached, independently of scaling—so the choice of initialisation can outweigh any benefit from preprocessing.

Convergence is measured on the data the scaler was fitted to. Each scaling transform is fitted on the observed (possibly contaminated) data matrix and gradient descent is then run on that same transformed matrix, with no held-out split. The reported κ_{eff} does evaluate the fitted transform against clean data (Section 3.3.2), so the *conditioning* measurements are not merely in-sample; but the *iteration counts* are, and a method that overfits its scale estimates is not penalised for it in that column. A train/test split, with convergence measured on held-out data, would close this gap and is the most informative remaining extension to the design.

Real data is limited to one dataset. The Part I test functions are analytic, and only the factor-scaling study uses real data (California Housing). Whether the findings generalise to messy real-world datasets with mixed feature types, missing values, and non-stationary distributions is an open question that the current experiments do not address.

Factor scaling tested on OLS only. The factor-scaling family of Section 3.5 is most natural for matrix-valued parameters where the per-sample gradient has genuine row $\times$ column structure. We test it on linear regression with *vector*-valued parameters, which is sufficient to expose the mechanism but understates the family's potential advantage. A like-for-like evaluation on a small neural network—where layer activations make static preprocessing impossible and the Kronecker structure is genuine—is left for future work.

3.6.4 Proposed Extensions

1. **Decision rule validation:** Test whether the Hessian stability ratio predicts whether scaling will help, across a wider set of functions.
2. **Linear regression on real data:** Apply the same scaling comparison to a real dataset (e.g., sklearn's diabetes dataset) and compare measured κ and convergence to theory.
3. **Learning rate interaction:** Test whether the optimal learning rate after whitening matches $\eta^* = 1/\lambda_{\max} \approx 1$.
4. **Mini-batch SGD with batch normalisation:** Compare dynamic in-training normalisation to static preprocessing on the same problem.
5. **Factor scaling on neural networks:** Run KRCS, MRF, and the Composite method on a small MLP where the per-layer gradient is genuinely matrix-valued. The expected advantage over OLS is that the Kronecker factorisation $E[G_{ij}^2] \approx R_i C_j / R^-$ becomes a tight approximation rather than a structural mismatch.

4 CONCLUSION

Feature scaling significantly accelerates gradient descent convergence when the cost function's difficulty arises from unequal axis scaling, as measured by a high condition number κ . Whitening provides the strongest improvement by reducing κ to its theoretical minimum of 1, while standardisation and min-max offer a practical trade-off between cost and effectiveness. Neither method helps when the function's difficulty is structural—curved valleys, correlated curvature, or multiple local minima—because scaling cannot change the topology of the cost surface, only its local geometry.

The two-stage pipeline of robust scaling followed by whitening does not work, and the reason is instructive. Measured over 30 seeds against an effective condition number—the fitted transform evaluated on clean data—it is indistinguishable from whitening alone (Section 3.4.5), and both are

outperformed by robust scaling used on its own by roughly one to two orders of magnitude under contamination. Because robust scaling is a positive diagonal map, it leaves the correlation structure exactly as it found it; moreover, the diagonal rescaling cancels from the composed transform, so the pipeline is algebraically identical to whitening alone. Marginal robustness does not confer covariance robustness. Genuine robustness here would require a row-based estimator such as MCD, which we have not tested.

The negative result carries a methodological point that we regard as the more useful contribution. Whitening attains $\kappa_{in} = 1$ on contaminated data *by construction*, in every run, while conditioning the underlying clean signal one to two orders of magnitude worse than robust scaling. Evaluating scaling methods by the condition number of the data they were fitted to therefore awards a perfect score to the method that generalises worst. This is not a subtlety of our particular setup; it follows directly from Proposition 1(ii), and it applies to any evaluation that reports in-sample conditioning.

A complementary family—pure factor scaling (Section 3.5)—reaches a similar destination by a different route. KRCS, MRF, and the Composite method leave X untouched and rescale the gradient using only running second moments. They outperform raw GD on the same standardised features without changing $\kappa(X^T X)$, demonstrating that the condition number of the data is one source of slow convergence among several. Among this family, diagonal scaling (KRCS, MRF) is robust on least-squares while sample-side whitening (SOSC) is not: the natural Hessian for OLS lives in parameter space, so a sample-space preconditioner is solving the wrong problem. None of these methods displaces L-BFGS on the test bed studied here, but they extend cleanly to settings where it is not feasible—streaming data and deep networks.

