Meta-LSQ: A Closed-Form Sobolev Inner Solver for Few-Shot Meta-Learning

Antonin Sulc*[1]

[1]Lawrence Berkeley National Laboratory
asulc@lbl.gov

Abstract

Gradient-based meta-learners such as model-agnostic meta-learning (MAML) adapt to a new task with an inner loop of gradient descent that has to be differentiated through, and they meta-learn every network weight. Meta-LSQ instead fits each task with one regularised least-squares solve over a fixed random Fourier basis and meta-learns only three scalars, a bandwidth, a ridge and a derivative weight. Since the derivative of a sinusoid is again a sinusoid, the same solve can also take derivative observations, and the fitted model has exact closed-form derivatives of any order. On 10-shot sinusoid regression Meta-LSQ trains in 14 s and ends with a 5.5× lower error than a first-order MAML baseline that trains 36× longer, and a Gaussian-process control suggests that most of this gain comes from sharing one kernel across tasks. In simulated beam tuning at DESY's ARES facility with model derivatives it is slightly more accurate than an ALPaCA-style model and well ahead of gradient descent. Over DESY's full range of target beams, again in simulation and from measurements alone, its best setting has a median error of 7.3 µm, against 14.4 for the published final beams of their reinforcement-learning agent, which has the lower median on their 300 trials of small targets. On real sensor-calibration data a learned prior mean puts it ahead of MAML on two of four targets, but an exact Gaussian process with the same change and an ALPaCA-style model stay more accurate on most targets, and it breaks down on non-smooth targets. On noiseless targets outside the span of the basis, in-sample criteria such as generalised cross-validation choose too small a ridge.

1 Introduction

In the classical pipeline a network is trained on each task separately, from a random start and by many steps of gradient descent, which needs far more than a handful of examples. A meta-learner is instead trained on many related tasks so that a new task can be fitted from a few examples. Model-Agnostic Meta-Learning (MAML) [1], the most widely used recipe, still fits each task by gradient descent, but only for a few steps from a meta-learned start. An inner loop takes T gradient steps on a task's support set, giving adapted weights $\theta_\tau = \theta_0 - \eta \sum_{t<T} \nabla_\theta \mathcal{L}^s_\tau(\theta_t)$, and an outer loop tunes the initialisation θ_0 so that the query loss after adaptation is small. The outer gradient has to differentiate through the T inner updates, kept on the graph, which needs double backpropagation. First-order variants [2] drop the second-order terms, and Jamal et al. [3] shorten the meta-gradient graph for long inner loops with a lazy teacher–student scheme. In all of these the meta-learned object is still the whole initialisation, so the outer search runs over as many dimensions as the network has weights, typically thousands.

If the inner loop is replaced by a convex solver, the iterative fit goes away and so does part of the second-order cost. ALPaCA [4] does closed-form Bayesian linear regression with a meta-learned prior on top of a learned encoder, and was evaluated on the same sinusoid benchmark we use; R2D2 [5] puts ridge regression on a convolutional backbone, and MetaOptNet [6] a convex SVM. The inner solve is one-shot and differentiable in all three, but the features still come from a deep network that is meta-learned along with it, so the number of meta-parameters does not go down.

Approach. We keep the closed-form solve but drop the learned feature extractor, so Meta-LSQ runs no gradient descent on a task at all. Each task is fitted by one regularised least-squares solve over a frozen random Fourier basis, the outer loop differentiates through that solve instead of through gradient steps, and only two or three scalars describing the basis are meta-learned.[1] Fixing the features costs expressivity, and the method is meant for smooth, low-dimensional families that the basis represents well (failures outside it are in Sec. 5.5).

The solve can also use derivative observations at little cost. The derivative of a sine feature is the same feature shifted by a quarter period (Fig. 1b), so a derivative constraint is linear in the same coefficients as a value constraint. Both go into one

*Corresponding Author.

[1]The code and the stored results behind every number and figure are at https://github.com/sulcantonin/metalsq-public.

Proceedings of the 8th Northern Lights Deep Learning Conference (NLDL), PMLR 352, 2027.
© 2027 Antonin Sulc. This is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).

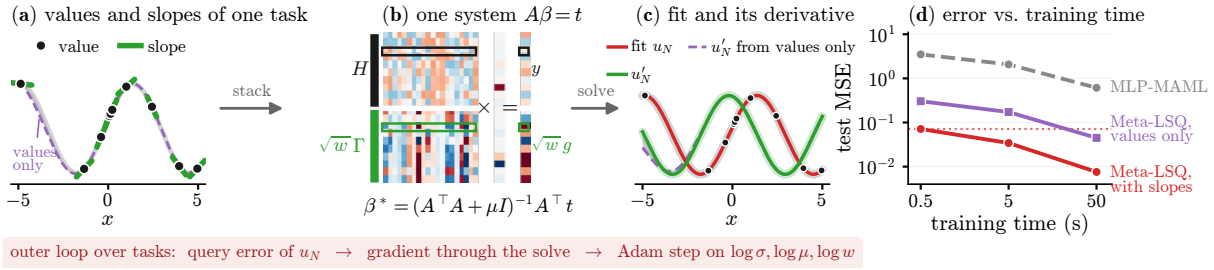


Figure 1. Meta-LSQ on one held-out sinusoid task. (a) The support set gives ten values and, at the same points, ten slopes. A value fixes a point of the curve and a slope fixes its direction there. Meta-LSQ fitted to the values alone (dashed) passes through every point but drifts from the truth where the points are sparse (shaded). **(b)** Each value is a row of H with target y_i and each slope a row of Γ with target g_i, in the same unknowns β; the two rows and targets of one point are outlined. A slope row is the value row shifted by $\pi/2$ and scaled by W_j (20 of N=200 features shown). **(c)** The fitted u_N and its closed-form derivative u'_N over the true task and its derivative (pale). The values-only fit of (a) also has a closed-form derivative (dashed), but it is wrong where the points are sparse; the slope rows correct it. Only σ, μ and w are meta-learned, by the outer loop shown at the bottom. **(d)** Test MSE against training time. Meta-LSQ-Sobolev after 0.5 s (dotted) is below MLP-MAML after 50 s (Table 2).

linear system, which we call *Sobolev fitting*. Fitting derivatives with an MLP (Sobolev training [7]) requires backpropagating through a backward pass, which roughly doubles graph depth and memory; in Meta-LSQ it adds Md rows to the linear system.

Contributions.

- Meta-LSQ (Sec. 4). It replaces MAML's inner loop with one least-squares solve on values and derivatives, and the learned initialisation with two or three scalars. The meta-gradient comes exactly from the solve, with no unrolling. The solve approximates gradient-enhanced kriging, so each scalar is a kernel length-scale or a noise ratio.
- An analysis of the derivative rows (Sec. 4.3). With ten support points, the ten slopes add only about three new independent directions to the ten values, but they lower the ratio of the largest to the tenth singular value more than a hundredfold. The individual coefficients stay unidentified, but on held-out points the fitted function's values and derivatives become three to five times more accurate.
- Experiments (Sec. 5) against MAML (a strong first-order variant, a matched training time, and MAML given the same derivatives), an ALPaCA-style model and gradient-enhanced kriging; on real sensor-calibration data; in a simulated beam-tuning loop; and on the cases where the method fails.
- A limit of standard ridge selection (Sec. 5.6). The ridge μ decides how far the fit may stray away from the data. With fewer observations than features and no noise, a tiny ridge fits the support points exactly, so methods that choose μ by how well those points are fitted, such as generalised cross-validation, can prefer the smallest μ even when the fit is far off elsewhere. The ridge then has to be set from the noise level instead. The meta-learned ridge does follow the noise level, and we give a rule for a minimum ridge in deployment.

2 Related Work

Closed-form inner solvers. The closest prior method is ALPaCA [4], whose learned encoder Meta-LSQ replaces with frozen features; R2D2 and MetaOptNet also learn their backbone (Sec. 1). iMAML [8] gets the outer gradient from the implicit function theorem instead of unrolling, and Lorraine et al. [9] approximate hypergradients the same way; here the inner optimum is the ridge solution, so they are exact up to round-off. Conditional Neural Processes [10] amortise the support-to-query mapping through a learned set-encoder, and Deep Kernel Transfer [11] learns a Gaussian-process (GP) kernel. Meta-LSQ learns no backbone or encoder, only three scalars of a fixed kernel family.

Derivative-augmented fitting and random features. Sobolev training [7] fits derivatives at double-backpropagation cost, and gradient-enhanced kriging (GEK) [12] does so in GP regression at $\mathcal{O}(M^3(d{+}1)^3)$, after Hermite RBF interpolation [13] and kriging with derivatives [14]. Random features are due to Rahimi and Recht [15], Tancik et al. [16] tied them to spectral bias, and extreme learning machines [17] and PIELM [18] fit closed-form read-outs over a frozen random basis. FastLSQ [19] solves one PDE at a time with sine features, and physics-informed networks [20] are retrained per instance; meta-learning the spectrum of such a fit across a family is, as far as we know, new.

