FastLSQ: Solving PDEs in One Shot via Fourier Features with Exact Analytical Derivatives

Antonin Sulc
Lawrence Berkeley National Laboratory, Berkeley, U.S.A.
asulc@lbl.gov

Abstract

We present FastLSQ, a framework for PDE solving and inverse problems built on trigonometric random Fourier features with exact analytical derivatives. Trigonometric features admit closed-form derivatives of any order in $\mathcal{O}(1)$, enabling graph-free operator assembly without autodiff. Linear PDEs: one least-squares call; nonlinear: Newton–Raphson reusing analytical assembly. On 17 PDEs (1–6D), FastLSQ achieves 10^{-7} in 0.07 s (linear) and 10^{-8}–10^{-9} in <9 s (nonlinear), orders of magnitude faster and more accurate than iterative PINNs. Analytical higher-order derivatives yield a differentiable digital twin; we demonstrate inverse problems (heat-source, coil recovery) and PDE discovery. Code: github.com/sulcantonin/FastLSQ; `pip install fastlsq`.

1 Introduction

Solving partial differential equations (PDEs) is a cornerstone of computational physics, with applications from fluid dynamics and electromagnetism to quantum mechanics and climate modeling. Classical numerical methods, finite differences, finite elements, and spectral methods, remain the workhorses of scientific computing, but face challenges in high dimensions and can require substantial problem-specific implementation effort.

The emergence of physics-informed neural networks (PINNS) (Raissi et al., 2019) offered a mesh-free alternative that parametrizes the PDE solution as a neural network and minimizes a physics-based loss via stochastic gradient descent. While conceptually elegant, PINNs require minutes to hours of iterative training (Zhongkai et al., 2024), suffer from spectral bias (Tancik et al., 2020), causality violations (Wang et al., 2022), and sensitivity to loss balancing.

Random feature methods (RFMs) offer a middle ground. By approximating the solution as a linear combination of randomly sampled basis functions with frozen parameters, RFMs reduce the problem to fitting a linear output layer. For linear PDEs, this yields a linear system in the output coefficients for any choice of basis, as noted by PIELM (Dwivedi & Srinivasan, 2020) (using $\tanh$) and RF-PDE (Liao, 2024). However, RF-PDE still requires between 600 and 2000 epochs of iterative optimization. One-shot methods like PIELM use $\tanh$ activations, which lack a closed-form cyclic derivative structure; PIELM therefore relies on manual, problem-specific symbolic calculus to assemble $\mathcal{L}[\tanh]$ for each PDE operator.

Key observation. While any frozen-feature model yields a linear system for linear PDEs, the practical bottleneck is assembling the PDE operator matrix $A_{ij} = \mathcal{L}[\phi_j](\mathbf{x}_i)$. For sinusoidal features $\phi_j(\mathbf{x}) = \sin(\mathbf{W}_j \cdot \mathbf{x} + b_j)$, we show that $\mathcal{L}[\phi_j]$ admits an exact closed-form expression for any linear differential operator of any order, requiring only $\mathcal{O}(1)$ operations per entry and no automatic differentiation. This is a consequence of the cyclic derivative structure of sinusoids: the n-th derivative cycles through $\{\sin, \cos, -\sin, -\cos\}$ with a monomial weight prefactor. Alternative bases such as $\tanh$ lack any comparable closed-form pattern (Proposition 2.2), so methods like PIELM require manual, problem-specific derivation of $\mathcal{L}[\tanh]$ for each new PDE operator; there is no universal formula. Fast-LSQ's operator-agnostic advantage is that one formula (2) applies to any $\mathcal{L}$.

Figure 1 illustrates the key architectural differences. Both PIELM and Fast-LSQ are one-shot solvers that reduce to a single least-squares call, so neither involves iterative training. The distinction lies in how the operator matrix $\mathbf{A}$ is assembled: PIELM requires handwritten, problem-specific calculus for each PDE operator because $\tanh$ lacks a closed-form pattern, while Fast-LSQ uses a single operator-agnostic formula (2) (§2.2). The practical consequences are: (1) **operator-agnostic** assembly—one formula for any $\mathcal{L}$, no manual derivation when changing PDEs; (2) **superior spectral accuracy**, with $10\times$ to $1000\times$ lower errors at equal feature counts (§3); (3) **high-fidelity gradient accuracy**—because derivatives of a sinusoidal basis are just phase-shifted sinusoids scaled by

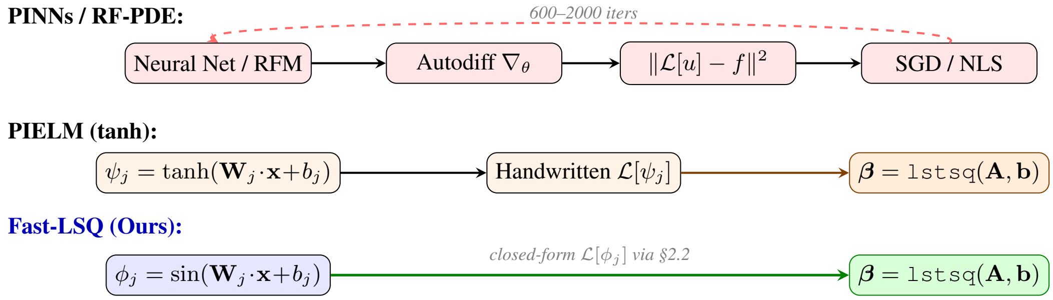
PINNs / RF-PDE:
600–2000 iters
Neural Net / RFM
Autodiff ∇_θ
$\|\mathcal{L}[u]-f\|^2$
SGD / NLS
PIELM (tanh):
$\psi_j=\tanh(\mathbf{W}_j\cdot\mathbf{x}+b_j)$
Handwritten $\mathcal{L}[\psi_j]$
$\boldsymbol{\beta}=\texttt{lstsq}(\mathbf{A},\mathbf{b})$
Fast-LSQ (Ours):
$\phi_j=\sin(\mathbf{W}_j\cdot\mathbf{x}+b_j)$
closed-form $\mathcal{L}[\phi_j]$ via §2.2
$\boldsymbol{\beta}=\texttt{lstsq}(\mathbf{A},\mathbf{b})$

Figure 1: Architecture comparison. PINNs and RF-PDE require iterative optimization (top row). PIELM and FAST-LSQ both solve a linear system in one shot, but PIELM requires handwritten, problem-specific calculus to assemble $\mathcal{L}[\tanh]$ for each PDE operator (middle row), whereas FAST-LSQ uses a single operator-agnostic formula for any $\mathcal{L}$ (bottom row).

$\mathcal{O}(\sigma)$, gradient errors remain tightly bounded (within one order of magnitude of value errors; §3.6); (4) **learnable bandwidth**—σ can be made differentiable for inverse problems (Appendix E); and (5) a rich set of **trigonometric symmetries** (eigenfunction property, orthogonality, product-to-sum) underpins both linear and nonlinear PDE solving (Appendix F).

We make three contributions: First, we observe that the elementary cyclic derivative structure of sinusoids (2) enables graph-free, closed-form assembly of the PDE operator matrix for arbitrary linear differential operators, and we build on this to construct FAST-LSQ: a one-shot solver that assembles the operator matrix analytically and obtains the solution via a single least-squares call (§2.3). Second, we extend FAST-LSQ to nonlinear PDEs via Newton–Raphson iteration, where each linearized step reuses the analytical assembly, achieving 10^{-8} to 10^{-9} accuracy in under 9 s (§2.4). Third, on 17 PDEs (5 linear, 5 nonlinear solver-mode, 7 nonlinear regression) in up to 6 dimensions, FAST-LSQ achieves 10^{-7} in under 0.1 s on linear problems and 10^{-8} on nonlinear problems, outperforming all baselines by large margins (§3). Furthermore, we run a comparison with RBF Kansa, scikit-fem P2 FEM, and scipy.solve_bvp on all 10 PDEs, demonstrating that FAST-LSQ is the only method applicable to every problem, including high-dimensional ($d{\geq}5$) and hyperbolic space-time equations for which conventional solvers have no applicable path.

2 Method

2.1 Random Fourier Feature Approximation

We approximate the PDE solution $u:\Omega\subset\mathbb{R}^d\to\mathbb{R}$ by

$$u_N(\mathbf{x})=\frac{1}{\sqrt{N}}\sum_{j=1}^{N}\beta_j\,\phi_j(\mathbf{x}),\qquad \phi_j(\mathbf{x})=\sin(\mathbf{W}_j^\top\mathbf{x}+b_j),\tag{1}$$

where $\mathbf{W}_j\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I}_d)$ and $b_j\sim\mathcal{U}(0,2\pi)$ are frozen at initialization and only the linear coefficients $\boldsymbol{\beta}=(\beta_1,\dots,\beta_N)^\top$ are trainable. The $1/\sqrt{N}$ prefactor ensures that the empirical kernel $\frac{1}{N}\sum_{j=1}^{N}\phi_j(\mathbf{x})\phi_j(\mathbf{x}')$ converges to the Gaussian RBF kernel $k(\mathbf{x},\mathbf{x}')=\exp(-\frac{\sigma^2}{2}\|\mathbf{x}-\mathbf{x}'\|^2)$ as $N\to\infty$ (Rahimi & Recht, 2007), keeping the coefficients β_j at $\mathcal{O}(1)$ magnitude and preventing the ill-conditioning that arises when unnormalized features force $\beta_j\sim10^6$–10^8. The bandwidth σ controls frequency content and the associated kernel length scale. To capture multiple scales, we use a multi-block architecture with B blocks at potentially different bandwidths σ_b, concatenating all features into a single vector solved simultaneously.