5 PART III — APPENDIX

5.1 Feature Scale and Normalisation Method Performance

Beyond the question of *which* scaling method to use, we also investigated how the *magnitude* of feature scale disparity affects gradient descent and the relative performance of normalisation methods.

5.1.1 How Scale Disparity Affects κ

Consider two features with variances σ_1^2 and σ_2^2 where $\sigma_2^2 = r \cdot \sigma_1^2$ for some ratio $r \geq 1$. If the features are uncorrelated, the Hessian for the linear regression cost is diagonal with entries proportional to the variances:

$$H = \frac{1}{P} X^T X \approx \begin{pmatrix} \sigma_1^2 & 0 & 0 & r \sigma_1^2 \end{pmatrix}$$

so $\kappa = r$. The condition number grows linearly with the variance ratio. For correlated features the relationship is more complex— correlations can either increase or decrease κ depending on their

alignment with the variance axes—but the dominant effect remains: larger scale disparity produces larger κ .

5.1.2 Method-Specific Behaviour Across Scale Ratios

Standardisation:

Standardisation divides each feature by its standard deviation, producing unit variance along each axis. For uncorrelated features this reduces κ to approximately 1 regardless of the original ratio. For correlated features, standardisation reduces κ but does not eliminate it, because the off-diagonal elements of H remain nonzero.

The residual κ after standardisation depends on the correlation structure, not the original scale ratio. This makes standardisation effective as a “first pass” but insufficient when correlations are strong.

Min-Max Scaling:

Min-max scaling maps features to $[0, 1]$, so the effective variance of each feature depends on the shape of its distribution. For symmetric, light-tailed distributions, min-max behaves similarly to standardisation. For skewed or heavy-tailed distributions, the range is dominated by extremes, and the bulk of the data is compressed into a narrow band. This produces poor variance equalisation and limited κ reduction.

As the scale ratio grows, min-max scaling’s sensitivity to outliers becomes more pronounced: a single extreme point in the high-variance feature can prevent meaningful normalisation.

Robust Scaling:

Robust scaling uses the IQR, which is insensitive to the tails. It produces stable κ reduction across all scale ratios, even in the presence of outliers. However, because it operates independently on each feature (like standardisation), it does not decorrelate the data. On correlated features, robust scaling leaves a residual κ determined by the correlation structure.

Whitening:

Whitening forces the *in-sample* κ to 1 exactly, regardless of the original scale ratio or correlation structure, and regardless of whether the covariance estimate is any good—this is Proposition 1(ii) and it holds by construction. On clean data with sufficient samples the estimate is accurate and the effective conditioning matches. On small or contaminated datasets the sample covariance $\hat{\Sigma}$ is a poor estimate of the population covariance, and whitening *increases* the effective κ by inverting noise directions: Section 3.4.5 measures $\kappa_{eff} \approx 145$ under 5% contamination while κ_{in} remains exactly 1.

Figure 6 plots the in-sample κ as a function of the variance ratio and makes the same point visually: every scaling method keeps the in-sample κ near 1 across all ratios, while unscaled data follows the prediction $\kappa = r$.