3 Background

Random Fourier features. A smooth function can be written as a sum of waves, and if the frequencies of the waves are fixed in advance, only their amplitudes are left to fit, which is a linear problem. Random Fourier features [15] work this way. We draw N frequencies W_j at random, give each a random phase $b_j \sim \mathcal{U}[0, 2\pi)$, and model the function as

$$u_N(x) = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} \beta_j \sin(W_j^\top x + b_j). \tag{1}$$

The waves are drawn once and frozen, and only the amplitudes $\beta \in \mathbb{R}^N$ are ever fitted. How smooth the fits are is decided by the spread of the frequencies. With $W_j \sim \mathcal{N}(0, \sigma^2 I)$, a small bandwidth σ gives slow waves and smooth fits, while a large one gives fast waves that can follow fine detail but need more data to pin down. By Bochner's theorem [21], for large N fitting β behaves like kernel regression with a Gaussian kernel of length-scale $1/\sigma$, so the usual intuitions about kernel bandwidths carry over. The model is nonlinear in x but linear in β, so the inner loop can be solved in closed form.

Closed-form derivatives of the basis. Differentiating a sine shifts its phase by a quarter period and multiplies it by the frequency, $\frac{d}{dx}\sin(Wx+b) = W\sin(Wx + b + \pi/2)$. Repeating this, a derivative of any order, written with a multi-index α in d dimensions, is

$$D^\alpha \sin z_j = W_j^\alpha \sin\left(z_j + |\alpha| \tfrac{\pi}{2}\right), \tag{2}$$

where $z_j = W_j^\top x + b_j$ and $W_j^\alpha = \prod_k W_{kj}^{\alpha_k}$, so every derivative costs one sine evaluation and is exact up to round-off, without the step-size error of finite differences. Its size grows with the factor W_j^α, so the bandwidth sets how large high orders get. FastLSQ [19] builds a calculus of differential and integral operators on this identity; we only need these two facts.

4 Meta-LSQ

Scope and assumptions. We assume throughout that **(A1)** the task functions are smooth and **(A2)** the support values are clean or nearly so. When, in addition, the functions lie close to the span of the basis, as the sinusoids do by construction, the best ridge is close to zero and the solve is almost exact, and this is the setting of our main results. Fitting derivatives as well as values needs derivative observations at the support points. Analytic generators, simulators and physical models usually provide them with the values (as a sensitivity, a force or a residual); when none are available we fit the values alone, as we do on the real data (Sec. 5.2).

4.1 Adaptation as a Single Least-Squares Solve

A task gives us M support points x_i, the value y_i at each and, when available, the gradient $g_i \in \mathbb{R}^d$. Write the model (1) as $u_N(x) = h(x)^\top \beta$ with $h_j(x) = \sin(W_j^\top x + b_j)/\sqrt{N}$. By (2) its partial derivatives are linear in the same β,

$$\partial_k u_N(x) = \gamma_k(x)^\top \beta, \tag{3}$$

with $\gamma_{k,j}(x) = W_{kj}\cos(W_j^\top x + b_j)/\sqrt{N}$, so a measured value and a measured slope at x_i are two linear equations in the same unknowns,

$$h(x_i)^\top \beta = y_i, \quad \gamma_k(x_i)^\top \beta = g_{ik} \quad (k \le d). \tag{4}$$

With the $h(x_i)^\top$ as the rows of H, the $\gamma_k(x_i)^\top$ for all i and k as the rows of Γ, and the g_{ik} in g in the same order, we stack them into one design matrix and one target vector,

$$A = \begin{bmatrix} H \\ \sqrt{w}\,\Gamma \end{bmatrix}, \qquad t = \begin{bmatrix} y \\ \sqrt{w}\,g \end{bmatrix}, \tag{5}$$

where the weight w says how much a slope counts relative to a value. Adaptation is then a single ridge solve,

$$\begin{aligned} \beta^*(\sigma, \mu, w) &= \arg\min_\beta \|A\beta - t\|_2^2 + \mu\|\beta\|_2^2 \\ &= (A^\top A + \mu I)^{-1} A^\top t, \end{aligned} \tag{6}$$

where the ridge μ keeps the amplitudes small in directions the data say nothing about, and σ enters through the frequencies $W = \sigma\hat{W}$. With $w = 0$ this is plain ridge regression on the values, and with derivatives the system only gets more rows, $M(d+1)$ instead of M.

The derivatives of the fit, $\gamma_k(x)^\top \beta^*$, are available in closed form either way. The slope rows matter when slopes are measured. A fit to the values alone passes through every y_i but can have the wrong slope at and between the points (Fig. 1a), and the rows in (4) rule such fits out at the cost of a taller A (Sec. 4.3 measures what they add). We call the method with derivative rows *Meta-LSQ-Sobolev* and the one with w=0 *value-only Meta-LSQ*.

Kernel view. For $N \to \infty$ the solve tends to kernel ridge regression with an RBF kernel of length-scale $1/\sigma$ and, with $w > 0$, to the posterior mean of a GP that also observes derivatives, that is, to GEK. With a Gaussian prior of variance σ_p^2 on β and Gaussian noise of variance σ_y^2 on the values and σ_g^2 on the derivatives, (6) is the most probable β when $\mu = \sigma_y^2/\sigma_p^2$ and $w = \sigma_y^2/\sigma_g^2$ (App. B). Meta-LSQ thus learns one kernel for the whole family, and

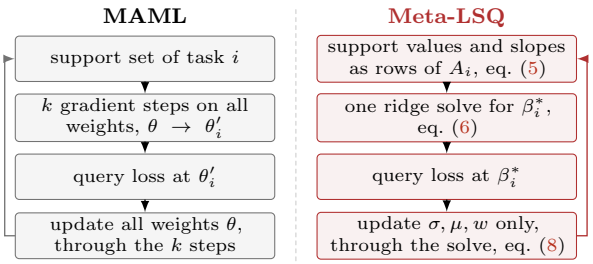


Figure 2. One meta-training step. MAML adapts with gradient steps on all weights and differentiates through them; Meta-LSQ adapts with one ridge solve over frozen features, differentiates through it by eq. (8) and learns three scalars.

the two ratios predict that μ should grow with value noise and w should fall with derivative noise, which we test in Secs. 5.4 and 5.5. The same model gives error bars on any value or derivative of the fit, but with a residual plug-in noise estimate the nominal 90% intervals cover only 31–64%, so we do not treat them as calibrated (App. J).

4.2 The Outer Objective

The three scalars should be chosen so that a single solve generalises across the family, so each training task is fitted on its support set and the fit is scored on the task's K_q query points,

$$\min_{\sigma,\mu,w>0} \; \mathbb{E}_\tau\big[\tfrac{1}{K_q}\|H_\tau^q(\sigma)\,\beta_\tau^*(\sigma,\mu,w) - y_\tau^q\|_2^2\big], \qquad (7)$$

in which only the three positive scalars are learned (two for value-only Meta-LSQ). The frozen draw $(\hat{W}, b)$ and the number of features N are set once at initialisation. N plays a role similar to the width of an MLP, and the test error hardly depends on it (App. L). We learn the logarithms of the scalars, which keeps them positive, rescale the frozen draw as $W = \sigma\hat{W}$ at every outer step, and use Adam (Fig. 2; Algorithm A.1 in App. A).

Gradient through the solve. Write $R = A^\top A + \mu I$, so that $R\beta^* = A^\top t$. Differentiating both sides with respect to a scalar $\theta \in \{\sigma, \mu, w\}$ (the implicit function theorem) gives

$$\partial_\theta \beta^* = R^{-1}\big(\partial_\theta(A^\top t) - (\partial_\theta R)\,\beta^*\big), \qquad (8)$$

for the ridge simply $\partial_\mu \beta^* = -R^{-1}\beta^*$. The query loss $\mathcal{L}$ of a task then has $\partial_\theta \mathcal{L} = (\nabla_{\beta^*}\mathcal{L})^\top \partial_\theta \beta^*$ (plus, for σ, a direct term through H^q). R^{-1} reuses the forward factorisation and nothing is unrolled.

Cost of an outer step. The solve costs $\mathcal{O}(M(d{+}1)N^2 + N^3)$, under a millisecond per task here. An outer step costs about as much as a MAML step, and Meta-LSQ trains faster in Table 1 because it needs a quarter as many steps (App. L).

4.3 Rank and Conditioning of the Stacked Design

Ten slopes can help a fit with 200 unknowns by adding new independent directions to the design or by making the system better conditioned, and we measure both.