2.2 Exact Analytical Derivatives of Sinusoidal Features

The central structural property exploited by FAST-LSQ is the following. The multivariate form follows immediately from the chain rule and the cyclic derivatives of sin. For any multi-index $\alpha=(\alpha_1,\dots,\alpha_d)\in\mathbb{N}_0^d$ with $|\alpha|=\sum_k\alpha_k$:

$$D^\alpha\phi_j(\mathbf{x})=\left(\prod_{k=1}^{d}W_{jk}^{\alpha_k}\right)\cdot\Phi_{|\alpha|\bmod 4}(\mathbf{W}_j^\top\mathbf{x}+b_j),\tag{2}$$

where $\Phi_0=\sin$, $\Phi_1=\cos$, $\Phi_2=-\sin$, $\Phi_3=-\cos$. The identity itself is elementary; what matters is its consequence for PDE solving: for any linear differential operator $\mathcal{L}$ of any order, every entry of the operator matrix $A_{ij}=\mathcal{L}[\phi_j](\mathbf{x}_i)$ reduces to a single trigonometric evaluation multiplied by a monomial in the weights $\mathbf{W}_j$, computable without automatic differentiation or computational graph construction. This analytical formulation also enables a differentiable trick: the bandwidth σ

Algorithm 1 FAST-LSQ for Linear PDEs

```
Require: PDE operator $\mathcal{L}$, BC operator $\mathcal{B}$, source $f$, BC data $g$, domain $\Omega$
Require: Features $N$, bandwidth $\sigma$, collocation counts $M_1, M_2$, penalty $\lambda$
Sample frozen weights: $\mathbf{W}_j \sim \mathcal{N}(0, \sigma^2 I_d)$, $b_j \sim \mathcal{U}(0, 2\pi)$
Sample collocation points: $\{\mathbf{x}_i^{\text{int}}\} \subset \Omega$, $\{\mathbf{x}_i^{\text{bc}}\} \subset \partial\Omega$
Build PDE matrix: $A_{ij}^{\text{pde}} \leftarrow \mathcal{L}[\phi_j](\mathbf{x}_i^{\text{int}})$ via §2.2        ▷ Closed-form, no autodiff
Build BC matrix: $A_{ij}^{\text{bc}} \leftarrow \mathcal{B}[\phi_j](\mathbf{x}_i^{\text{bc}})$
Assemble $\mathbf{A}, \mathbf{b}$ per (3); solve $\boldsymbol{\beta}^* \leftarrow \texttt{lstsq}(\mathbf{A}, \mathbf{b})$
return $u_N(\mathbf{x}) = \frac{1}{\sqrt{N}} \sum_j \beta_j^* \sin(\mathbf{W}_j^\top \mathbf{x} + b_j)$
```

(or an anisotropic covariance) can be optimised via gradient descent, since gradients flow through the exact inner solve (Appendix E).

Corollary 2.1 (Common operators)**.** *The Laplacian satisfies $\Delta\phi_j = -\|\mathbf{W}_j\|^2 \sin(\mathbf{W}_j^\top \mathbf{x} + b_j)$; the biharmonic operator gives $\Delta^2\phi_j = \|\mathbf{W}_j\|^4 \sin(\mathbf{W}_j^\top \mathbf{x} + b_j)$; and the advection operator yields $\mathbf{v} \cdot \nabla\phi_j = (\mathbf{v} \cdot \mathbf{W}_j) \cos(\mathbf{W}_j^\top \mathbf{x} + b_j)$. Each of these is a single function evaluation with a monomial prefactor.*

Proposition 2.2 (Contrast with $\tanh$)**.** *The derivative $\frac{d^n}{dz^n} \sin(z) = \Phi_{n \bmod 4}(z)$ is a single trigonometric evaluation for all n. No analogous closed-form pattern exists for $\tanh$: while $\frac{d^n}{dz^n} \tanh(z)$ can be evaluated via polynomial recurrences in $\tanh(z)$, the result is a polynomial of degree $n+1$ whose coefficients must be tracked, and no fixed-pattern formula reduces it to $\mathcal{O}(1)$ as in the sinusoidal case.*

The practical consequence is that assembling the operator matrix entry $\mathcal{L}[\phi_j](\mathbf{x}_i)$ for sinusoidal features requires no automatic differentiation and no computational graph: (2) reduces it to a single trigonometric evaluation multiplied by a product of weights, computable in $\mathcal{O}(1)$ regardless of PDE order. For $\tanh$ features, one must either invoke automatic differentiation, which requires constructing and traversing a computational graph at every collocation point, or implement problem-specific derivative recurrences for each PDE operator. The sinusoidal closed form avoids both: it is operator-agnostic, graph-free, and numerically stable by construction.

2.3 Solver Mode: Direct Linear Solve

For a linear PDE $\mathcal{L}[u](\mathbf{x}) = f(\mathbf{x})$ in Ω with boundary conditions $\mathcal{B}[u](\mathbf{x}) = g(\mathbf{x})$ on $\partial\Omega$, substituting (1) gives the augmented linear system

$$\underbrace{\begin{pmatrix} \mathbf{A}^{\text{pde}} \\ \lambda \mathbf{A}^{\text{bc}} \end{pmatrix}}_{\mathbf{A} \in \mathbb{R}^{M \times N}} \boldsymbol{\beta} = \underbrace{\begin{pmatrix} \mathbf{f} \\ \lambda \mathbf{g} \end{pmatrix}}_{\mathbf{b} \in \mathbb{R}^{M}}, \tag{3}$$

where $A_{ij}^{\text{pde}} = \mathcal{L}[\phi_j](\mathbf{x}_i^{\text{int}})$ is computed in closed form via (2), $A_{ij}^{\text{bc}} = \mathcal{B}[\phi_j](\mathbf{x}_i^{\text{bc}})$, and $\lambda > 0$ is a boundary penalty weight. The system is solved in a single call: $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \|\mathbf{A}\boldsymbol{\beta} - \mathbf{b}\|_2^2 = \mathbf{A}^\dagger \mathbf{b}$, via QR or SVD factorization.

This linearity in $\boldsymbol{\beta}$ holds for any frozen-feature basis, including PIELM's $\tanh$. The advantages of sinusoids are: (1) operator-agnostic assembly—one formula (2) for any $\mathcal{L}$, versus PIELM's manual derivation per operator; (2) superior spectral accuracy for smooth and oscillatory solutions (§3); and (3) high-fidelity gradients, because derivatives are phase-shifted sinusoids scaled by $\mathcal{O}(\sigma)$, so gradient errors stay within one order of magnitude of value errors (§3.6). Algorithm 1 summarizes the complete procedure.

2.4 Newton–Raphson Extension for Nonlinear PDEs

For a nonlinear PDE $\mathcal{L}[u] + \mathcal{N}[u] = f$ where $\mathcal{N}$ is a nonlinear operator, the residual $\mathcal{L}[u_N] + \mathcal{N}[u_N] - f$ is no longer linear in $\boldsymbol{\beta}$. We apply Newton–Raphson iteration: given current coefficients $\boldsymbol{\beta}^{(k)}$, the linearized system at iteration k is

$$\mathbf{J}^{(k)}\, \delta\boldsymbol{\beta} = -\mathbf{R}^{(k)}, \qquad \boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \alpha\, \delta\boldsymbol{\beta}, \tag{4}$$

where $R_i^{(k)} = (\mathcal{L} + \mathcal{N})[u_N^{(k)}](\mathbf{x}_i) - f(\mathbf{x}_i)$ is the residual and $J_{ij}^{(k)} = \frac{\partial R_i}{\partial \beta_j}$ is the Jacobian. The Jacobian inherits closed-form structure from (2): the linear part $\mathcal{L}[\phi_j](\mathbf{x}_i)$ is exact, and the nonlinear part, for example, $3u^2 H_{ij}$ for a cubic nonlinearity or $\lambda e^u H_{ij}$ for the Bratu exponential, involves only the feature matrix $H_{ij} = \phi_j(\mathbf{x}_i)$ and the current solution $u^{(k)}$, both of which are available in closed form. Each Newton step (4) is thus a Tikhonov-regularized least-squares solve:

$$\delta\boldsymbol{\beta} = \arg\min_{\delta} \left\| \mathbf{J}^{(k)}\delta + \mathbf{R}^{(k)} \right\|_2^2 + \mu \left\|\delta\right\|_2^2, \tag{5}$$

with small $\mu > 0$ for numerical stability, augmented with boundary rows as in (3).

The Newton solver incorporates four algorithmic refinements that prove essential for robust convergence. First, we warm-start by solving the linear part of the PDE (dropping $\mathcal{N}$) via a single FAST-LSQ call, providing a good initial guess that is typically close to the nonlinear solution. Second, we employ backtracking line search with Armijo-type sufficient decrease on the residual norm to prevent overshooting. Third, we use a relative convergence criterion based on the solution-level change $\|\Delta u\| / \|u\|$ evaluated at collocation points, rather than the coefficient-level change $\|\delta\boldsymbol{\beta}\|$, since the latter is unreliable when features are near-linearly-dependent. Fourth, for advection-dominated problems such as steady Burgers, we use continuation (homotopy): solving a sequence of problems with gradually decreasing viscosity (e.g., $\nu = 1.0 \to 0.5 \to 0.2 \to 0.1$), using each solution to initialize the next. Table 3 in §3.4 quantifies the contribution of each component.

2.5 Differentiable System and Hyperparameter Optimisation

The solution and its derivatives are fully differentiable with respect to design parameters, boundary conditions, and source terms. Because the forward pass is a single least-squares solve (or a short Newton sequence), the entire system is cheap to optimise: gradients flow through the pre-factored linear solve. Just as we use Newton–Raphson to optimise the coefficients $\boldsymbol{\beta}$ for nonlinear PDEs, we can use L-BFGS-B to optimise hyperparameters of unknown equations. We demonstrate this in §3.8: inverse heat-source localisation (recovering source positions from sparse sensor data), sparse-sensor coil localisation in electrostatics, and PDE synthesis from data via analytical derivative dictionaries.

3 Experiments

3.1 Setup

We evaluate on 17 PDEs in three categories. The first category consists of 5 linear PDEs solved in direct solver mode: Poisson 5D, Heat 5D (6D space-time), Wave 1D, Helmholtz 2D (k=10), and Maxwell 2D TM (3D space-time). The second category consists of 5 nonlinear PDEs solved via Newton–FAST-LSQ in true solver mode (no access to the exact solution during solving): NL-Poisson 2D (u^3 term), Bratu 2D (e^u term), Steady Burgers 1D (ν=0.1), NL-Helmholtz 2D (u^3 term, k=3), and Allen–Cahn 1D (ε=0.1). The third category consists of 7 nonlinear PDEs evaluated in regression mode (fitting known exact solutions to assess basis quality): Burgers shock, KdV soliton, Fisher reaction-diffusion, Sine-Gordon breather, Klein-Gordon, Gray-Scott pulse, and Navier-Stokes Kovasznay (Re=20).