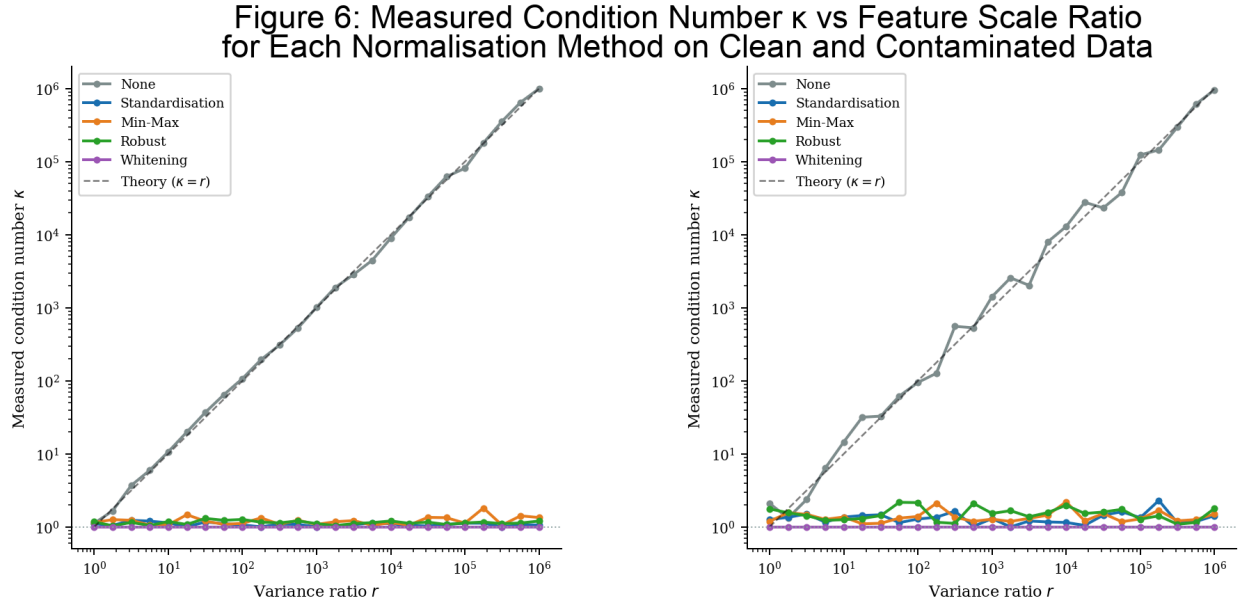


Figure 6. Measured (in-sample) condition number κ (log–log scale) as a function of the feature variance ratio r , for each normalisation method on clean (a) and 5%-outlier-contaminated (b) data ($P = 500$). The dashed line is the theoretical prediction $\kappa = r$ for unscaled data. On clean data, unscaled κ follows $\kappa = r$ while every scaling method keeps the in-sample κ near 1; under contamination the unscaled κ still reaches 10^6 while the scaled methods' in-sample κ remains near 1—exactly the pattern that makes in-sample κ misleading (see Table 1 for the κ_{eff} values, which rank the methods differently).**

5.2 Mathematical Derivations

5.2.1 Proof of Proposition 1

(i) Under (A3), $E(W) = \frac{1}{2P} \|XW - d\|_2^2 = \frac{1}{2P} (W^\top X^\top XW - 2d^\top XW + d^\top d)$. Differentiating twice gives $\nabla^2 E(W) = \frac{1}{P} X^\top X = \hat{\Sigma}$, independent of W . Under (A1) the columns are centred, so $\frac{1}{P} X^\top X$ is the sample covariance rather than merely the second-moment matrix. Under (A2) $\hat{\Sigma} > 0$, so E is strictly convex and the minimiser is unique.

(ii) By (A2), $\hat{\Sigma}$ is symmetric positive definite, so it admits $\hat{\Sigma} = V\Lambda V^\top$ with V orthogonal and $\Lambda > 0$ diagonal; hence $\Lambda^{-1/2}$ is well defined. For $\tilde{X} = XV\Lambda^{-1/2}$,

$$\tilde{H} = \frac{1}{P} \tilde{X}^\top \tilde{X} = \Lambda^{-1/2} V^\top \left(\frac{1}{P} X^\top X \right) V \Lambda^{-1/2} = \Lambda^{-1/2} V^\top V \Lambda V^\top V \Lambda^{-1/2} = I_n,$$

using $V^\top V = I$. All eigenvalues equal 1, so $\kappa(\tilde{H}) = 1/1 = 1$.

(iii) With $\tilde{H} = I$ the error recurrence (Eq. (5)) becomes $e_{t+1} = (I - \eta I)e_t = (1 - \eta)e_t$, which is zero for $\eta = 1$ regardless of e_0 . $\square$

Remark. Step (ii) is purely algebraic and holds for *any* positive definite $\hat{\Sigma}$, including one estimated from contaminated or insufficient data. This is the formal content of the caveat in Section 2.1.5: attaining $\kappa = 1$ in-sample is guaranteed by construction and therefore carries no information about whether the transform generalises. The quantity with empirical content is $\kappa(\hat{\Sigma}^{-1/2} \Sigma \hat{\Sigma}^{-1/2})$, which equals 1 exactly when $\hat{\Sigma}$ is a positive scalar multiple of Σ (the identity requires $\hat{\Sigma} = \Sigma$).