Observation 1 (Slope rows add few new directions)**.** In exact arithmetic, $M(d{+}1) \le N$ value and slope rows at distinct points are almost surely linearly independent, so the rank could grow from M to $M(d{+}1)$. Numerically, what counts is the effective rank, the number of singular values above a relative tolerance. At a small bandwidth the features are smooth, a slope is almost determined by nearby values, and the effective rank stays well below $M(d{+}1)$.

We measured both with ten support points at the learned bandwidth (Fig. 5c). The ten value rows are essentially independent. The ten slope rows add only about three more independent directions (relative tolerance 10^{-6}), but they lower the ratio of the largest to the tenth singular value from $10^{4.7}$ to $10^{2.3}$. The full stacked design stays numerically singular, and neither design determines the 200 coefficients, yet with the slope rows the fitted function is about five times more accurate on random targets in the span of the basis, for its values and its derivatives (Fig. 5d), and three to four times on the sinusoid tasks (App. L).

5 Experiments

All experiments run single-threaded on CPU, and we only compare times within one experiment. Unless we say otherwise we report the mean over 5 seeds, where a seed redraws the frozen basis and the training and test tasks (further protocol details are in App. L).

5.1 Accuracy and Training Cost

Setup. We use the usual sinusoid family $y = a\sin(x + \phi)$ with $a \sim \mathcal{U}[0.1, 5]$, $\phi \sim \mathcal{U}[0, \pi]$ and $x \in [-5, 5]$, with $K{=}10$ support and $K_q{=}50$ query points at test time and 200 held-out tasks. We compare with a *standard MLP-MAML* (two hidden layers of 40 units, one inner step, 8000 iterations) and a *strong MLP-MAML*, a larger first-order network with five inner steps trained for up to 60k iterations. We also run MLP-MAML with a Sobolev loss on the support gradients (*MAML + Sob.*), an *ALPaCA-style* model and GEK. *Meta-LSQ* has $N{=}200$ frozen features and is trained for 2000 iterations; Meta-LSQ-Sobolev is also given the support gradients (App. L).

Table 1. 10-shot sinusoid regression, 200 test tasks, meta-iterations in parentheses. Mean ± std over 5 seeds (3 for ALPaCA-style and MAML + Sob.; one run per budget for strong MAML). "learn.": learnable parameters; "ms/t": per-task adaptation time. GEK uses an RBF kernel, ‡fitted per task by marginal likelihood or *shared and meta-learned like Meta-LSQ's scalars (App. D). Bold: best without per-task search.

method	learn.	train s	ms/t	test MSE
MAML std. (8k)	1 761	51	0.5	0.70 ± 0.13
MAML strong (10k)	8 513	130	1.8	0.328
" (30k)		399		0.144
" (60k)		512		0.104
MAML + Sob. (8k)	1 761	78	1.0	0.216 ± 0.031
ALPaCA-style (8k)	1 721	12	n/a	0.026 ± 0.017
GEK, per task‡	n/a	n/a	2.9	0.013 ± 0.008
GEK, shared (2k)*	3	n/a	n/a	0.026 ± 0.014
Meta-LSQ w=0 (2k)	2	11	0.4	0.060 ± 0.043
Meta-LSQ-Sob. (2k)	3	14	0.9	$\mathbf{0.019 \pm 0.016}$

Results. Meta-LSQ-Sobolev reaches a test MSE of 0.019 (Table 1). The strongest MAML baseline we trained, with about 8500 weights, is still 5.5× worse after training 36× longer, and its error was still falling. The three learned scalars come out similar for different basis draws (App. L).

Controls for the derivative information. Meta-LSQ-Sobolev gets support derivatives and MLP-MAML does not, so we ran three controls. Without derivatives Meta-LSQ is still 12× better than standard MAML, and MAML given the same derivatives through a Sobolev loss improves but stays 11× worse than Meta-LSQ-Sobolev. The ALPaCA-style model is about as accurate as Meta-LSQ, and value-only Meta-LSQ trained as long matches it in float64 but not in float32 (App. L). The derivatives help both methods but do not account for the gap to MAML.

Comparison with Gaussian processes. The closest comparator is GEK, a GP given the same values and derivatives. Fitted per task it is slightly more accurate (0.013, within seed spread of Meta-LSQ), and a kernel matched to the sinusoid family solves this benchmark almost exactly, but every task then needs its own marginal-likelihood search. When we instead share and meta-learn its three hyperparameters exactly as Meta-LSQ does, the exact GP reaches 0.026 and Meta-LSQ is lower on most seeds (App. D). This suggests that most of the gain over MAML comes from learning one kernel for all tasks and adapting with a closed-form solve. The random features are at least as accurate here.

Matched wall-clock budgets. Iterations are not comparable units of work, so we also give each method a fixed window of single-thread training time (0.5, 5 or 50 s), train it for as many steps as fit in the window and then test it (Table 2). A step costs about 4 ms for Meta-LSQ-Sobolev and 5 ms for MLP-MAML, so Meta-LSQ takes about a third more steps, and it also gains far more per step. After half a second (about 115 steps) Meta-LSQ-Sobolev is 8.6× more accurate than MLP-MAML after fifty seconds (about 9100 steps). Training with the ridge floor costs accuracy here, so elsewhere we apply it only at deployment (App. F).

Table 2. Test MSE after a fixed single-thread wall-clock budget (mean ± population std over 3 seeds, all evaluated on the same 200 test tasks). The floored row applies the ridge floor $\mu \geq 10^{-2}$ of Sec. 5.5 during meta-training.

method	0.5 s	5 s	50 s
MLP-MAML	$3.50 \pm .06$	$2.07 \pm .05$	$0.61 \pm .06$
Meta-LSQ w=0	$.304 \pm .076$	$.173 \pm .106$	$.045 \pm .009$
Meta-LSQ-Sob.	$\mathbf{.071} \pm .017$	$\mathbf{.034} \pm .018$	$\mathbf{.0076} \pm .0011$
+ floor	$.144 \pm .034$	$.085 \pm .033$	$.020 \pm .009$

5.2 Real Data: Sensor Calibration under Drift

Sensor drift. The response of low-cost metal-oxide gas sensors also depends on temperature, humidity and ageing, so the curve that converts a reading into a gas concentration drifts and is re-fitted from a few reference measurements. In the UCI Air Quality data [22], a device with five such sensors in a polluted urban area logged hourly readings next to a certified reference analyser for about a year (9357 hours). For one sensor and gas, a week of readings follows a noisy calibration curve that moves from week to week, so each week is a few-shot task, with no derivatives measured. Because the drift follows temperature and humidity, every method except the per-week quadratic may use both as inputs next to the reading x. Meta-LSQ adds to the sine features H a trend Φ with rows $(1, \text{inputs}, x^2)$ and fits each week by

$$\min_{\beta,\gamma} \ \|H\beta + \Phi\gamma - y\|^2 + \mu\|\beta\|^2 + \mu_\tau\|\gamma - \bar{\gamma}\|^2, \quad (9)$$

where $\bar{\gamma}$, like σ, μ and μ_τ, is meta-learned across weeks. With no data the fit is the typical calibration curve $\Phi\bar{\gamma}$. Ten noisy points move γ only as far as μ_τ allows, and the sines add the week's nonlinear correction. Our first design had no trend, so each week started from zero. MAML's learned initialisation and ALPaCA's learned prior mean play the role of $\bar{\gamma}$, and the GP gets the same trend as its mean.

Setup. Each ISO week is a task whose calibration curve has to be recovered from K=10 hours and is

Table 3. Gas-sensor calibration: mean normalised query MSE on held-out, later weeks (5 seeds, identical draws for every method). Meta-LSQ and the exact RBF GP have the learned prior mean and trend, and the inputs of every method except "quad." are chosen by cross-validation. "ALPaCA" is ALPaCA-style with a learned prior mean; "lin." and "quad." are per-week least squares. *In units of 10^{-4}. Bold: lowest at the printed precision. Spreads, cost and the design without prior mean in App. E.

target	Meta-LSQ	GP	ALPaCA	MAML	lin.	quad.
CO	0.25	**0.20**	**0.20**	0.28	0.34	0.40
NO_2	0.78	0.69	**0.42**	1.61	0.84	1.74
benzene*	1.6	**0.57**	2.2	1.4	483	3.0
NO_x	2.57	**0.62**	0.63	0.72	1.56	2.28

scored on 50 others. Weeks are split chronologically (60/40), so the test weeks (November to April) come after all training weeks, and half of their hours are colder than 95% of the training hours. We use the four sensor/gas pairs with enough weeks (CO, NO_2, benzene and NO_x, 53–56 each). We chose the design above after exploratory runs that also saw the test weeks. Whether each method uses temperature and humidity, and Meta-LSQ's ridge floor, are then picked by four-fold cross-validation over blocks of training weeks.