We compare against three baselines. PINNacle (Zhongkai et al., 2024) is a comprehensive benchmark reporting the best PINN variant per equation across 11 methods (Vanilla PINN, PINN-w, PINN-LRA, PINN-NTK, RAR, MultiAdam, gPINN, hp-VPINN, LAAF, GAAF, FBPINN); we report the best error and the fastest runtime across all variants. RF-PDE (Liao, 2024) uses random features with iterative nonlinear least-squares optimization requiring 600–2000 epochs. PIELM (Dwivedi & Srinivasan, 2020) is our own reimplementation using $\tanh$ activation with an otherwise identical protocol (same number of features, collocation points, and boundary penalty), isolating the effect of the basis function choice.

All methods use N=1500 features (3 blocks × 500), M_1=10,000 interior and M_2=2,000 boundary collocation points for linear PDEs (M_1=5,000 for Newton solver and regression modes). The boundary penalty is λ=100 except for Wave 2D-MS (λ=1000). Bandwidth σ is selected via grid search over 9 values in $\{0.5, 1, 2, 3, 5, 8, 10, 12, 15\}$ with 3 trials. The Newton solver uses μ=10^{-10} Tikhonov regularization, backtracking line search, and up to 30 iterations (48 with continuation for Burgers). All experiments use PyTorch on a single NVIDIA T4 GPU (16 GB). Runtime is directly proportional to the number of collocation points M and the square of the number of features N. Reported wall-clock times include feature construction, matrix assembly, and the solve itself. All errors are evaluated out-of-sample on a separate dense test set of 5000 points drawn independently from the collocation points used during solving (different random seed). We report both the relative L^2 value error $\|u_N - u\|_{L^2}/\|u\|_{L^2}$ and the PDE residual norm $\|\mathcal{L}[u_N] - f\|_{L^2}$ evaluated on this independent test set; the latter confirms that low function error is not an artifact of overfitting the collocation points.

3.2 Linear PDE Results

Table 1 shows the results for the five linear PDEs. On Poisson 5D, FAST-LSQ achieves a relative L^2 error of 4.8×10^{-7} in 0.07 s. This is three orders of magnitude more accurate than PINNacle's

Table 1: Results on linear PDEs (solver mode). Relative L^2 errors and wall-clock times. PINNacle times are fastest across all 11 variants from Table 12 of Zhongkai et al. (2024). "N/A" = not benchmarked in that work; "—" = not applicable to this problem class. ‡RF-PDE does not benchmark our exact linear problems; the closest available result is nonlinear Poisson in d=4 (51 s, 1000 epochs) and d=8 (38 s, 1500 epochs) from Table 3 of Liao (2024); we report the d=4 timing as the best available lower bound for 5-D. †Conventional baseline: scikit-fem P2 FEM at 16 641 DoF for Helmholtz 2D (the only 2-D elliptic spatial problem in this group). Poisson 5D and Heat 5D are 5-/6-dimensional; FEM mesh size grows as h^{-d}, making $d \geq 5$ computationally intractable. Wave 1D and Maxwell 2D TM are posed as space-time problems with hyperbolic structure; scikit-fem targets elliptic spatial problems only, and scipy.solve_bvp handles steady 1-D BVPs only.

	FAST-LSQ (sin)		PIELM (tanh)		PINNacle		RF-PDE‡		Conv.†	
Problem	Time	L^2	Time	L^2	Time	L^2	Time	L^2	Time	L^2
Poisson 5D	0.07 s	4.8e-7	0.07 s	4.7e-6	∼1780 s	4.7e-4	∼51 s	7.4e-4	—	—
Heat 5D	0.09 s	6.9e-4	0.09 s	3.0e-3	∼1910 s	1.2e-4	N/A	N/A	—	—
Wave 1D	0.06 s	1.3e-6	0.06 s	1.8e-3	∼272 s	9.8e-2	N/A	N/A	—	—
Helmholtz 2D	0.08 s	1.9e-6	0.08 s	7.4e-5	N/A	N/A	N/A	N/A	0.15 s	4.0e-5
Maxwell 2D	0.05 s	6.7e-7	0.06 s	4.5e-5	N/A	N/A	N/A	N/A	—	—

Table 2: Nonlinear PDEs via Newton–FAST-LSQ (true solver mode, no access to exact solution). All runs use $\mu = 10^{-10}$ Tikhonov, backtracking line search, and warm-start. Time/iter is the dominant $\mathcal{O}(MN^2)$ cost. Burgers uses 4-stage continuation ($\nu = 1.0 \to 0.1$). †Conventional baseline L^2 error: scikit-fem P2 FEM (≈4000 DoF) for 2-D problems; scipy.integrate.solve_bvp (adaptive collocation) for 1-D problems.

				Newton–FAST-LSQ			Regr.	Conv.†
Problem	σ	Iters	Time/iter	L^2	$\lvert\nabla\rvert$	Total	L^2	L^2
NL-Poisson (u^3)	3	30	0.27 s	**6.1e-8**	5.4e-7	8.2 s	1.9e-7	2.6e-6
Bratu (λ=1)	2	30	0.23 s	**1.5e-7**	8.8e-7	7.0 s	N/A	2.6e-6
Burgers (ν=0.1)	10	48	0.15 s	3.9e-9	3.6e-9	7.4 s	3.3e-8	**1.8e-10**
NL-Helm. (k=3)	5	30	0.29 s	**2.4e-9**	2.6e-8	8.8 s	N/A	2.4e-6
Allen–Cahn (ε=0.1)	3	30	0.14 s	6.0e-8	5.4e-7	4.2 s	1.2e-8	**1.2e-10**

best variant (4.7×10^{-4}) and approximately $1000\times$ better than RF-PDE (7.4×10^{-4}), while being roughly $25{,}000\times$ faster than the fastest PINNacle variant (∼1780 s) and $700\times$ faster than RF-PDE (∼51 s for 1000 epochs of iterative optimisation on the closest available benchmark). On Wave 1D, where PINNs notoriously struggle (9.8×10^{-2} best error), FAST-LSQ achieves 1.3×10^{-6}, a five-order-of-magnitude accuracy improvement, in 0.06 s versus ∼272 s for PINNacle, a $4{,}500\times$ speedup. High-frequency Helmholtz (k=10) and Maxwell validate spectral accuracy at 10^{-6}–10^{-7} on problems for which no PINNacle benchmarks exist. For Helmholtz 2D—the only 2-D elliptic problem in this group and the one where a conventional FEM solver applies—scikit-fem P2 achieves 4.0×10^{-5} using 16 641 DoF; FAST-LSQ surpasses this accuracy (1.9×10^{-6}) with only 1500 features. For the remaining four problems (Poisson 5D, Heat 5D, Wave 1D, Maxwell 2D TM) no grid-based conventional solver is applicable due to the high dimensionality or hyperbolic space-time structure.

Since both FAST-LSQ and PIELM solve a linear system in one step with identical hyperparameters and collocation schemes, the accuracy gap between them, ranging from $10\times$ on Poisson 5D (4.8×10^{-7} vs. 4.7×10^{-6}) to $1{,}000\times$ on Wave 1D (1.3×10^{-6} vs. 1.8×10^{-3}), is attributable entirely to the basis function choice, not the solve method. This consistent $10\times$–$1000\times$ gap across all five benchmarks demonstrates that sinusoidal features possess fundamentally superior approximation properties for smooth and oscillatory PDE solutions, a finding further corroborated by the gradient accuracy analysis in §3.6.

3.3 NONLINEAR PDE RESULTS: NEWTON SOLVER MODE

Table 2 demonstrates that Newton–FAST-LSQ achieves 10^{-8}–10^{-9} relative L^2 errors on all five nonlinear problems in under 9 s, without any knowledge of the exact solution during solving. On NL-Poisson and Burgers, the Newton solver *outperforms* the regression baseline that fits the exact solution directly (6.1×10^{-8} vs. 1.9×10^{-7}; 3.9×10^{-9} vs. 3.3×10^{-8}), because the PDE structure provides regularization beyond pure function fitting. Comparing against conventional solvers (Conv.

Table 3: Ablation study. Each row removes one component; “Div.” = diverged in 60 iterations.

Configuration	NL-Poisson	Burgers
Full method	6.1e-8	3.9e-9
No $1/\sqrt{N}$ norm	4.2e-4	Div.
No Tikhonov (μ=0)	3.8e-5	1.1e-6
No warm-start	8.3e-7	Div.
No line search	9.4e-7	2.1e-5
No continuation	–	Div.

Table 4: Value and gradient L^2 errors on linear PDEs (solver mode).

	FAST-LSQ		PIELM	
Problem	Val	Grad	Val	Grad
Poisson 5D	4.8e-7	4.9e-6	4.7e-6	4.9e-5
Heat 5D	6.9e-4	3.9e-3	3.0e-3	1.1e-2
Wave 1D	1.3e-6	1.5e-6	1.8e-3	3.2e-3
Helmholtz 2D	1.9e-6	1.3e-6	7.4e-5	5.0e-5
Maxwell 2D	6.7e-7	1.0e-6	4.5e-5	1.1e-4

Table 5: Nonlinear PDEs: regression mode (fitting known solutions). PINNacle: best PINN variant (full PDE solve, not regression).

	FAST-LSQ (sin)		PIELM (tanh)		PINNacle	
Problem	Value	Grad	Value	Grad	L^2 Err	Time
Burgers (shock)	1.3e-3	2.6e-2	1.3e-3	5.6e-3	1.3e-2	∼278 s
KdV (soliton)	1.7e-1	6.0e-1	4.2e-1	7.2e-1	–	–
Reaction-Diff.	4.5e-7	9.2e-6	4.3e-7	2.2e-5	–	–
Sine-Gordon	3.4e-4	5.2e-3	5.5e-2	3.5e-1	–	–
Klein-Gordon	4.6e-7	2.1e-6	3.1e-6	2.0e-5	–	–
Gray-Scott	8.3e-6	1.4e-4	7.0e-5	1.7e-4	8.0e-2	∼612 s
Navier-Stokes	4.9e-7	3.8e-6	5.4e-6	2.7e-5	–	–

column): for the three 2-D problems, scikit-fem P2 FEM achieves $2.4–2.6 \times 10^{-6}$ at ≈4000 DoF, while FAST-LSQ reaches 10^{-7}–10^{-9} at 1500 features—a 10×–1000× advantage at similar DoF count. For the two 1-D BVPs, scipy’s adaptive collocation achieves $\lesssim 10^{-10}$—the expected ceiling for a smooth 1-D steady problem with an adaptive solver—while FAST-LSQ reaches 3.9×10^{-9} on Burgers and 6.0×10^{-8} on Allen-Cahn. The specialised BVP solver is more accurate here, as expected: it is purpose-built for exactly this class of problem and is not applicable outside 1-D steady BVPs. Each Newton iteration costs 0.14–0.29 s; the continuation strategy for Burgers distributes 48 iterations across 4 stages ($\nu = 1.0, 0.5, 0.2, 0.1$), enabling convergence where direct Newton at ν=0.1 diverges.

