LANCE: LAYER-AWARE PRECONDITIONING FOR GAUSS–NEWTON TRAINING OF PINNS

Anonymous authors
Paper under double-blind review

ABSTRACT

Physics-informed neural networks (PINNs) minimise the squared norm of a collocation residual, a deterministic nonlinear least-squares problem whose curvature is inherited from a differential operator and whose condition number reaches 10^{15}. First-order and limited-memory quasi-Newton methods stall on it, and a line of work has moved PINN training towards Newton and natural-gradient methods that differ mainly in how much of the curvature matrix they represent. Within that line, we ask which part of the curvature is worth representing exactly when only a few iterations of the inner linear solve can be afforded per step by measuring the curvature layer by layer. Across nine PDEs and networks of up to 2.6×10^5 parameters, the input and output layers hold 0.3 to 15% of the parameters and 14 to 42% of the curvature energy. No scalar per layer preconditioner can exploit this concentration, because scaling a block by a constant leaves its internal conditioning unchanged. LANCE forms the exact Gauss–Newton blocks of those layers and uses them to precondition a fixed-budget conjugate-gradient (CG) solve while the other layers stay matrix-free. Once the damping has fallen, ten CG iterations capture 0.03 to 0.06 of the attainable model decrease without the preconditioner and 0.38 to 0.97 with it. As a result, the final relative L^2 error is 4.5 to 25 times lower than without the preconditioner, at up to 1.4 times the time per step.

1 INTRODUCTION

Training a physics-informed neural network (Raissi et al., 2019) is not the kind of optimisation problem that modern deep-learning optimisers were designed for. There is no data distribution to sample and no minibatch noise, only one deterministic objective, the squared norm of a residual evaluated at a fixed set of collocation points, so the only obstacle between an initialisation and a solution is the conditioning of that objective. Because the residual applies a differential operator to the network, a k-th order operator stretches the spectrum of the curvature matrix by the $2k$-th power of the frequencies the solution contains, and measured condition numbers on standard benchmarks exceed 10^{15} (Rathore et al., 2024; De Ryck et al., 2024; Cao & Zhang, 2025). Adam (Kingma & Ba, 2015) and L-BFGS (Liu & Nocedal, 1989) remain the default pair (Markidis, 2021; Kiyani et al., 2025). Adam's diagonal preconditioner cannot rotate coordinates, whereas the ill-conditioning here is not axis-aligned, and the $m \approx 20$ secant pairs of L-BFGS cannot cover fifteen orders of magnitude.

This places PINN training much closer to classical nonlinear least squares than to empirical risk minimisation, and a steady line of work has replaced the default pair by Newton and natural-gradient methods (Müller & Zeinhofer, 2023; Dangel et al., 2024; Rathore et al., 2024; Chen et al., 2024; Jnini et al., 2026; Jnini & Vella, 2025b). The members of that line share the step, almost always a damped Gauss–Newton or natural-gradient step, and differ in *how much of the curvature matrix they represent*. Energy natural gradient descent assembles the matrix densely (Müller & Zeinhofer, 2023). KFAC-PINN keeps the diagonal blocks and approximates each by a Kronecker product (Dangel et al., 2024). Nyström-preconditioned Newton–CG keeps a low-rank sketch (Rathore et al., 2024; Frangella et al., 2023). Fully matrix-free methods keep nothing and touch the matrix only through products (Jnini et al., 2026; Jnini & Vella, 2025b). Figure 1 draws these choices side by side.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

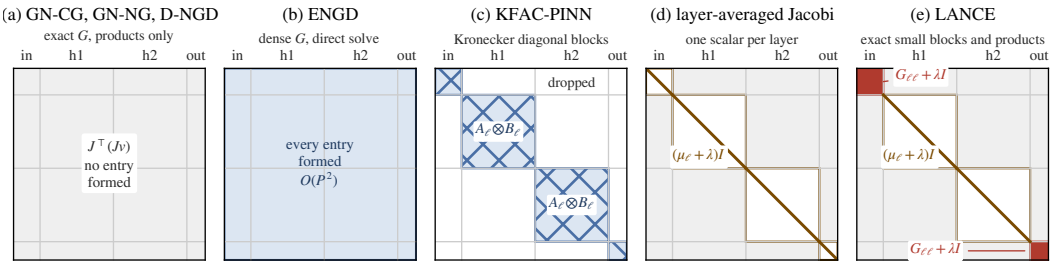


Figure 1: What each method represents of $G + \lambda I$, by layer, for a four-layer network. Hatched entries enter only through products $J^\top(Jv)$. LANCE (e) keeps the exact system and Krylov solve of (a) and adds the exact blocks of the small curvature-dense layers.

The choice is then a trade-off between memory and fidelity, which the methods above span from dense to matrix-free, each representing every layer in the same way. Exact Gauss–Newton products for a PINN residual cost 1.0 to 1.3 gradients, so what limits a step is how many Krylov iterations one is willing to spend before moving. Under a fixed and small budget the question becomes which part of the curvature is worth representing exactly by measurement.

Comparing each layer's share of the trace of the Gauss–Newton matrix with its share of the parameters yields a curvature density ρ_ℓ. Across nine PDEs and tanh multilayer perceptrons from 2.2×10^3 to 2.6×10^5 parameters trained under a hard-constraint ansatz, the first and last layers hold 0.3 to 15% of the parameters and 14 to 42% of the curvature energy, with ρ_ℓ between 1.3 and 110 against at most 1.2 on average for a hidden layer (Appendices G and L), and those layers are small enough for their exact Gauss–Newton blocks to be formed and factorised at negligible cost. A scalar per layer would be cheaper still, but rescaling a block by a constant leaves the condition number inside it unchanged, and that internal conditioning is what a fixed-budget conjugate-gradient solve loses.

LANCE, for Layer-Aware Natural Curvature Exploitation, runs a fixed budget of preconditioned conjugate-gradient (CG) iterations on the scalar-damped Gauss–Newton system with exact matrix-free products. The preconditioner holds the exact blocks of the layers whose measured curvature density exceeds their parameter density and whose size is below a threshold, and one scalar for every other layer. Neither J nor G is ever assembled.

Contributions.

1. We decompose the distance between the step taken and the Newton step into a *truncation deficit*, the fraction of the attainable quadratic-model decrease that K Krylov iterations leave behind, and the *Gauss–Newton model error*, which every method in this family shares (Figure 2b). The deficit concentrates in two small layers and no preconditioner of the solve can touch the model error, so the remedy is a layer-selective preconditioner. Layer-wise damping, the other natural response, removes progress instead (Appendix N).
2. The curvature density $\rho_{t,\ell}$ selects the exactly represented blocks at every iteration from a per-point gradient pass costing 2.0 to 5.3 gradients. Any scalar-per-layer preconditioner leaves within-block conditioning unchanged, whereas the exact block clusters the spectrum, and we verify both numerically at full size (§4.1).
3. At $K = 10$ the loop captures 0.03 to 0.06 of the attainable model decrease without the preconditioner and 0.38 to 0.97 with it, against 0.97 to 1.00 for the exact block diagonal over all layers. On the seven of nine PDEs that at least one method solves LANCE lowers the final error by 4.5 to 25 times over the identical unpreconditioned loop, on 33 of 35 seeds, for 1.06 to 1.38 times the time per step. Where the dense Gauss–Newton solve is affordable LANCE closes 20 to 75% of the logarithmic distance to it, and the same blocks used as a step instead of a preconditioner are worse than no preconditioning on seven of eight rows (§5.4). The advantage persists to $P = 2.6 \times 10^5$, and we report the budgets and problems on which it is absent.

Table 1: Curvature methods for PINNs, by what they represent of G. P parameters, N residual terms, K Krylov iterations, r sketch rank, m secant pairs, S the selected layers, $\mathcal{E}$ the eligible ones ($P_\ell \le P_{\max}$), PCG preconditioned CG. Step cost in gradient-equivalents at the rates of §5.1.

Method	Represents of G	Solve	Memory	Step
Adam	diagonal second moment	elementwise	$\mathcal{O}(P)$	1
L-BFGS	rank-$2m$ secant model	two-loop	$\mathcal{O}(mP)$	1–3
ENGD	dense, assembled	direct	$\mathcal{O}(P^2)$	$\mathcal{O}(P^2N+P^3)$
KFAC-PINN	Kronecker diagonal blocks	block	$\mathcal{O}(\sum_\ell P_\ell)$	few
NNCG	Hessian, Nyström sketch	CG	$\mathcal{O}(rP)$	$\approx 2K$
GN-NG	nothing, exact products	CG	$\mathcal{O}(P+N)$	$\approx 1.2K$
D-NGD	dual $JJ^\top$, dense or sketched	direct or PCG	$\mathcal{O}(P+N^2)$ or $\mathcal{O}(P+rN)$	$\mathcal{O}(N^2P+N^3)$ or $\approx 1.2K$
LANCE	exact blocks on S, scalars elsewhere	PCG	$\mathcal{O}(P+N+\sum_{\mathcal{E}} P_\ell^2)$	$\approx 1.2K+4$

2 Background

2.1 PINN training as a nonlinear least-squares problem

Let u_θ be a tanh multilayer perceptron with parameters $\theta \in \mathbb{R}^P$ and let $\mathcal{D}$ be the differential operator of the PDE. Given N collocation points $\{x_i\}_{i=1}^N \subset \Omega$ and forcing f, we assemble one stacked residual

$$r(\theta) = \tfrac{1}{\sqrt{N}}\big[\mathcal{D}[u_\theta](x_1) - f(x_1),\ \ldots,\ \mathcal{D}[u_\theta](x_N) - f(x_N)\big]^\top \in \mathbb{R}^N, \tag{1}$$

so that $\mathcal{L}(\theta) = \frac{1}{2}\|r(\theta)\|_2^2$ is half the mean squared residual and $\nabla_\theta \mathcal{L} = J^\top r$ with $J = \nabla_\theta r \in \mathbb{R}^{N\times P}$. Boundary and initial conditions are imposed exactly by a hard-constraint ansatz (Lagaris et al., 1998; Sukumar & Srivastava, 2022), which removes the penalty weights whose balancing is a documented source of training pathologies (Wang et al., 2021; 2022) and leaves a single homogeneously scaled residual. Training is full batch. Differentiating twice gives $\nabla^2\mathcal{L} = J^\top J + \sum_i r_i \nabla_\theta^2 r_i$, and dropping the second term leaves the Gauss–Newton matrix $G = J^\top J$, which is positive semidefinite by construction, agrees with the Hessian at an interpolating solution, and admits cheap products (Schraudolph, 2002; Nocedal & Wright, 2006). Levenberg damping makes $G + \lambda I$ positive definite (Levenberg, 1944; Marquardt, 1963). The problem is deterministic, so the gradient noise that stochastic methods are built around (Robbins & Monro, 1951; Bottou et al., 2018) is absent, and the condition number inside the damped diagonal block of a single layer reaches 10^9 within forty iterations (Appendix H).

2.2 The curvature family tree

Natural gradient descent (Amari, 1998) preconditions by a metric on the model output rather than the parameters, and for the L^2 energy metric of energy natural gradient descent that metric coincides with the Gauss–Newton matrix of Eq. (1) (Müller & Zeinhofer, 2023), so on PINNs the Gauss–Newton and natural-gradient branches meet. What remains to choose is the representation, and Table 1 places the methods on it.

2.3 Matrix-free Gauss–Newton products

The only curvature primitive the inner solver needs is Gv. Parameter differentiation commutes with the linear part of $\mathcal{D}$, so each row of J applies the input-derivative graph defining the residual to a parameter tangent, and composing a Jacobian–vector product with a vector–Jacobian product gives

$$Gv = J^\top(Jv) = \mathrm{VJP}_\theta\big[\mathrm{JVP}_\theta(r, v)\big], \tag{2}$$

the classical exact Gauss–Newton product (Pearlmutter, 1994; Schraudolph, 2002; Martens, 2010), used for PINNs by Jnini et al. (2026) and Jnini & Vella (2025b). Its cost relative to a gradient depends on mode, backend and architecture (Griewank & Walther, 2008; Dagréou et al., 2024), so we measure it in §5.1 and obtain 1.0 to 1.3× (Appendix C.5).

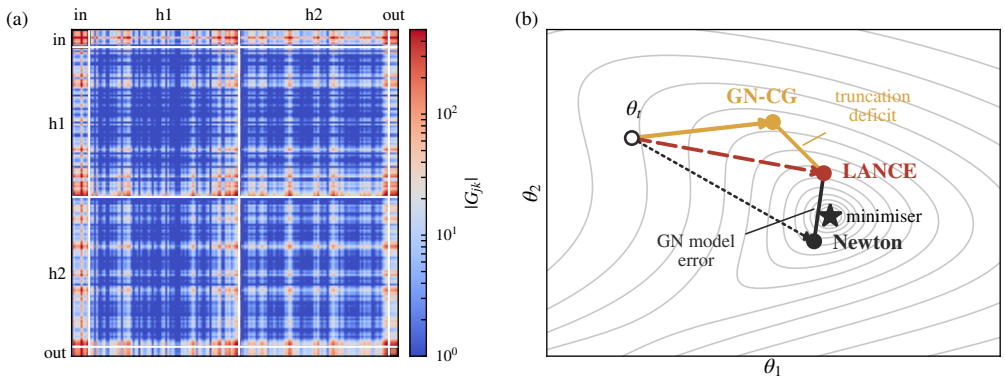


Figure 2: (a) $|G_{jk}|$ of a hard-constrained Helmholtz PINN with a 1–16–16–16–1 network at initialisation, by layer. (b) The two gaps on a two-parameter least-squares toy (Appendix E). GN-CG is the first CG iterate, LANCE the damped Gauss–Newton direction $d^\star$, Newton the full-Hessian step.

3 The location of the curvature

Each step solves the scalar-damped system

$$(G + \lambda I)\, d = \nabla_\theta \mathcal{L}(\theta) \tag{3}$$

approximately, by K iterations of conjugate gradient started from zero. Two distinct errors separate the resulting step from a Newton step. Writing $q(d) = \nabla\mathcal{L}^\top d - \frac{1}{2} d^\top (G + \lambda I) d$ for the quadratic model and $d^\star$ for its maximiser, the *truncation deficit* is $1 - q(d_K)/q(d^\star)$, the part of the attainable model decrease that the budget forgoes. The *Gauss–Newton model error* is the discrepancy between $d^\star$ and the Newton direction, caused by dropping $\sum_i r_i \nabla^2 r_i$. Figure 2b shows both. Only the first can be reduced without changing the curvature model, and §5.3 shows that it is large under a realistic budget.

Layer-resolved curvature. Partition the parameter indices into the L layers, with P_ℓ parameters in layer ℓ. The layer energy share, block trace and curvature density are

$$E_\ell = \frac{\tau_\ell}{\sum_{\ell'} \tau_{\ell'}}, \qquad \tau_\ell = \sum_{j \in \ell} G_{jj} = \sum_{i=1}^{N} \left\| \nabla_{\theta_\ell} r_i \right\|_2^2, \qquad \rho_\ell = \frac{E_\ell}{P_\ell / P}, \tag{4}$$

so ρ_ℓ compares a layer's share of $\mathrm{tr}(G)$, the sum of the eigenvalues, with its share of the parameters. A per-point gradient pass evaluates $\nabla_\theta r_i$ over the collocation points in chunks by a vectorised reverse-mode sweep and accumulates squared entries, returning $\mathrm{diag}(G)$ and, for every layer with at most $P_{\max}$ parameters, the exact block $G_{\ell\ell} = \sum_i \nabla_{\theta_\ell} r_i \nabla_{\theta_\ell} r_i^\top$, without forming J, at $\mathcal{O}(cP)$ transient memory for chunk size c and $\mathcal{O}(NP_\ell^2)$ extra cost.

Figure 2a shows the result on a small network, whose input and output layers hold 8.3% of the parameters and 48% of the curvature energy. At the width of our experiments the two layers hold 2.3% of the parameters and 40% of the energy on Helmholtz, with ρ_ℓ of 15 and 23 against 0.6 to 0.7 for the hidden layers, and Appendices G and L reproduce the ordering on nine PDEs at widths 32 to 256 and depths up to 8. The concentration is caused by the network and its initialisation. For a residual that depends on the parameters through one forward evaluation, the trace of a layer's block is the sum over collocation points of the squared backpropagated sensitivity times one plus the squared norm of the layer input. Under Glorot initialisation both factors are of the same order in every layer of a tanh network with one or two inputs, so every layer holds a similar trace while the end layers hold a factor of the width fewer parameters. Replacing the PDE residual by the plain regression residual leaves the shares unchanged to within 0.02 on all nine problems (Appendix C.4).

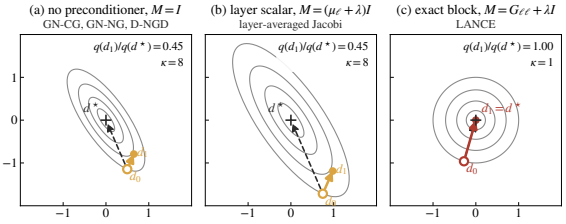


Figure 3: The damped system as CG sees it, for two parameters of one curvature-dense layer, in the coordinates $z = M^{1/2}d$. A layer scalar (b) magnifies (a) uniformly. The exact block (c) makes the first iterate equal to $d^\star$.

4 LANCE

4.1 The layer-aware block preconditioner

LANCE preconditions Eq. (3) with

$$M_t = \mathrm{blkdiag}\Big(\big[G_{t,\ell\ell} + \lambda_t I\big]_{\ell \in S_t},\ \big[(\mu_{t,\ell} + \lambda_t) I\big]_{\ell \notin S_t}\Big), \qquad S_t = \{\ell : \rho_{t,\ell} > 1,\ P_\ell \le P_{\max}\}, \quad \mu_{t,\ell} = \tau_{t,\ell}/P_\ell, \tag{5}$$

the block-diagonal part of the damped system matrix on the small high-density layers and its block-averaged diagonal elsewhere. Applying M_t^{-1} costs two triangular solves per exact block, $\mathcal{O}(\sum_{\ell \in S_t} P_\ell^2)$, and an elementwise scaling of the other layers. The Cholesky factorisation, once per step, is $\mathcal{O}(\sum_{\ell \in S_t} P_\ell^3)$, negligible at $P_{\max} = 512$.

Two elementary facts, proved in Appendix C and verified numerically in Appendix H, explain why the exact block achieves what a layer scalar cannot. Write $A = G + \lambda I$ and $B = M^{-1/2} A M^{-1/2}$ for the matrix that preconditioned CG effectively iterates on. First, a scalar per layer cannot change conditioning inside a layer. If $M = \mathrm{blkdiag}(c_\ell I)$ then $B_{\ell\ell} = A_{\ell\ell}/c_\ell$, whose condition number equals that of $A_{\ell\ell}$ for every choice of the constants. Second, an exact block clusters its part of the spectrum at one. If $M_{\ell\ell} = A_{\ell\ell}$ for $\ell \in S$ then $B_{\ell\ell} = I$, and the eigenvalues of B restricted to S lie in $[1 - \|E\|_2, 1 + \|E\|_2]$ with E the off-block coupling. Figure 3 shows both facts in two dimensions.

The global condition number of B is *larger* under LANCE than under no preconditioner, by two to three orders of magnitude, and the captured fraction still improves on every problem. The two are compatible because unpreconditioned CG never leaves $\mathrm{range}(J^\top)$, so the $P - N$ eigenvalues of A equal to λ that dominate κ never influence it (Appendix C). We therefore report the captured fraction and treat κ as a diagnostic (Appendix H).

4.2 The training loop

Each step solves Eq. (3) by K iterations of preconditioned CG with M_t, started from zero, each iteration using one product (2) and one application of M_t^{-1}. The k-th iterate maximises q over the preconditioned Krylov subspace and $q(0) = 0$, so every iterate is a descent direction (Nocedal & Wright, 2006) and a small fixed K is safe to combine with a line search. We set $\lambda_0 = 10^{-3}$, use $\lambda \leftarrow 0.7\lambda$ after a full Armijo step and $\lambda \leftarrow 1.5\lambda$ otherwise, clipped to $[10^{-8}, 10^{1}]$, with $c_1 = 10^{-4}$ and at most 30 halvings, $K = 10$ except $K = 15$ on low-viscosity Burgers, and refresh M_t every iteration. Algorithm 1 in Appendix D gives the complete update. The ablation without preconditioning, GN-CG, sets $M_t = I$ and is otherwise identical. One step costs one per-point pass, one gradient, K products with K preconditioner applications and a few loss-only line-search evaluations, about $(\pi + 1 + cK + \ell)$ gradient-equivalents with c the cost of a product, 1.0 to 1.3 at the measured rates, and π that of the pass, 2.0 to 5.3, so roughly 15 to 22 at $K = 10$ against 13 to 16 for GN-CG. Working memory is $\mathcal{O}(P + N)$ plus $\sum P_\ell^2$ over the layers with $P_\ell \le P_{\max}$.