Results. ALPaCA-style is the most accurate on NO_2, the GP on benzene, and the two are level on CO and NO_x (Table 3). Meta-LSQ is below MAML on every seed for CO and NO_2 and level with it on benzene. Our first design (the reading alone, no prior mean or trend, N=200 and a floor of $10\mu^*$) has 30%, 12% and 25× higher error on these three. On NO_x it fails on three of five seeds, extrapolating on the colder test weeks. The last cross-validation block was hard for every method (1.07 to 2.14, Meta-LSQ 1.41), so it gave no warning specific to Meta-LSQ. This is roughly what the kernel view of Sec. 4.1 leads us to expect. Without slope rows Meta-LSQ approximates the shared GP, and with ten points per week the exact kernel is more accurate and cheaper to train (1.4 s against 5–9 s, since Meta-LSQ has N=1000 features to build; App. E). What transfers across weeks is the typical calibration curve, which the other methods learn as well; Meta-LSQ's advantages need derivative or operator rows, which these data lack.

5.3 Closed-Loop Beam Tuning in Simulation

We also put the solve inside a tuning loop, on the ARES experimental area at DESY [23]. An electron beam passes three quadrupoles, magnetic lenses that focus it in one plane and defocus it in the other, and two steerers, small dipoles that deflect it, and then hits a screen (Fig. 3). The screen gives four numbers, the horizontal and vertical position and width of the beam, and tuning means finding the five magnet strengths that bring them to a requested target in few measurements. The answer changes from trial to trial with the incoming beam and the misalignments of the quadrupoles, which the tuner does not know. Kaiser et al. [24] compared an optimiser trained by reinforcement learning with Bayesian optimisation on 300 such trials. We simulate the beamline with Cheetah [25], use their trials, starting setting and budget of 150 measurements, and score the mean absolute error of the four beam parameters at the best setting found.

Meta-training on simulated machines. Each beam parameter gets its own Meta-LSQ model, a function from the five magnet settings (scaled to $[-1, 1]$) to that parameter, with N=500 features. A meta-training task is one simulated machine, with incoming beam and misalignments drawn from the ranges of DESY's environment, measured at 60 settings around a random setting; we use 400 such tasks and none of the 300 trials. As on the sinusoids, the solve is fitted on a support set (30 settings with derivatives, 50 without) and scored on the rest, and the scalars are learned by Adam. With derivatives the support rows also hold the slopes and second derivatives of the beam parameter with respect to the magnets, taken from the differentiable simulation, which stands in for a model of the real beamline; the second-derivative rows get a weight of their

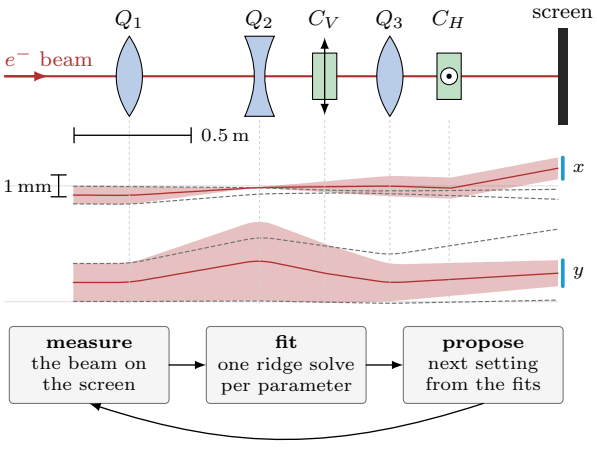


Figure 3. The ARES beamline to scale (quadrupoles drawn as lenses that focus or defocus in x, steerers as boxes marked with the plane of their kick and widened), the beam along it, and the loop of the surrogate methods. The strips show the beam's centre $\pm 2\sigma$ in x and y (magnified, bar 1 mm) on the trial of Fig. 4, at the starting setting (grey dashes) and at Meta-LSQ's tuned setting (red); the cyan bars are the target at the screen.

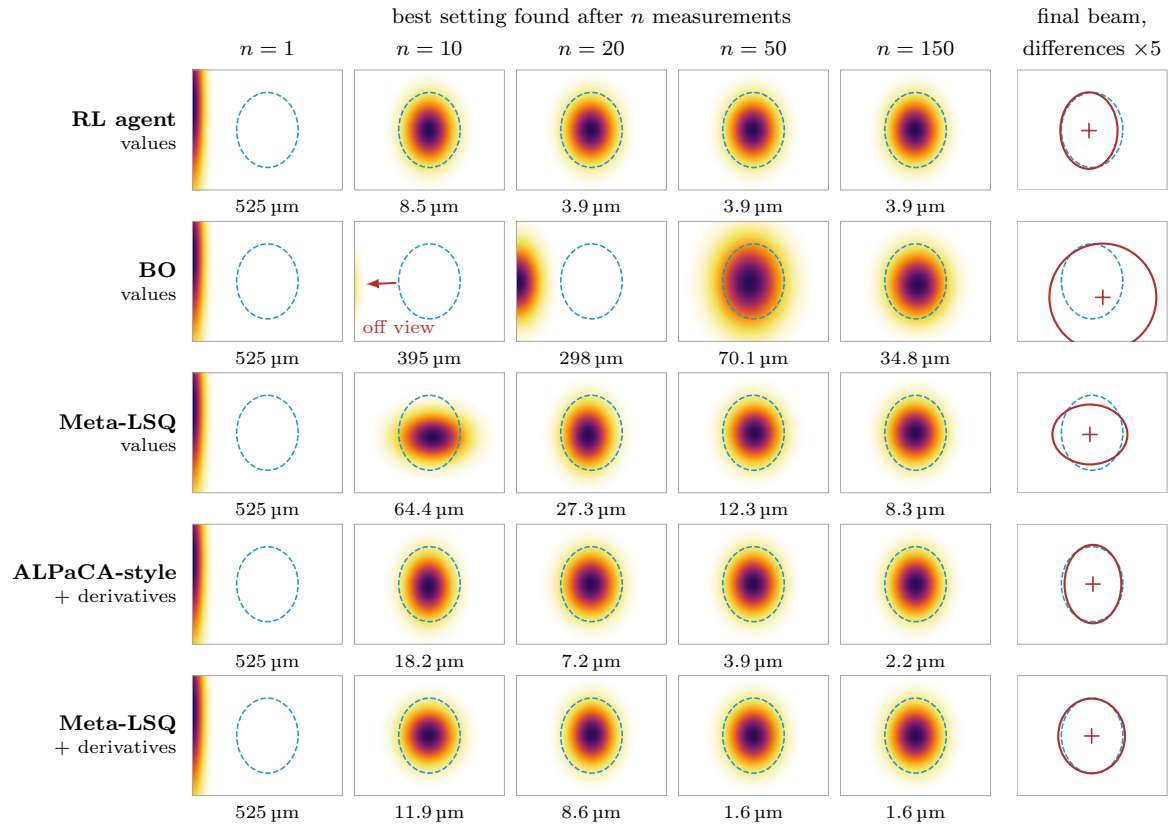


Figure 4. Tuning on the trial where Meta-LSQ with derivatives ends closest to its median: the simulated screen at the best setting after n measurements, with the mean absolute error of the four beam parameters; the dashed ellipse is the target (2σ). Last column: the final beam (solid) against the target, differences magnified five times.

own, a fourth scalar.

Tuning. Nothing is trained during a trial. For each of the 150 measurements the loop of Fig. 3 refits the four models to all measurements so far, one solve each with the learned scalars fixed, and measures next the setting whose predicted beam is closest to the target, within a small window around the last setting (App. K). ALPaCA-style models get the same information in the same loop, each variant was fixed on 60 development tasks, and DESY's agent and their BoTorch [26] optimisation run in our re-implementation of their protocol. Their optimisation uses measurements only, so with derivatives Meta-LSQ is compared with ALPaCA-style and with gradient descent.

From measurements alone the two surrogates are level (Table 4) and both are about four times more accurate than the Bayesian optimisation, while the agent has the lowest median. Its mean is above the surrogates' because we hold its magnets to their limits, which DESY's simulation did not do, and without that it is 11.8 µm (App. K). With model derivatives Meta-LSQ has the lowest error in Table 4 and removes 99% of the starting error in 16 measurements, slightly ahead of ALPaCA-style. Gradient descent on the measured error with the same derivatives ends at 14.2 µm, worse than both surrogates from values alone, so the derivatives help mainly through the fitted model. Figure 4 shows one trial.