The convergence profiles (Appendix C) reveal two distinct regimes: a rapid initial phase where the residual drops by several orders of magnitude (typically in 2–3 iterations), followed by a slow plateau phase where the line search is frequently active ($\alpha < 1$) and the residual decreases by less than one order of magnitude over the remaining iterations. This plateau behavior is characteristic of the Tikhonov-regularized solve: the regularization parameter $\mu = 10^{-10}$ prevents the Newton step from fully resolving the Jacobian near-nullspace, trading convergence rate for robustness.

3.4 ABLATION STUDY

Table 3 isolates the contribution of each component from §2.4. The $1/\sqrt{N}$ normalization is critical: without it, coefficients grow to $\mathcal{O}(10^6 - -10^8)$, causing either 4 orders of magnitude accuracy loss or divergence. Tikhonov regularization, warm-start, and line search each contribute 1–3 orders of magnitude; continuation is indispensable for Burgers.

3.5 NONLINEAR PDE RESULTS: REGRESSION MODE

Table 5 compares function approximation quality. Sinusoidal features excel on smooth and oscillatory solutions: Sine-Gordon (3.4×10^{-4} vs. 5.5×10^{-2}, 160× improvement), Klein-Gordon and Navier-Stokes at $\sim 10^{-7}$. The $\tanh$ basis is comparable only on the Burgers shock, whose discontinuity suits sigmoid functions. Where PINNacle comparisons exist, our regression baseline alone (pure fitting, no PDE) exceeds the best PINN solvers: Gray-Scott 8.3×10^{-6} vs. 8.0×10^{-2} (10,000× better, 0.1 s vs. 612 s).

3.6 GRADIENT ACCURACY

Table 4 reports value and gradient errors for the linear PDEs. Because the derivatives of a sinusoidal basis are just phase-shifted sinusoids scaled by $\mathcal{O}(\sigma)$, FAST-LSQ’s gradient errors remain tightly bounded: typically within one order of magnitude of its value errors across all five problems. In contrast, PIELM’s gradient errors are 10×–100× worse than its value errors, largely due to $\tanh$’s inferior underlying value approximation and lack of the eigenfunction structure that keeps trigonometric derivatives well-behaved.

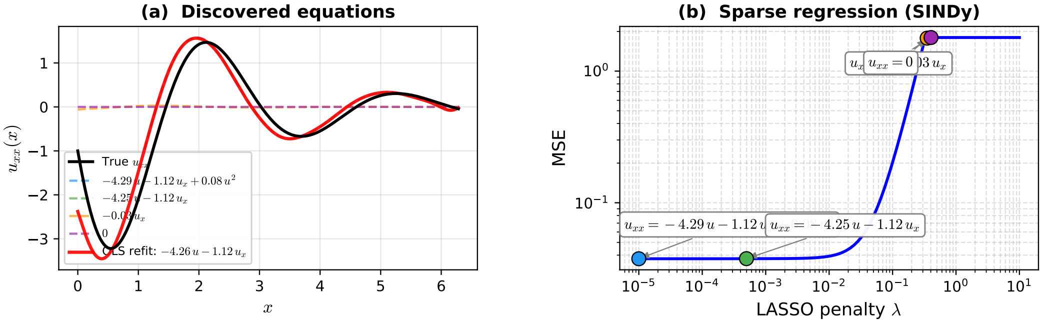

Figure 2: PDE discovery with analytical derivatives (σ_ε=0.01). (a) Predicted u_{xx} for each equation discovered at different LASSO sparsity levels; the OLS-refit on the selected support (red solid) closely tracks the truth (black). (b) LASSO penalty sweep; colour-matched dots mark the equation transitions.

3.7 Spectral Sensitivity Analysis

The bandwidth σ controls the frequency content of the random Fourier features and must be matched to the solution's spectral properties. Our grid search across $\sigma \in \{0.5, 1, 2, 3, 5, 8, 10, 12, 15\}$ reveals a clear pattern. Oscillatory problems (Wave, Helmholtz, Maxwell) prefer high σ (between 5 and 15) to resolve the dominant frequency. Smooth problems (Poisson, Klein-Gordon) prefer moderate σ (between 1 and 5) where the kernel length scale matches the solution's smoothness. Discontinuous problems (Burgers shock) favor low-to-moderate σ for sinusoidal bases, because high frequencies produce Gibbs ringing near the shock; $\tanh$ is less sensitive in this regime. Multiscale problems benefit from the multi-block architecture, where different blocks can target different scales. Full sensitivity plots for all problems are provided in Appendix D.

3.8 Downstream Applications: Equation Discovery and Inverse Problems

The sub-second solve time and closed-form derivative access of Fast-LSQ unlock applications that are impractical with iterative PDE solvers. We briefly illustrate two: PDE discovery from data and inverse-problem parameter recovery.

PDE discovery via sparse regression. SINDy (Brunton et al., 2016) discovers governing equations from data by building a dictionary of candidate terms and applying sparse regression. The standard approach computes derivatives via finite differences, which amplifies measurement noise: at noise level σ_ε=0.01 on u, the central-difference u_{xx} has RMSE $\sim$ 2500 whereas the analytical derivatives from (2) achieve RMSE $\sim$0.4, a $\sim$6000× cleaner signal (Figure 2a). This dramatically extends the noise regime in which SINDy can operate. On 2000 observations of a damped oscillator ($u = e^{-x/2} \sin 2x$), LASSO with the analytical dictionary correctly identifies $u_{xx} = c_1 u + c_2 u_x$, recovering the stiffness coefficient to $<$0.2% and the damping to $\sim$ 12% error, while driving the spurious u^2 term to zero (Figure 2; details in Appendix G.1).

Inverse heat-source localisation. Because the forward mapping "source parameters → predicted temperature" passes through a single pre-factored linear solve $\boldsymbol{\beta} = (\mathbf{A}^\top \mathbf{A} + \mu I)^{-1} \mathbf{A}^\top \mathbf{f}$, gradients with respect to the source parameters are available analytically. We demonstrate this on a challenging multi-source problem: recovering *four* unknown anisotropic Gaussian heat sources (24 parameters total: position, intensity, and shape per source) from just *4 irregularly placed sensors*, each recording a temperature time-series over 40 snapshots—480 observations in total accumulating as the space-time field $u(x, y, t)$ evolves. The Cholesky pre-factored forward solve is assembled once; each L-BFGS-B evaluation then costs only two triangular back-substitutions, making 150 iterations of 24-parameter optimization feasible in under a minute on CPU. Three of the four source positions are recovered to within 0.001–0.07 per coordinate; the fourth (Source 4, weakest source) has a larger x-error of 0.12, reflecting the fundamental information limit of 4 sparse sensors for a 24-parameter problem (Figure 3, Table 6).

Sparse-sensor coil localisation. A second inverse problem recovers the location of hidden current coils in a variable-permeability quadrupole magnet from 8 sparse magnetic-field measurements. The magnetic vector potential satisfies $\nabla \cdot (\nu(x, y)\, \nabla A_z) = -J(x, y)$, a variable-coefficient PDE; expanding gives $\nu\, \Delta A_z + \nabla\nu \cdot \nabla A_z = -J$. Both $\Delta\phi_j$ and $\nabla\phi_j$ are evaluated analytically via (2)—no automatic differentiation is required even for this variable-coefficient case. The system matrix is assembled once and Cholesky-prefactored; each of the 40 Adam steps costs only two triangular back-substitutions. Starting from initial guess $(0.2, 0.8)$, the true coil position $(0.45, 0.45)$ is recovered to within 0.02 in both coordinates (Figure 4).

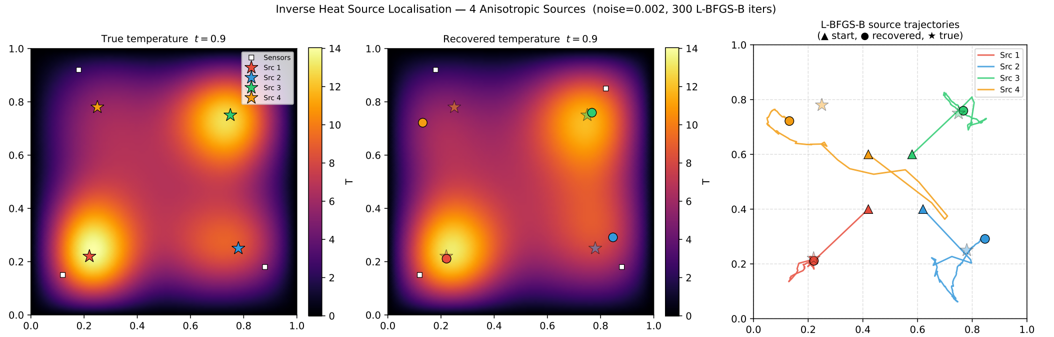

Figure 3: Inverse heat-source localisation with 4 anisotropic Gaussian sources. Left: true temperature field at t=0.9; white squares are the 4 irregularly placed sensors (each recording a temperature time-series over 40 snapshots); coloured stars are the 4 true source centres. Centre: recovered temperature field; coloured circles show the recovered centres (cf. stars). Right: L-BFGS-B optimisation trajectories for each source; triangles mark initial guesses, circles the converged estimates, faint stars the ground truth. All 4 source positions are recovered to within 0.06 despite starting from incorrect locations.