5 Experiments

Setup. Every number comes from the released notebooks, run on one NVIDIA A100 with JAX 0.11.1 (Bradbury et al., 2018) in float64 and Optax for the baselines. Unless a scaling study varies it, the model is a tanh multilayer perceptron with three hidden layers of width 64,

Table 2: Final relative L^2 error, geometric mean over five seeds, $P = 8{,}513$ to $9{,}089$, and the geometric mean of the per-seed GN-CG to LANCE ratio. L-BFGS runs for $1{,}000$ iterations. Bold marks the smallest error. Per-seed minima and maxima are in Table 11, times in Table 10.

Problem	Adam	L-BFGS	GN-CG	LANCE	ratio
Burgers (ν=$0.01/\pi$)	2.39e-03	8.31e-04	3.26e-04	**1.85e-05**	17.6×
Burgers (ν=$0.002/\pi$)	1.86e-03	6.41e-04	1.15e-04	**1.30e-05**	8.8×
Helmholtz-1D (k=20)	3.48e-01	1.54e-01	2.54e-02	**3.50e-03**	7.2×
Poisson-2D	4.78e-04	7.74e-06	3.57e-05	**2.18e-06**	16.4×
Allen–Cahn	1.22e-03	7.97e-04	6.89e-04	**2.79e-05**	24.7×
Wave-1D	6.42e-04	**6.39e-04**	6.74e-03	1.49e-03	4.5×
Helmholtz-2D (k=10)	1.36e+00	9.81e-01	7.99e-01	**6.95e-01**	1.2×
Poisson-5D	1.22e-01	7.83e-02	7.49e-02	**9.40e-03**	8.0×
Poisson-10D	9.99e-01	**1.14e-01**	1.15e-01	1.15e-01	1.0×

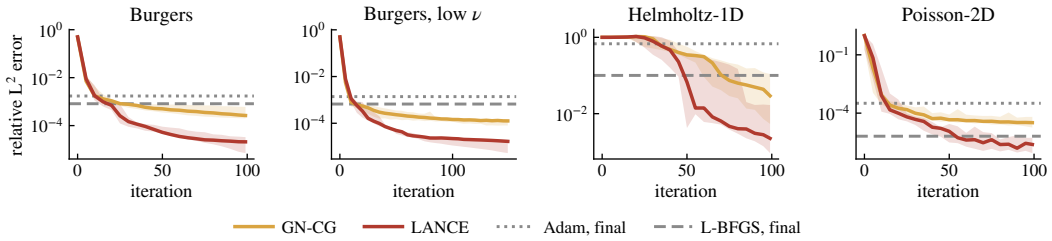


Figure 4: Relative L^2 error against iteration, median and min–max band over five seeds. Rules mark the median final Adam and L-BFGS errors.

$P = 8{,}513$ for the one-input Helmholtz problem and $P = 8{,}577$ to $9{,}089$ otherwise (Appendix E). Nine hard-constrained problems with manufactured solutions are used, viscous Burgers at $\nu \in \{0.01/\pi, 0.002/\pi\}$, Helmholtz-1D with $k = 20$, Poisson-2D as the elliptic control (Shin et al., 2020), Allen–Cahn, the wave equation, Helmholtz-2D with $k = 10$, and Poisson in five and ten dimensions, with $N = 1{,}024$ residual terms on Helmholtz-1D and $4{,}096$ elsewhere. Each of five seeds regenerates the collocation set and the initialisation, and all optimisers share θ_0 within a seed. The baselines are Adam at 10^{-3} for $10{,}000$ to $15{,}000$ iterations, which is 2.5 to 8 times the LANCE wall-clock, and full-batch L-BFGS for $1{,}000$ iterations, 0.7 to 1.7 times the LANCE wall-clock on eight problems and 8 times on Poisson-10D (Table 10). GN-CG and LANCE run 100 iterations at $K = 10$, or 150 at $K = 15$ on low-viscosity Burgers.

5.1 Exactness and cost of the primitives

Against dense references the matrix-free product (2) agrees to a relative error of at most 2.1×10^{-15} at full size and the diagonal and blocks of the per-point pass to 10^{-15} (Appendix E). One product costs 1.0 to 1.3 gradients and the per-point pass 2.0 to 5.3 (Table 9), so the cost model of §4.2 puts a LANCE step at 1.2 to 1.3 times a GN-CG step at $K = 10$, and the measured wall-clock ratios over the nine problems are 1.06 to 1.38 (Table 10).

5.2 Benchmark results

Table 2 reports all nine problems and Figure 4 the traces of the first four. On the seven problems on which some method reaches a geometric-mean error below 10^{-2}, LANCE lowers the final error of the identical unpreconditioned loop by factors of 18, 8.8, 7.2, 16, 25, 4.5 and 8.0, on Burgers, low-ν Burgers, Helmholtz-1D, Poisson-2D, Allen–Cahn, Wave-1D and Poisson-5D, and improves 33 of the 35 seeds (Appendix F). Against Adam, which runs for 2.5 to 8 times the LANCE wall-clock, the factors are 13 to 220 on six of the seven. Against L-BFGS at 0.7 to 1.7 times the LANCE wall-clock they are 29 to 49 on Burgers, low-ν Burgers, Helmholtz-1D and Allen–Cahn, 8.3 on Poisson-5D and 3.5 on Poisson-2D. Wave-1D is the exception. There Adam and L-BFGS are ahead of LANCE by 2.3 times and GN-CG trails them by ten. On Helmholtz-2D every method ends near a relative error of one, and on Poisson-10D L-BFGS, GN-CG and LANCE stop within one per cent of the

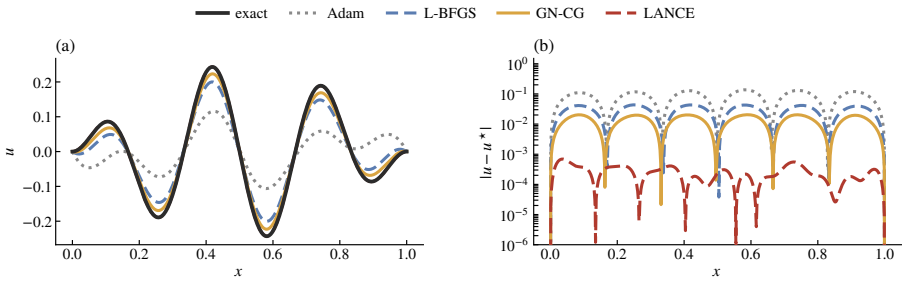


Figure 5: Helmholtz-1D, seed 4. (a) The exact solution and the four trained networks. (b) Their pointwise error. Final relative errors 0.68, 0.23, 0.11 and 2.3×10^{-3} for Adam, L-BFGS, GN-CG and LANCE. The other problems are in Figures 8 and 9.

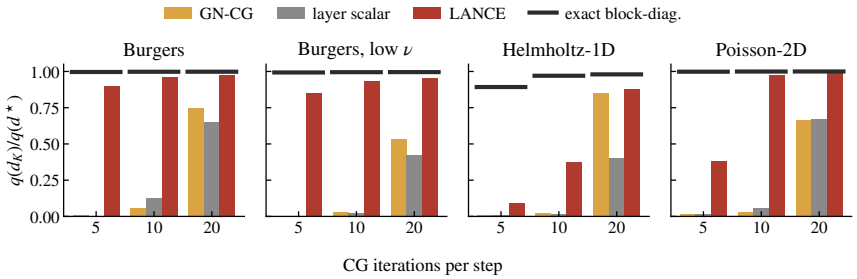


Figure 6: Captured fraction $q(d_K)/q(d^\star)$ after forty GN-CG iterations, at the damping then in force. Black rules mark the exact block diagonal over all layers.

same error. On Poisson-5D every method ends between 4×10^{-2} and 1.4×10^{-1} on every seed except LANCE, which does so on two seeds and reaches 1.2×10^{-3} to 3.0×10^{-3} on the other three (Appendix F). A preconditioner changes the direction of the inner solve and cannot move the loop off a stationary point it has settled into, and the stall on Helmholtz-2D and Poisson-10D matches the documented difficulty of PINN training at high frequency and in high dimension (Wang et al., 2021; Rathore et al., 2024).

On Helmholtz-1D, L-BFGS leaves the initial plateau after 440 to 920 iterations on four of the five seeds, Adam ends between 0.06 and 1.1 depending on the seed, and every Gauss–Newton run spends thirty to sixty iterations on the plateau while the full step is rejected and λ rises towards its cap. LANCE leaves the plateau at about the same iteration as GN-CG and then descends faster. Figure 5 shows the trained solutions on one seed. GN-CG recovers the oscillation with its peaks 8 to 22 per cent low and L-BFGS 18 to 45 per cent low, LANCE lies on the exact solution, and Adam does not resolve it.

5.3 The truncation deficit at the operating budget

At initialisation unpreconditioned CG-10 already captures all of the attainable decrease on all four problems, so there is nothing for a preconditioner to improve at the first step. After forty GN-CG iterations the damping sits within a factor of 1.5 of its floor of 10^{-8} on three problems and at 3×10^{-2} on Helmholtz-1D, which is still leaving its plateau. At that damping unpreconditioned CG-10 captures 0.06, 0.03, 0.03 and 0.03 of the attainable decrease on Burgers, low-ν Burgers, Helmholtz-1D and Poisson-2D, the layer scalar 0.13, 0.02, 0.02 and 0.06, LANCE 0.96, 0.93, 0.38 and 0.97, and the exact block diagonal over all layers 0.97 to 1.00 (Figure 6). The two blocks close 94 to 97% of that gap on three problems and 37% on Helmholtz-1D, where the curvature has begun to migrate to the hidden layers.

The full budget sweep (Appendix I) records where the advantage ends. LANCE reaches 0.93 to 0.97 by $K = 10$ on the three small-damping problems, whereas the unpreconditioned solve needs $K = 40$ to reach 0.76 to 0.90 and overtakes LANCE in captured fraction only on Helmholtz-1D at

Table 3: Final relative L^2 error by curvature representation, geometric mean over three seeds, 100 iterations. CG to tol. stops at a relative residual of 10^{-2} or 50 iterations, dual CG spends $K = 10$ in residual space, the block-diagonal step takes the solved blocks as the step, the dense solve factorises $G + \lambda I$. Best in bold.

Problem	GN-CG	CG to tol.	dual CG	layer scalar	LANCE	block-diag. step	dense solve
width 32, $P = 2{,}209$ to $2{,}241$, three seeds							
Helmholtz-1D	8.10e-03	4.71e-04	3.40e-01	6.27e-02	1.14e-03	1.89e-02	**1.66e-04**
Burgers	2.63e-04	5.69e-05	3.94e-04	4.56e-04	2.09e-05	1.51e-02	**7.78e-06**
Burgers, low ν	2.88e-04	5.38e-05	3.93e-04	5.12e-04	2.62e-05	1.52e-02	**1.17e-05**
Poisson-2D	5.43e-05	1.34e-06	6.90e-05	7.32e-05	1.31e-05	1.10e-04	**1.27e-06**
width 64, $P = 8{,}513$ to $8{,}577$, three seeds							
Helmholtz-1D	1.84e-02	4.11e-03	2.53e-01	1.99e-01	6.32e-03	8.98e-03	**9.84e-05**
Burgers	2.48e-04	3.67e-05	2.40e-04	2.52e-04	2.83e-05	1.31e-02	**8.50e-06**
Burgers, low ν	2.37e-04	4.06e-05	2.14e-04	2.54e-04	2.72e-05	1.28e-02	**1.17e-05**
Poisson-2D	3.65e-05	1.45e-06	4.00e-05	4.67e-05	2.21e-06	7.37e-05	**3.42e-07**

Table 4: Final relative L^2 error against the CG budget K at 100 iterations, geometric mean over ten seeds, width 64. Bold marks the smaller error in each column.

Problem		K=2	K=5	K=10	K=20	K=40
Helmholtz-1D	GN-CG	1.0e+00	8.7e-01	2.6e-02	1.5e-03	**2.3e-04**
	LANCE	**8.4e-01**	**1.2e-02**	**4.3e-03**	**4.9e-04**	2.7e-04
Burgers	GN-CG	4.2e-02	4.6e-03	3.7e-04	7.3e-05	**1.7e-05**
	LANCE	**1.1e-02**	**3.3e-05**	**1.9e-05**	**1.9e-05**	1.9e-05
Poisson-2D	GN-CG	7.5e-04	1.9e-04	3.1e-05	5.7e-06	**1.1e-06**
	LANCE	**3.6e-04**	**2.6e-06**	**2.2e-06**	**2.1e-06**	1.6e-06

that budget. Unpreconditioned CG never leaves $\mathrm{range}(J^\top)$, where $d^\star$ lives, whereas an exact block spreads the $P - N$ eigenvalues at λ from $\ker J$ into an interval that later iterations must resolve (Appendix C), so the blocks buy their gain at small K with a cost at large K, which is why the advantage in §5.5 is confined to $K \leq 20$.

5.4 Comparison with other curvature representations

Table 3 runs the curvature rows of Table 1 through the same loop, so that the only difference between columns is what is represented of G and how it is used. CG to tolerance is the solve of GN-NG, dual CG a fixed budget spent in the residual space of D-NGD, the block-diagonal step the exact-factor limit of KFAC-PINN and the dense solve the representation of ENGD (Appendix J). The dense solve is the most accurate representation, and LANCE closes 20 to 75% of the logarithmic distance from the unpreconditioned loop to it (Appendix J), while keeping $\mathcal{O}(P + N)$ memory against $\mathcal{O}(P^2)$ and a sixtieth of the time per run at $P = 8.5 \times 10^3$. The layer scalar is no better than no preconditioner on any row, within a factor of 1.8 of it on six of the eight and 8 and 11 times worse on Helmholtz-1D, since a scalar reweights the layers but cannot improve the conditioning inside any block (§4.1). The block-diagonal step is worse than the unpreconditioned loop on seven of the eight rows, by 2.0 to 57 times, because the off-block coupling it discards is not small (Appendix H), the exception being Helmholtz-1D at width 64, where it is 2 times better than the loop and 1.4 times worse than LANCE. Whether the stored blocks replace the system or precondition it therefore matters as much as how much of G is stored. Solving to a relative residual of 10^{-2} spends 13 to 35 products per step against a fixed 10, and with them it is ahead of LANCE by 1.5 to 10 times on Helmholtz-1D and Poisson-2D and behind by 1.3 to 2.7 on the two Burgers problems, whereas at every matched budget up to $K = 20$ LANCE is ahead of the unpreconditioned solve (Table 4). The dual solve at the same budget minimises a different norm and trails the primal one on six of the eight rows (Appendix J).

5.5 Budget, ablations and scale

Table 4 sweeps the budget over ten seeds. LANCE is ahead at every budget up to $K = 20$, by 74, 140 and 73 times at $K = 5$ on Helmholtz-1D, Burgers and Poisson-2D, by 6.1, 19 and 14 at $K = 10$ and by 3.0, 3.9 and 2.8 at $K = 20$. At $K = 40$ GN-CG is ahead by 16, 11 and 41%, for the reason

given in §5.3. A preconditioner can only recover what the budget loses, and the useful operating range is $5 \leq K \leq 20$.

The block-size limit $P_{\max}$ isolates each block (Table 21). At $P_{\max} = 64$ nothing is eligible and LANCE is the layer scalar, which trails GN-CG by 9.6 times on Helmholtz-1D. Admitting the output layer at $P_{\max} = 128$ and the input layer at 256 lowers the error by 2.3 and 9.2 times on Burgers and by 1.3 and 10 on Poisson-2D, and both Helmholtz-1D layers enter at 128 for a factor of 58. At $P_{\max} = 8192$ the rule admits a hidden block late in training on most Helmholtz-1D and Poisson-2D seeds, for a further 2.6 and 1.2 at a factorisation cost that is no longer negligible. $\rho_{\mathrm{th}} = 0.5$ halves the Helmholtz-1D error by keeping its input layer selected to the end, $\rho_{\mathrm{th}} = 2$ changes at most 25%, and $\rho_{\mathrm{th}} = 4$ raises the Helmholtz-1D and Poisson-2D errors by 3.3 and 4.2 times. The initial damping matters least, except that Burgers prefers $\lambda_0 \leq 10^{-4}$ by up to 3.6 times and Poisson-2D $\lambda_0 \geq 10^{-4}$ by up to 8.5 (Appendix K).

Scaling the network to widths 128 and 256 and depths 5 and 8 takes P to 2.6×10^5. The two small layers then hold 0.3 to 1.5% of the parameters and still 14 to 42% of the curvature energy, and LANCE is ahead of GN-CG on all fourteen configurations, by 2.1 to 27 times, for 1.05 to 1.32 times the wall-clock (Appendix L).

6 Discussion and limitations

Ten CG iterations reproduce the damped Gauss–Newton direction exactly at initialisation and recover a few per cent of it once the damping has fallen, and the missing part sits in the two smallest layers, whose blocks are cheap because they are small. The same measurement explains two negative results, layer-wise damping proportional to $\rho_{t,\ell}$, which removes progress rather than truncation deficit and is 2.8 and 15 times worse than LANCE on Helmholtz-1D and Poisson-2D, and the layer scalar (Appendix N).

The evidence covers full-batch, hard-constrained training of one network family on one software stack, with our own implementations of the compared representations rather than the published codes. Hard constraints leave a single homogeneously scaled residual, so ρ_ℓ measures the network and the operator rather than penalty weights, and soft boundary penalties move no layer share by more than 0.01 (Appendix O). The blocks help where the small layers hold the curvature and where the budget is losing something, and each condition fails somewhere in the campaign. The curvature migrates into the hidden layers as the activations grow, on Helmholtz-1D from 0.39 to 0.05 of the trace in a hundred LANCE iterations against 0.41 to 0.20 on Burgers, so the faster the loop moves, the sooner the small blocks stop carrying the curvature, and the full block diagonal then shows headroom that would need cheap approximations of the 4160×4160 hidden blocks (Dangel et al., 2024). The deficit itself is not specific to the operator, since the regression residual develops it too once the damping has fallen, with LANCE ahead of GN-CG by 6 to 9 times, but the operator makes it far larger at any fixed damping (Appendix C.4). The input layer's density falls with the input dimension, by a factor of 4.7 between $d = 1$ and $d = 10$ on Poisson (Appendix L). Once the loop has settled no preconditioner of the solve helps, and where ten unpreconditioned iterations already capture the step, as on Poisson-1D and at $K = 40$, the blocks cost up to a factor of three. The deficit diagnostic says where the headroom is and does not rank the gains (Appendix M).

7 Conclusion

Once exact Gauss–Newton products are cheap, the question about curvature methods for PINNs becomes which part of the matrix deserves an exact representation under a fixed CG budget, rather than how much of it to store. By measurement the answer is the input and output layers, which are small, hold a disproportionate share of the curvature, and cannot be served by a scalar per layer. Preconditioning a fixed-budget CG solve with their exact blocks lowers final error by 4.5 to 25 times over the identical unpreconditioned loop on the seven problems that at least one method solves, at 1.06 to 1.38 times the time per step, whereas the same blocks used as a step are worse than nothing on seven of eight rows, so the blocks pay only as a preconditioner of the exact system.

Reproducibility statement

The complete training loop is Algorithm 1, the experimental protocol including every hyperparameter, seed and test grid is Appendix E, and the formal statements behind §4.1 are proved in Appendix C. Every number, table and figure in the paper was produced by the single notebook described in Appendix P, together with a companion notebook for the measurements of Appendix C.4 whose training code is copied from it, and their result files are turned into the LaTeX tables and the figures by released scripts without manual editing. The notebooks, the scripts, every result file and an offline test of the notebook's linear algebra against an independent NumPy oracle are included in the supplementary material and in an anonymised repository at `https://anonymous.4open.science/r/lance-B58F`, which will be made public upon acceptance.

Ethics statement

This work studies optimisation algorithms for solving partial differential equations with neural networks. It uses no human subjects, no personal data and no released dataset, all problems being manufactured solutions generated by the code itself. We see no specific ethical concern beyond the general one that faster PDE solvers are dual-use in the same way as any numerical method.