5.2.2 Proof of Optimal Convergence Rate

Starting from the decoupled error recurrence (Eq. (7)), the convergence rate is

$$\rho = \max_i |1 - \eta \lambda_i|$$

To minimise ρ over η , we set the convergence rates of the fastest and slowest eigendirections equal:

$$1 - \eta \lambda_{\min} = -(1 - \eta \lambda_{\max})$$

Solving for η :

$$\eta^* = \frac{2}{\lambda_{\max} + \lambda_{\min}}$$

Substituting back:

$$\rho_{\text{opt}} = 1 - \eta^* \lambda_{\min} = 1 - \frac{2\lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{\kappa - 1}{\kappa + 1}$$

For $\kappa \gg 1$:

$$\rho_{\text{opt}} \approx 1 - \frac{2}{\kappa}$$

The number of iterations to reduce the error by a factor of δ is therefore:

$$T = \frac{\ln(1/\delta)}{\ln(1/\rho_{\text{opt}})} \approx \frac{\kappa}{2} \ln\left(\frac{1}{\delta}\right)$$

so the iteration count required for a fixed error reduction grows linearly with κ . This is a statement about the bound, not a measurement; see Section 3.3.3 on why our experiments bound rather than confirm the constant.

5.2.3 Derivation: What Robust Scaling Contributes to Whitening

Let $X \in R^{p \times n}$ be the data matrix and let X_{out} denote the same data contaminated with a fraction ϵ of outlier rows. The sample covariance is

$$\hat{\Sigma} = \frac{1}{p} X^T X$$

Under contamination, the sample covariance becomes

$$\hat{\Sigma}_{out} = (1 - \epsilon) \hat{\Sigma}_{clean} + \epsilon \hat{\Sigma}_{outlier} + \text{cross terms}$$

The outlier covariance $\hat{\Sigma}_{outlier}$ typically has eigenvalues much larger than those of $\hat{\Sigma}_{clean}$, inflating λ_{max} of the combined estimate. When whitening is applied to $\hat{\Sigma}_{out}$, the transformation $\hat{\Sigma}_{out}^{-1/2}$ over-corrects along the outlier directions and under-corrects along the true data directions, yielding an effective $\kappa > 1$ on the bulk of the data.

Robust scaling replaces each feature's mean and standard deviation with the median and IQR, which have a *breakdown point* of 25% for that feature's marginal scale: the estimate remains bounded as long as fewer than 25% of observations in that column are outliers. Formally, if the per-column outlier fraction satisfies $\epsilon < 0.25$, the IQR-based scale estimate $\hat{s}_i$ satisfies

$$|\hat{s}_i - s_i| = O(P^{-1/2})$$

regardless of the outlier magnitudes, where s_i is the population IQR. This contrasts with the standard deviation, where a single outlier of magnitude M inflates the estimate by $O(M/\sqrt{P})$, without bound in M .

Why this does not transfer to the covariance.

It is tempting to conclude that whitening $\hat{\Sigma}_{rob} = \frac{1}{P} \tilde{X}_{rob}^T \tilde{X}_{rob}$ therefore inherits a 25% breakdown point. It does not. Writing $D = \text{diag}(\hat{s})$, robust scaling is the diagonal map $\tilde{X}_{rob} = (X - 1m^T)D^{-1}$, so

$$\hat{\Sigma}_{rob} = D^{-1} \hat{\Sigma}_c D^{-1}, \quad \text{corr}(\hat{\Sigma}_{rob}) = \text{corr}(\hat{\Sigma}_c).$$

A positive diagonal congruence cannot change a correlation matrix. It does change the *eigenvectors* in general, however, so eigenvector preservation is not what makes the pipeline redundant. What does is algebraic: the whitening stage is fitted to $D^{-1} \hat{\Sigma}_c D^{-1}$, and that diagonal rescaling cancels from the composed transform, so the pipeline's effective condition number equals whitening's exactly. The

breakdown point of the covariance estimate is $1/P$ both before and after robust scaling: one row placed far enough along an arbitrary direction moves the leading eigenvector arbitrarily.