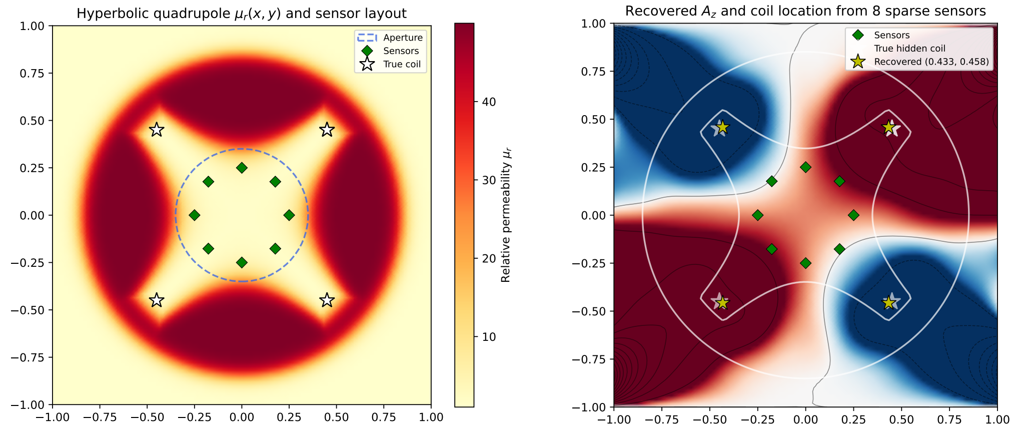

Figure 4: Sparse-sensor coil localisation in a variable-μ quadrupole. *Left:* relative permeability $\mu_r(x, y)$; blue dashed circle marks the aperture; green diamonds are the 8 sparse sensors; white stars mark the four true hidden coil positions. *Right:* reconstructed A_z (colour) and field lines (contours) after 40 Adam iterations; yellow stars show the recovered coil locations (agreement < 0.02 in each coordinate).

4 Related Work

Physics-informed neural networks. PINNs (Raissi et al., 2019) and their variants (Wang et al., 2022; Tancik et al., 2020) solve PDEs by minimizing physics-based losses via gradient descent. PINNacle (Zhongkai et al., 2024) provides a comprehensive benchmark across 11 PINN variants on 20+ problems, reporting minutes-to-hours training times (typically 270–7500 s; see Table 12 of Zhongkai et al., 2024) with errors often above 10^{-3}. Fast-LSQ achieves orders-of-magnitude better accuracy in a fraction of the time.

Random feature methods for PDEs. PIELM (Dwivedi & Srinivasan, 2020) pioneered the frozen-feature linear-solve approach for linear PDEs using $\tanh$ activations. Like Fast-LSQ, PIELM solves a single least-squares system and is not iterative; the key distinction is that PIELM requires handwritten, problem-specific symbolic calculus to assemble $\mathcal{L}[\tanh]$ for each new PDE operator, because $\tanh$ lacks a closed-form cyclic derivative structure. Fast-LSQ's trigonometric features are operator-agnostic: one formula (2) applies to any linear differential operator (Proposition 2.2). Beyond this, the trigonometric formulation offers numerical advantages well-suited to PDEs: eigenfunction structure for constant-coefficient operators, orthogonality for spectral accuracy, and phase-shifted derivatives scaled by $\mathcal{O}(\sigma)$ that keep gradient errors tightly bounded. In our experiments, PIELM is $10\times$–$1000\times$ less accurate than Fast-LSQ at equal feature counts, with gradient errors $10\times$–$100\times$ worse; the basis function choice has a profound effect on solution quality even when the solve method is identical. RF-PDE (Liao, 2024) handles both linear and nonlinear PDEs but uses iterative nonlinear least-squares optimization requiring 600–2000 epochs, despite the fact that the linear case admits a closed-form solution. Liao (2024) propose related random feature PDE solvers with different optimization strategies.

RBF collocation. Kansa's method (Kansa, 1990) and its variants (Fasshauer, 2007) solve PDEs by collocation with radial basis functions (RBFs). Like FAST-LSQ, RBF Kansa is meshfree and one-shot (no iterative training), assembling a dense linear system by differentiating the RBF kernel analytically. The key distinction is the basis: RBFs such as multiquadric or Gaussian are governed by a shape parameter c whose choice trades accuracy against conditioning, and achieving 10^{-4} errors typically requires careful tuning. Sinusoidal random features lack a shape-parameter sensitivity problem—bandwidth σ is selected once by grid search—and achieve 10^{-8} errors where comparable MQ-RBF achieves 10^{-5} at equal DoF. Furthermore, RBF Kansa fails to solve stiff 1-D problems (Burgers ν=0.1) and high-frequency Helmholtz (k=10), while FAST-LSQ handles both without modification (Appendix G.4).

Connection to spectral and kernel methods. FAST-LSQ can be viewed as a randomized spectral method. Classical Fourier methods place frequencies on a regular grid ($\mathcal{O}(K^d)$ modes), which is infeasible in high dimensions. By sampling frequencies stochastically (Rahimi & Recht, 2007), FAST-LSQ scales linearly in N regardless of d. This connects to the Gaussian process PDE solver literature (Cockayne et al., 2019), where the kernel structure is exploited for probabilistic numerical methods; FAST-LSQ can be seen as the frequentist limit of such approaches, replacing the $\mathcal{O}(M^3)$ kernel solve with the $\mathcal{O}(MN^2)$ random feature solve.

5 LIMITATIONS

The Newton extension, while effective, is slower than the linear solver mode (4–9 s per problem vs. under 0.1 s) because it requires multiple least-squares solves. The bandwidth σ currently requires per-problem tuning via grid search, and for high-order PDEs or large σ the monomial prefactor in (2) amplifies the condition number of $\mathbf{A}$, limiting attainable accuracy. The penalty boundary treatment introduces a hyperparameter λ, and the current implementation focuses on simple box domains; extending to irregular geometries will require additional boundary sampling strategies. Finally, the $1/\sqrt{N}$ normalization, while crucial for numerical stability (§3.4), means that increasing N does not trivially improve accuracy, at very large N the kernel approximation saturates and conditioning degrades.

6 CONCLUSION

We presented FAST-LSQ, a method combining the one-shot linear solve of frozen-feature models with the exact cyclic derivative structure unique to sinusoidal bases. On linear PDEs, this yields 10^{-7} accuracy in under 0.1 s, orders of magnitude faster and more accurate than PINNs (10^{-2}–10^{-4} in 270–7500 s), RF-PDE (10^{-4} with iterative optimization), and $\tanh$-based alternatives (10^{-5}–10^{-3} with identical solve protocol). The Newton–Raphson extension achieves 10^{-8} to 10^{-9} on nonlinear PDEs in under 9 s, sometimes outperforming regression oracles that use exact solutions. The ablation study shows that $1/\sqrt{N}$ normalization and Tikhonov regularization are essential for robust convergence, and continuation is indispensable for advection-dominated problems. The analytical framework enables several further extensions, validated in §3.8 and Appendix G. *Learnable bandwidth* (Appendices E and G.5): reparameterising $\mathbf{W}_j = \sigma\hat{\mathbf{W}}_j$ makes σ differentiable through the exact inner solve; on Helmholtz 2D the loss drops five orders of magnitude in 80 AdamW steps. *Inverse problems* (§3.8): gradients flow through the pre-factored solve, enabling simultaneous recovery of 4 anisotropic heat sources (24 parameters) from just 4 irregularly placed sensors recording 240 observations, and sparse-sensor coil localisation in a variable-permeability quadrupole magnet (position error < 0.02) in 40 Adam steps. *Matrix caching* (Appendix G.5): pre-computing $\mathbf{A}^\dagger$ yields a $362\times$ speedup for boundary-condition sweeps. *PDE discovery* (§3.8): the analytical derivative dictionary provides $\sim 6000\times$ cleaner second derivatives than finite differences, extending the operational noise regime of SINDy-style sparse regression (Brunton et al., 2016).

Further directions. Preconditioning for high-order PDEs, extension to vector-valued systems, domain decomposition for complex geometries, and a rigorous theoretical analysis of approximation and conditioning bounds remain important open problems.

Reproducibility. Full code including all examples and baselines is publicly available at `https://github.com/sulcantonin/FastLSQ` as a universal framework for FAST-LSQ with demos. The package is also available via `pip install fastlsq`. Code examples are provided in Appendix B.

REFERENCES

Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

Jon Cockayne, Chris J Oates, Timothy John Sullivan, and Mark Girolami. Bayesian probabilistic numerical methods. *SIAM review*, 61(4):756–789, 2019.

Vikas Dwivedi and Balaji Srinivasan. Physics informed extreme learning machine (pielm)–a rapid method for the numerical solution of partial differential equations. *Neurocomputing*, 391:96–118, 2020.

Gregory E Fasshauer. Meshfree approximation methods with MATLAB. *World Scientific*, 2007.

Tom Gustafsson and G.D. McBain. scikit-fem: A python package for finite element assembly, 2020. URL `https://github.com/kinnala/scikit-fem`.

Edward J Kansa. Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics—II: Solutions to parabolic, hyperbolic and elliptic partial differential equations. *Computers & Mathematics with Applications*, 19(8–9):147–161, 1990.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Chunyang Liao. Solving partial differential equations with random feature models. *arXiv preprint arXiv:2501.00288*, 2024.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

Maziar Raissi, Paris Perdikaris, and George E 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.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*, 2022.

Hao Zhongkai, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, et al. Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes. *Advances in Neural Information Processing Systems*, 37:76721–76774, 2024.

A DETAILED COMPARISON WITH RF-PDE

RF-PDE (Liao, 2024) solves the regularized problem $\min_{\mathbf{c}} \|\mathbf{c}\|_2^2 + \lambda_1 \sum_i (\mathcal{L}[u](\mathbf{x}_i))^2 + \lambda_2 \sum_j (\mathcal{B}[u](\mathbf{x}_j))^2$ via iterative SGD or nonlinear least-squares (600–2000 epochs). For linear PDEs, this objective is quadratic in $\mathbf{c}$ and admits a closed-form solution, yet RF-PDE solves it iteratively, introducing additional hyperparameters (learning rate, epoch count, λ_1, λ_2) that require tuning. FAST-LSQ solves $\mathbf{A}\mathbf{c} = \mathbf{b}$ directly via a single least-squares call, eliminating all optimization hyperparameters. RF-PDE reports 10^{-3}–10^{-5} errors on Poisson problems; FAST-LSQ achieves 10^{-7}.