AI use statement

Generative AI tools assisted in writing code, namely the experiment notebook, the table and figure scripts and the offline tests, in scaling the experiments to the full-scale campaign reported here once its content and protocol had been decided by the authors, and in writing, namely restructuring and editing the manuscript. The research question, the method, the mathematical claims and the design and interpretation of the experiments are the authors', who read all AI-assisted code, executed the campaign, regenerated every numerical claim from the released scripts, and take responsibility for the final content of this work.

References

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

Owe Axelsson and Gunhild Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48(5):499–523, 1986.

Michele Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs. Software, 2018. URL `https://github.com/jax-ml/jax`. Version 0.11.1.

Wei Cai, Xiaoguang Li, and Lizuo Liu. A phase shift deep neural network for high frequency approximation and wave problems. *SIAM Journal on Scientific Computing*, 42(5):A3285–A3312, 2020. doi: 10.1137/19M1310050. URL `https://doi.org/10.1137/19M1310050`.

Wenbo Cao and Weiwei Zhang. An analysis and solution of ill-conditioning in physics-informed neural networks. *Journal of Computational Physics*, 520:113494, 2025. doi: 10.1016/j.jcp.2024.113494. URL `https://doi.org/10.1016/j.jcp.2024.113494`.

Zhuo Chen, Jacob McCarran, Esteban Vizcaino, Marin Soljačić, and Di Luo. TENG: Time-evolving natural gradient for solving PDEs with deep neural nets toward machine precision. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 7143–7162. PMLR, 2024. URL `https://proceedings.mlr.press/v235/chen24ad.html`.

Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust Region Methods*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000. doi: 10.1137/1.9780898719857. URL https://doi.org/10.1137/1.9780898719857.

Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. How to compute Hessian-vector products? ICLR Blogposts, May 2024. URL https://iclr-blogposts.github.io/2024/blog/bench-hvp/.

Felix Dangel, Johannes Müller, and Marius Zeinhofer. Kronecker-factored approximate curvature for physics-informed neural networks. In *Advances in Neural Information Processing Systems*, volume 37, pp. 34582–34636. Curran Associates, Inc., 2024. doi: 10.52202/079017-1091. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/3d27d607586984908900eaa8ce19c96c-Abstract-Conference.html.

Tim De Ryck, Florent Bonnet, Siddhartha Mishra, and Emmanuel de Bézenac. An operator preconditioning perspective on training in physics-informed machine learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=WWlxFtR5sV.

Zachary Frangella, Joel A. Tropp, and Madeleine Udell. Randomized Nyström preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 44(2):718–752, 2023. doi: 10.1137/21M1466244. URL https://doi.org/10.1137/21M1466244.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2013.

Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2 edition, 2008. doi: 10.1137/1.9780898717761. URL https://doi.org/10.1137/1.9780898717761.

Andrés Guzmán-Cordero, Felix Dangel, Gil Goldshlager, and Marius Zeinhofer. Improving energy natural gradient descent through Woodbury, momentum, and randomization. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113870–113900. Curran Associates, Inc., 2025. doi: 10.52202/085713-3798. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a5809a25c65040fd43e146126b45b150-Abstract-Conference.html.

Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, pp. 8571–8580. Curran Associates, Inc., 2018.

Anas Jnini and Flavio Vella. Hessian-free natural gradient descent for physics informed machine learning. Withdrawn submission to the Thirteenth International Conference on Learning Representations, 2025a. URL https://openreview.net/forum?id=Oqk1Ui6m0n.

Anas Jnini and Flavio Vella. Dual natural gradient descent for scalable training of physics-informed neural networks. *Transactions on Machine Learning Research*, 2025b. ISSN 2835-8856. URL https://openreview.net/forum?id=GDHVRy6SDd.

Anas Jnini, Flavio Vella, and Marius Zeinhofer. Gauss–Newton natural gradient descent for physics-informed computational fluid dynamics. *Computers & Fluids*, 307:106955, 2026. doi: 10.1016/j.compfluid.2025.106955. URL https://www.sciencedirect.com/science/article/pii/S0045793025004153.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Elham Kiyani, Khemraj Shukla, Jorge F. Urbán, Jérôme Darbon, and George Em Karniadakis. Optimizing the optimizer for physics-informed neural networks and Kolmogorov–Arnold networks. *Computer Methods in Applied Mechanics and Engineering*, 446:118308, 2025. doi: 10.1016/j.cma.2025.118308. URL https://doi.org/10.1016/j.cma.2025.118308.

Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M. Kirby, and Michael W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26548–26560. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/df438e5206f31600e6ae4af72f2725f1-Abstract.html.

Isaac Elias Lagaris, Aristidis C. Likas, and Dimitrios I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9 (5):987–1000, 1998. doi: 10.1109/72.712178. URL https://doi.org/10.1109/72.712178.

Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944. doi: 10.1090/qam/10666. URL https://doi.org/10.1090/qam/10666.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528, 1989.

Stefano Markidis. The old and the new: Can physics-informed deep-learning replace traditional linear solvers? *Frontiers in Big Data*, 4:669097, 2021. doi: 10.3389/fdata.2021.669097. URL https://doi.org/10.3389/fdata.2021.669097.

Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. doi: 10.1137/0111030. URL https://doi.org/10.1137/0111030.

James Martens. Deep learning via Hessian-Free optimization. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 735–742, Haifa, Israel, 2010. URL https://icml.cc/Conferences/2010/papers/458.pdf.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL https://jmlr.org/papers/v21/17-678.html.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417, 2015.

James Martens and Ilya Sutskever. Training deep and recurrent networks with Hessian-Free optimization. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller (eds.), *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pp. 479–535. Springer, Berlin, Heidelberg, 2 edition, 2012. doi: 10.1007/978-3-642-35289-8_27. URL https://doi.org/10.1007/978-3-642-35289-8_27.

Johannes Müller and Marius Zeinhofer. Achieving high accuracy with PINNs via energy natural gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25471–25485. PMLR, 2023. URL https://proceedings.mlr.press/v202/muller23b.html.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, 2 edition, 2006. doi: 10.1007/978-0-387-40065-5. URL https://doi.org/10.1007/978-0-387-40065-5.

Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994. doi: 10.1162/neco.1994.6.1.147. URL https://doi.org/10.1162/neco.1994.6.1.147.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5301–5310. PMLR, 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training PINNs: A loss landscape perspective. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42159–42191. PMLR, 2024. URL https://proceedings.mlr.press/v235/rathore24a.html.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003.

Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002. doi: 10.1162/08997660260028683. URL https://doi.org/10.1162/08997660260028683.

Yeonjong Shin, Jérôme Darbon, and George Em Karniadakis. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. *Communications in Computational Physics*, 28(5):2042–2074, 2020. doi: 10.4208/cicp.OA-2020-0193. URL https://doi.org/10.4208/cicp.OA-2020-0193.

Trond Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983. doi: 10.1137/0720042. URL https://doi.org/10.1137/0720042.

N. Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022. doi: 10.1016/j.cma.2021.114333. URL https://doi.org/10.1016/j.cma.2021.114333.

Charles F. Van Loan and Nikos Pitsianis. Approximation with Kronecker products. In Marc S. Moonen, Gene H. Golub, and Bart L. R. De Moor (eds.), *Linear Algebra for Large Scale and Real-Time Applications*, volume 232 of *NATO ASI Series*, pp. 293–314. Springer, Dordrecht, 1993. doi: 10.1007/978-94-015-8196-7_17.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43 (5):A3055–A3081, 2021. doi: 10.1137/20M1318043. URL https://doi.org/10.1137/20M1318043.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. doi: 10.1016/j.jcp.2021.110768. URL https://www.sciencedirect.com/science/article/pii/S002199912100663X.

Zhi-Qin John Xu, Lulu Zhang, and Wei Cai. On understanding and overcoming spectral biases of deep neural network learning methods for solving PDEs. *Journal of Computational Physics*, 530:113905, 2025. doi: 10.1016/j.jcp.2025.113905. URL https://doi.org/10.1016/j.jcp.2025.113905.

A NOTATION AND STANDING ASSUMPTIONS

Throughout, $\theta \in \mathbb{R}^P$ collects all network parameters, $r(\theta) \in \mathbb{R}^N$ is the stacked collocation residual of Eq. (1), $J = \nabla_\theta r \in \mathbb{R}^{N\times P}$, $G = J^\top J$, and $A_\lambda = G + \lambda I$ for a damping parameter $\lambda > 0$. The parameter index set is partitioned into L layers, layer ℓ holding P_ℓ parameters, so that $\sum_\ell P_\ell = P$. For a matrix $B \in \mathbb{R}^{P\times P}$ we write $B_{\ell k}$ for the sub-block indexed by layers ℓ and k. For a symmetric positive-definite B, $\kappa(B) = \lambda_{\max}(B)/\lambda_{\min}(B)$. The quadratic model of the damped system at the current iterate is

$$q(d) = \nabla_\theta \mathcal{L}^\top d - \tfrac{1}{2} d^\top A_\lambda d, \qquad d^\star = \arg\max_d q(d) = A_\lambda^{-1} \nabla_\theta \mathcal{L}, \tag{6}$$

and d_K denotes the K-th iterate of (preconditioned) conjugate gradient started from $d_0 = 0$. The *captured fraction* is $q(d_K)/q(d^\star) \in (0, 1]$, and the *truncation deficit* is its complement. A layer parameterisation $\theta_\ell = (W_\ell, b_\ell)$ of a tanh multilayer perceptron of widths n_0–n_1–$\cdots$–n_L gives $P_\ell = n_\ell n_{\ell-1} + n_\ell$.

We assume throughout that $\lambda > 0$, so that $A_\lambda \succ 0$ and conjugate gradient is well defined, that training is full batch, and that boundary and initial conditions are imposed by a hard-constraint ansatz, so that r carries a single homogeneously scaled PDE residual and no penalty weights.

B EXTENDED RELATED WORK

The main text compresses the literature into Figure 1 and Table 1. This appendix gives the longer account, organised by the question each line of work answers.

B.1 WHY PINN TRAINING IS HARD

The difficulty was identified early and has been attributed to several distinct mechanisms. Wang et al. (2022) relate slow convergence to the strongly separated eigenvalues of the neural tangent kernel (Jacot et al., 2018) induced by the differential operator. Wang et al. (2021) show that competing residual terms produce gradients whose magnitudes differ by orders of magnitude, which motivates loss balancing and, in our setting, is removed entirely by the hard-constraint ansatz. Krishnapriyan et al. (2021) show that solutions well within the approximation capacity of the network remain inaccessible to standard training on convection- and reaction-dominated problems, which locates the difficulty in the optimiser rather than the model. Rathore et al. (2024) measure the spectrum of the loss Hessian directly and report condition numbers that make first-order convergence rates vacuous, and De Ryck et al. (2024) and Cao & Zhang (2025) trace the ill-conditioning to the operator itself. Shin et al. (2020) give consistency results for the elliptic and parabolic cases that we use as the reason to treat Poisson as a control rather than a challenge.

A separate strand concerns the approximation side. Rahaman et al. (2019), Cai et al. (2020) and Xu et al. (2025) document the spectral bias of standard networks towards low frequencies, which is why an oscillatory Helmholtz target at $k = 20$ is a discriminating test and a smooth Poisson target is not.

B.2 HARD CONSTRAINTS

Imposing boundary and initial conditions by construction goes back to Lagaris et al. (1998) and was generalised to complex geometries by Sukumar & Srivastava (2022). We adopt it for a reason specific to this paper rather than for accuracy. A penalised objective has a residual vector whose blocks carry different physical units and arbitrary weights, so the layer-resolved curvature energy of Eq. (4) would measure the weighting as much as the geometry. With hard constraints, r is a single homogeneously scaled PDE residual and ρ_ℓ is a property of the operator and the architecture alone.

B.3 SECOND-ORDER METHODS FOR NEURAL NETWORKS

Hessian-free optimisation (Martens, 2010; Martens & Sutskever, 2012) established the pattern this paper follows, namely truncated conjugate gradient on a damped Gauss–Newton system with exact matrix-free products (Pearlmutter, 1994; Schraudolph, 2002), with Levenberg damping (Levenberg, 1944; Marquardt, 1963) and a trust-region or line-search globalisation (Steihaug, 1983; Conn et al.,

2000; Nocedal & Wright, 2006). Martens (2020) sets out the relationship between the Gauss–Newton matrix, the Fisher information and the natural gradient of Amari (1998). KFAC (Martens & Grosse, 2015) introduced the Kronecker factorisation of layer blocks that Dangel et al. (2024) adapt to PINNs. The cost of the curvature product relative to a gradient is not determined by complexity analysis alone (Griewank & Walther, 2008; Dagréou et al., 2024), which is why we measure it.

B.4 Curvature methods for PINNs

Energy natural gradient descent (Müller & Zeinhofer, 2023) assembles the L^2 energy metric, which for the residual of Eq. (1) coincides with the Gauss–Newton matrix, and solves the resulting system directly. It reaches accuracies unreachable by first-order methods at moderate P and is limited by its $\mathcal{O}(P^2)$ memory. TENG (Chen et al., 2024) applies natural-gradient least-squares solves inside a time-stepping scheme and reaches near machine precision on evolution problems. Guzmán-Cordero et al. (2025) exploit the primal–dual structure by a Woodbury identity in kernel space. Dangel et al. (2024) keep only the diagonal blocks and approximate each by a Kronecker product. Rathore et al. (2024) propose NNCG, a Nyström-preconditioned Newton–CG applied to the full Hessian after first-order and quasi-Newton training, using the randomised preconditioner of Frangella et al. (2023). Among the matrix-free methods, GN-NG (Jnini et al., 2026) combines the Gauss–Newton product with conjugate gradient and a line search, the withdrawn HF-NGD submission (Jnini & Vella, 2025a) adds Lanczos and Woodbury preconditioning with Levenberg–Marquardt damping, and D-NGD (Jnini & Vella, 2025b) moves the system to residual space and solves it directly when N is moderate and by Nyström-preconditioned CG otherwise.

Every one of these methods treats the parameter space *uniformly*. The curvature is represented densely, or block-diagonally, or by a low-rank sketch, or not at all, and the same choice is made for every layer. The measurement of Appendix G says that the curvature is not uniform across layers, so a uniform representation either overpays on the hidden layers or underpays on the small ones. The low-rank preconditioners of Rathore et al. (2024) and Jnini & Vella (2025b) and the block preconditioner here are complementary. A Nyström sketch captures the leading eigendirections of the whole matrix, which are shared across layers, whereas the exact block captures the internal structure of one layer, which a rank-r sketch with $r \ll P$ cannot resolve. Combining them is the obvious next experiment and we have not run it.

B.5 Preconditioning

The block-Jacobi preconditioner is classical (Saad, 2003; Benzi, 2002; Golub & Van Loan, 2013), and the new part here is the selection rule. A classical block-Jacobi preconditioner takes all diagonal blocks, which is exactly the variant that Table 21 shows to be worth a further factor of 1.0 to 2.6, at a factorisation cost that Table 17 puts at six times the LANCE run time when the 4160×4160 blocks are solved at every iteration. The contribution is the measurement that says which blocks earn their cost, and Proposition 1, which says that the cheap substitute for a block cannot work. The convergence theory we rely on is standard (Hestenes & Stiefel, 1952; Nocedal & Wright, 2006), with the caveat of Proposition 3 that the usual condition-number summary (Axelsson & Lindskog, 1986) says nothing on a matrix whose bulk spectrum sits at the damping level.

C Proofs

The two facts stated informally in §4.1 are the following.

Proposition 1 (Scalar preconditioning is invariant within a block)**.** *Let $A_\lambda = G + \lambda I$ be partitioned by layer and let $M = \mathrm{blkdiag}(c_\ell I)$ with $c_\ell > 0$. Then the ℓ-th diagonal block of $M^{-1/2} A_\lambda M^{-1/2}$ equals $(A_\lambda)_{\ell\ell}/c_\ell$, and therefore $\kappa\big((M^{-1/2} A_\lambda M^{-1/2})_{\ell\ell}\big) = \kappa\big((A_\lambda)_{\ell\ell}\big)$ for every ℓ and every choice of $\{c_\ell\}$.*

Proposition 2 (Exact blocks cluster the spectrum)**.** *Let $M = \mathrm{blkdiag}\big((A_\lambda)_{\ell\ell}\big)_{\ell \in S} \oplus \mathrm{blkdiag}(c_\ell I)_{\ell \notin S}$ and $B = M^{-1/2} A_\lambda M^{-1/2}$. Then $B_{\ell\ell} = I$ for every $\ell \in S$, and the eigenvalues of the principal submatrix of B indexed by $\bigcup_{\ell \in S} \ell$ lie in $[1 - \|E\|_2,\, 1 + \|E\|_2]$, where E is the off-diagonal part of that submatrix.*

C.1 Scalar preconditioning is invariant within a block

Proof of Proposition 1. Let $M = \mathrm{blkdiag}(c_\ell I_{P_\ell})$ with every $c_\ell > 0$. Then $M \succ 0$ and $M^{-1/2} = \mathrm{blkdiag}(c_\ell^{-1/2} I_{P_\ell})$, so for any ℓ, k,

$$\big(M^{-1/2} A_\lambda M^{-1/2}\big)_{\ell k} = c_\ell^{-1/2} (A_\lambda)_{\ell k} c_k^{-1/2}. \tag{7}$$

Taking $k = \ell$ gives $(A_\lambda)_{\ell\ell}/c_\ell$. Since the eigenvalues of αB are α times those of B for any $\alpha > 0$, and $(A_\lambda)_{\ell\ell} \succ 0$ as a principal submatrix of a positive-definite matrix,

$$\kappa\big((M^{-1/2} A_\lambda M^{-1/2})_{\ell\ell}\big) = \kappa\big((A_\lambda)_{\ell\ell}/c_\ell\big) = \kappa\big((A_\lambda)_{\ell\ell}\big), \tag{8}$$

independently of $\{c_\ell\}$. □

Remark 1. *A layer scalar still does something. Off-diagonal blocks are scaled by $c_\ell^{-1/2} c_k^{-1/2}$, so the* relative *weight of layers against one another changes, which is why a layer-averaged Jacobi preconditioner sometimes helps a little and sometimes hurts. The proposition rules out any improvement of the conditioning* inside *a block. When the truncation deficit is generated inside the curvature-dense blocks, as §5.3 and Appendix I show it is, no choice of $\{c_\ell\}$ can remove it.*

C.2 Exact blocks cluster the spectrum

Proof of Proposition 2. Let S be the selected set and $\mathcal{I} = \bigcup_{\ell \in S} \ell$ the union of their index sets. On $\mathcal{I}$ the preconditioner is $M_{\mathcal{I}\mathcal{I}} = \mathrm{blkdiag}\big((A_\lambda)_{\ell\ell}\big)_{\ell \in S}$, which is positive definite because each $(A_\lambda)_{\ell\ell}$ is. Since M is block diagonal with respect to the layer partition and $\mathcal{I}$ is a union of layers, $(M^{-1/2})_{\mathcal{I}\mathcal{I}} = (M_{\mathcal{I}\mathcal{I}})^{-1/2}$, so for $\ell \in S$,

$$B_{\ell\ell} = (A_\lambda)_{\ell\ell}^{-1/2} (A_\lambda)_{\ell\ell} (A_\lambda)_{\ell\ell}^{-1/2} = I_{P_\ell}. \tag{9}$$

Hence $B_{\mathcal{I}\mathcal{I}} = I + E$ where $E = B_{\mathcal{I}\mathcal{I}} - I$ has zero diagonal blocks and collects only the couplings between distinct layers of S. Both $B_{\mathcal{I}\mathcal{I}}$ and E are symmetric, so Weyl's inequality gives $\lambda_i(B_{\mathcal{I}\mathcal{I}}) \in [1 - \|E\|_2,\ 1 + \|E\|_2]$ for every i. □

Remark 2. *When S contains a single layer, $E = 0$ and the corresponding part of the spectrum is exactly $\{1\}$. When $S = \{1, \dots, L\}$, that is the exact block-diagonal preconditioner, $B = I + E$ globally and $\|E\|_2$ measures the total off-block coupling. This is the quantity reported in the last column of Table 14, which is 1.7×10^3 to 4.4×10^3 under a layer scalar and 3.00 under the exact block diagonal.*

C.3 Unpreconditioned conjugate gradient sees only the range of $J^\top$

The following observation explains why the global condition number of the damped system is an uninformative summary in this setting, and why we report the captured fraction instead.