What the diagonal bound does give is control of $\lambda_{\max}$. Since $\text{tr}(\hat{\Sigma}_{\text{rob}}) = \sum_i \lambda_i$ is the sum of the (now near-unit) diagonal entries, $\lambda_{\max}$ is bounded above by that trace. The pipeline therefore removes the failure mode in which one contaminated feature inflates $\lambda_{\max}$ and collapses the admissible step size via

Eq. (8). It provides no corresponding lower bound on $\lambda_{\min}$, which is the eigenvalue that $\hat{\Sigma}^{-1/2}$ amplifies.

This asymmetry is why the pipeline's benefit should be expected to depend on how contamination is distributed across features rather than to hold in general. Genuine covariance robustness requires estimators that downweight or exclude entire rows, such as MCD (Rousseeuw and Van Driessen 1999) or the M-estimators surveyed by Maronna et al. (2019); comparing the pipeline against those is left for future work.

5.2.4 Derivation of Hessian as Covariance Matrix

For the LMS cost (Eq. (13)), expanding the gradient:

$$\nabla E(W) = \frac{1}{P} \sum_{p=1}^P (x^p{}^T W - d^p) x^p = \frac{1}{P} X^T (XW - d)$$

The Hessian is:

$$H = \frac{\partial}{\partial W} \nabla E(W) = \frac{1}{P} X^T X$$

which is the sample covariance matrix of the inputs (assuming zero-mean data). The eigenvalues of H are therefore the variances along the principal directions of the input distribution.

5.3 Additional Plots

Figure 7 collects the remaining diagnostics referenced in the main text.