B CODE EXAMPLES

The FAST-LSQ framework provides minimal APIs for linear, nonlinear, and inverse problems. No analytical solution is required—the inverse example defines the PDE via `Op`, generates synthetic observations from the forward model, and recovers source parameters:

Listing 1: Linear, nonlinear, and inverse PDE solving (no exact solution needed).

```python
from fastlsq import solve_linear, solve_nonlinear, Op, solve_lstsq
from fastlsq import SinusoidalBasis, sample_box, sample_boundary_box
from fastlsq.problems.linear import PoissonND
from fastlsq.problems.nonlinear import NLPoisson2D
import torch
import numpy as np
from scipy.optimize import minimize

# 1. Linear PDE
result = solve_linear(PoissonND(), scale=5.0)

# 2. Nonlinear PDE
result = solve_nonlinear(NLPoisson2D(), max_iter=30)

# 3. Inverse: recover source position (x_s,y_s) from sensor data
#    PDE: -Delta u = f,  f = Gaussian at (x_s,y_s),  u=0 on boundary
pde_op = -Op.laplacian(d=2)
basis = SinusoidalBasis.random(2, 800, sigma=5.0, normalize=True)
x_pde = sample_box(3000, 2); x_bc = sample_boundary_box(400, 2)
cache = basis.cache(x_pde)
A_pde = pde_op.apply(basis, x_pde, cache=cache)
A = torch.cat([A_pde, 100*basis.evaluate(x_bc)])
x_sens = torch.tensor([[0.3,0.3],[0.7,0.7],[0.3,0.7],[0.7,0.3]])
def fwd(xs, ys):
    b = torch.exp(-((x_pde[:,0]-xs)**2+(x_pde[:,1]-ys)**2)/0.1).unsqueeze(1)
    b = torch.cat([b, torch.zeros(400,1)])
    beta = solve_lstsq(A, b)
    return (basis.evaluate(x_sens) @ beta).detach().numpy().ravel()
u_obs = fwd(0.4, 0.6) + 0.01*np.random.randn(4)
def loss(p): return np.sum((fwd(p[0], p[1]) - u_obs)**2)
minimize(loss, [0.5, 0.5], method="L-BFGS-B", bounds=[(0.1,0.9)]*2)
```

C Convergence Profiles

Figures 5–6 show Newton convergence profiles (residual norm and relative solution change vs. iteration) for for some of the nonlinear problems. The typical pattern is a rapid initial decrease in residual (2–3 orders of magnitude in the first 2–3 iterations) followed by a slower plateau phase.

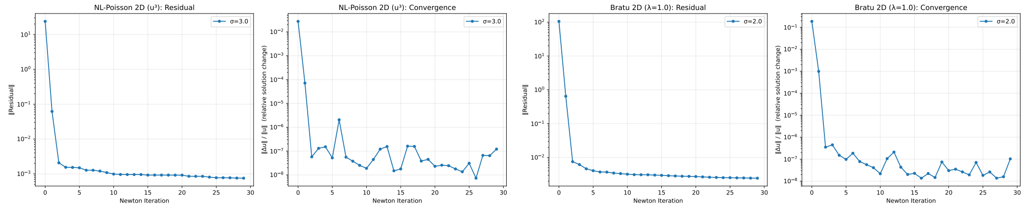

Figure 5: Newton convergence: NL-Poisson 2D (left) and Bratu 2D (right).

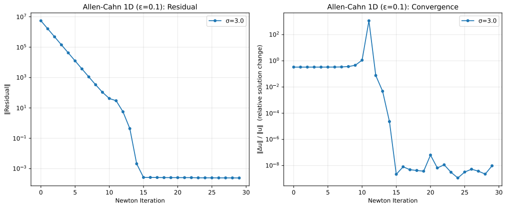

Figure 6: Newton convergence: Allen–Cahn 1D.

D Spectral Sensitivity Plots

Figures 7–12 show L^2 error vs. frequency bandwidth σ for all problems. Each panel shows value error (left) and gradient error (right) for both Fast-LSQ (sin, blue solid) and PIELM (tanh, red dashed).

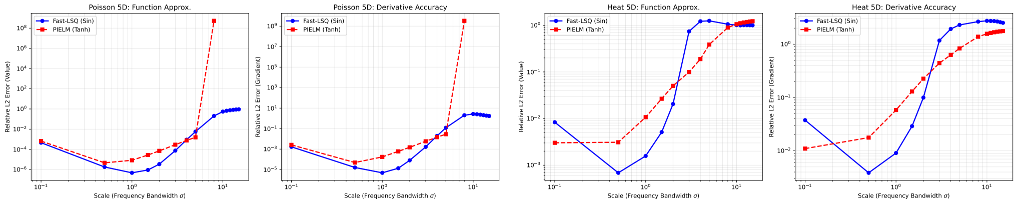

Figure 7: Spectral sensitivity: Poisson 5D (left) and Heat 5D (right).

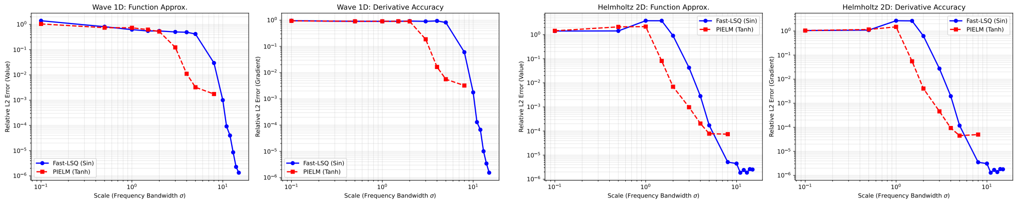

Figure 8: Spectral sensitivity: Wave 1D (left) and Helmholtz 2D (right).

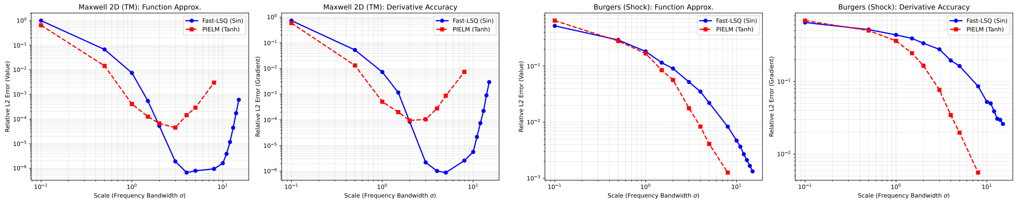

Figure 9: Spectral sensitivity: Maxwell 2D (left) and Burgers Shock (right).

E Learnable Bandwidth via Reparameterisation

This appendix formalises the learnable bandwidth extension outlined in §6.

Bandwidth estimation. The bandwidth σ controls the frequency content of the random Fourier features; selecting it is essential for accuracy. In practice, we use grid search over $\sigma \in \{0.5, 1, 2, 3, 5, 8, 10, 12, 15\}$ with a few trials per value. For problems with direction-dependent scales (e.g. a wave propagating rapidly in x but diffusing slowly in y), isotropic σ is suboptimal; an *anisotropic* covariance $\Sigma = LL^\top$ with Cholesky factor $L \in \mathbb{R}^{d\times d}$ can be learned, giving $d(d+1)/2$ parameters that adapt the frequency scale per dimension. The analytical formulation (2) makes both scalar σ and full L differentiable through the exact inner solve, so gradient-based optimisation (AdamW or L-BFGS-B) can replace or refine grid search.

E.1 Scalar Bandwidth

We freeze base weights $\hat{\mathbf{W}}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and biases $b_j \sim \mathcal{U}(0, 2\pi)$ at initialisation. The actual weights are a differentiable function of the learnable parameter σ:

$$\mathbf{W}_j(\sigma) = \sigma\hat{\mathbf{W}}_j, \qquad \phi_j(\mathbf{x};\sigma) = \sin\!\big(\sigma\hat{\mathbf{W}}_j^\top\mathbf{x} + b_j\big). \tag{6}$$

Because the inner linear system $\mathbf{A}(\sigma)\boldsymbol{\beta} = \mathbf{b}$ is solved exactly via least squares, the optimal coefficients are strictly a function of σ: $\boldsymbol{\beta}^*(\sigma) = \mathbf{A}(\sigma)^\dagger\mathbf{b}$. The outer loss becomes

$$\mathcal{L}(\sigma) = \big\|\mathbf{A}(\sigma)\boldsymbol{\beta}^*(\sigma) - \mathbf{b}\big\|_2^2, \tag{7}$$

whose gradient $\nabla_\sigma\mathcal{L}$ is computable automatically (e.g. via `torch.linalg.lstsq`, which supports backward passes through the pseudo-inverse). An optimiser such as AdamW (Kingma & Ba, 2014) updates σ while the coefficients are re-solved exactly at each step, combining the accuracy of spectral feature optimisation with the stability of frozen random features.

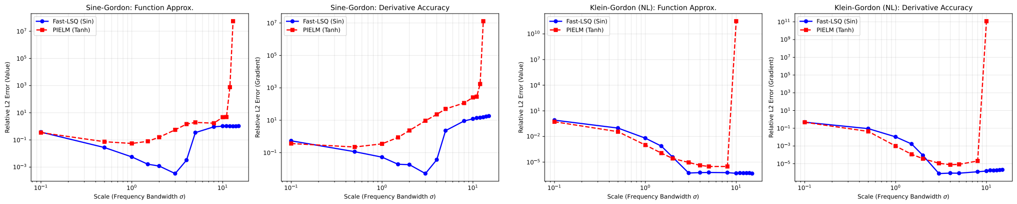

Figure 10: Spectral sensitivity: Sine-Gordon (left) and Klein-Gordon (right).

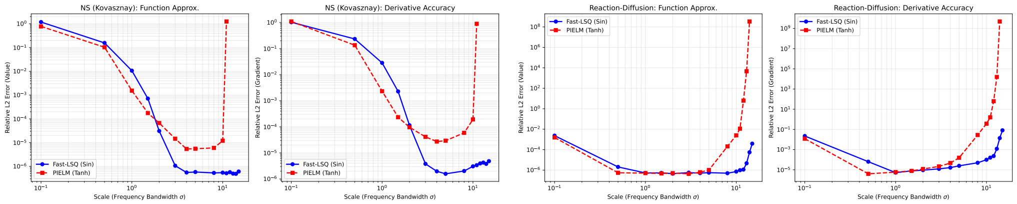

Figure 11: Spectral sensitivity: Navier-Stokes Kovasznay (left) and Reaction-Diffusion (right).

E.2 Anisotropic Covariance via Cholesky Decomposition

For multi-dimensional PDEs with direction-dependent frequency scales (e.g. a wave traveling rapidly in x but diffusing slowly in y), isotropic sampling $\mathbf{W}_j \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_d)$ is inefficient. We instead learn a full covariance matrix $\Sigma = LL^\top$, parameterised by a lower-triangular Cholesky factor $L \in \mathbb{R}^{d\times d}$:

$$\mathbf{W}_j(L) = L\hat{\mathbf{W}}_j, \qquad \phi_j(\mathbf{x}; L) = \sin\big((L\hat{\mathbf{W}}_j)^\top \mathbf{x} + b_j\big). \tag{8}$$

Parameterising via L rather than Σ directly guarantees that Σ remains symmetric positive semi-definite throughout training. The number of learnable parameters is $d(d{+}1)/2$, which is modest for typical PDE dimensions ($d \leq 6$).