Proposition 3. *Let $b = \nabla_\theta \mathcal{L} = J^\top r$ and $A_\lambda = J^\top J + \lambda I$ with $\lambda > 0$. Then $b \in \mathrm{range}(J^\top)$, $A_\lambda \,\mathrm{range}(J^\top) \subseteq \mathrm{range}(J^\top)$, and consequently*

$$\mathcal{K}_K(A_\lambda, b) = \mathrm{span}\{b, A_\lambda b, \dots, A_\lambda^{K-1} b\} \subseteq \mathrm{range}(J^\top), \qquad d^\star = A_\lambda^{-1} b \in \mathrm{range}(J^\top). \tag{10}$$

In particular, when $N < P$ every unpreconditioned conjugate-gradient iterate lies in a subspace of dimension at most N, and the $P - \mathrm{rank}(J)$ eigenvalues of A_λ equal to λ that come from $\ker J$ never influence the iterates.

Proof. $b = J^\top r$ lies in $\mathrm{range}(J^\top)$ by construction. If $v = J^\top w$ then $A_\lambda v = J^\top (J J^\top w) + \lambda J^\top w = J^\top (J J^\top w + \lambda w) \in \mathrm{range}(J^\top)$, which gives invariance and, by induction, the statement about $\mathcal{K}_K$. Since $A_\lambda \succ 0$ it is invertible, and its restriction to the invariant subspace $\mathrm{range}(J^\top)$ is invertible on that subspace, so $A_\lambda^{-1} b$ lies in it. □

Remark 3. *Two consequences of Proposition 3 are used in the paper. The first is that $\kappa(A_\lambda)$ is dominated by the ratio $\lambda_{\max}(G)/\lambda$ arising from the damped null space, which the solver never explores, so it predicts nothing about K-step behaviour. The measurements agree. Under the*

LANCE preconditioner κ of the transformed matrix is two to three orders of magnitude larger *than without it on every problem (Table 14), while the captured fraction improves on every problem at the operating budget (Tables 15 and 16). The second is that a preconditioner M that is not a polynomial in A_λ breaks the invariance, since $M^{-1}\mathrm{range}(J^\top)$ need not be contained in $\mathrm{range}(J^\top)$, and this costs iterations. Every component of an iterate along* $\ker J$ *lowers the quadratic model, since q restricted to* $\ker J$ *is $-\frac{\lambda}{2}\|d\|^2$, and $d^\star$ has no such component. Write $B = M^{-1/2}A_\lambda M^{-1/2} = M^{-1/2}GM^{-1/2} + \lambda M^{-1}$. The first term has rank at most N, so by interlacing at least $P-N$ eigenvalues of B lie within the spectrum of λM^{-1}, which on an exact-block layer is the set of values $\lambda/(\sigma+\lambda)$ over the eigenvalues σ of $G_{\ell\ell}$, a spread over several orders of magnitude, in place of the single value λ that A_λ has on* $\ker J$*. The preconditioned residual $M^{-1}b$ has components on those directions, so the preconditioned iteration inherits a de-clustered part of the spectrum that the unpreconditioned one never touched and must spend iterations resolving it. This is why the unpreconditioned solve, confined to an invariant subspace of dimension at most N on which its spectrum has at most N distinct values, overtakes the preconditioned one once K is large enough (Table 15), and why the advantage of the blocks is a property of small budgets.*

C.4 Why the curvature concentrates at the ends of the network

The concentration reported in Appendix G is a property of the network and its initialisation, and it has a short explanation. The measurements behind it come from a companion notebook whose training code is copied from the campaign notebook (Appendix P), and Tables 5 to 7 report them. Write the per-point gradient with respect to the weight matrix of layer ℓ as the outer product $\delta_{\ell,i}\, a_{\ell-1,i}^\top$ of the backpropagated sensitivity, which equals the gradient with respect to the bias, and the layer input. For a residual that depends on the parameters through one forward evaluation this is exact, and the trace of the layer's block is

$$\operatorname{tr} G_{\ell\ell} = \sum_{i=1}^{N} \|\delta_{\ell,i}\|_2^2 \left(1 + \|a_{\ell-1,i}\|_2^2\right), \tag{11}$$

the one from the biases and the norm from the weights. For the PDE residual the derivative terms of the operator add further outer products, and Eq. (11) holds for the weight part up to a factor R_ℓ that we measure as 0.98 on Helmholtz-1D and 1.67 to 1.77 on Burgers and Poisson-2D (Table 5).

Table 5: The two factors of Eq. (11) at initialisation, width 64, seed 0, $N = 1{,}024$, means over the collocation points. R_ℓ is the ratio of the measured weight trace of layer ℓ to $\sum_i \|\delta_{\ell,i}\|^2\|a_{\ell-1,i}\|^2$.

Problem	**residual**	$\|a_{\ell-1}\|^2$ by layer	$\|\delta_\ell\|^2$ by layer	bias share of $\operatorname{tr} G$	R_ℓ by layer
Helmholtz-1D	pde	0.35, 0.56, 0.59, 0.53	$5.5, 6.4, 8.4, 4.9$	0.70	0.98, 0.98, 0.98, 0.98
Helmholtz-1D	regression	0.35, 0.56, 0.59, 0.53	$3.6, 4.2, 5.5, 3.2 \times 10^{-5}$	0.70	1.00, 1.00, 1.00, 1.00
Burgers	pde	0.69, 1.16, 0.89, 0.87	$4.3, 5.1, 5.6, 2.1 \times 10^{-3}$	0.43	1.76, 1.74, 1.72, 1.71
Burgers	regression	0.69, 1.16, 0.89, 0.87	$3.9, 4.6, 5.0, 1.9 \times 10^{-4}$	0.51	1.00, 1.00, 1.00, 1.00
Poisson-2D	pde	0.69, 1.28, 0.94, 0.83	$1.0, 1.2, 1.3, 0.5 \times 10^{-3}$	0.41	1.77, 1.67, 1.67, 1.73
Poisson-2D	regression	0.69, 1.28, 0.94, 0.83	$2.4, 2.8, 3.0, 1.1 \times 10^{-6}$	0.55	1.00, 1.00, 1.00, 1.00

At initialisation both factors are of the same order in every layer. Glorot initialisation keeps the sensitivities within a factor of three across layers, and a tanh network with one or two inputs starts in a regime in which every layer input has squared norm between 0.35 and 1.3, so the biases carry 41 to 70% of the trace and each layer's trace lies within a factor of 3.2 of every other's. The parameter counts differ by a factor of the width, so the density $\rho_\ell \approx (P/L)/P_\ell$ is of order n for the two end layers and below one for the hidden ones. The shares are independent of the width at leading order, and ρ_{in} and ρ_{out} grow as $n^{0.98}$ and $n^{0.9}$ from width 16 to 512 on Helmholtz-1D and Burgers, which is the trend across widths in Table 13.

The operator plays no part in where the curvature sits. Replacing the PDE residual by the plain regression residual $u_\theta - u^\star$ changes no layer share, averaged over three seeds, by more than 0.019 on the nine problems at widths 32 to 128 and depths 3 and 5 (Table 6 shows width 64), and in the same study replacing it by a first-derivative residual changes none by more than 0.024 and removing the hard-constraint factor changes none by more than 0.032. What the operator changes is the scale of every sensitivity, by a factor of 7.5 to 1.5×10^5 in the trace at width 64, and the conditioning inside each block (Appendix H).

Table 6: Layer energy shares and densities at initialisation for the PDE residual and the regression residual on the same networks, width 64, three hidden layers, $N = 1{,}024$, means over three seeds.

	PDE residual			**regression residual** $u_\theta - u^\star$		
Problem	E_ℓ	ρ_{in}	ρ_{out}	E_ℓ	ρ_{in}	ρ_{out}
Burgers ($\nu{=}0.01/\pi$)	0.27, 0.31, 0.28, 0.15	11.9	19.4	0.27, 0.30, 0.28, 0.15	12.2	19.4
Burgers ($\nu{=}0.002/\pi$)	0.27, 0.31, 0.28, 0.15	11.9	19.4	0.27, 0.30, 0.28, 0.15	12.2	19.4
Helmholtz-1D ($k{=}20$)	0.21, 0.27, 0.33, 0.18	14.1	23.8	0.21, 0.27, 0.33, 0.18	14.1	23.8
Poisson-2D	0.27, 0.31, 0.28, 0.14	12.0	18.6	0.28, 0.31, 0.27, 0.14	12.6	18.6
Allen–Cahn	0.26, 0.31, 0.28, 0.15	11.6	19.4	0.27, 0.30, 0.28, 0.15	12.2	19.4
Wave-1D	0.26, 0.32, 0.28, 0.14	11.6	18.6	0.27, 0.31, 0.27, 0.14	12.1	18.6
Helmholtz-2D ($k{=}10$)	0.28, 0.30, 0.27, 0.14	12.7	18.7	0.28, 0.31, 0.27, 0.14	12.6	18.6
Poisson-5D	0.21, 0.33, 0.31, 0.15	4.8	20.4	0.21, 0.32, 0.31, 0.15	4.9	20.4
Poisson-10D	0.21, 0.28, 0.36, 0.15	2.7	21.2	0.21, 0.28, 0.36, 0.15	2.7	21.2

The same explanation says when the concentration fails (Table 7). Halving the Glorot scale moves 62% of the trace to the output layer and doubling it moves 44% to the input layer. LeCun and He initialisation, whose hidden-layer inputs have squared norms of 6 to 17, leave the input layer with a density of 1.6 to 2.1 while the output layer stays above 40, and in the activation study of the companion notebook the activations that do not vanish at zero do the same, the input layer's density falling to 1.3 under GELU and to 0.0 under softplus while the output layer takes 0.62 to 0.65 of the trace. The rule therefore selects both end layers under Glorot-type initialisation of tanh, sine or linear networks, which give the same shares to within 0.002, and the output layer alone otherwise.

Table 7: Helmholtz-1D, width 64, PDE residual, means over three seeds. Halving or doubling the Glorot scale, or a fan-in initialisation, moves the trace between the end layers, and the output layer stays dense under every scheme.

Initialisation	E_{in}	E_{out}	ρ_{in}	ρ_{out}	$\max_{\text{hid}} \rho_\ell$	weights' share	$\|a_{\ell-1}\|^2$ by layer
Glorot normal (the paper)	0.21	0.18	14.1	23.8	0.68	0.30	0.35, 0.55, 0.59, 0.56
Glorot normal $\times$ 0.5	0.02	0.62	1.2	81.1	0.60	0.03	0.35, 0.14, 0.04, 0.01
Glorot normal $\times$ 2	0.44	0.10	29.1	12.5	0.51	0.51	0.35, 2.04, 6.50, 13.59
LeCun normal	0.02	0.36	1.6	47.6	0.71	0.87	0.35, 10.38, 8.22, 6.29
He normal	0.03	0.32	2.1	42.4	0.72	0.92	0.35, 15.44, 16.89, 17.30
Glorot uniform	0.25	0.16	16.4	20.5	0.65	0.31	0.35, 0.64, 0.65, 0.59
NTK parameterisation	0.28	0.17	18.8	21.8	0.61	0.07	0.35, 0.55, 0.59, 0.56

During training the layer inputs grow with the weights, the weights' share of the trace rises, and the curvature moves into the hidden layers. On Helmholtz-1D the first hidden layer's squared output norm, averaged over the collocation points, grows from 0.5 to 21 in a hundred LANCE iterations, the weights' share of the trace rises from 0.30 to 0.92 and the end layers fall from 0.39 to 0.05 of it, whereas on Burgers the norm reaches 7.7 and the end layers keep 0.20. This is the migration of §6, and it occurs for the regression residual as well, where the same quantities on Helmholtz-1D reach 23, 0.94 and 0.03. The truncation deficit that the blocks recover is not specific to the operator either. The same loop on the regression residual develops it once the damping has fallen, and LANCE then ends ahead of GN-CG by 9.3 times on Helmholtz-1D and 5.7 on Burgers. At a fixed damping the operator makes the deficit far larger. After forty GN-CG iterations at $\lambda = 10^{-3}$ the unpreconditioned solve captures 0.03 and 0.06 of the attainable decrease at $K = 10$ on the PDE residual of Helmholtz-1D and Poisson-2D against 0.99 and 1.00 on the regression residual, and the damped input block of Helmholtz-1D has condition number 4.5×10^7 against 5.2×10^2.

C.5 The matrix-free product, worked through one PDE

Equation (2) is standard but its content for a PINN residual is easy to lose. We write it out for the one-dimensional Helmholtz problem, where every step can be checked by hand.

Let NN_θ be the network, $\phi(x) = x(1-x)$ the hard-constraint factor, and $u_\theta = \phi\,\mathrm{NN}_\theta$. The residual at collocation point x_i is

$$r_i(\theta) = \tfrac{1}{\sqrt{N}}\Big(\partial_x^2 u_\theta(x_i) + k^2 u_\theta(x_i) - f(x_i)\Big). \tag{12}$$

The quantity the solver needs is $Gv = J^\top(Jv)$ for a direction $v \in \mathbb{R}^P$. The first factor is a directional derivative of the residual map,

$$(Jv)_i = \lim_{\varepsilon \to 0} \frac{r_i(\theta + \varepsilon v) - r_i(\theta)}{\varepsilon} = \tfrac{1}{\sqrt{N}} \Big(\partial_x^2 \big[\nabla_\theta u_\theta(x_i) \cdot v \big] + k^2 \, \nabla_\theta u_\theta(x_i) \cdot v \Big), \tag{13}$$

where the exchange of ∂_x^2 and ∇_θ is legitimate because u_θ is smooth in both arguments and the operator is linear, which is the commutation invoked in §2.3. In words, Eq. (13) says that the directional derivative of the residual is the residual operator *applied to the directional derivative of the solution field*. One forward-mode pass propagates the tangent $\nabla_\theta u_\theta \cdot v$ through the same derivative graph that computes u_θ and its spatial derivatives, so the cost is a small multiple of one forward pass and no per-parameter work appears.

The second factor is the adjoint. For $w \in \mathbb{R}^N$,

$$J^\top w = \sum_{i=1}^{N} w_i \, \nabla_\theta r_i(\theta), \tag{14}$$

which one reverse-mode pass computes for the whole sum at once, again without materialising any row of J. Composing the two with $w = Jv$ gives Gv exactly, in floating point to the accuracy reported in Table 9, at a measured 1.0 to 1.3 gradient evaluations.

Nothing in this construction is specific to the operator beyond smoothness, so the same two passes serve the Burgers nonlinearity $u\,\partial_x u$ and the Allen–Cahn term $u - u^3$, with the chain and product rules supplying the extra factors. The construction gives Gv and nothing else. It gives neither $\mathrm{diag}(G)$ nor $G_{\ell\ell}$, which require the separate per-point pass of Algorithm 1. That pass is where the additional 2.0 to 5.3 gradients of the LANCE step are spent, and it is the reason the method is described as paying a few extra gradient-like passes for the selection and the blocks.

D THE COMPLETE ALGORITHM

Defaults. $K = 10$ (except $K = 15$ on low-viscosity Burgers), $\lambda_0 = 10^{-3}$, $P_{\max} = 512$, $\rho_{\mathrm{th}} = 1$, $c_1 = 10^{-4}$, $\beta_{\mathrm{down}} = 0.7$, $\beta_{\mathrm{up}} = 1.5$, $[\lambda_{\min}, \lambda_{\max}] = [10^{-8}, 10^{1}]$, at most 30 halvings. M_t is refreshed at every iteration. GN-CG is the same loop with $M_t = I$.

Scaling of the step. With $\mathcal{L} = \frac{1}{2}\|r\|^2$ the solution of Eq. (3) tends to the Gauss–Newton step as $\lambda \to 0$, so the unit trial step is the Gauss–Newton step itself. Doubling the trial step, which is what solving the same system for the loss $\|r\|^2$ amounts to, changes the final LANCE error by between 0.8 and 1.5 times (Table 24).

Table 8: Per-operation time and memory costs. Multiples are the values measured in §5.1 on the reported stack, π is the measured cost of the per-point pass, c its chunk size, ℓ the number of line-search evaluations and $\mathcal{E}$ the eligible layers, $P_\ell \le P_{\max}$.

Component	Time (gradient-equivalents)	Extra memory
Loss and gradient, one `value_and_grad`	$1\times$	$\mathcal{O}(P+N)$
Per-point gradient pass, $\mathrm{diag}(G)$ and blocks	$\pi = 2.0$–$5.3\times$	$\mathcal{O}(cP) + \sum_{\mathcal{E}} P_\ell^2$
One damped Gauss–Newton product	1.0–1.3×	$\mathcal{O}(P+N)$
Cholesky of the exact blocks	$\mathcal{O}(\sum_{\ell \in S} P_\ell^3)$ flops	—
PCG solve, fixed K	$\approx 1.2K\times$	$\mathcal{O}(P)$
One full step	$\approx (\pi + 1 + 1.2K + \ell)\times$	$\mathcal{O}(P+N) + \sum_{\mathcal{E}} P_\ell^2$
Dense G (diagnostic only)	$\lceil P/256 \rceil$ batched JVPs, $\mathcal{O}(NP^2)$	$\mathcal{O}(P^2)$

Cost model.

Dense Gauss–Newton diagnostics. Applying JVP to blocks of 256 identity columns assembles G column-block-wise, $J_{:,\mathcal{C}} = \mathrm{JVP}_\theta(r, E_{\mathcal{C}})$ and $G = J^\top J$, at a few hundred megabytes of peak memory instead of the 8.9 GB of a one-shot `jacfwd` at $P \approx 8.5 \times 10^3$. Dense assembly is $\mathcal{O}(P^2)$

Algorithm 1 LANCE, layer-aware exact Gauss–Newton training.

Require: initial parameters θ_0, residual $r : \mathbb{R}^P \to \mathbb{R}^N$, CG steps K, damping λ_0, block-size limit $P_{\max}$, density threshold ρ_{th}, Armijo constant c_1, damping factors $\beta_{\text{down}} < 1 < \beta_{\text{up}}$, damping range $[\lambda_{\min}, \lambda_{\max}]$

Ensure: trained parameters θ_T

$\lambda \leftarrow \lambda_0$
for $t = 0$ **to** $T - 1$ **do**
  $D_t, \{B_{t,\ell}\} \leftarrow \textsc{PerPointGradientPass}(r, \theta_t, P_{\max})$ ▷ Stage 0
  $\tau_{t,\ell} \leftarrow \sum_{j \in \ell} D_{t,j}$; $\rho_{t,\ell} \leftarrow \dfrac{\tau_{t,\ell} / \sum_{\ell'} \tau_{t,\ell'}}{P_\ell / P}$; $\mu_{t,\ell} \leftarrow \tau_{t,\ell} / P_\ell$
  $S_t \leftarrow \{\ell : \rho_{t,\ell} > \rho_{\text{th}},\ P_\ell \le P_{\max}\}$
  $M_t \leftarrow \textsc{BlockDiag}(\{B_{t,\ell} + \lambda I\}_{\ell \in S_t},\ \{(\mu_{t,\ell} + \lambda) I\}_{\ell \notin S_t})$ ▷ Cholesky of the exact blocks
  $g_t \leftarrow \nabla_\theta \mathcal{L}(\theta_t)$ ▷ Stage 1, one `value_and_grad`
  $d_t \leftarrow \text{PCG}(v \mapsto J_t^\top (J_t v) + \lambda v,\ g_t,\ M_t^{-1},\ K)$ ▷ Stage 2, K products
  $\alpha \leftarrow 1$ ▷ Stage 3, Armijo backtracking
  while $\mathcal{L}(\theta_t - \alpha d_t) > \mathcal{L}(\theta_t) - c_1 \alpha\, g_t^\top d_t$ **and** not exhausted **do**
    $\alpha \leftarrow \alpha / 2$
  end while
  $\theta_{t+1} \leftarrow \theta_t - \alpha d_t$ ▷ Stage 4
  $\lambda \leftarrow \text{clip}\big(\beta_{\text{down}} \lambda \text{ if } \alpha = 1 \text{ else } \beta_{\text{up}} \lambda,\ \lambda_{\min}, \lambda_{\max}\big)$ ▷ Stage 5
end for
return θ_T

function $\textsc{PerPointGradientPass}(r, \theta, P_{\max})$ ▷ never forms J or G
  $D \leftarrow 0 \in \mathbb{R}^P$; $B_\ell \leftarrow 0 \in \mathbb{R}^{P_\ell \times P_\ell}$ for all ℓ with $P_\ell \le P_{\max}$
  for each chunk $\mathcal{C}$ of collocation points **do**
    $J_{\mathcal{C}} \leftarrow \big[\nabla_\theta r_i(\theta)\big]_{i \in \mathcal{C}}$ ▷ batched reverse-mode AD
    $D \leftarrow D + \text{diag}(J_{\mathcal{C}}^\top J_{\mathcal{C}})$ ▷ sum of squares, $\mathcal{O}(|\mathcal{C}|P)$
    $B_\ell \leftarrow B_\ell + J_{\mathcal{C},\ell}^\top J_{\mathcal{C},\ell}$ for all such ℓ ▷ $\mathcal{O}(|\mathcal{C}|P_\ell^2)$
  end for
  return $D,\ \{B_\ell\}$
end function

in memory, is used only for diagnostics, and is never on the training path. From it we compute the eigenspectrum, the layer energies of Eq. (4), the exact direction $d^\star$, the direction cosine $\cos(d_K, d^\star)$ and the captured fraction $q(d_K)/q(d^\star)$.