Table 4. Beam tuning on DESY's 300 ARES trials: error of the best setting after 150 measurements (µm) and measurements needed to remove 99% of the starting error (median, with the share of trials that get there). One training run per method; protocol in App. K.

method	uses	median	mean	to 99%
RL agent [24]	values	**4.4**	17.1	39 (62%)
BO [24]	values	30.7	38.0	– (1%)
ALPaCA-style	values	7.5	**13.2**	– (46%)
Meta-LSQ	values	7.8	**13.2**	– (44%)
Gradient descent	+ deriv.	14.2	27.7	– (37%)
ALPaCA-style	+ deriv.	1.9	9.8	20 (78%)
Meta-LSQ	+ deriv.	**1.6**	**9.6**	**16** (79%)

On the first 150 trials we also took the derivatives from an imperfect model, with no misalignments and the incoming beam off by up to 20% (App. K). Meta-LSQ and ALPaCA-style then end near 11 µm, worse than Meta-LSQ from values alone (8.8 µm on the same trials) and only slightly ahead of gradient descent (12 µm). The meta-training tasks have no readout noise and no magnet hysteresis, so we leave both out of the evaluation. On a machine each reversal of a magnet also costs time, and the surrogate

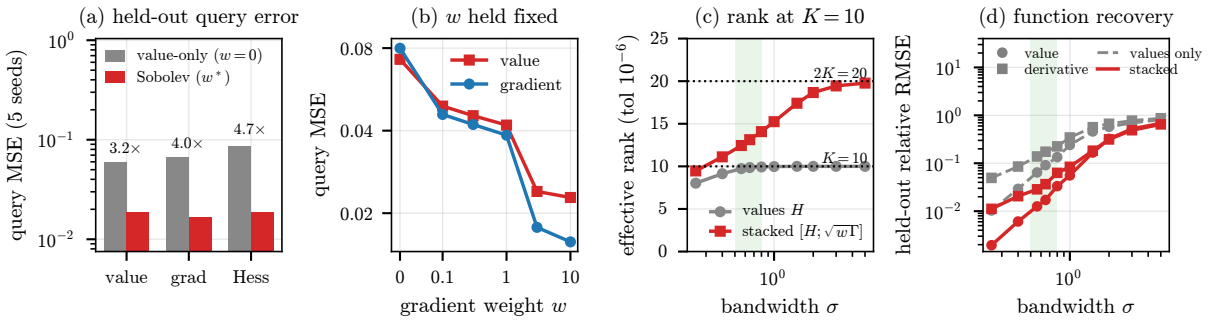


Figure 5. Effect of the derivative rows. **(a)** On held-out tasks they lower query value, gradient and Hessian error (5 seeds), though nothing above first order is supervised. **(b)** Meta-training with w held fixed: mean value and gradient error fall as w grows (3 seeds). **(c)** Effective rank at the 10-shot support set (200 draws): the derivative rows add about three directions at the learned bandwidth (shaded) and approach $2K$ only at large σ (Observation 1). **(d)** Held-out relative RMSE of an in-span random target, values-only against stacked fit (μ=10^{-5}, mean over 200 draws).

Table 5. Simulated tuning on DESY's grid of targets by the larger target beam size: error after 150 measurements (µm, median / mean), 300 targets per band. Agent and BO from the final beams DESY published, ALPaCA is the ALPaCA-style model. The last row weights the bands by their share of the grid.

size (mm)	share	agent	BO	ALPaCA	Meta-LSQ
below 0.5	49%	6.7/10	29/46	6.5/7.5	6.6/8.9
0.5–1	23%	18/152	49/80	7.7/11	7.7/12
1–1.5	9%	69/258	109/147	7.9/8.6	7.9/8.4
1.5–2	19%	278/1222	189/236	8.3/21	8.3/21
whole grid		14/295	42/99	7.3/11	7.3/12

loop reverses a quadrupole about 240 times per run against 8 for the agent, so with a 30 s cycle per reversal the agent removes 99% of the starting error sooner (4.1 against 6.6 minutes, medians; App. K).

DESY's trials ask for beam sizes of at most 0.5 mm. On their simulated grid of targets over the full range (sizes up to 2 mm), for which they published the final beams of the agent and the Bayesian optimisation, the same values-only loops end at a median of 7.3 µm, against 14.4 and 41.9. The agent is level with the surrogates below 0.5 mm, and its error grows to 278 µm for the largest targets, where the surrogates stay near 8 µm (Table 5). The published agent is scored on the last beam of its run and the surrogates on their best setting. Scored on its best setting in our own run, the agent is ahead of Meta-LSQ below 0.5 mm and behind it in the other bands (App. K).

5.4 Derivative and Operator Rows

Figure 5a and b repeat the analysis of Sec. 4.3 for the meta-learned solve on held-out tasks. Derivative rows lower the query error of the values and of every derivative order we checked; the Hessian error falls by 4.7×, although only first derivatives are supervised. With w held fixed during meta-training, a larger w gives a lower error (App. L).

Noisy and missing derivatives. As $w=\sigma_y^2/\sigma_g^2$ predicts, noisy derivatives drive the learned w down by more than an order of magnitude, while derivatives that are only missing at some support points leave it roughly unchanged (App. C).

Operator and integral rows. Any linear operator $\mathcal{L}$ built from derivatives and integrals maps a feature $\sin(Wx+b)$ to a closed-form function, so the same solve takes rows $\mathcal{L}[\sin(Wx+b)]$ with the right-hand side of an equation as targets, plus rows for the boundary conditions. For the Helmholtz equation $-u''+k^2u=f$ these rows are $(W^2+k^2)\sin(Wx+b)$, and a new problem (k,f) is solved by one ridge solve, with no training. The bandwidth and ridge are meta-learned from the equation residual at held-out points of the training problems, so no solution is ever used as a label. Against the exact solution the median error is 1.8×10^{-7}, four times lower than choosing the bandwidth for each problem at a fixed ridge; choosing it per problem at the learned ridge lowers the error further, to 5.7×10^{-8}. Integral rows turn a Volterra integro-differential equation into one linear system as well, and gradients through the solve recover its memory parameter from noisy observations to within 4% (App. G).

5.5 Out-of-Class Transfer and Failure Modes

The sinusoids lie in the span of the basis by construction, so the results above are a best case. Table 6 tests models meta-trained on sinusoids alone on two new families: **tanh**, $y=a\tanh(b(x-\phi))$, which is smooth but outside the span, and **triangle**, $y=a\,\mathrm{tri}((x-\phi)/T)$, which has kinks and any period.

Table 6. Out-of-class transfer after meta-training on sinusoids: query MSE per task, mean and median over 200 test tasks (5 seeds; 3 for t_2). The Meta-LSQ-Sobolev columns use the learned ridge μ^*, a floor $\mu \geq 10^{-2}$ at test time, and the floor with a Student-t frequency draw. A few blowups dominate the mean.

	MLP-MAML		μ^*		$\mu \geq 10^{-2}$		t_2 freq.	
family	mean	med	mean	med	mean	med	mean	med
sin	0.70	0.47	**.019**	**3**e-5	.056	1.7e-3	0.28	.044
tanh	6.0	4.6	45	0.11	0.56	.055	**0.18**	**.037**
triangle	**4.0**	3.2	18000	32	63	9.2	14	**3.0**

By the median, Meta-LSQ-Sobolev with its learned ridge beats MLP-MAML on tanh by about 40×, but its mean is far worse. The learned ridge is close to zero, so on a few unfamiliar tasks the fit goes far off where there are no support points, and these tasks dominate the mean. A more stable solver does not remove them, so they come from extrapolation (App. F). A floor $\mu \geq 10^{-2}$ at test time removes most of them, at a cost on sinusoids that stays far below MLP-MAML's error, and Meta-LSQ-Sobolev then beats MLP-MAML on tanh in the mean as well. On triangles MLP-MAML keeps by far the lower mean, and only a heavier-tailed (Student-t) frequency draw brings the median to MLP-MAML's level.

Setting the floor. Noise on the support values raises the learned ridge by nearly three orders of magnitude, as $\mu = \sigma_y^2/\sigma_p^2$ predicts, so a floor amounts to assuming some noise; with it Meta-LSQ-Sobolev stays well below MLP-MAML at every noise level (App. C). One fixed floor does not suit targets with different noise levels. On our first real-data design a floor of 10 to 30 times the learned ridge did best on the held-out weeks, and for the final design cross-validation chose 1 to 3 times (App. E).

5.6 Selecting the Ridge

We also tried to choose the ridge from the support data alone, by generalised cross-validation (GCV) and leave-out cross-validation.

Observation 2 (In-sample criteria at a vanishing ridge)**.** Suppose the design has fewer rows than features and full row rank, and the support targets are noiseless. As $\mu \to 0$ the fit passes through every support point, whatever it does away from them. A criterion built only from the residual at the support points and its degrees of freedom therefore cannot see how the fit extrapolates. GCV, for one, tends to a finite value as $\mu \to 0$, so it has no reason to avoid the smallest ridge (sketch in App. F).

GCV chose the smallest ridge on every sinusoid task and on most tanh tasks, the best choice for sinusoids and by far the worst for tanh. Leave-out cross-validation matches the floor on typical tasks but misses the rare blowups (App. F). Any kernel or random-feature model fitted to a few noiseless points shares this limit.