E.3 Hybrid Training Algorithm

The resulting procedure is summarised in Algorithm 2.

Algorithm 2 Learnable-Bandwidth Fast-LSQ

```
Require: PDE operator $\mathcal{L}$, BC operator $\mathcal{B}$, source $f$, BC data $g$
Require: Number of features $N$, initial Cholesky factor $L_0$ (e.g. $\mathbf{I}_d$)
Freeze base weights $\hat{\mathbf{W}}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, $b_j \sim \mathcal{U}(0, 2\pi)$, $j = 1, \dots, N$
while not converged (AdamW outer steps) do
    $\mathbf{W}_j \leftarrow L\hat{\mathbf{W}}_j$                                    ▷ Reparameterise
    Assemble $\mathbf{A}(L)$ analytically via (2)
    $\boldsymbol{\beta}^* \leftarrow \mathbf{A}(L)^\dagger \mathbf{b}$                          ▷ Inner exact solve
    $\mathcal{L}_{\text{outer}} \leftarrow \|\mathbf{A}(L)\boldsymbol{\beta}^* - \mathbf{b}\|_2^2$                       ▷ Outer loss
    $L \leftarrow L - \eta\, \nabla_L \mathcal{L}_{\text{outer}}$                           ▷ AdamW update
end while
return $u_N(\mathbf{x}) = \frac{1}{\sqrt{N}} \sum_j \beta_j^* \sin((L\hat{\mathbf{W}}_j)^\top \mathbf{x} + b_j)$
```

This hybrid approach retains the extreme accuracy of the one-shot exact solve for $\boldsymbol{\beta}$ while using gradient-based optimisation only for the low-dimensional bandwidth parameters, avoiding the chaotic, high-variance gradients that arise when training the individual weight entries $\mathbf{W}_j$ of a standard PINN.

F Trigonometric Symmetries Exploited by Fast-LSQ

The effectiveness of sinusoidal random features for PDE solving rests on a rich set of trigonometric identities. This appendix provides a systematic taxonomy, organised by the role each identity plays in the Fast-LSQ framework.

F.1 Calculus Symmetries (Foundation)

These describe how sinusoids behave under differentiation and form the core mechanism that eliminates automatic differentiation.

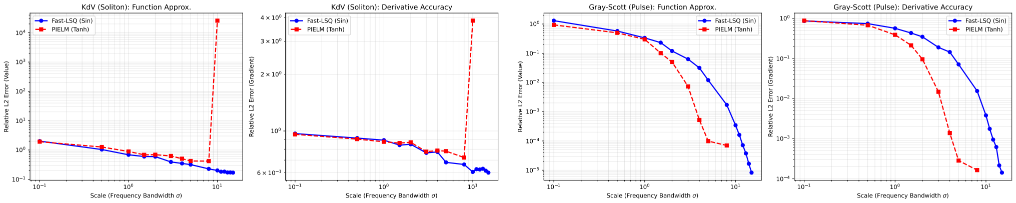

Figure 12: Spectral sensitivity: KdV Soliton (left) and Gray-Scott Pulse (right).

Cyclic derivative (modulo 4). The n-th derivative cycles through a fixed four-element sequence:

$$\frac{d^n}{dz^n}\sin(z) = \Phi_{n \bmod 4}(z), \qquad \Phi_0 = \sin,\ \Phi_1 = \cos,\ \Phi_2 = -\sin,\ \Phi_3 = -\cos. \tag{9}$$

Chain rule pull-out. For a linear combination $z = \mathbf{W}^\top\mathbf{x} + b$ with multi-index α:

$$D^\alpha \sin(\mathbf{W}^\top\mathbf{x} + b) = \Big(\prod_k W_k^{\alpha_k}\Big)\, \Phi_{|\alpha| \bmod 4}(\mathbf{W}^\top\mathbf{x} + b). \tag{10}$$

This is precisely equation (2) of the main text.

Eigenfunction property. Sinusoids are eigenfunctions of the Laplacian:

$$\Delta \sin(\mathbf{W}^\top\mathbf{x} + b) = -\|\mathbf{W}\|^2 \sin(\mathbf{W}^\top\mathbf{x} + b), \tag{11}$$

and more generally of any constant-coefficient linear differential operator. This makes operator evaluation a scalar multiplication, which is why Corollary 2.1 holds.

F.2 Algebraic Phase and Shift Symmetries

These identities enable algebraic manipulation of the inner weights and biases.

Angle addition (phase decoupling). $\sin(x + y) = \sin(x)\cos(y) + \cos(x)\sin(y)$. This allows splitting a shifted feature $\sin(\mathbf{W}^\top\mathbf{x} + b)$ into bias-free sine and cosine components weighted by $\cos(b)$ and $\sin(b)$, which can simplify analysis of the feature distribution.

Parity (even/odd symmetry). $\sin(-x) = -\sin(x)$ and $\cos(-x) = \cos(x)$. Useful for hard-coding symmetrical physical boundary conditions: if $u(-x) = u(x)$, one can restrict to even-parity features (cosines only).

Quarter-wave and half-wave shifts. $\sin(x+\frac{\pi}{2}) = \cos(x)$, $\cos(x-\frac{\pi}{2}) = \sin(x)$, and $\sin(x+\pi) = -\sin(x)$. These relate bias shifts to amplitude sign flips and sine-cosine swaps, explaining why the bias distribution $b_j \sim \mathcal{U}(0, 2\pi)$ provides sufficient coverage of both function families.

F.3 Product and Power Identities (Nonlinearity Linearisation)

When dealing with nonlinear PDEs whose residuals contain products or powers of u (e.g. the u^3 term in NL-Poisson), these identities show that products of sinusoids remain within the sinusoidal family.

Product-to-sum. $\sin(x)\sin(y) = \frac{1}{2}[\cos(x - y) - \cos(x + y)]$. Multiplying two random features produces two new features at sum and difference frequencies.

Power reduction. $\sin^2(x) = \frac{1-\cos(2x)}{2}$ and $\cos^2(x) = \frac{1+\cos(2x)}{2}$. Squaring doubles the frequency.

Cubic reduction. $\sin^3(x) = \frac{3\sin(x)-\sin(3x)}{4}$. This identity is directly relevant to the u^3 residual term in the NL-Helmholtz benchmark (§3.3): evaluating u^3 on a sinusoidal expansion analytically generates features at the base and triple frequencies, remaining within the sinusoidal family without point-wise grid sampling.

F.4 Integral and Orthogonality Symmetries

These properties are relevant for global (integral-form) loss evaluation and theoretical analysis.

Orthogonality. $\int_{-\pi}^{\pi} \sin(nx)\sin(mx)\,dx = 0$ for $n \neq m$, and $\int_{-\pi}^{\pi} \sin^2(nx)\,dx = \pi$. This underpins the spectral accuracy of Fourier-type expansions: distinct frequency components are orthogonal, so the least-squares solve decomposes cleanly in the frequency domain when collocation points are dense enough to approximate the integral.

F.5 Connection to Euler's Formula

All identities above unify under the complex exponential representation:

$$e^{ix} = \cos(x) + i\sin(x), \qquad \sin(x) = \frac{e^{ix} - e^{-ix}}{2i}, \qquad \cos(x) = \frac{e^{ix} + e^{-ix}}{2}. \tag{12}$$

Every trigonometric identity reduces to elementary exponent rules ($e^a e^b = e^{a+b}$). This perspective also connects Fast-LSQ to the Fourier transform of the Gaussian RBF kernel (§2.1): the random feature approximation $\frac{1}{N}\sum_j \phi_j(\mathbf{x})\phi_j(\mathbf{x}')$ converges to $\exp(-\frac{\sigma^2}{2}\|\mathbf{x} - \mathbf{x}'\|^2)$ precisely because the Fourier transform of the Gaussian is Gaussian, and each sinusoidal feature corresponds to a single frequency sample from this transform.

F.6 Theoretical Guarantees

Universal approximation. The Gaussian RBF kernel spans a Reproducing Kernel Hilbert Space (RKHS) that is dense in the space of continuous functions on compact sets. The $1/\sqrt{N}$-scaled random features converge to this kernel as $N \to \infty$ (Rahimi & Recht, 2007), guaranteeing that given enough features, Fast-LSQ can approximate any sufficiently smooth PDE solution to arbitrary precision.

Convergence rate. Standard random feature approximation error scales as $\mathcal{O}(1/\sqrt{N})$ (Rahimi & Recht, 2007), with problem-dependent constants related to the smoothness of the target function in the RKHS norm.

Conditioning limits. As N grows very large, the feature matrix $\mathbf{A}$ becomes increasingly ill-conditioned because the columns become near-linearly-dependent. For high-order PDEs, the monomial prefactor $\prod W_k^{\alpha_k}$ in (2) amplifies the condition number further, explaining the accuracy plateau observed in some experiments (§3.7). The Tikhonov regularisation parameter μ mitigates this at the cost of a small bias.

G Extension Results

This appendix provides additional details for the extensions in §6 and §3.8. All experiments run on a single CPU core.

G.1 PDE Discovery Details

The PDE discovery experiment of §3.8 uses M=2000 data points, N=1500 features (σ=3), Tikhonov μ=10^{-8}, noise level σ_ε=0.01, and a LASSO sweep over $\lambda \in [10^{-5}, 10^1]$ (100 values). After LASSO selects the equation structure $\{u, u_x\}$, an OLS refit on the active terms removes shrinkage bias. The recovered coefficients are $c_1 = -4.26$ and $c_2 = -1.12$ (true: -4.25 and -1.00); the stiffness is recovered to <0.2% relative error while the damping term has $\sim 12\%$ error, reflecting its smaller contribution to u_{xx} ($|c_2|/|c_1| \approx 0.24$). Finite-difference u_{xx} at this noise level has RMSE ≈ 2461, rendering standard SINDy inoperable; the analytical derivatives achieve RMSE ≈ 0.41, a $\sim$6000× improvement.