E Full-scale experimental protocol

Environment. Google Colab with a single NVIDIA A100, JAX 0.11.1 (Bradbury et al., 2018) with `jax_enable_x64`, Optax for the baseline optimisers. Training is full batch. Wall-clock time is measured around each optimiser call. Just-in-time compilation is triggered by the first call for a given problem, method and shape, which falls on seed 0, so every wall-clock figure in the paper is a median over seeds, and the primitive timings of Table 9 are taken after a warm-up call.

Networks. Every model is a tanh multilayer perceptron with scalar output and, unless the scaling study varies it, three hidden layers of width 64. The input layer has $64(d+1)$ parameters, each hidden layer 4,160 and the output layer 65, so $P = 8{,}513$, $8{,}577$, $8{,}769$ and $9{,}089$ for $d = 1, 2, 5, 10$. Weights are Glorot-normal, biases zero, and the initialisation is regenerated by every seed. The scaling study of Appendix L uses widths 128 and 256 and depths 5 and 8.

Discretisation and evaluation. The residual has $N = 1{,}024$ interior terms on Helmholtz-1D and $N = 4{,}096$ elsewhere, rising to $8{,}192$ and $16{,}384$ in the two largest scaling configurations. Collocation points are drawn independently and uniformly and are shared across all methods within a seed. The relative L^2 error $\|u - u^\star\|_2 / \|u^\star\|_2$ is evaluated on fixed test sets independent of the training points, $1{,}001$ points on Helmholtz-1D, a 101×51 mesh on the time-dependent problems, a 101×101

mesh on the two-dimensional elliptic problems and 4,000 fixed random points in $[0.05, 0.95]^d$ for Poisson in five and ten dimensions. All optimisers start from the same θ_0 within a seed and configuration.

Problems. Every problem is hard-constrained by the ansatz $u_\theta = g + \varphi\,\mathrm{NN}_\theta$ with φ vanishing on the constrained part of the boundary, and its forcing f is the exact operator applied to $u^\star$. (i) Viscous Burgers on $(x,t) \in [-1,1] \times [0,1]$, $u_t + uu_x - \nu u_{xx} = f$, with $u^\star = e^{-t}\sin(\pi x)$, $\nu \in \{0.01/\pi,\, 0.002/\pi\}$, $g = \sin(\pi x)$ and $\varphi = t(1-x^2)$. (ii) Helmholtz-1D, $u'' + 400u = f$ on $[0,1]$, $u^\star = x(1-x)\sin(6\pi x)$, $\varphi = x(1-x)$. (iii) Poisson-2D, $-\Delta u = f$ on the unit square, $u^\star = \sin(\pi x)\sin(\pi y)$, $\varphi = x(1-x)y(1-y)$. (iv) Allen–Cahn on $[-1,1] \times [0,1]$, $u_t - \varepsilon u_{xx} - (u - u^3) = f$ with $\varepsilon = 10^{-3}$, $u^\star = e^{-t}\sin(\pi x)$, $g = \sin(\pi x)$, $\varphi = t(1-x^2)$. (v) Wave-1D on $[0,1]^2$, $u_{tt} - c^2 u_{xx} = f$ with $c = 2$, $u^\star = \sin(\pi x)\cos(2\pi t)$, $g = \sin(\pi x)$, $\varphi = x(1-x)t^2$. (vi) Helmholtz-2D, $\Delta u + 100u = f$ on the unit square, $u^\star = \sin(3\pi x)\sin(3\pi y)$, $\varphi = x(1-x)y(1-y)$. (vii) Poisson-dD for $d \in \{5, 10\}$, $-\Delta u = f$ on $[0,1]^d$, $u^\star = \prod_i \sin(\pi x_i)$, $\varphi = \prod_i x_i(1-x_i)$.

Baselines and budgets. Adam at learning rate 10^{-3} for 10,000 iterations on the two Burgers problems and 15,000 elsewhere. Full-batch L-BFGS, the Optax implementation with a zoom line search, for 1,000 iterations, with the error after 1,500 reported in Table 11. GN-CG and LANCE for 100 iterations at $K = 10$, except low-ν Burgers at 150 iterations and $K = 15$, with $\lambda_0 = 10^{-3}$ and the damping adaptation of Algorithm 1. The curvature-method comparison of §5.4 runs every representation for 100 iterations from the same θ_0 with the same damping schedule and Armijo search, on three seeds at widths 32 and 64. The robustness sweeps of §5.5 use ten seeds at width 64 and 100 iterations.

The two-parameter toy of Figure 2b. The residual is $r(\theta) = \big(\theta_1 - 1 + (\theta_2 - 1)^2,\ (\theta_2 - 1) + \frac{1}{5}(\theta_1 - 1)^2\big)$, the current iterate $\theta_t = (-0.4, 1.67)$ and the damping $\lambda = 10^{-3}$. GN-CG is the first unpreconditioned CG iterate from zero, LANCE the solution $d^\star$ of the damped system, Newton the step with the full Hessian $G + \sum_i r_i \nabla^2 r_i$.

Exactness and cost of the primitives. On seven problems at $P \approx 200$, where every quantity can be formed densely, the product, the diagonal and the exact blocks agree with their dense references to a relative error of at most 6.5×10^{-16}, the nearest-Kronecker routine recovers an exactly Kronecker matrix to 2.3×10^{-16} and returns positive semidefinite factors for a positive definite matrix, preconditioned CG run to $K = P$ under each of the four preconditioners recovers the dense solution of Eq. (3) to 10^{-9} or better, and the quadratic model $q(d_k)$ is non-decreasing in k. Table 9 reports the product at full size, on the seed-0 initialisation, averaged over 30 repetitions after compilation and device synchronisation. The pass ratio is the cost of the chunked reverse-mode sweep of Algorithm 1 relative to one gradient at a chunk size of 256. It depends on the chunk size and on how the backend compiles the batched sweep, so it should be re-measured on any other stack.

Table 9: Exactness and wall-clock cost of the two matrix-free primitives at full size, relative to one full-batch loss-gradient evaluation.

Problem	P	N	**product rel. error**	**gradient (ms)**	**product (ms, ratio)**	**per-point pass (ms, ratio)**
Burgers (ν=0.01/π)	8577	4096	2.1e-15	3.78	4.58 (1.21×)	12.45 (3.29×)
Burgers (ν=0.002/π)	8577	4096	2.0e-15	3.74	4.59 (1.23×)	12.03 (3.22×)
Helmholtz-1D (k=20)	8513	1024	5.9e-16	1.48	1.48 (1.00×)	3.00 (2.02×)
Poisson-2D	8577	4096	1.9e-15	2.46	3.29 (1.34×)	13.02 (5.30×)

Run-to-run variability. Rounding differences between GPU kernels change the trajectories of individual runs, so per-seed final errors on the plateau problems (Helmholtz-1D, Poisson-2D) are reproducible only to within an order of magnitude, whereas the seed-averaged results and every statement in the paper are stable under a change of kernels.

Table 10: Wall-clock time per run in seconds, median over five seeds, for the runs of Table 2. Adam runs 10,000 to 15,000 iterations, L-BFGS 1,000, and the Gauss–Newton methods 100, or 150 on low-ν Burgers. The last two columns are the LANCE to GN-CG and the L-BFGS to LANCE time ratios.

Problem	**Adam**	**L-BFGS**	**GN-CG**	**LANCE**	LANCE/GN-CG	L-BFGS/LANCE
Burgers (ν=0.01/π)	36.7	10.8	6.1	7.8	1.29×	1.38×
Burgers (ν=0.002/π)	36.9	11.1	12.3	15.1	1.23×	0.73×
Helmholtz-1D (k=20)	21.5	5.3	2.5	3.2	1.24×	1.70×
Poisson-2D	36.7	7.8	4.5	6.2	1.38×	1.26×
Allen–Cahn	54.6	10.9	6.0	7.0	1.17×	1.55×
Wave-1D	36.7	7.8	4.5	5.5	1.22×	1.41×
Helmholtz-2D (k=10)	37.5	8.1	4.7	5.6	1.20×	1.45×
Poisson-5D	51.3	10.9	6.6	7.4	1.13×	1.47×
Poisson-10D	79.4	84.1	9.6	10.1	1.06×	8.33×

Table 11: Per-seed minimum and maximum of the final relative L^2 error behind Table 2, and the geometric mean of L-BFGS after 1,500 iterations, which take 1.0 to 2.2 times the LANCE wall-clock and 13 times on Poisson-10D.

Problem	**Adam**	**L-BFGS**	**GN-CG**	**LANCE**	**L-BFGS 1,500 its**
Burgers (ν=0.01/π)	1.2e-03–1.3e-02	7.3e-04–1.1e-03	2.0e-04–6.0e-04	7.1e-06–3.3e-05	4.35e-04
Burgers (ν=0.002/π)	1.1e-03–9.0e-03	4.7e-04–8.0e-04	8.0e-05–1.4e-04	5.3e-06–1.9e-05	4.31e-04
Helmholtz-1D (k=20)	6.2e-02–1.1e+00	4.4e-02–1.1e+00	5.4e-03–1.1e-01	9.3e-04–1.7e-02	4.12e-02
Poisson-2D	2.3e-04–2.0e-03	6.3e-06–1.1e-05	1.8e-05–6.4e-05	8.8e-07–5.4e-06	4.76e-06
Allen–Cahn	1.0e-03–1.6e-03	3.1e-04–1.6e-03	4.8e-04–9.2e-04	1.4e-05–5.6e-05	5.34e-04
Wave-1D	4.6e-04–9.9e-04	4.4e-04–9.4e-04	4.1e-03–1.1e-02	1.6e-04–3.7e-03	3.53e-04
Helmholtz-2D (k=10)	1.2e+00–1.5e+00	4.4e-01–2.0e+00	4.4e-01–1.2e+00	4.9e-01–1.0e+00	1.02e+00
Poisson-5D	1.2e-01–1.4e-01	3.9e-02–1.0e-01	3.7e-02–1.1e-01	1.2e-03–1.0e-01	5.29e-02
Poisson-10D	1.0e+00–1.0e+00	1.1e-01–1.2e-01	1.1e-01–1.2e-01	1.1e-01–1.2e-01	1.14e-01

F EXTENDED FULL-SCALE RESULTS

Wall-clock. Table 10 gives the times behind Table 2. The LANCE to GN-CG ratio at the same iteration count is 1.06 to 1.38, with the largest values on the two-input problems whose per-point pass is the most expensive relative to a gradient (Table 9) and the smallest on the five- and ten-dimensional Poisson problems. The cost model of Table 8 predicts 1.2 to 1.3 at $K = 10$ from the measured primitive rates, within 0.1 of the measured ratio on each of the four problems on which the rates were measured. L-BFGS at 1,000 iterations takes 0.7 to 1.7 times the LANCE wall-clock on eight problems and 8.3 times on Poisson-10D, and at 1,500 iterations 1.0 to 2.2 times and 13 times, so the comparison of Table 2 gives L-BFGS equal or more time than LANCE on every problem except low-ν Burgers, where it gives 0.7 of it. Adam runs for 2.5 to 8 times the LANCE wall-clock.

Per-seed picture. Figure 7 shows every seed, and Table 11 the per-seed extremes of every method. On the seven problems that some method solves the per-seed factors range from 0.77 to 91, and the two seeds below one are on Helmholtz-1D, where the length of the initial plateau varies with the initialisation and LANCE ends at 7.0×10^{-3} against 5.4×10^{-3}, and on Poisson-5D, where both methods stall at 1.0×10^{-1}. On Helmholtz-2D the factors are 0.43 to 1.8 and on Poisson-10D 1.0. On Poisson-5D LANCE leaves the stall on three of the five seeds, reaching 1.2×10^{-3} to 3.0×10^{-3} where GN-CG stays between 3.7×10^{-2} and 1.1×10^{-1} on every seed, and on the other two seeds the two methods end together near 10^{-1}. L-BFGS at 1,500 iterations reaches 4.8×10^{-6} on Poisson-2D, within 2.2 of LANCE, and 3.5×10^{-4} on Wave-1D, ahead of it, and stays 12 to 33 times behind on Burgers, low-ν Burgers, Helmholtz-1D and Allen–Cahn.

The selected set. The per-point pass measures $\rho_{t,\ell}$ at every iteration and the rule re-selects S_t from it. At initialisation the densities of the input and output layers exceed 1.3 on every configuration of the campaign and no hidden layer is eligible, so S_0 is the input and output layers throughout (Appendix G). Logged at every iteration on five seeds of the four primary problems, both layers stay selected at every iteration on the two Burgers problems. On Poisson-2D the input layer is selected

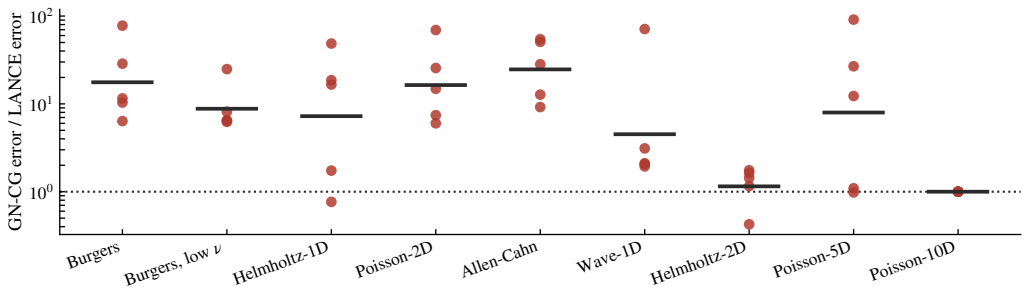


Figure 7: Per-seed ratio of the final GN-CG error to the final LANCE error on the nine problems of Table 2, with the geometric mean as a bar. On the seven solved problems 33 of the 35 seeds lie above one.

at 99% of the iterations and the output layer at 62%, two seeds dropping it within the first four iterations. On Helmholtz-1D the densities equalise during training, the input layer is selected at 95% of the iterations and the output layer at 46%, leaving S_t between iterations 17 and 95 on every seed, and on one seed the input layer leaves too. With $P_{\max} = 8192$ every layer is eligible and the rule admits a hidden block by the last iteration on six of ten Helmholtz-1D seeds, nine of ten Poisson-2D seeds and one Burgers seed, which is the source of the further gain of that setting in Table 21.

The stalled problems. On Helmholtz-2D with $k = 10$ every method ends at a relative error near one. On Poisson-10D, L-BFGS, GN-CG and LANCE stop between 0.11 and 0.12 on every seed. Three different methods stopping at the same error indicates a common stationary point of the residual, and on Poisson-10D the unpreconditioned solve already captures 1.000 of the attainable model decrease at $K = 1$ (Appendix I), so there is no truncation deficit for a preconditioner to act on. On Poisson-5D the loop is still moving, the full step is accepted at only 29 to 40% of the iterations (Table 23), and the preconditioner makes the difference between leaving the stall and not on three of five seeds. The difficulty of PINN training at high frequency and in high dimension is documented (Wang et al., 2021; Rathore et al., 2024).

Trained solutions. Figures 8 and 9 extend Figure 5 to the other eight problems, on seed 0 of each, with the budgets of Table 2, and Table 12 gives the relative errors of the runs shown. Poisson-5D and Poisson-10D are drawn on the (x_1, x_2) plane through the centre of the domain, the other coordinates at 0.5. The two stalled rows are those of the paragraph above, on which every method ends near a relative error of one on Helmholtz-2D and L-BFGS, GN-CG and LANCE end within one per cent of the same error on Poisson-10D. The Poisson-5D seed shown is one of the three on which LANCE leaves the stall.

Table 12: Relative L^2 error on the test sets of Appendix E of the seed-0 runs shown in Figures 8 and 9.

Problem	Adam	L-BFGS	GN-CG	LANCE
Burgers (ν=0.01/π)	1.42e-03	7.33e-04	2.00e-04	**1.73e-05**
Burgers (ν=0.002/π)	1.38e-03	8.03e-04	1.43e-04	**1.76e-05**
Poisson-2D	2.01e-03	6.49e-06	2.54e-05	**1.71e-06**
Allen–Cahn	1.01e-03	8.13e-04	6.99e-04	**2.47e-05**
Wave-1D	**6.56e-04**	6.85e-04	7.54e-03	3.66e-03
Helmholtz-2D (k=10)	1.29e+00	**4.37e-01**	1.15e+00	7.07e-01
Poisson-5D	1.17e-01	1.03e-01	1.14e-01	**1.89e-03**
Poisson-10D	9.99e-01	**1.14e-01**	1.14e-01	1.14e-01

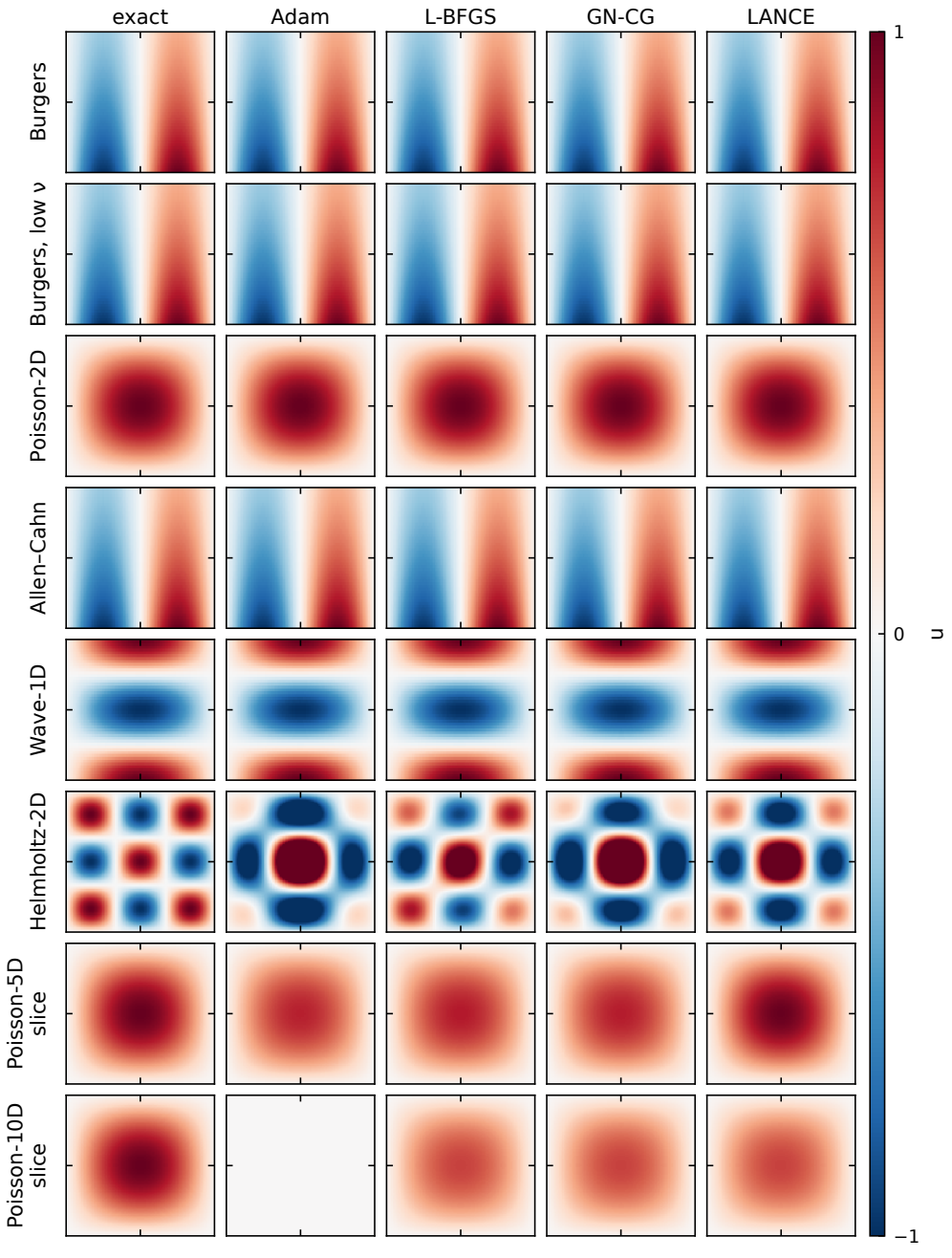


Figure 8: Trained solutions on seed 0. Each row is one problem of Table 2, the first column the exact solution and the others the networks trained by the four optimisers, all on one colour scale. Horizontal axis x, on $[-1, 1]$ for Burgers and Allen–Cahn and $[0, 1]$ otherwise, vertical axis t on the time-dependent problems and y or x_2 otherwise. Helmholtz-1D is Figure 5.