6 Discussion and Conclusions

We have presented Meta-LSQ, a meta-learner for smooth, low-dimensional few-shot regression, the setting of our main experiments, that fits each task by one linear solve. Derivative and operator observations enter that solve as extra rows. It works best when the task family is close to the span of the basis, and with a ridge floor it also beats MLP-MAML on smooth out-of-span tanh functions. The comparison with the shared-hyperparameter GP suggests that most of the improvement over MAML comes from sharing a three-scalar Sobolev kernel across tasks. In simulated beam tuning the solve is a competitive surrogate from measurements alone and, given model derivatives, slightly ahead of ALPaCA-style and well ahead of gradient descent. On real calibration data without derivatives it needed a learned prior mean to pass MAML, and an exact GP and an ALPaCA-style model stay more accurate, as expected without slope rows. The ALPaCA-style encoder transfers better out of class (App. L), non-smooth targets remain the main weakness, and derivative estimates degrade in high dimension (App. I). With one-hot labels the solve is a closed-form classifier, so far tested on synthetic tasks only (App. H). Integrals of the features are closed-form too, so future work could fit the solve to integral observations, such as readings averaged over a time window or a quantity known only as its total over a region. On a real beamline the inputs would also need each magnet's last direction of motion, because of hysteresis.

Acknowledgments

We thank Innovatrics for supporting this work.

LLM Disclosure

We used Claude (Anthropic; Opus 5.5) in Claude Code to write the experiment code, to help with the methodology, experimental design and interpretation of results, and to edit the manuscript, including the formal statements and proof sketches. AI tools were not used for synthetic data, conceptual frameworks, hypotheses, translation, data cleaning or qualitative data analysis. All code is publicly available on the author's GitHub, https://github.com/sulcantonin/metalsq-public.

References

[1] C. Finn, P. Abbeel, and S. Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 1126–1135.

[2] A. Nichol, J. Achiam, and J. Schulman. "On first-order meta-learning algorithms". In: *arXiv preprint arXiv:1803.02999* (2018).

[3] M. A. Jamal, L. Wang, and B. Gong. "A lazy approach to long-horizon gradient-based meta-learning". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 6557–6566.

[4] J. Harrison, A. Sharma, and M. Pavone. "Meta-learning priors for efficient online Bayesian regression". In: *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics (WAFR 2018)*. 2020, pp. 318–337.

[5] L. Bertinetto, J. F. Henriques, P. H. S. Torr, and A. Vedaldi. "Meta-learning with differentiable closed-form solvers". In: *International Conference on Learning Representations*. 2019.

[6] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. "Meta-learning with differentiable convex optimization". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 10649–10657.

[7] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Świrszcz, and R. Pascanu. "Sobolev training for neural networks". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4278–4287.

[8] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. "Meta-learning with implicit gradients". In: *Advances in Neural Information Processing Systems*. 2019, pp. 113–124.

[9] J. Lorraine, P. Vicol, and D. Duvenaud. "Optimizing millions of hyperparameters by implicit differentiation". In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. 2020, pp. 1540–1552.

[10] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. M. A. Eslami. "Conditional neural processes". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 1704–1713.

[11] M. Patacchiola, J. Turner, E. J. Crowley, M. O'Boyle, and A. J. Storkey. "Bayesian meta-learning for the few-shot setting via deep kernels". In: *Advances in Neural Information Processing Systems*. 2020, pp. 16108–16118.

[12] L. Laurent, R. Le Riche, B. Soulier, and P.-A. Boucard. "An overview of gradient-enhanced metamodels with applications". In: *Archives of Computational Methods in Engineering* 26.1 (2019), pp. 61–106.

[13] Z. Wu. "Hermite–Birkhoff Interpolation of Scattered Data by Radial Basis Functions". In: *Approximation Theory and its Applications* 8.2 (1992), pp. 1–10.

[14] M. D. Morris, T. J. Mitchell, and D. Ylvisaker. "Bayesian Design and Analysis of Computer Experiments: Use of Derivatives in Surface Prediction". In: *Technometrics* 35.3 (1993), pp. 243–255.

[15] A. Rahimi and B. Recht. "Random features for large-scale kernel machines". In: *Advances in Neural Information Processing Systems*. 2007, pp. 1177–1184.

[16] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. "Fourier features let networks learn high frequency functions in low dimensional domains". In: *Advances in Neural Information Processing Systems*. 2020, pp. 7537–7547.

[17] G. Huang, G.-B. Huang, S. Song, and K. You. "Trends in extreme learning machines: A review". In: *Neural Networks* 61 (2015), pp. 32–48.

[18] V. Dwivedi and B. Srinivasan. "Physics informed extreme learning machine (PIELM)–a rapid method for the numerical solution of partial differential equations". In: *Neurocomputing* 391 (2020), pp. 96–118.

[19] A. Sulc. "FastLSQ: Solving PDEs in one shot via Fourier features with exact analytical derivatives". In: *arXiv preprint arXiv:2602.10541* (2026).

[20] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[21] S. Bochner. "Monotone Funktionen, Stieltjessche Integrale und harmonische Analyse". In: *Mathematische Annalen* 108.1 (1933), pp. 378–410.

[22] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia. "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario". In: *Sensors and Actuators B: Chemical* 129.2 (2008), pp. 750–757.

[23] E. Panofski, R. W. Assmann, F. Burkart, U. Dorda, L. Genovese, F. Jafarinia, S. M. Jaster-Merz, M. Kellermeier, W. Kuropka, F. Lemery, B. Marchetti, D. Marx, F. Mayet, T. Vinatier, and S. Yamin. "Commissioning Results and Electron Beam Characterization with the S-Band Photoinjector at SINBAD-ARES". In: *Instruments* 5.3 (2021), p. 28.

[24] J. Kaiser, C. Xu, A. Eichler, A. Santamaria Garcia, O. Stein, E. Bründermann, W. Kuropka, H. Dinter, F. Mayet, T. Vinatier, F. Burkart, and H. Schlarb. "Reinforcement learning-trained optimisers and Bayesian optimisation for online particle accelerator tuning". In: *Scientific Reports* 14.1 (2024), p. 15733.

[25] J. Kaiser, C. Xu, A. Eichler, and A. Santamaria Garcia. "Bridging the gap between machine learning and particle accelerator physics with high-speed, differentiable simulations". In: *Physical Review Accelerators and Beams* 27.5 (2024), p. 054601.

[26] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization". In: *Advances in Neural Information Processing Systems*. 2020, pp. 21524–21538.

Algorithm A.1 Meta-LSQ training

Require: frozen $\hat{W} \in \mathbb{R}^{d\times N}$, $\hat{W}_{kj} \sim \mathcal{N}(0,1)$; phases $b_j \sim \mathcal{U}[0, 2\pi)$
Require: log-parameters $\ell_\sigma, \ell_\mu, \ell_w$; Adam
for outer step $= 1, \dots, T_{\text{outer}}$ **do**
  sample B tasks $(X_i, y_i, G_i, X_i^q, y_i^q)$
  $\sigma \leftarrow e^{\ell_\sigma}$, $\mu \leftarrow e^{\ell_\mu}$, $w \leftarrow e^{\ell_w}$, $W \leftarrow \sigma\hat{W}$
  for $i = 1, \dots, B$ **do**
    assemble H_i, Γ_i by (2); build A_i, t_i
    $\beta_i \leftarrow \texttt{solve}(A_i^\top A_i + \mu I,\ A_i^\top t_i)$ // differentiable
    $\mathcal{L}_i \leftarrow \|H_i^q \beta_i - y_i^q\|^2 / K_q$
  end for
  Adam-step on $\nabla_{\ell_\sigma,\ell_\mu,\ell_w} \frac{1}{B}\sum_i \mathcal{L}_i$
end for
return $(\sigma^*, \mu^*, w^*, \hat{W}, b)$

A Training Algorithm

B Posterior, and the Meaning of μ and w

Place a Gaussian prior $\beta \sim \mathcal{N}(0, \sigma_p^2 I)$ and independent Gaussian likelihoods on the two observation types, $y_i = h(x_i)^\top\beta + \varepsilon_i^y$ with $\varepsilon_i^y \sim \mathcal{N}(0, \sigma_y^2)$ and $g_{ik} = \gamma_k(x_i)^\top\beta + \varepsilon_{ik}^g$ with $\varepsilon_{ik}^g \sim \mathcal{N}(0, \sigma_g^2)$. The negative log-posterior is, up to constants, $\frac{1}{2}\big(\sigma_y^{-2}\|H\beta - y\|^2 + \sigma_g^{-2}\sum_k \|\Gamma_k\beta - g_k\|^2 + \sigma_p^{-2}\|\beta\|^2\big)$, and multiplying by $2\sigma_y^2$ gives the stacked objective (6) with $w = \sigma_y^2/\sigma_g^2$ and $\mu = \sigma_y^2/\sigma_p^2$. The posterior is $\beta \mid t \sim \mathcal{N}(\beta^*, \sigma_y^2(A^\top A + \mu I)^{-1})$. Any predictive derivative $D^\alpha u_N(x_\star) = c^{(\alpha)}(x_\star)^\top\beta$ is linear in β, so it is Gaussian with variance $\sigma_y^2\, c^{(\alpha)\top}(A^\top A + \mu I)^{-1}c^{(\alpha)}$, one $\mathcal{O}(N^2)$ product against the cached factor. The ridge fixes only the ratio σ_y^2/σ_p^2; the absolute scale needs an estimate of σ_y^2. Because (1) omits the factor $\sqrt{2}$, the induced GP has signal variance $\sigma_p^2/2$.