– Non-Quadratic Functions, SGD vs GD, and Condition Number Heatmap

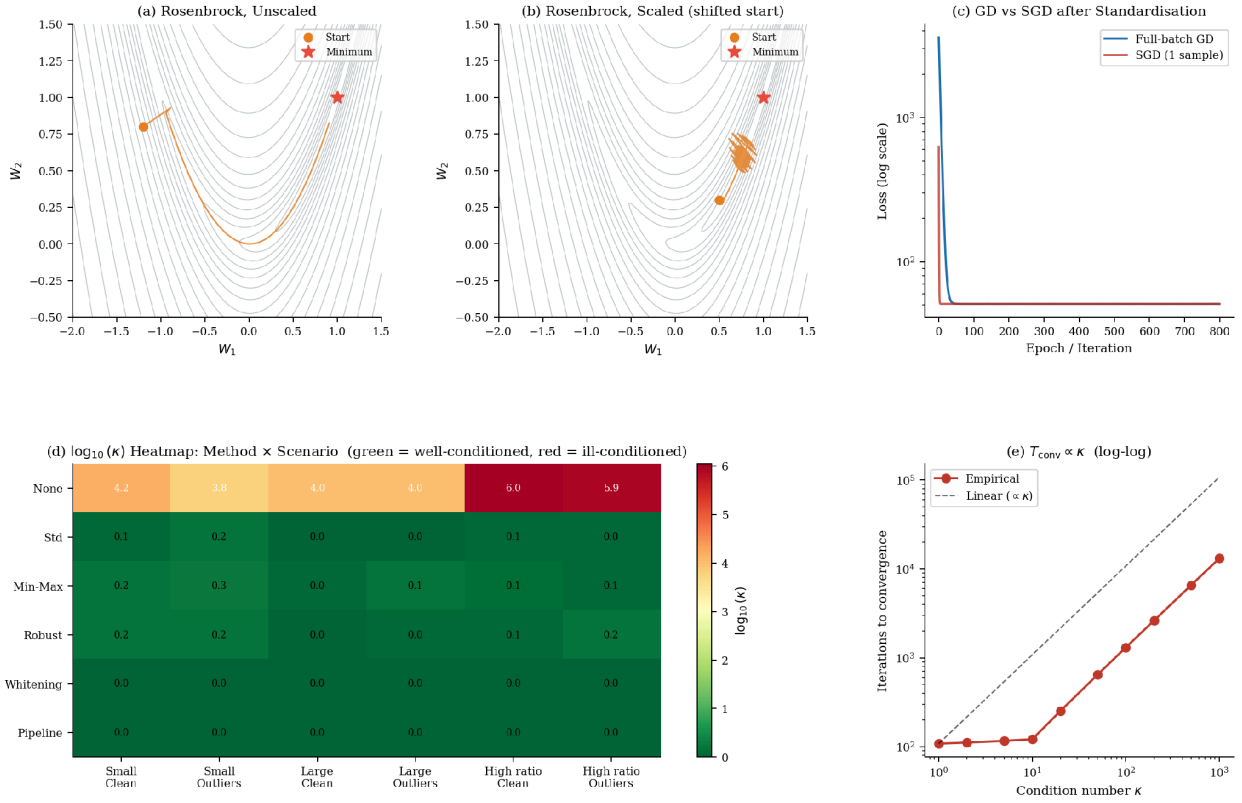


Figure 7. Additional analyses. **(a–b)** Gradient descent trajectories on the Rosenbrock function, illustrating that scaling repositions the start point into a locally better-conditioned neighbourhood rather than reducing a fixed κ —a different mechanism than for quadratic functions. **(c)** Full-batch GD vs SGD after standardisation: SGD exhibits noisy but competitive convergence, with gradient noise preventing premature stagnation. **(d)** Heatmap of $\log_{10}(\kappa)$ across all methods and six scenarios (dataset size $\times$ outlier condition $\times$ variance ratio): green cells indicate well-conditioned problems ($\kappa \approx 1$); red cells indicate ill-conditioning. Note that this panel plots in-sample κ , which is 1 by construction for whitening-based methods regardless of contamination; see Section 3.4.5 for the corresponding κ_{eff} values, which rank the methods differently. **(e)** Empirical consistency with the bound $T_{\text{conv}} \propto \kappa$ on a log–log scale (Eq. (12)); the points flatten below $\kappa \approx 10^2$, consistent with the bound’s asymptotic character.

REFERENCES

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. “Layer Normalization.” *arXiv Preprint arXiv:1607.06450*.

Sep 2026

Vol 11. No 6.

- Bottou, Léon, Frank E. Curtis, and Jorge Nocedal. 2018. "Optimization Methods for Large-Scale Machine Learning." *SIAM Review* 60 (2): 223–311.
- Boyd, Stephen, and Lieven Vandenbergh. 2004. *Convex Optimization*. Cambridge University Press.
- Golub, Gene H., and Charles F. Van Loan. 2013. *Matrix Computations*. 4th ed. Johns Hopkins University Press.
- 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 (ICML)*, 448–56.
- LeCun, Yann, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. 1998. "Efficient BackProp." In *Neural Networks: Tricks of the Trade*, edited by Genevieve B. Orr and Klaus-Robert Müller, vol. 1524. Lecture Notes in Computer Science. Springer.
- Maronna, Ricardo A., R. Douglas Martin, Victor J. Yohai, and Matías Salibián-Barrera. 2019. *Robust Statistics: Theory and Methods (with r)*. 2nd ed. Wiley.
- Martens, James, and Roger Grosse. 2015. "Optimizing Neural Networks with Kronecker-Factored Approximate Curvature." *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2408–17.
- Nocedal, Jorge, and Stephen J. Wright. 2006. *Numerical Optimization*. 2nd ed. Springer.
- Rousseeuw, Peter J. 1984. "Least Median of Squares Regression." *Journal of the American Statistical Association* 79 (388): 871–80.
- Rousseeuw, Peter J., and Katrien Van Driessen. 1999. "A Fast Algorithm for the Minimum Covariance Determinant Estimator." *Technometrics* 41 (3): 212–23.
- Saad, Yousef. 2003. *Iterative Methods for Sparse Linear Systems*. 2nd ed. SIAM.
- Santurkar, Shibani, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. 2018. "How Does Batch Normalization Help Optimization?" *Advances in Neural Information Processing Systems (NeurIPS)* 31: 2483–93.
- Shanker, M., and et al. 2019. "Influence of Feature Scaling on Convergence of Gradient Iterative Algorithm." *IOP Conference Series: Materials Science and Engineering* 1213.
- Shazeer, Noam, and Mitchell Stern. 2018. "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost." *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 4596–604.
- Wright, Stephen J., and Benjamin Recht. 2022. *Optimization for Data Analysis*. Cambridge University Press.