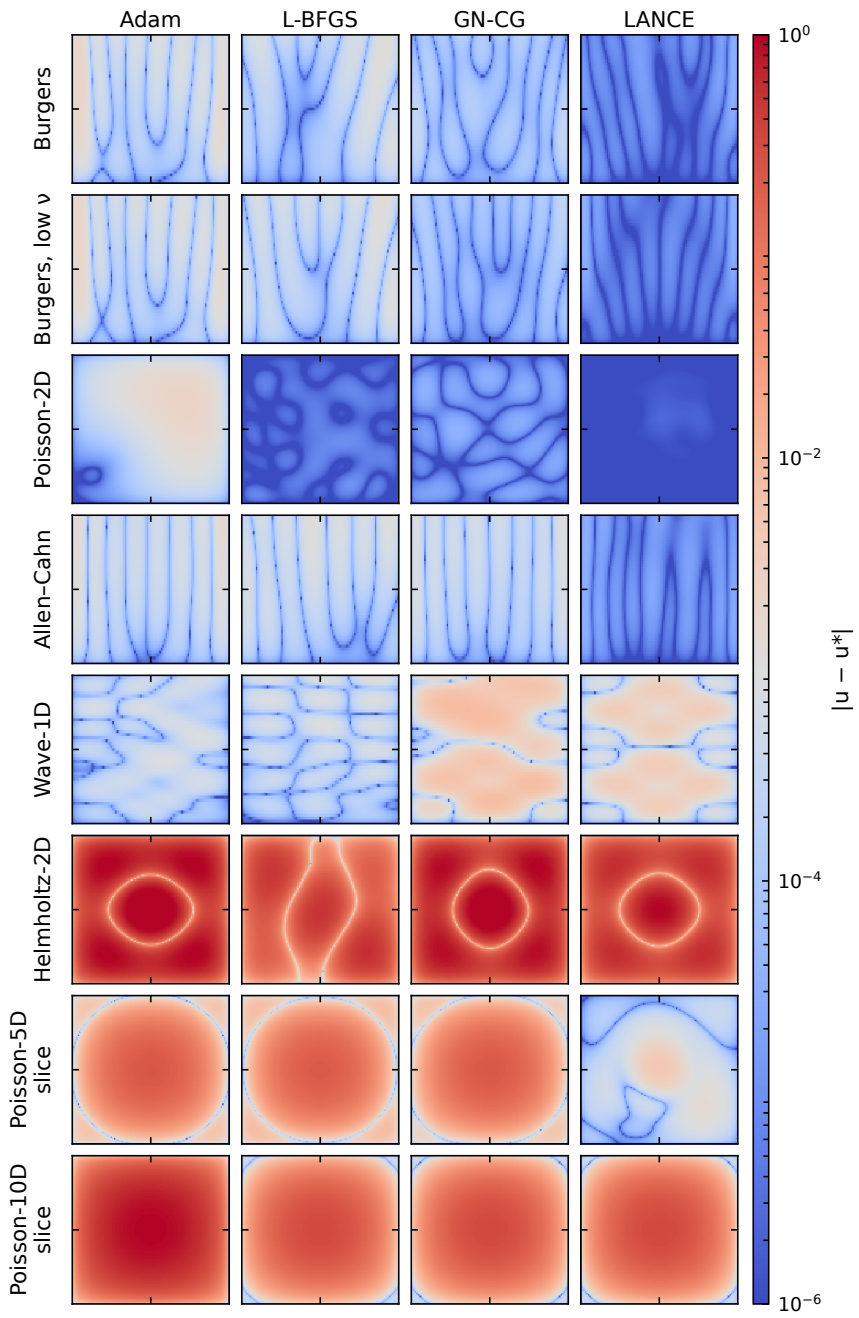


Figure 9: Pointwise error $|u_\theta - u^\star|$ of the networks of Figure 8, on one logarithmic scale for every panel.

G CURVATURE CONCENTRATION ACROSS PROBLEMS AND ARCHITECTURES

The method rests on the claim that the curvature of a PINN concentrates in the input and output layers, which §3 states for one architecture on one problem. Table 13 tests it on the nine PDEs of the campaign at widths 32, 64 and 128 with three and five hidden layers, averaging the layer statistics of Eq. (4) at initialisation over five Glorot initialisations, with $N = 1{,}024$ collocation points fixed per configuration. The two Burgers problems coincide at initialisation, so one is shown.

Table 13: Curvature concentration at initialisation, mean over five initialisations. P_{sm}/P is the fraction of the parameters held by the input and output layers together and E_{sm} their share of $\mathrm{tr}(G)$. A curvature density $\rho_\ell > 1$ means the layer holds more curvature than its share of the parameters.

Problem	width	depth	P	P_{sm}/P	E_{sm}	$\rho_{\mathrm{in}}, \rho_{\mathrm{out}}$	$\max_{\mathrm{hid}} \rho_\ell$
Burgers	32	3	2241	5.8%	41%	5.6, 11.1	0.71
	64	3	8577	3.0%	39%	10.5, 21.0	0.67
	128	3	33537	1.5%	34%	17.4, 36.8	0.70
	32	5	4353	3.0%	25%	6.8, 12.9	0.97
	64	5	16897	1.5%	25%	13.7, 25.4	0.93
	128	5	66561	0.8%	22%	22.2, 47.7	0.84
Helmholtz-1D	32	3	2209	4.4%	37%	7.5, 10.4	0.74
	64	3	8513	2.3%	40%	14.8, 23.0	0.67
	128	3	33409	1.2%	39%	34.0, 33.0	0.65
	32	5	4321	2.2%	25%	10.0, 13.6	0.97
	64	5	16833	1.1%	26%	21.4, 26.3	0.85
	128	5	66433	0.6%	24%	39.2, 46.9	0.90
Poisson-2D	32	3	2241	5.8%	41%	5.7, 11.1	0.71
	64	3	8577	3.0%	39%	10.6, 20.7	0.69
	128	3	33537	1.5%	34%	17.4, 37.4	0.70
	32	5	4353	3.0%	25%	7.0, 12.2	0.99
	64	5	16897	1.5%	25%	13.8, 24.7	0.95
	128	5	66561	0.8%	22%	22.0, 48.6	0.85
Allen-Cahn	32	3	2241	5.8%	40%	5.5, 11.2	0.72
	64	3	8577	3.0%	39%	10.2, 21.1	0.68
	128	3	33537	1.5%	33%	16.8, 37.0	0.71
	32	5	4353	3.0%	25%	6.8, 12.8	0.98
	64	5	16897	1.5%	25%	13.3, 25.6	0.93
	128	5	66561	0.8%	22%	21.6, 47.7	0.85
Wave-1D	32	3	2241	5.8%	40%	5.4, 11.3	0.72
	64	3	8577	3.0%	38%	10.1, 20.9	0.70
	128	3	33537	1.5%	33%	16.5, 37.7	0.72
	32	5	4353	3.0%	24%	6.7, 12.4	0.99
	64	5	16897	1.5%	24%	13.1, 24.6	0.96
	128	5	66561	0.8%	21%	20.7, 48.7	0.87
Helmholtz-2D	32	3	2241	5.8%	42%	6.0, 11.1	0.70
	64	3	8577	3.0%	41%	11.3, 20.7	0.67
	128	3	33537	1.5%	36%	19.0, 37.0	0.68
	32	5	4353	3.0%	25%	7.2, 12.6	0.97
	64	5	16897	1.5%	26%	14.7, 24.9	0.93
	128	5	66561	0.8%	23%	24.1, 48.8	0.83
Poisson-5D	32	3	2337	9.6%	38%	3.0, 9.5	0.75
	64	3	8769	5.1%	36%	4.5, 22.4	0.71
	128	3	33921	2.6%	37%	8.7, 46.5	0.66
	32	5	4449	5.1%	29%	4.0, 16.6	0.97
	64	5	17089	2.6%	22%	5.1, 27.1	0.93
	128	5	66945	1.3%	23%	10.9, 53.2	0.89
Poisson-10D	32	3	2497	15.4%	36%	1.3, 13.0	0.85
	64	3	9089	8.5%	36%	2.9, 18.7	0.78
	128	3	34561	4.4%	37%	5.1, 43.6	0.70
	32	5	4609	8.4%	23%	1.3, 17.6	1.05
	64	5	17409	4.4%	22%	3.2, 25.5	1.03
	128	5	67585	2.3%	22%	5.3, 55.6	0.89

The input and output layers hold between 0.6% and 15% of the parameters and between 21% and 42% of the curvature energy, 33 to 42% with three hidden layers and 21 to 29% with five. Their densities never fall below 1.3, so the selection rule $\rho_{t,\ell} > 1$ includes both at initialisation in every configuration, the closest call being the input layer of Poisson-10D at width 32, whose 352 parameters dilute its share. Averaged over initialisations the largest hidden-layer density is 1.05, on Poisson-10D with five hidden layers at width 32, and on individual initialisations at five hidden layers a hidden layer reaches 1.0 to 1.3 in eleven of the 24 configurations, which is immaterial to the rule because every hidden layer has more than $P_{\max}$ parameters. The concentration in energy is nearly independent of width while the parameter share of the small layers halves with each dou-

bling, so their density grows with width, which is the regime in which the exact blocks are cheapest. It weakens with depth because the additional hidden layers take a share of the trace, although the density ordering is unchanged. It weakens with input dimension, since the input layer grows linearly in d while its share of the trace does not, from $\rho_{\text{in}} = 15$ on Helmholtz-1D to 2.9 on Poisson-10D at width 64, while the output layer keeps a density above 9 everywhere and above 18 at width 64. Table 22 extends the measurement to width 256 and depth 8, and Appendix C.4 explains the concentration.

H BLOCK-LEVEL SPECTRAL DIAGNOSTICS

Propositions 1 and 2 are statements about the diagonal blocks of $B = M^{-1/2} A_\lambda M^{-1/2}$. Table 14 evaluates them at full size, $P = 8{,}513$ to $8{,}577$, on the four primary problems at initialisation and at the iterate reached after forty unpreconditioned iterations, at the damping then in force. G is assembled by batched Jacobian–vector products, $M^{-1/2}$ is formed block by block, and every eigenvalue is computed densely.

Table 14: Spectral diagnostics of the transformed matrix $B = M^{-1/2}(G + \lambda I)M^{-1/2}$ at width 64, seed 0. $\kappa(B)$ is the global condition number over the positive spectrum, $\kappa(B_{\text{in}})$ to $\kappa(B_{\text{out}})$ the condition numbers of the four diagonal blocks, and $\|E\|_2$ the spectral norm of the off-block part $B - \text{blkdiag}(B)$.

Problem, stage	**preconditioner**	$\kappa(B)$	$\kappa(B_{\text{in}})$	$\kappa(B_{\text{h1}})$	$\kappa(B_{\text{h2}})$	$\kappa(B_{\text{out}})$	$\|E\|_2$
Burgers, init (λ=10^{-3})	none	3.3e+04	7.3e+03	1.2e+04	1.1e+04	3.6e+03	2.4e+01
	layer scalar	3.7e+05	7.3e+03	1.2e+04	1.1e+04	3.6e+03	3.2e+03
	LANCE	3.1e+07	1.0e+00	1.2e+04	1.1e+04	1.0e+00	2.8e+03
	exact block diag.	4.4e+04	1.0e+00	1.0e+00	1.0e+00	1.0e+00	3.0e+00
Burgers, after 40 (λ=10^{-8})	none	7.5e+09	2.1e+09	2.8e+09	1.9e+09	7.0e+08	5.4e+01
	layer scalar	9.2e+10	2.1e+09	2.8e+09	1.9e+09	7.0e+08	3.9e+03
	LANCE	9.6e+12	1.0e+00	2.8e+09	1.9e+09	1.0e+00	3.5e+03
	exact block diag.	9.4e+09	1.0e+00	1.0e+00	1.0e+00	1.0e+00	3.0e+00
Burgers, low ν, init (λ=10^{-3})	none	3.3e+04	7.3e+03	1.2e+04	1.1e+04	3.5e+03	2.4e+01
	layer scalar	3.7e+05	7.3e+03	1.2e+04	1.1e+04	3.5e+03	3.2e+03
	LANCE	3.1e+07	1.0e+00	1.2e+04	1.1e+04	1.0e+00	2.8e+03
	exact block diag.	4.4e+04	1.0e+00	1.0e+00	1.0e+00	1.0e+00	3.0e+00
Burgers, low ν, after 40 (λ=10^{-8})	none	8.1e+09	2.3e+09	3.1e+09	2.0e+09	7.4e+08	5.8e+01
	layer scalar	9.9e+10	2.3e+09	3.1e+09	2.0e+09	7.4e+08	3.9e+03
	LANCE	3.9e+12	1.0e+00	3.1e+09	2.0e+09	1.0e+00	3.5e+03
	exact block diag.	1.0e+10	1.0e+00	1.0e+00	1.0e+00	1.0e+00	3.0e+00
Helmholtz-1D, init (λ=10^{-3})	none	3.6e+07	7.1e+06	9.3e+06	1.2e+07	7.2e+06	2.7e+04
	layer scalar	9.4e+08	7.1e+06	9.3e+06	1.2e+07	7.2e+06	4.4e+03
	LANCE	5.5e+10	1.0e+00	9.3e+06	1.2e+07	1.0e+00	4.1e+03
	exact block diag.	4.3e+07	1.0e+00	1.0e+00	1.0e+00	1.0e+00	3.0e+00
Helmholtz-1D, after 40 (λ=3×10^{-2})	none	1.6e+07	1.7e+06	3.8e+06	8.2e+06	2.7e+06	2.9e+05
	layer scalar	3.5e+08	1.7e+06	3.8e+06	8.2e+06	2.7e+06	3.2e+03
	LANCE	1.2e+10	1.0e+00	3.8e+06	8.2e+06	1.0e+00	2.9e+03
	exact block diag.	2.1e+07	1.0e+00	1.0e+00	1.0e+00	1.0e+00	3.0e+00
Poisson-2D, init (λ=10^{-3})	none	7.6e+03	1.6e+03	2.6e+03	2.5e+03	8.6e+02	5.4e+00
	layer scalar	5.5e+04	1.6e+03	2.6e+03	2.5e+03	8.6e+02	1.7e+03
	LANCE	3.6e+06	1.0e+00	2.6e+03	2.5e+03	1.0e+00	1.4e+03
	exact block diag.	1.0e+04	1.0e+00	1.0e+00	1.0e+00	1.0e+00	3.0e+00
Poisson-2D, after 40 (λ=10^{-8})	none	1.2e+10	3.3e+09	4.8e+09	3.5e+09	1.1e+09	1.1e+02
	layer scalar	1.7e+11	3.3e+09	4.8e+09	3.5e+09	1.1e+09	2.1e+03
	LANCE	6.8e+12	1.0e+00	4.8e+09	3.5e+09	1.0e+00	1.9e+03
	exact block diag.	1.4e+10	1.0e+00	1.0e+00	1.0e+00	1.0e+00	3.0e+00

Proposition 1. The within-block condition numbers under the layer scalar are identical to those without any preconditioner, to a maximum relative difference of 3.6×10^{-7} over the 32 combinations of problem, stage and layer, the residual being the rounding of eigenvalues at condition numbers near 10^9. A scalar per layer therefore leaves untouched exactly the quantity that limits a truncated solve inside a curvature-dense block, and that quantity is large. Inside the input block it is 7×10^3 on Burgers and 7×10^6 on Helmholtz-1D at initialisation, and after forty iterations, with the damping at its floor, it reaches 2×10^9 on Burgers and 3×10^9 on Poisson-2D.

Proposition 2. Under LANCE the diagonal blocks of the selected layers have condition number 1 with every eigenvalue within 2×10^{-5} of one, and under the exact block diagonal every block does. The off-block coupling norm is 1.4×10^3 to 4.1×10^3 under LANCE, 8 to 18% below the layer scalar, and 3.00 under the full block diagonal on every problem and at both stages, which is the largest value the off-block part of a positive semidefinite matrix with four unit diagonal blocks can have.

The global condition number. Under LANCE, $\kappa(B)$ is two to three orders of magnitude larger than under no preconditioner on every problem and at both stages, from 3.3×10^4 to 3.1×10^7 on Burgers at initialisation and from 1.6×10^7 to 1.2×10^{10} on Helmholtz-1D after forty iterations, while the captured fraction at the operating budget improves on every problem (Table 15). The two are compatible. By Proposition 3 the bulk of the spectrum of A_λ sits at λ, coming from $\ker J$, which unpreconditioned conjugate gradient never explores, so $\kappa(A_\lambda)$ describes a part of the problem that the solver never sees. Preconditioning rescales those inert directions and inflates κ without affecting what the iterates do at small K, and Remark 3 explains what it does at large K. We therefore report the captured fraction $q(d_K)/q(d^\star)$, which is the operational quantity, and treat κ as a diagnostic of the transformation rather than of convergence.

I TRUNCATION DEFICIT ACROSS BUDGETS AND PROBLEMS

Tables 15 and 16 extend the diagnostic of §5.3 to the nine PDEs of the campaign and six budgets, at width 64 after forty unpreconditioned iterations, which is the point at which the initial plateau has been left and curvature has developed. G is assembled by batched Jacobian–vector products as in Appendix D, and every entry is the ratio of the quadratic-model value at the K-th iterate to its maximum $q(d^\star)$. At initialisation the unpreconditioned solve captures 1.000 at $K = 10$ on every one of the nine problems, so the diagnostic is informative only once curvature has developed.

Table 15: Captured fraction $q(d_K)/q(d^\star)$ on the four primary problems, width 64, seed 0, after forty GN-CG iterations at the damping λ then in force. Figure 6 draws the $K \in \{5, 10, 20\}$ columns.

Problem	preconditioner	K=1	K=2	K=5	K=10	K=20	K=40
Burgers (ν=0.01/π) (λ=10^{-8})	none	0.000	0.000	0.007	0.059	0.751	0.898
	layer scalar	0.000	0.000	0.006	0.127	0.652	0.898
	LANCE	0.097	0.539	0.898	0.961	0.974	0.976
	exact block diag.	0.974	0.986	0.996	0.998	0.998	0.998
Burgers (ν=0.002/π) (λ=10^{-8})	none	0.000	0.000	0.004	0.028	0.533	0.755
	layer scalar	0.000	0.000	0.004	0.023	0.420	0.783
	LANCE	0.006	0.624	0.852	0.934	0.951	0.953
	exact block diag.	0.964	0.983	0.993	0.995	0.996	0.996
Helmholtz-1D (k=20) (λ=3×10^{-2})	none	0.000	0.004	0.010	0.025	0.854	0.996
	layer scalar	0.000	0.004	0.008	0.017	0.404	0.870
	LANCE	0.001	0.014	0.091	0.376	0.875	0.939
	exact block diag.	0.682	0.756	0.893	0.970	0.980	0.987
Poisson-2D (λ=10^{-8})	none	0.004	0.005	0.013	0.027	0.667	0.897
	layer scalar	0.004	0.006	0.013	0.059	0.669	0.879
	LANCE	0.038	0.062	0.384	0.975	0.990	0.995
	exact block diag.	0.996	0.997	0.998	0.999	1.000	1.000

The operating budget. After forty iterations the damping is at or within a factor of 1.5 of its floor of 10^{-8} on Burgers, low-ν Burgers and Poisson-2D and the transformed system is correspondingly ill-conditioned (Table 14). At $K = 10$ the unpreconditioned solve captures 0.03 to 0.06 on the four primary problems and LANCE 0.38 to 0.97, and at $K = 5$ the two differ by factors of 120 and 240 on the two Burgers problems, where the exact blocks act from $K = 2$ onwards. The exact block diagonal captures 0.68 to 0.996 from the first iteration. On the additional PDEs that any method solves the picture is the same, LANCE raising the captured fraction at $K = 10$ from 0.16 to 0.97 on Allen–Cahn and from 0.07 to 0.83 on Wave-1D, and on Helmholtz-2D, where no method solves the problem, from 0.16 to 0.43, against 0.97 to 1.00 for the full block diagonal. On Poisson-10D every preconditioner captures 1.000 from $K = 1$, because the loop has stalled at an iterate where the truncated solve is already exact, which is the situation of step 3 in Appendix M. On Poisson-5D the unpreconditioned solve captures 0.49 at $K = 2$ and 1.000 from $K = 10$, and the exact block

Table 16: The same diagnostic on the additional PDEs of Table 2.