C Noise Ablations: the Ridge and w

With noise of std $\epsilon \in \{0, 0.1, 0.25, 0.5\}$ on the support values only (8000 meta-training steps for both methods, 3 seeds), the learned ridge rises from 1.9×10^{-6} to 8.6×10^{-5}, 4.2×10^{-4} and 1.2×10^{-3}, and Meta-LSQ-Sobolev reaches a clean-query test MSE of 0.0049–0.055 against about 0.65 for MLP-MAML. With noise of std $\epsilon_g \in \{0, 0.1, 0.25, 0.5\}$ on the support gradients only, the learned w falls from 10.9 to 0.29, 0.20 and 0.18. When only 2, 5 or all 10 support points carry a clean gradient row (1200 steps, 3 seeds), w stays at 3.8, 4.9 and 4.6.

Table E.1. Ridge floor on held-out weeks, applied at test time: mean / worst-week normalised query MSE (5 seeds pooled for the maximum).

floor	CO	NO_2	benzene	NO_x
μ^*	.38 / 5.8	1.03 / 5.4	.0013 / .020	.95 / 8.0
$3\mu^*$	.35 / 4.1	.92 / 3.5	.0021 / .040	.90 / 7.3
$10\mu^*$	.32 / 2.4	.86 / 2.9	.0037 / .071	.85 / 4.3
$30\mu^*$	.32 / 1.6	.84 / 2.9	.0066 / .096	.85 / 4.6
$100\mu^*$	.35 / 1.5	.89 / 2.6	.012 / .22	.94 / 4.7
$\mu \geq 10^{-2}$	.32 / 1.7	.85 / 2.9	.051 / .48	.84 / 4.3

D Gaussian-Process Comparators

All GPs get the same support values and derivatives as Meta-LSQ-Sobolev (200 tasks, 5 seeds). Fitted per task by marginal likelihood, an RBF kernel reaches 0.013 (2.9 ms per task), and a single spectral-mixture component, whose sample paths approach $A\sin(\omega x + \phi)$ as its spectral variance goes to zero, reaches 3.2×10^{-10} (102 ms per task). The shared GP has an RBF kernel with three hyperparameters (length-scale, value noise, derivative noise), learned by Adam on the query loss with Meta-LSQ's tasks, initialisation, learning rate and budget, then frozen. It reaches 0.026 after 2000 steps and 0.013 after 6000, against 0.019 and 0.0043 for Meta-LSQ-Sobolev, which is lower on 4 of 5 and on all 5 paired seeds.

E Real Data: Protocol and Full Results

Each task is one ISO week of hourly readings, with the reference analyser as target: PT08.S1 to CO, PT08.S4 to NO_2, PT08.S2 to benzene and PT08.S3 to NO_x, with 53–56 weeks of at least 60 valid hours (the NMHC reference has only eight weeks). The first 60% of weeks are for training, and the support (10) and query (50) hours of a week are a seeded split, identical for every method. Our first design gave Meta-LSQ the sensor reading as its only input, N=200 features and coefficients shrunk toward zero, and the GP the reading alone with a zero mean.

The ridge floor. Table E.1 applies a floor $\mu = c\,\mu^*$ at test time to the first design. On the noisy targets the error is flat or improving up to $c \approx 30$; on the almost noiseless benzene it degrades with c, and the fixed floor of Table 6 costs 40×.

Design of Sec. 5.2. The task set drops hours without a temperature or humidity reading. Meta-LSQ has N=1000 features with one bandwidth per input. With $A = [H, \Phi]$, $\bar{\beta} = (0, \bar{\gamma})$ and D diagonal with μ on the sine block and μ_τ on the trend,

the minimiser of (9) is $\bar{\beta} + D^{-1}A^\top(AD^{-1}A^\top + I)^{-1}(y - A\bar{\beta})$, a 10×10 solve. The GP has one length-scale per input and the learned trend as its mean, and ALPaCA-style is the encoder of Sec. 5.1 with a meta-learned prior mean on its last-layer weights. Cross-validation chose temperature and humidity for Meta-LSQ and the GP except on benzene, for ALPaCA-style on every target and for MAML only on NO_2, and floors of $3\mu^*$, μ^*, $3\mu^*$ and $3\mu^*$. On NO_x Meta-LSQ's test error per seed is 0.73, 0.78, 2.79, 5.35 and 3.19, and 56% of the test hours are colder than the 5th percentile of the training temperatures. Over the four targets (one thread) Meta-LSQ has 6–10 meta-learned parameters, trains in 5.4–8.7 s and adapts to a week in 0.29–0.49 ms; the GP has 5–9, 1.3–1.4 s and 0.06–0.07 ms, ALPaCA-style 1842, 6.9–7.0 s and 0.09–0.10 ms, and MAML 1761–1841, 8.8–9.2 s and 0.12–0.13 ms. We reached this design in two exploratory rounds that also scored the test weeks; the numbers come from one run of the design fixed afterwards, to which ALPaCA-style and the timings were added later without changing any other row.

F Extrapolation and the Ridge

Sketch of Observation 2. Write $A = U\,\mathrm{diag}(s)\,V^\top$, $c = U^\top t$ and $\rho_i = \mu/(s_i^2 + \mu)$. The in-sample residual is $\sum_i \rho_i^2 c_i^2$ and the residual degrees of freedom are $\sum_i \rho_i$. Full row rank means every $s_i > 0$, so both vanish as $\mu \to 0$, while $\mathrm{GCV} = n\sum_i \rho_i^2 c_i^2/(\sum_i \rho_i)^2$, with n rows, tends to $n\sum_i c_i^2 s_i^{-4}/(\sum_i s_i^{-2})^2$.

Solver accuracy and the tail. A stabilised least-squares solve on $[A; \sqrt{\mu}I]$ instead of the normal equations changes the in-class test MSE by less than 1% and the blowup ratio $\max|\text{prediction}|/\max|\text{target}|$ by at most 3.2 where it reaches 980, so the tail comes from extrapolation. Raising μ cuts it (Table F.1). Leave-out cross-validation over μ (7 support points to fit, 3 to score) and GCV are compared with the floor in Table F.2. With μ pinned at 10^{-2} during meta-training, one of three runs diverged by 2×10^4 steps.

G Operator and Integral Rows

Helmholtz. For $-u'' + k^2 u = f$ on $[0, 2\pi]$ with homogeneous Dirichlet conditions, $k \sim \mathcal{U}[1, 10]$ and f a random sum of low-frequency sinusoids, the rows are $(W^2 + k^2)\sin(Wx + b)$, and (σ, μ) are meta-learned from the PDE residual at held-out collocation points. Scored against the exact solution (5 seeds), the shared pair reaches a median error of 1.8×10^{-7} against 8.0×10^{-7} for choosing among seven bandwidths per task at ridge 10^{-6}; given the learned ridge, per-task selection reaches 5.7×10^{-8}.

Volterra. Integrals of the sine features are also closed-form, so

$$u'(x) + \lambda\int_0^x u(s)\,ds = f(x), \quad u(0) = 0,\ x \in [0, 1], \tag{10}$$

is one linear system. Given f and noisy observations of u at 60 points, Adam through the solve recovers $\lambda \in \{0.5, 1, 2, 4\}$ to within 0.3–3.9% under 5% observation noise.

H Few-Shot Classification

One-hot labels turn (6) into a closed-form one-vs-rest classifier. On synthetic 5-way Gaussian-blob tasks in d=2 (3 seeds) it reaches 89.0% (5-shot) and 84.0% (1-shot), against 85.1% and 80.5% for MLP-MAML.

I Log-Density Fits and Dimension

Fitting the log-density of a Gaussian mixture from samples with values and gradients (N=1500, bandwidth selected on held-out data, one seed), the value error stays at or below 6×10^{-2} for d up to 64, while gradient and Hessian errors reach 0.89 and 0.96 at d=64.

J Posterior Calibration

For value-only Meta-LSQ with support noise of 0, 0.1 and 0.25 (300 tasks, 3 seeds), nominal 90% intervals with the residual plug-in $\hat{\sigma}_y^2 = \frac{1}{M}\|H\beta^* - y\|^2$ cover 31–64%; σ_y^2=μ covers 27–68% and σ_y^2=1 about 100%.