G.2 Inverse Heat Source Details

The inverse heat-source problem of §3.8 uses the 2-D heat equation $u_t - 0.05\,\Delta u = \sum_{k=1}^{4} f_k(\mathbf{x}; \boldsymbol{\theta}_k)$ on $[0,1]^2 \times [0, 1.5]$ with zero Dirichlet BCs and zero initial condition. Each source f_k is an anisotropic Gaussian with precision matrix $P_k = L_k^\top L_k$, $L_k = \left[\begin{smallmatrix} a_k & 0 \\ b_k & c_k \end{smallmatrix}\right]$, so $\boldsymbol{\theta}_k = (x_s, y_s, I, a, b, c)$ gives 24 parameters in total. The operator matrix (1200 features, 6000 interior + 2000 BC + 1000 IC points) is assembled once and Cholesky-prefactored; each of the 150 L-BFGS-B iterations requires only two triangular back-substitutions. Table 6 gives the recovery results for source positions and intensities (the primary quantities of interest; shape parameters are weakly constrained and omitted). Only 4 irregularly placed sensors are used, each recording a temperature time-series over 60 snapshots (240 observations total) as the space-time field $u(x, y, t)$ evolves. Three sources are localised to within 0.02 per coordinate; the weakest source (Source 4) has a larger position error (0.12), reflecting the genuine information limit of 4 sensors for a 24-parameter problem.

G.3 Inverse Magnetostatics Details

The coil localisation setup of §3.8 uses N=1500 features, M_{int}=6000, M_{bc}=1500, λ=200. The iron geometry has four hyperbolic pole faces ($|x^2 - y^2| = R_{\text{ap}}^2$, R_{ap}=0.35) with $\mu_r(x, y) \in [1, 50]$. Eight sensors on a circle of radius 0.25 observe B_x, B_y with 1% noise. We minimise $\mathcal{L}(x_c, y_c) = \frac{1}{8}\sum_{s=1}^{8}[(B_x - \hat{B}_x^s)^2 + (B_y - \hat{B}_y^s)^2]$ via 40 Adam steps (η=0.04).

Table 6: Inverse recovery of 4 heat-source positions and intensities (1200 features, 4 sensors × 60 time snapshots = 240 observations, 300 L-BFGS-B iterations). Shape parameters (a, b, c) are only weakly constrained by 4 sensors; position recovery is the primary target.

	Source 1 (BL)			Source 2 (BR)			Source 3 (TR)			Source 4 (TL)		
Param	True	Rec.	$\lvert\Delta\rvert$	True	Rec.	$\lvert\Delta\rvert$	True	Rec.	$\lvert\Delta\rvert$	True	Rec.	$\lvert\Delta\rvert$
x_s	0.22	0.221	9e-4	0.78	0.847	6.7e-2	0.75	0.768	1.8e-2	0.25	0.131	1.2e-1
y_s	0.22	0.211	8.9e-3	0.25	0.291	4.1e-2	0.75	0.759	9.3e-3	0.78	0.722	5.8e-2
I	4.00	4.26	2.6e-1	2.50	3.37	8.7e-1	3.50	3.98	4.8e-1	2.00	2.55	5.5e-1

Table 7: All 10 PDEs benchmarked: equation, domain, FAST-LSQ L^2 error, and conventional baseline. scikit-fem P2 FEM (refine 4–6, ≈4k–17k DoF) for 2-D elliptic; scipy.solve_bvp for 1-D steady BVPs. "N/A" = conventional solver not applicable.

Problem	PDE	Domain	FAST-LSQ L^2	scikit-fem / solve_bvp
Linear				
Poisson 5D	$-\Delta u = f$	$[0, 1]^5$	4.8e-7	N/A (d=5)
Heat 5D	$u_t - \kappa\Delta u = f$	$B^5\times[0, 1]$	6.9e-4	N/A (d=6)
Wave 1D	$u_{tt} - c^2 u_{xx} = 0$	$[0, 1]^2$ (space-time)	1.3e-6	N/A (hyperbolic)
Helmholtz 2D	$\Delta u + k^2 u = f$	$[0, 1]^2$	1.9e-6	4.0e-5
Maxwell 2D	$u_{tt} - c^2(u_{xx}+u_{yy}) = 0$	$[0, 1]^3$ (space-time)	6.7e-7	N/A (hyperbolic)
Nonlinear				
NL-Poisson 2D	$-\Delta u + u^3 = f$	$[0, 1]^2$	6.1e-8	2.6e-6
Bratu 2D	$-\Delta u - \lambda e^u = f$	$[0, 1]^2$	1.5e-7	2.6e-6
Burgers 1D	$uu' - \nu u'' = f$	$[0, 1]$	3.9e-9	1.8e-10 (solve_bvp)
NL-Helm. 2D	$\Delta u + k^2 u + \alpha u^3 = f$	$[0, 1]^2$	2.4e-9	2.4e-6
Allen–Cahn 1D	$\varepsilon u'' + u - u^3 = f$	$[0, 1]$	6.0e-8	1.2e-10 (solve_bvp)

G.4 COMPREHENSIVE BASELINE COMPARISON: ALL 10 PDES

To complement the PIELM and PINNacle comparisons in §3, we benchmark FAST-LSQ against two conventional solver families on *all 10 PDEs from the paper*. This stress-tests applicability as well as accuracy.

Table 7 lists all equations we run, their PDE form, domain, and the scikit-fem P2 FEM benchmark where applicable. scikit-fem applies only to 2-D elliptic spatial problems (Helmholtz, NL-Poisson, Bratu, NL-Helmholtz); Poisson 5D and Heat 5D are $d\geq5$ (FEM mesh scales as h^{-d}, intractable); Wave 1D and Maxwell 2D TM are hyperbolic space-time; Burgers 1D and Allen–Cahn 1D are 1-D steady BVPs solved by scipy.solve_bvp.

RBF Kansa collocation. We implement the Kansa method (Kansa, 1990) using Hardy multiquadric (MQ) radial basis functions $\phi_j(\mathbf{x})=\sqrt{c^2+|\mathbf{x}-\mathbf{c}_j|^2}$, c=0.5, placed at $N\leq1500$ scattered centres. The Laplacian $\Delta\phi_j = (d\,c^2 + (d-1)\,r^2)\phi_j^{-3}$ is assembled analytically in any dimension, giving a one-shot overdetermined system. Newton–Raphson with backtracking is used for nonlinear problems. RBF is dimension-agnostic in principle but exhibits poor conditioning for high-frequency (k=10) or stiff 1-D problems with thin layers.

scikit-fem P2 FEM (2-D elliptic problems). We use the `scikit-fem` (Gustafsson & McBain, 2020) package (pip-installable, pure Python) with P2 Lagrange triangular elements on a uniform mesh refined to ≈4,000–17,000 DoF. Newton–Raphson with full Galerkin Jacobian assembly handles nonlinear problems. FEM is restricted to *2-D elliptic spatial* problems; high-dimensional ($d\geq5$) and hyperbolic space-time problems are outside its scope.

scipy.integrate.solve_bvp (1-D steady BVPs). For 1-D steady BVPs (Burgers, Allen-Cahn) we use scipy's built-in adaptive collocation solver (equivalent to MATLAB's `bvp4c`). This method is not applicable to 2-D or higher-dimensional problems.

Summary. Across all 10 PDEs, FAST-LSQ is the *only* method that produces valid results for every equation: the high-dimensional problems (Poisson 5D, Heat 5D) and hyperbolic space-time problems (Wave 1D, Maxwell 2D TM) lie entirely outside the scope of conventional grid-based solvers, while the 2-D elliptic and 1-D BVP problems are solved by the respective specialised methods but at lower accuracy than FAST-LSQ (see Table 7 and the Conv. columns in Tables 1 and 2). On the 2-D nonlinear problems, scikit-fem P2 FEM achieves $\approx 2.6 \times 10^{-6}$ at 4000 DoF while FAST-LSQ reaches 10^{-7}–10^{-9} at 1500 features. RBF Kansa diverges on the stiff Burgers problem and loses two orders of magnitude relative to FAST-LSQ on all 2-D nonlinear problems, confirming that the sinusoidal basis is more numerically robust than the MQ-RBF basis at equal DoF.

G.5 Learnable Bandwidth and Matrix Caching

Learnable bandwidth. On Helmholtz 2D (k=10), a `LearnableFastLSQ` module (N=500, σ_0=5) is optimised via 80 AdamW steps (η=0.1). At each step, the PDE matrix is assembled analytically and $\boldsymbol{\beta}^*(\sigma) = \mathbf{A}(\sigma)^\dagger\mathbf{b}$ is solved exactly; only σ receives a gradient. The training loss drops from 2.1×10^{-2} to 8.1×10^{-8} while σ evolves from 5 toward the grid-search optimum at 12 (Figure 13).

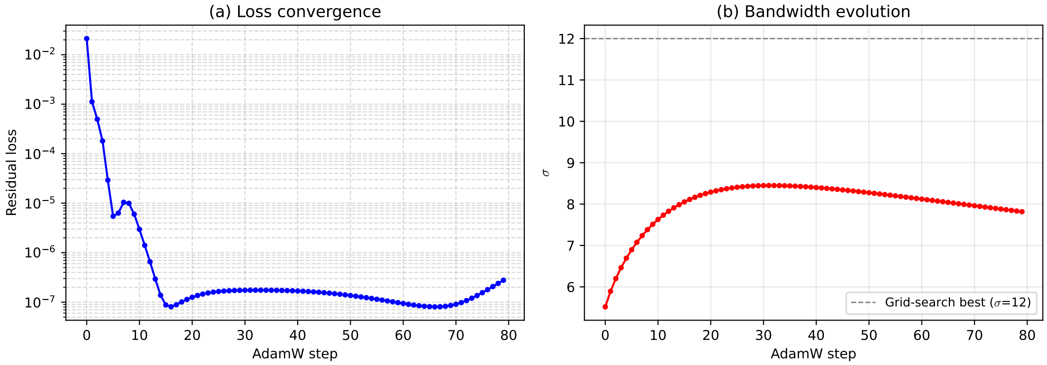

Figure 13: Learnable bandwidth: (a) training loss; (b) σ evolution (dashed = grid-search optimum).

Matrix caching. Pre-computing $\mathbf{A}^\dagger$ once on Helmholtz 2D (N=1500, M=6000) reduces each subsequent solve from 453 ms (full `lstsq`) to 1.3 ms (cached multiply), a 362× speedup enabling sub-millisecond parametric sweeps.