Problem	preconditioner	K=1	K=2	K=5	K=10	K=20	K=40
Allen–Cahn (λ=10^{-8})	none	0.000	0.001	0.031	0.162	0.869	0.960
	layer scalar	0.000	0.001	0.031	0.257	0.767	0.962
	LANCE	0.014	0.035	0.852	0.967	0.983	0.984
	exact block diag.	0.987	0.993	0.998	0.999	0.999	0.999
Wave-1D (λ=3×10^{-7})	none	0.001	0.007	0.042	0.067	0.742	0.838
	layer scalar	0.001	0.008	0.037	0.078	0.700	0.819
	LANCE	0.003	0.021	0.166	0.832	0.945	0.976
	exact block diag.	0.986	0.995	0.998	0.999	0.999	1.000
Helmholtz-2D (k=10) (λ=6×10^{-1})	none	0.007	0.024	0.039	0.163	0.682	0.883
	layer scalar	0.011	0.030	0.064	0.372	0.647	0.881
	LANCE	0.011	0.031	0.075	0.432	0.681	0.902
	exact block diag.	0.762	0.872	0.937	0.968	0.983	0.993
Poisson-5D (λ=10^{-2})	none	0.045	0.492	0.987	1.000	1.000	1.000
	layer scalar	0.045	0.551	0.990	1.000	1.000	1.000
	LANCE	0.046	0.544	0.991	0.997	1.000	1.000
	exact block diag.	0.818	0.867	0.909	0.976	1.000	1.000
Poisson-10D (λ=10^{-8})	none	1.000	1.000	1.000	1.000	1.000	1.000
	layer scalar	1.000	1.000	1.000	1.000	1.000	1.000
	LANCE	1.000	1.000	1.000	1.000	1.000	1.000
	exact block diag.	1.000	1.000	1.000	1.000	1.000	1.000

diagonal is the slower one there, 0.82 at $K = 1$ and 0.98 at $K = 10$, which is the de-clustering of Remark 3 on a system with no deficit to recover.

Large budgets. By $K = 20$ the unpreconditioned solve captures 0.53 to 0.85 on the four primary problems against 0.88 to 0.99 for LANCE, and by $K = 40$ it captures 0.76 to 0.996 and overtakes LANCE on Helmholtz-1D, 0.996 against 0.94, while LANCE and the full block diagonal plateau between 0.94 and 1.00. Remark 3 gives the reason. The unpreconditioned iteration is confined to $\mathrm{range}(J^\top)$, an invariant subspace of dimension at most N on which the spectrum of A_λ has at most N distinct values and on which $d^\star$ lies, whereas a block preconditioner spreads the $P - N$ eigenvalues at λ into an interval and gives the iterates components along $\ker J$ that only lower q and must be cancelled by later iterations. The exact blocks therefore help at small K and hurt at large K, and the crossover in Table 4, between $K = 20$ and $K = 40$, is where the two effects balance.

The layer scalar. On Helmholtz-1D the layer scalar captures 0.017 at $K = 10$ against 0.025 for no preconditioner, whereas on Burgers it captures 0.13 against 0.06 and on Helmholtz-2D 0.37 against 0.16. Proposition 1 fixes the conditioning inside every block, but the scalars still reweight the layers against one another, and Remark 1 says that this can go either way. Both directions occur in the tables, and the Helmholtz-1D direction is the one behind the 8 to 11 times worse errors of the layer scalar in Tables 3 and 21.

J Comparison with other curvature representations

Section 5.4 compares seven representations of the curvature at widths 32 and 64. Table 17 repeats that table with two additions, the Kronecker-factored step and the median wall-clock time per run. This appendix records what each column implements, since several are stand-ins for published methods rather than the methods themselves.

What each column implements. All columns share the residual, the initialisation, the Armijo search and the damping adaptation of Algorithm 1, and differ only in the linear-algebra step. *GN-CG* is $K = 10$ unpreconditioned CG on Eq. (3). *CG to tolerance* runs the same matrix-free CG until the relative residual falls below 10^{-2} or 50 iterations are spent, which is the solve of GN-NG (Jnini et al., 2026) in its representational choice, exact products and a convergence criterion rather than a fixed budget, and is given up to five times the products of the other Krylov columns. *Dual CG* spends the same $K = 10$ iterations on the residual-space system $(JJ^\top + \lambda I)y = r$ and recovers $d = J^\top y$. Its Krylov space is the image under $J^\top$ of the dual one and therefore coincides with the primal Krylov space, but CG in the dual minimises the $(JJ^\top + \lambda I)$-norm of the dual error rather than maximising q, so at a fixed K its iterate is not the q-optimal one, and the table shows

Table 17: Table 3 with the Kronecker-factored step, damped exactly as described below, and the median wall-clock time per run over the problems and seeds of each block. Geometric mean over three seeds, 100 iterations, one A100.

Problem	GN-CG	CG to tol.	dual CG	layer scalar	LANCE	Kronecker step	block-diag. step	dense solve
width 32, $P = 2{,}209$ *to* $2{,}241$								
Helmholtz-1D	8.10e-03	4.71e-04	3.40e-01	6.27e-02	1.14e-03	1.05e+00	1.89e-02	1.66e-04
Burgers	2.63e-04	5.69e-05	3.94e-04	4.56e-04	2.09e-05	3.88e-02	1.51e-02	7.78e-06
Burgers, low ν	2.88e-04	5.38e-05	3.93e-04	5.12e-04	2.62e-05	4.12e-02	1.52e-02	1.17e-05
Poisson-2D	5.43e-05	1.34e-06	6.90e-05	7.32e-05	1.31e-05	6.07e-02	1.10e-04	1.27e-06
median time per run (s)	3	4	4	3	3	7	2	16
width 64, $P = 8{,}513$ *to* $8{,}577$								
Helmholtz-1D	1.84e-02	4.11e-03	2.53e-01	1.99e-01	6.32e-03	1.05e+00	8.98e-03	9.84e-05
Burgers	2.48e-04	3.67e-05	2.40e-04	2.52e-04	2.83e-05	7.72e-02	1.31e-02	8.50e-06
Burgers, low ν	2.37e-04	4.06e-05	2.14e-04	2.54e-04	2.72e-05	7.71e-02	1.28e-02	1.17e-05
Poisson-2D	3.65e-05	1.45e-06	4.00e-05	4.67e-05	2.21e-06	1.11e-01	7.37e-05	3.42e-07
median time per run (s)	4	5	4	3	3	11	17	190

it behind the primal solve on six of the eight rows, by up to 42 times on Helmholtz-1D, and within 10% of it on the two Burgers problems at width 64. D-NGD (Jnini & Vella, 2025b) solves the dual system to convergence with a Nyström preconditioner, and at these N the converged dual step is the exact damped Gauss–Newton step of the dense-solve column, so the column measures what a fixed budget spent in the dual costs and not the published method. *Layer scalar* and *LANCE* are as in §4. *Kronecker step* forms every exact block by the per-point pass, replaces each weight block by its nearest Kronecker product $A \otimes C$ in the Frobenius norm, the leading singular pair of the Van Loan rearrangement obtained by power iteration, solves $(A \otimes C + \lambda I)d = g$ through the eigendecompositions of A and C, solves the biases by the damped diagonal, and takes the result as the step. The factors of a positive semidefinite block are positive semidefinite (Van Loan & Pitsianis, 1993), which the self-test of Part 0 checks and which holds to rounding on every block of the campaign, the smallest factor eigenvalue being -10^{-13}. The nearest-Kronecker approximation itself is close in norm, with a relative Frobenius error of at most 0.03 at initialisation and 0.05 to 0.41 after forty iterations. Damping each factor by $\sqrt{\lambda}$ instead, as a Kronecker-factored implementation without factor rescaling would, adds a damping of order $\sqrt{\lambda}(\|A\| + \|C\|)$ to the block, 2.5×10^3 times λ on the input layer of Helmholtz-1D at initialisation, and gives relative errors of 0.07 to 1 on every row. With the exact damping the column reads 0.04 to 1.05, so the damping convention is not the cause and the failure is that of the approximation as a step. The column has the structure of the KFAC-PINN update (Dangel et al., 2024) but not its estimator of the factors, which builds them from forward and backward statistics of the operator's Taylor-mode derivatives rather than from the exact block, so it says nothing about the published method. *Block-diagonal step* solves every exact block and takes the result as the step, the limit of any Kronecker-factored method as its factors become exact. *Dense solve* assembles G by batched Jacobian–vector products and factorises $G + \lambda I$, the representation of energy natural gradient descent (Müller & Zeinhofer, 2023).

The cost argument. At width 32 the dense solve costs 16 seconds per run against 3 for LANCE, because assembling G is $\mathcal{O}(NP^2)$ at every iteration, and at width 64 the matrix alone is 580 MB in float64 with an $\mathcal{O}(P^3)$ factorisation per iteration, and the run takes 190 seconds against 3 for LANCE. The block-diagonal step factorises the two 4160×4160 hidden blocks at every iteration and takes 17 seconds at width 64. LANCE is unchanged at $\mathcal{O}(P + N) + \sum_{P_\ell \le P_{\max}} P_\ell^2$, which is $65^2 + 192^2$ entries at width 64.

The block-diagonal step. The same diagonal blocks appear in LANCE and in the block-diagonal step, and only their role differs. Used as the step, the block-diagonal solve discards the off-block coupling entirely, and Table 14 shows that this coupling is as large as it can be. Once every block is normalised to the identity, the off-block part has spectral norm 3.00, the maximum for four unit blocks, on every problem and at both stages. Used as a preconditioner inside a Krylov solve of the exact system, the same blocks improve the iterates while the coupling is still represented exactly through the products. The result is a step that is worse than the unpreconditioned $K = 10$ solve by 2.0 to 57 times on seven of the eight rows. The exception is Helmholtz-1D at width 64, where the step is 2 times better than the unpreconditioned solve and 1.4 times worse than LANCE. The full block diagonal of Table 15 is compatible with this, because it measures the captured fraction of the

exact quadratic model under block preconditioning and so describes the preconditioned solve and not a step that drops the coupling. A Kronecker-factored method that drops the coupling inherits the same limitation, with the Kronecker approximation on top of it.

Solving to a tolerance. The CG-to-tolerance column spends 13 to 35 products per step on average, 15, 18, 18 and 35 at width 32 on Helmholtz-1D, Burgers, low-ν Burgers and Poisson-2D and 13, 20, 20 and 31 at width 64, against the fixed 10 of the other Krylov columns, and the count grows over a run as the damping falls. With those products it is ahead of the fixed budget on every row, ahead of LANCE on Helmholtz-1D and Poisson-2D at both widths by 1.5 to 10 times, and behind LANCE on the two Burgers problems by 1.3 to 2.7 times. It is the strongest of the matrix-free alternatives, at a per-step cost that is not fixed. The matched-budget comparison is Table 18, where LANCE is ahead of the unpreconditioned solve at every budget up to $K = 20$.

The distance to the dense solve. The fraction of the logarithmic distance that LANCE closes, quoted in §5.4, is $(\log e_{\text{GN-CG}} - \log e_{\text{LANCE}})/(\log e_{\text{GN-CG}} - \log e_{\text{dense}})$ on the geometric means of Table 3. It is 0.50, 0.72, 0.75 and 0.38 at width 32 and 0.20, 0.64, 0.72 and 0.60 at width 64 on Helmholtz-1D, Burgers, low-ν Burgers and Poisson-2D.

Per seed. LANCE is ahead of the unpreconditioned loop on every row of Table 3 in geometric mean and on 22 of the 24 seeds. The two exceptions are one Poisson-2D seed at width 32, where it is behind by 1.8, and one Helmholtz-1D seed at width 64, behind by 1.3, the same seed as in Table 2.

K CG BUDGET AND HYPERPARAMETER ROBUSTNESS

All sweeps in this appendix are at full scale, width 64, 100 iterations, ten seeds, one knob varied at a time around the defaults of Appendix D. Every table entry is the geometric mean over seeds of the final relative L^2 error.

Krylov budget. Table 18 repeats Table 4 to three significant figures, from which the ratios quoted in §5.5 can be recovered. The per-step cost is linear in K, so at a fixed iteration count the $K = 40$ column uses four times the products of the $K = 10$ column, and the comparison at fixed wall-clock would move further in favour of the intermediate budgets, where LANCE at $K = 5$ is already ahead of GN-CG at $K = 10$ by 2.2, 11 and 12 times at a lower per-step cost, the pass costing less than the five saved products. And the $K = 40$ column, where GN-CG is ahead by 16, 11 and 41% on Helmholtz-1D, Burgers and Poisson-2D, is the large-budget regime of Table 15 in which the unpreconditioned solve has captured 0.76 to 0.996 of the attainable decrease on its own and the de-clustering of Remark 3 is the remaining effect.

Table 18: Table 4 to three significant figures. Final relative L^2 error against the Krylov budget K at 100 iterations, geometric mean over ten seeds, width 64.

Problem		K=2	K=5	K=10	K=20	K=40
Helmholtz-1D	GN-CG	9.99e-01	8.74e-01	2.64e-02	1.46e-03	**2.34e-04**
	LANCE	**8.37e-01**	**1.19e-02**	**4.35e-03**	**4.91e-04**	2.70e-04
Burgers	GN-CG	4.22e-02	4.64e-03	3.69e-04	7.32e-05	**1.68e-05**
	LANCE	**1.14e-02**	**3.27e-05**	**1.93e-05**	**1.90e-05**	1.87e-05
Poisson-2D	GN-CG	7.48e-04	1.87e-04	3.10e-05	5.71e-06	**1.14e-06**
	LANCE	**3.60e-04**	**2.57e-06**	**2.21e-06**	**2.06e-06**	1.61e-06

Damping, threshold and block-size limit. Tables 19 to 21 vary the three remaining knobs.

Initial damping. Varying λ_0 over four orders of magnitude changes the final error by a factor of 2.3 on Helmholtz-1D with no systematic direction. On Burgers the error rises with λ_0, from 5.3×10^{-6} at 10^{-5} to 2.8×10^{-5} at 10^{-2}, so the default 10^{-3} leaves a factor of 3.6 on that problem, and on Poisson-2D $\lambda_0 = 10^{-5}$ is 8.5 times worse than the default while the other four settings agree within 2.3. The adaptation of Algorithm 1 lowers λ by at most a factor of 0.7 per accepted full step,

Table 19: Sensitivity to the initial damping λ_0, geometric mean over ten seeds.

Problem	1e-5	1e-4	1e-3	1e-2	1e-1
Helmholtz-1D	2.8e-03	1.9e-03	4.3e-03	2.9e-03	2.2e-03
Burgers	5.3e-06	9.9e-06	1.9e-05	2.8e-05	2.2e-05
Poisson-2D	1.9e-05	5.1e-06	2.2e-06	4.0e-06	2.9e-06

Table 20: Sensitivity to the density threshold ρ_{th}, geometric mean over ten seeds. A threshold above the density of a small layer removes it from S_t.

Problem	0.5	1.0	2.0	4.0
Helmholtz-1D	2.0e-03	4.3e-03	5.1e-03	1.4e-02
Burgers	1.9e-05	1.9e-05	2.1e-05	2.8e-05
Poisson-2D	2.3e-06	2.2e-06	2.8e-06	9.4e-06

so a large λ_0 lengthens the initial phase of over-damped steps by a dozen or more iterations out of a hundred.

Density threshold. Thresholds of 0.5 and 1.0 agree within 15% on Burgers and Poisson-2D. On Helmholtz-1D the lower threshold halves the error, because it keeps the input layer in S_t to the last iteration on every seed, where the default has dropped both layers by then on four of ten seeds. Raising the threshold to 2.0 and 4.0 removes the Helmholtz-1D input layer at the last iteration on seven and eight of ten seeds, and the error rises by 1.2 and 3.3. Burgers changes by at most 1.5, and Poisson-2D rises by 1.2 at 2.0 and by 4.2 at 4.0, where the input layer has left S_t on five seeds. The rule degrades as soon as the threshold removes a layer whose density still exceeds the others', and the Helmholtz-1D result says that on the problem whose densities equalise during training the default of one is slightly too high.

Block-size limit. The limit isolates the effect of each block. At $P_{\max} = 64$ no layer is eligible and LANCE is the layer scalar, which on Helmholtz-1D is 9.6 times worse than the unpreconditioned solve at the same budget (Table 18). At $P_{\max} = 128$ both small layers of Helmholtz-1D enter and its error falls by a factor of 58, while on the two-input problems only the output layer enters and the error falls by 2.3 and 1.3. At $P_{\max} = 256$ their input layer enters and the error falls by a further 9.2 and 10. Each exact block is individually responsible for a measurable part of the improvement and the input block carries most of it. At $P_{\max} = 8192$ every layer is eligible, and the rule admits a hidden block once its density crosses one, which by the last iteration has happened on six Helmholtz-1D seeds, nine Poisson-2D seeds and one Burgers seed (Appendix F). The error falls by a further 2.6 on Helmholtz-1D and 1.2 on Poisson-2D and is unchanged on Burgers, at a factorisation cost of $\mathcal{O}(4160^3)$ per admitted hidden block per iteration that is no longer negligible, and that a cheap approximation of the hidden blocks would be needed to avoid.

L Scaling in width, depth and input dimension

Two limits of the method are named in §6. The exact blocks help only where the small layers hold the curvature, and for PDEs with many inputs the first layer is no longer small. Both are measurable, and the full-scale campaign measures the first up to $P = 2.6 \times 10^5$.

Width and depth. Figure 10b shows the premise of the method at every size tested. As the width grows from 64 to 256 the two small layers shrink from 2.3 to 3% of the parameters to 0.6 to 0.8%, and to 0.3 to 0.4% at depth 5, while their share of the curvature energy stays at 34 to 42% for three hidden layers and their densities reach 81 and 108 at width 256 with five hidden layers, so the concentration strengthens with width and the cost of the exact blocks stays fixed at $\mathcal{O}(P_{\mathrm{in}}^2 + P_{\mathrm{out}}^2)$. Depth dilutes the share, to 22 to 27% at five hidden layers and 14 to 20% at eight, because the additional hidden layers take a share of the trace, and at depth 8 the largest hidden-layer density reaches 1.0 to 1.2. The density ordering between the small layers and the hidden layers is

Table 21: Sensitivity to the block-size limit $P_{\max}$, geometric mean over ten seeds. The input layer has 128 parameters on Helmholtz-1D and 192 on the two-input problems, the output layer 65, and the hidden layers 4,160, so the columns admit the layers one at a time. At $P_{\max} = 8192$ every layer is eligible.

Problem	64	128	256	512	8192
Helmholtz-1D	2.5e-01	4.3e-03	4.3e-03	4.3e-03	1.7e-03
Burgers	4.2e-04	1.8e-04	1.9e-05	1.9e-05	2.0e-05
Poisson-2D	3.0e-05	2.2e-05	2.2e-06	2.2e-06	1.9e-06

Table 22: Scaling in width and depth at full scale, geometric mean over three seeds, 100 iterations at $K = 10$. P_{sm}/P is the fraction of the parameters in the input and output layers, E_{sm} their share of $\mathrm{tr}(G)$ at initialisation and $\max_{\mathrm{hid}} \rho_\ell$ the largest hidden-layer density, all averaged over seeds. The ratio is the GN-CG error divided by the LANCE error, and the time column the median LANCE to GN-CG wall-clock ratio.