K Beam Tuning on ARES

Protocol. We rebuilt the simulated ARES section of DESY's rl-vs-bo code in Cheetah (Gaussian beam moments, no noise, an unbounded screen). The five magnet settings are scaled to $[-1, 1]$, every method starts at $(10, -10, 0, 10, 0)$ (in m^{-2} and mrad), and each move stays within ±5% of a magnet's range around the last setting, as in DESY's Bayesian optimisation. A surrogate step fits one model per beam parameter to all measurements so far, predicts the beam at 2000 random settings in that window and measures the one predicted closest to the target (refined by gradient descent with

Table E.2. Gas-sensor calibration on the task set of Sec. 5.2: mean normalised query MSE on held-out weeks, ± std over 5 seeds. "First design" rows use the sensor reading alone, no prior mean, N=200 and a floor of $10\mu^*$; the other rows use the inputs (and, for Meta-LSQ, the floor) chosen by cross-validation.

method	CO	NO_2	benzene	NO_x
Meta-LSQ, first design	0.32 ± 0.05	0.87 ± 0.08	4.0e-3 ± 2.9e-3	0.84 ± 0.07
Meta-LSQ, prior mean and trend	0.25 ± 0.03	0.78 ± 0.08	1.6e-4 ± 1.9e-4	2.57 ± 1.9
GP RBF, first design	0.30 ± 0.03	0.84 ± 0.06	3.8e-3 ± 2.8e-3	0.81 ± 0.05
GP RBF, prior mean and trend	0.20 ± 0.01	0.69 ± 0.10	5.7e-5 ± 3.2e-5	0.62 ± 0.02
ALPaCA-style, prior mean	0.20 ± 0.01	0.42 ± 0.04	2.2e-4 ± 1.3e-4	0.63 ± 0.03
MLP-MAML	0.28 ± 0.02	1.61 ± 0.97	1.4e-4 ± 1.2e-4	0.72 ± 0.03
per-task linear	0.34 ± 0.05	0.84 ± 0.20	4.8e-2 ± 0.9e-2	1.56 ± 0.50
per-task quadratic	0.40 ± 0.18	1.74 ± 0.89	3.0e-4 ± 2.2e-4	2.28 ± 1.4

Table F.1. Per-task MSE on the triangle family across ridge values (600 tasks pooled over 3 seeds).

ridge μ	median	p90	p99	max	top-1%
$\mu^* \approx 10^{-5}$	34	3900	2.6e5	4.8e6	76%
10^{-3}	12	282	4400	1.7e4	43%
10^{-2}	7.7	124	762	3800	31%
10^{-1}	**7.1**	**74**	**182**	**582**	**15%**

Table F.2. Ridge policies on 1000 pooled per-task fits per family (5 seeds × 200 tasks, float64). This run is separate from the one behind Table 6 and pools its medians, so the two differ slightly.

family	policy	mean	median	p99
sin	tuned μ^*	0.019	2.7e-5	0.39
	floor 10^{-2}	0.054	1.5e-3	1.0
	leave-out CV	0.024	5.2e-5	0.47
	GCV	**0.0062**	**7.8e-7**	**0.091**
tanh	tuned μ^*	44	0.100	1070
	floor 10^{-2}	**0.57**	**0.057**	**8.8**
	leave-out CV	12	0.065	211
	GCV	1948	0.227	4.2e4
triangle	tuned μ^*	1.8e4	27.8	2.6e5
	floor 10^{-2}	**63**	8.8	**914**
	leave-out CV	5419	**7.1**	7.4e4
	GCV	1.5e6	54.5	3.2e7

derivatives; from values alone the predicted log error is lowered by 0.1 times the range of the measured log errors times the posterior standard deviation, normalised to at most 1, to favour unexplored settings). The surrogates, gradient descent and the Bayesian optimisation keep each quadrupole's polarity fixed (strengths from 0 to $\pm 72\,\mathrm{m}^{-2}$ in our re-implementation), whereas the agent may reverse it, and its best setting does so on 110 of the 300 trials. Meta-LSQ uses N=500 and 1500 Adam steps, ALPaCA-style two tanh layers of 40 and 8000 steps; both are meta-trained on 400 simulated tasks of 60 settings. Each model's variant (values, slopes, second derivatives) was chosen on 60 development tasks. Gradient descent steps from the best setting along the normalised gradient of the distance to the target, halving the step when a measurement does not improve. DESY's agent runs from its published weights; our copy repeats the moves of their published runs at every step and their errors (median 4.4, mean 11.3 µm). Their simulation lets the magnets leave their limits, which the agent does on 28 of 300 trials; holding the magnets to the limits raises its mean from 11.8 to 17.1 µm and changes its median little (4.3 to 4.4 µm).

Limits. Each method was trained once. The ALPaCA-style architecture was not tuned, whereas Meta-LSQ's settings come from earlier development on this task, and the development tasks were drawn from ranges chosen after looking at DESY's trials.

Imperfect model. The imperfect model of Sec. 5.3 has no misalignments and an incoming beam whose sizes and divergences are off by up to 20%, position by 0.2 mm, angle by 20 µrad and energy by 2%; the readings come from the true machine and only the derivatives from the model.

Machine time. With a 1 s readout, a 10 s ramp over a magnet's full range, 1 s settling and a 30 s cycle per reversal to clear hysteresis, the agent removes 95% of the starting error in 2.2 minutes and Meta-LSQ with derivatives in 3.3, and 99% in 4.1 and 6.6 (medians); without the cycle Meta-LSQ needs 0.6 minutes for 99%.

Targets over the full range. DESY's grid has 160,000 targets (positions within ±2 mm, sizes 20 µm to 2 mm, one incoming beam). We drew 300 from each band of Table 5. Run by us and scored on its best setting, the agent reaches 5.5, 15, 34 and 185 µm, ahead of Meta-LSQ in the first band and behind it on 82%, 95% and 97% of the targets in the others.

L Protocol and Further Results

Protocol. All experiments run single-threaded on CPU; the rows of Table 1 come from two machines,

so their times are indicative. A seed redraws the frozen basis and the training and test tasks. The outer loop starts at σ=1, μ=10^{-4}, w=1.

Baselines and training. The standard MLP-MAML has two hidden layers of 40 ReLU units and one inner step at 10^{-2} (Adam at 10^{-3}, 8000 iterations). The strong one is first-order MAML with three hidden layers of 64, 5 inner steps at 5×10^{-3} and gradient clipping. Meta-LSQ has N=200 features, Adam at 5×10^{-3} for 2000 iterations on batches of 4 tasks, and Meta-LSQ-Sobolev gets the analytic gradients $g_i = a\cos(x_i+\phi)$. The test MSE is 0.024, 0.021 and 0.023 for $N \in \{50, 100, 200\}$ (3 seeds). A solve takes 0.81 ms and an outer step 9.6 ms against 8.1 ms for MLP-MAML. The learned scalars are $\sigma = 0.65 \pm 0.09$, $\mu^* \approx 10^{-5}$ and $w = 10.9 \pm 3.4$ over 3 basis draws.

Controls. The ALPaCA-style model (two tanh layers of 40, a closed-form Bayesian last layer, 8000 steps on values) reaches 0.026 against 0.023 for Meta-LSQ-Sobolev on matched seeds, and value-only Meta-LSQ trained as long reaches 0.019 in float64 and 0.041 in float32, the precision of our other runs. Out of class, ALPaCA-style gives 0.30 on tanh against 0.64 for floored Meta-LSQ.

Derivative rows. At σ=0.65 (200 draws, tolerance 10^{-6}) the effective rank is 9.9 with values and 13.1 of 20 with slopes, and the ratio of the largest to the tenth singular value drops from $10^{4.7}$ to $10^{2.3}$. The coefficient error is about as large as the coefficients in both designs, while the held-out error of the fit falls from 0.091 to 0.017 for values and from 0.17 to 0.037 for derivatives. With w fixed in meta-training at $\{0, 0.1, 0.3, 1, 3, 10\}$ (3 seeds), the error falls from 0.073 to 0.023 for values, though not on every seed.

Bandwidth matrix and spectrum. On $y = a\sin(\omega_1 x_1 + \omega_2 x_2 + \phi)$ with $\omega_1 \sim \mathcal{U}[0.5, 1.5]$ and $\omega_2 \sim \mathcal{U}[3, 5]$, a diagonal bandwidth matrix is 3.9× better than a scalar and a triangular one a further 1.8× (5 of 5 seeds). With the floor, a Student-t_2 frequency draw cuts the triangle median from 7.99 to 2.97 (MLP-MAML: 3.15) but raises the in-class median from 0.0015 to 0.044 (3 seeds).