Problem	**width, depth,** N	P	P_{sm}/P	E_{sm}	$\max_{\mathrm{hid}} \rho_\ell$	GN-CG	LANCE	**ratio**	**time**
Helmholtz-1D	64, 3, 4096	8513	2.3%	39%	0.68	1.97e-02	9.44e-03	2.1×	1.13×
	128, 3, 4096	33409	1.2%	39%	0.63	2.87e-02	6.07e-03	4.7×	1.11×
	256, 3, 4096	132353	0.6%	42%	0.60	3.99e-02	8.29e-03	4.8×	1.19×
	64, 5, 4096	16833	1.1%	26%	0.85	1.15e-02	2.83e-03	4.0×	1.06×
	128, 5, 8192	66433	0.6%	25%	0.92	1.70e-02	4.15e-03	4.1×	1.19×
	256, 5, 16384	263937	0.3%	26%	0.82	6.77e-02	1.16e-02	5.8×	1.25×
	64, 8, 4096	29313	0.7%	20%	1.03	5.19e-03	1.75e-03	3.0×	1.05×
Burgers	64, 3, 4096	8577	3.0%	41%	0.64	2.23e-04	2.44e-05	9.1×	1.13×
	128, 3, 4096	33537	1.5%	34%	0.69	3.90e-04	1.85e-05	21.1×	1.19×
	256, 3, 4096	132609	0.8%	37%	0.68	4.11e-04	5.93e-05	6.9×	1.22×
	64, 5, 4096	16897	1.5%	27%	0.94	2.13e-04	1.03e-05	20.7×	1.12×
	128, 5, 8192	66561	0.8%	22%	0.84	3.97e-04	1.48e-05	26.8×	1.29×
	256, 5, 16384	264193	0.4%	24%	0.84	2.43e-04	4.70e-05	5.2×	1.32×
	64, 8, 4096	29377	0.9%	14%	1.15	2.30e-04	8.62e-06	26.7×	1.16×

nevertheless preserved in every configuration, so the rule selects the same two layers at initialisation at every size.

The advantage at scale. LANCE is ahead of GN-CG on all fourteen configurations, by 2.1 to 27 times, including both problems at $P = 2.6 \times 10^5$ with $N = 16{,}384$, by 5.8 on Helmholtz-1D and 5.2 on Burgers, where its wall-clock overhead is 1.25 and 1.32. The smallest gain, 2.1, is Helmholtz-1D at width 64 with three hidden layers, where the per-seed ratios of Table 2 range from 0.77 to 49 and three seeds do not pin the geometric mean down. On Burgers the largest gains are at the deepest and at the third-largest configurations, 27 at depth 8 and 27 at $P = 6.7 \times 10^4$, so the effect does not decay with size in either direction.

Input dimension. Table 23 shows the opposite direction, in three distinct regimes. The input layer grows linearly in d while its share of the trace does not, so ρ_{in} at initialisation falls from 12.6 at $d = 1$ to 2.7 at $d = 10$ while ρ_{out} stays between 19 and 25. During training the rule follows the measurement. At $d \leq 2$ the input layer is in S_t at every iteration, at $d = 3$ at 94% of them and at $d = 5$ at 78%. At $d = 10$ the input layer has 704 parameters, above $P_{\max} = 512$, and is never eligible, so LANCE preconditions the output layer alone from the start. Raising $P_{\max}$ to 1024 admits it at 8% of the iterations, and the three seeds end at the same error to four digits (Table 23). The rule removes a block once the measurement says the block no longer carries disproportionate curvature, or once the block is too large to factorise, whatever the block's position in the network.

Gains by dimension. At $d = 2$, 3 and 5 LANCE wins, by 21 times at $d = 2$ and 1.6 at $d = 3$. At $d = 5$ the loop struggles without stalling. GN-CG accepts the full step at 20 to 42% of the iterations, ends with the damping between 4.9 and its cap of 10, and stops at training losses of 1.9 to 2.5 on every seed, whereas LANCE leaves that regime on one of the three seeds and reaches a loss of 6.4×10^{-3} and an error of 3.2×10^{-3}, the same pattern as the three escaping seeds of Table 2. At $d = 10$ both loops stop at the same iterate on every seed, at training losses of 0.25 to 0.32 that are far from zero, with the full step accepted at every iteration, the damping at its floor of 10^{-8}

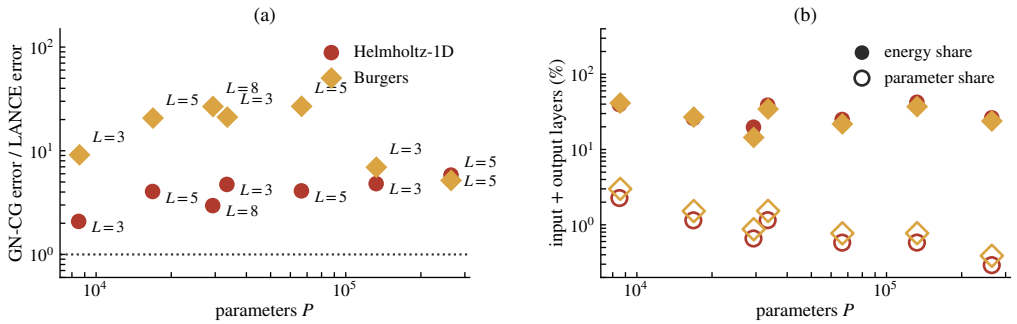


Figure 10: The data of Table 22. (a) GN-CG to LANCE error ratio against P, labelled by the number of hidden layers L. (b) The curvature-energy share (filled markers) and the parameter share (hollow markers) of the input and output layers against P, with the same marker per problem as in (a).

Table 23: Input-dimension scaling on Poisson-dD at width 64, $N = 4{,}096$, 100 iterations at $K = 10$, geometric mean over three seeds. P_{in} is the size of the input layer, which grows linearly in d while its curvature density at initialisation falls. The last two columns are the fraction of LANCE iterations at which the input layer is in S_t and the fraction of iterations at which the full step is accepted, for GN-CG and for LANCE. The last row repeats $d = 10$ with the input layer made eligible.

d	P	P_{in}	ρ_{in}	ρ_{out}	GN-CG	LANCE	input in S_t	full steps
1	8513	128	12.58	24.50	2.99e-07	8.69e-07	100%	77% / 61%
2	8577	192	12.07	18.67	4.28e-05	2.01e-06	100%	98% / 62%
3	8641	256	6.90	21.35	2.47e-04	1.55e-04	94%	79% / 60%
5	8769	384	4.80	20.43	1.09e-01	3.32e-02	78%	29% / 40%
10	9089	704	2.69	21.28	1.13e-01	1.13e-01	0%	100% / 100%
10, $P_{\max}$=1024	9089	704	2.69	21.28	1.13e-01	1.13e-01	8%	100% / 100%

and the unpreconditioned solve capturing 1.000 of the attainable decrease at $K = 1$ (Table 16), so there is neither a truncation deficit nor any progress for a preconditioner to act on. The optimiser has stopped at a poor stationary point of the residual, a documented pathology of PINNs in higher dimension.

At $d = 1$ there is nothing to fix. On the one-dimensional Poisson problem GN-CG reaches 3.0×10^{-7} and LANCE 8.7×10^{-7}, so the preconditioner costs a factor of three. GN-CG accepts the full step at 66 to 84% of the iterations and drives the damping to its floor, so its ten unpreconditioned iterations already capture the step, which is the same phenomenon as the initialisation diagnostic of §5.3. When the truncation deficit is negligible, reshaping the Krylov space can only perturb a solve that was already accurate.

Together with Appendix J, this evidence marks out where the method applies. LANCE helps when the truncation deficit is large and concentrated in small layers, which is the convection-dominated and oscillatory regime where PINN training is hardest. It is mildly harmful when the deficit is small, which is the easy elliptic regime where any reasonable optimiser succeeds, and it cannot help where the Gauss–Newton loop has settled at a stationary point. The deficit and the density are both measurable before committing to a long run.

M When to expect LANCE to help

The evidence in Appendices J to L is two-sided. LANCE improves on the unpreconditioned loop by an order of magnitude and more on some problems, ties on others, and is behind at large budgets and where the budget was not limiting the step. Because the governing quantities are measurable before a long run, a practitioner can learn something about which case they are in without trying it. The following procedure costs one per-point pass plus one dense diagnostic on a proxy network.

1. *The loop.* Run a few tens of GN-CG iterations. If the error stops falling far from the accuracy of interest while the full step is accepted at every iteration and the damping sits at its floor, the loop has stalled at a stationary point of the residual, as on Poisson-10D in Table 2, and no preconditioner of the inner solve will change that. A loop that keeps rejecting the full step with the damping at its cap, as on Poisson-5D, has not settled, and there the preconditioner made the difference on three of five seeds.
2. *The concentration.* Run the per-point gradient pass at initialisation and compute ρ_ℓ from Eq. (4). If no layer with $P_\ell \le P_{\max}$ has $\rho_\ell > 1$, S_t is empty and LANCE reduces to the layer scalar, which Tables 3 and 21 show to be no better than nothing and sometimes far worse. Every configuration in this paper passes this test at initialisation, the closest call being Poisson-10D at width 32 with $\rho_{\text{in}} = 1.3$, and the input layer's density falls with the input dimension while its size grows past $P_{\max}$, so it is the test most likely to fail on high-dimensional problems.
3. *The deficit at the intended budget.* Train for a few tens of iterations with GN-CG, assemble G on a proxy network small enough to allow it, and compute $q(d_K)/q(d^\star)$ for the unpreconditioned solve. If it is already above about 0.7 at the intended K, the budget is not what limits the step and there is little for any preconditioner to recover. This is the $d = 1$ Poisson case of Table 23, it is what the initialisation diagnostic of §5.3 reports at iteration zero on every problem, and it is what Table 16 reports after forty iterations on Poisson-10D, where the stalled loop leaves a captured fraction of 1.000 at $K = 1$. The test is taken along the unpreconditioned trajectory and can miss a gain that comes from leaving that trajectory. On Poisson-5D it also reports 1.000 at $K = 10$, yet LANCE, whose iterates differ from the first step, leaves the stall on three of five seeds.
4. *The part of the deficit that blocks can reach.* Compute the same fraction under the LANCE preconditioner and under the exact block diagonal over all layers. The gap between the two is the part of the deficit that lives in the hidden blocks and that LANCE does not address, and it says where that part is. It does not predict the size of the end-to-end effect. Helmholtz-1D, where LANCE captures 0.38 against 0.97 for the full block diagonal, gains 7.2 times, and Burgers, where it captures 0.96 against 1.00, gains 18, but Wave-1D at 0.83 against 1.00 gains 4.5. How a gain in one iterate's captured fraction compounds over a run is not something a one-iterate diagnostic can tell.

Applied to the nine problems of Tables 15 and 16, the procedure sorts Poisson-10D out at steps 1 and 3 and, wrongly, Poisson-5D at step 3. Ranking the remaining seven by the captured fraction that the two small blocks recover at $K = 10$ puts the five on which they recover more than 0.8 of the attainable decrease, Poisson-2D, Allen–Cahn, Burgers, low-ν Burgers and Wave-1D, at gains of 4.5 to 25 times in final error, and the two on which they recover 0.33 to 0.37, Helmholtz-2D and Helmholtz-1D, at 1.2 and 7.2. The two ranges overlap, Helmholtz-1D gaining more than Wave-1D from less than half the recovered fraction, and within a group the order is not reproduced either. The diagnostic identifies the regime and says where the remaining deficit lies, and it does not rank the problems within it. Steps 1 and 2 are free, since the per-point pass runs at every LANCE iteration anyway. Steps 3 and 4 need a dense G and are therefore restricted to a proxy network, which assumes that the ordering transfers across width. Table 22 shows that the concentration does and that the improvement does not do so monotonically.

N NEGATIVE RESULTS

This appendix records the variants that did not work, because the reasons are informative and because two of them are the natural first things to try, together with one convention that makes no difference. All are measured by the released campaign, the three of Table 24 on the four primary problems over five seeds with the protocol of Table 2.

Layer-wise Levenberg–Marquardt damping. Replacing the scalar λ by a per-layer $\lambda_\ell = \lambda\rho_{t,\ell}$, which damps the curvature-dense layers more, leaves low-ν Burgers within 21% of LANCE, is 1.7 times worse on Burgers, and raises the error by 2.8 times on Helmholtz-1D and 15 times on Poisson-2D, where it keeps only a factor of 2.5 and 1.1 of LANCE's 7.2 and 16 over the unpreconditioned loop. Figure 2b shows why. Damping shortens the step towards the gradient direction, which is the

Table 24: Final relative L^2 error of three variants of the LANCE loop, geometric mean over five seeds, alongside GN-CG and LANCE from the same runs. Layer-wise damping replaces λI by $\lambda\rho_{t,\ell}I$ on layer ℓ, identity outside S_t replaces $(\mu_{t,\ell}+\lambda)I$ by I on the unselected layers, and the doubled step multiplies the trial step by two, which is the convention in which the loss is $\|r\|^2$ with the same system matrix.

Problem	GN-CG	LANCE	layer-wise damping	identity outside S_t	doubled step
Burgers (ν=0.01/π)	3.26e-04	1.85e-05	3.09e-05	2.44e-05	2.09e-05
Burgers (ν=0.002/π)	1.15e-04	1.30e-05	1.58e-05	1.70e-05	1.20e-05
Helmholtz-1D (k=20)	2.54e-02	3.50e-03	9.96e-03	9.85e-02	2.75e-03
Poisson-2D	3.57e-05	2.18e-06	3.38e-05	2.26e-06	3.36e-06

correct response to Gauss–Newton *model* error, whereas what the small layers carry is truncation deficit. Damping them more removes progress rather than model error.

Dropping the mean-diagonal scaling of the hidden layers. Replacing $(\mu_{t,\ell}+\lambda_t)I$ by I on the layers outside S_t is 1.0 to 1.3 times worse than LANCE on Burgers, low-viscosity Burgers and Poisson-2D and 28 times worse on Helmholtz-1D, where it ends four times behind the unpreconditioned loop, so the scaling stays.

The doubled step. Multiplying the trial step by two, so that the unit step is twice the Gauss–Newton step and the first halving of the line search returns it, which is what solving Eq. (3) for the loss $\|r\|^2$ amounts to, changes the LANCE error by between 0.79 and 1.54 times, better on Helmholtz-1D and low-viscosity Burgers and worse on Burgers and Poisson-2D, so the results do not depend on this convention.

Layer-averaged Jacobi preconditioning over all layers. Setting $M_t = \mathrm{blkdiag}((\mu_{t,\ell}+\lambda_t)I)$ over every layer is the $P_{\max} = 64$ column of Table 21 and the layer-scalar column of Table 3. On Helmholtz-1D it is 9.6 to 11 times worse than no preconditioner at width 64 and 8 times worse at width 32, and on the other problems it is within a factor of 1.8 of no preconditioner and better than it by at most 4%. Proposition 1 explains both observations. The preconditioner cannot touch the conditioning inside any block, and its only effect is to reweight layers against one another, which by Remark 1 can go either way, and on Helmholtz-1D goes the wrong way (Table 15).

The blocks as a step. The exact blocks used as a step instead of as a preconditioner are worse than the unpreconditioned loop on every problem at width 32 (Appendix J), which is the negative result behind the framing of the paper.

O SOFT CONSTRAINTS

Every other experiment imposes the boundary and initial conditions by the hard-constraint ansatz of Appendix E. Table 25 repeats the width-64 comparison of GN-CG and LANCE on Helmholtz-1D and Poisson-2D with the conditions imposed by a penalty instead. The network output is used directly as u_θ, and the residual stacks the PDE residual on the N interior points with $w_b\,(u_\theta - u^\star)$ on N_b boundary points, the two end points on Helmholtz-1D and 256 points spread over the four sides of the square on Poisson-2D, each part scaled by the square root of its own count. The layer statistics of Eq. (4) are taken on that stacked residual, and the protocol is otherwise that of Table 2, over three seeds.

The layer shares do not move. Adding the boundary rows changes no share by more than 0.007 and the densities of the two end layers by at most 0.5, at either weight. On Poisson-2D the penalty formulation is harder for both loops and the gain of the preconditioner grows, from 18 times under hard constraints to 79 times at $w_b = 10$ and 210 times at $w_b = 1$. On Helmholtz-1D at $k = 20$ neither loop leaves a relative error of 0.5 to 0.9 within the hundred iterations at either weight, so the penalty formulation stalls both loops there, as Helmholtz-2D does in Table 2.

Table 25: Hard against soft constraints, width 64, three seeds, 100 iterations at $K = 10$. Layer shares and densities at initialisation, geometric-mean final errors and the geometric mean of the per-seed GN-CG to LANCE ratio.

Problem	**constraint**	E_ℓ (in, h_1, h_2, out)	ρ_{in}	ρ_{out}	GN-CG	LANCE	**ratio**
Helmholtz-1D	hard	0.21, 0.27, 0.33, 0.18	14.1	23.8	1.84e-02	6.32e-03	2.9×
	soft, w_b=1	0.21, 0.27, 0.33, 0.18	13.9	24.0	8.19e-01	7.91e-01	1.0×
	soft, w_b=10	0.21, 0.27, 0.33, 0.18	13.9	24.0	8.16e-01	6.70e-01	1.2×
Poisson-2D	hard	0.27, 0.31, 0.28, 0.14	12.1	18.7	3.69e-05	2.01e-06	18×
	soft, w_b=1	0.28, 0.31, 0.27, 0.14	12.4	18.4	6.94e-02	3.31e-04	210×
	soft, w_b=10	0.27, 0.31, 0.27, 0.14	12.2	18.8	4.43e-03	5.59e-05	79×

P THE FULL-SCALE CAMPAIGN

Every number, table and figure in the paper was produced by a single self-contained notebook released with the supplementary material, run on one A100 on Google Colab, together with two scripts that turn its result files into the LaTeX fragments and figures used here without manual editing. The notebook is organised in thirteen parts, and its result files are included in the supplement so that every table can be regenerated without rerunning it. The measurements of Appendix C.4 come from a companion notebook whose training code is copied from this one without change and which is released alongside it with its own result files.

Part 0, self-test. On seven problems at $P \approx 200$ it checks the matrix-free product, the per-point diagonal and blocks, the gradient, the nearest-Kronecker routine and the definiteness of its factors, and preconditioned CG run to $K = P$ against dense references, and checks that the quadratic model is monotone in K (Appendix E).

Part 1, reproduction and diagnostics. The four primary configurations of Table 2 over five seeds and four optimisers, the primitive costs of Table 9, and the dense diagnostics of Figure 6 and Table 15, with G assembled by batched Jacobian–vector products. Runtime about twenty-five minutes.

Part 2, curvature representations. The Krylov and replacing representations of Table 17, at widths 32 and 64 over three seeds, from shared initialisations with a shared damping schedule and line search. Runtime about fifteen minutes.

Part 3, scaling. Widths 64, 128 and 256 and depths 3, 5 and 8 on Helmholtz-1D and Burgers, taking P from 8.5×10^3 to 2.6×10^5 with N from 4,096 to 16,384, three seeds, recording the layer statistics, the final error and the wall-clock (Table 22). Runtime about one hour, of which the two largest configurations take most.

Part 4, additional PDEs and robustness. Allen–Cahn, the wave equation, Helmholtz-2D and Poisson in five and ten dimensions over five seeds and four optimisers (the lower half of Table 2), and the sweeps of Appendix K over ten seeds, 24 runs per seed and problem. Runtime about one and three-quarter hours.

Part 5, the remaining studies. The curvature concentration of Table 13 on nine PDEs at widths 32 to 128 and depths 3 and 5, the block-level spectral diagnostics of Table 14 at $P = 8.5 \times 10^3$, the captured fraction on the additional PDEs (Table 16), Poisson in $d \in \{1, 2, 3, 5, 10\}$ with the selected set, the step length and the damping logged at every iteration (Table 23), the negative results of Table 24, the selected set and the CG-to-tolerance iteration counts logged at every iteration on the primary problems (Appendices F and J), and the dense Gauss–Newton matrix of the 593-parameter network of Figure 2a. Runtime about half an hour.

Part 6, trained solutions. The four optimisers of Table 2 on one seed of Helmholtz-1D, with the trained networks evaluated on the test grid for Figure 5. Runtime about one minute.

Parts 7 to 11, additions. The block-diagonal step and the dense solve at width 64 (Table 3) and the definiteness and Frobenius error of the nearest-Kronecker factors on every block (Part 7, about

forty minutes), Poisson-10D with the input layer eligible (Part 8, one minute), L-BFGS for $1{,}500$ iterations on the nine problems with the error traced every ten iterations, from which the $1{,}000$-iteration column of Table 2 is read (Part 9, about thirty-five minutes), the soft-constrained runs of Appendix O (Part 10, five minutes), and the Kronecker step with exact damping (Part 11, five minutes).

Part 12, trained solutions on every problem. The four optimisers of Table 2 on seed 0 of the eight problems other than Helmholtz-1D, with the budgets of Appendix E, evaluated on the test grids of Appendix E and, for Poisson-5D and Poisson-10D, on a 101×101 grid of the (x_1, x_2) plane at $x_3 = \cdots = x_d = 0.5$, for Figures 8 and 9 and Table 12. Runtime about fifteen minutes.

Because all problems employ hard constraints, the campaign trains a single PDE residual using one software stack. We then compare ENGD, KFAC-PINN, GN-NG and D-NGD, using our implementations of the curvature representations for that residual (Appendix J).