LEARNING WHILE SEARCHING: INVERSE DESIGN THROUGH ONLINE REFINEMENT AND OFFLINE REPLAY

Anonymous authors
Paper under double-blind review

ABSTRACT

Predicting a physical system's response from its parameters is a well-established forward problem. Inverse design is harder when the mapping is nonlinear or non-unique and each simulation is expensive. We introduce ORO (Online Refinement, Offline Reuse), a framework that learns while it searches by coupling a forward surrogate with an inverse design network. Online, the inverse network proposes a candidate for a target whose design is unknown; the simulated response corrects the surrogate, and gradients through the corrected surrogate move the inverse network toward the target, with neither simulator derivatives nor an acquisition function. Offline, the accumulated design–response pairs retrain both networks, with achieved responses serving as inverse-training targets whose designs are known, so unsuccessful proposals continue to inform learning at no extra simulation cost. On three synthetic simulators that separate non-uniqueness from ill-conditioning, at a matched budget of 2,000 simulator calls per target including initialization, ORO attains lower mean and median error than genetic search on the true simulator, Bayesian optimization, and genetic search on an offline-trained surrogate, with median errors 5.5, 51.4 and 1.4 times lower than direct genetic search. Removing offline reuse causes learning to stall. Carrying its dataset across twenty targets, ORO matches the accuracy of a 30,000-sample offline surrogate using under 10% of that budget on two of the benchmarks. On a band-pass filter designed with a commercial full-wave solver, it reaches a compliant design in about 1,200 simulations, against 16,300 for a surrogate-based genetic approach.

1 INTRODUCTION

Engineering design often begins with a desired behavior: an optical spectrum, a radio-frequency S-parameter response, a mechanical deformation, or a fluid-flow characteristic. Given a geometry, material configuration, and operating conditions, established solvers can predict the resulting response. This forward problem is well defined, even when its numerical solution is expensive. Inverse design reverses the task: it seeks physical parameters that produce a specified response, a formulation shared across electromagnetics, photonics, mechanics, and fluid dynamics (Lalau-Keraly et al., 2013; Wu et al., 2024).

Three difficulties make this substantially harder than forward prediction. The inverse relation is often non-unique: different configurations may produce indistinguishable responses, so a network trained to reproduce individual design labels faces conflicting supervision even though any of several designs would satisfy the specification. Response-based evaluation is therefore more appropriate than requiring recovery of a particular parameter vector (Liu et al., 2018). Even a uniquely solvable problem can be nonlinear, nonconvex, and poorly conditioned, with narrow curved valleys and sensitivities differing by orders of magnitude. Finally, physical and manufacturing constraints restrict feasible designs, and a requested response may not be exactly attainable; the practical objective is an admissible design with sufficiently small response error.

These challenges compound when the forward model is an expensive black box. A high-fidelity simulation may take minutes or hours, limiting both optimization and training-data generation, and broad sampling wastes much of that budget on responses far from the specification. Learning introduces a further difficulty: designs proposed during optimization can leave the region covered by the training data, where a surrogate may predict an attractive response for a design that performs

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

poorly under the true simulator—a well-known failure mode in model-based optimization (Trabucco et al., 2021). The central problem is thus to coordinate candidate selection and model learning under a limited evaluation budget.

Classical optimization solves inverse design by repeatedly generating and evaluating candidates. Genetic algorithms search complex design spaces without simulator gradients and have been applied to photonic and electromagnetic design for many years (Sanchez-Dehesa et al., 2004), but they have two limitations relevant here. Every candidate costs one simulation, and although each simulation returns a full response curve, the optimizer reduces it to a single fitness value, so most of the acquired information is discarded. In addition, each search targets one specification, and the evaluations it consumed do not transfer to the next. When simulator gradients are available, adjoint optimization computes sensitivities for many design variables from few field solves, but it requires access to sensitivity information unavailable in the black-box setting considered here.

Surrogate-assisted methods instead learn an approximation of the forward process and optimize through it, as in the neural-adjoint method (Ren et al., 2020). A fixed surrogate, however, inherits the coverage of its training set, which must be generated before the specification is known: accuracy is distributed uniformly while the search concentrates in one region. Adaptive sampling addresses the choice of evaluations explicitly. Bayesian optimization maintains a probabilistic model and selects queries through an acquisition rule balancing predicted performance and uncertainty (Snoek et al., 2012; Eriksson et al., 2019), and AutoTandemML combines predictive-uncertainty sampling with tandem networks (Grbcic et al., 2025). These methods reuse observations effectively, but select new evaluations through a separate criterion that must itself be maximized over the design space at every round, and that criterion is target-agnostic: it seeks where the forward model is uncertain, not where the requested design lies. Physics-driven generative optimization couples generation with learning more directly (Jiang & Fan, 2019), but obtains its training gradients from adjoint simulations.

Proposed approach. We propose *Online Refinement, Offline Reuse* (ORO), a framework that learns while it searches. It maintains a differentiable forward surrogate S_ϕ, an inverse design network D_θ, and an accumulating dataset of simulator observations. During online refinement, the inverse network is fed the target performance and generates $x_t = D_\theta(y^\star)$; the simulator returns $y_t = F(x_t)$. This single observation first updates the surrogate to better predict the proposed design's response. With the surrogate then held fixed, the inverse network is updated by differentiating $\ell(S_\phi(D_\theta(y^\star)), y^\star)$. The simulator supplies ground-truth responses; the surrogate supplies the approximate gradients that move the next proposal.

This arrangement has a consequence that distinguishes ORO from the adaptive-sampling methods above: *no acquisition function exists*. Because the inverse network is conditioned on the target, the next query is its forward pass, and it moves between rounds only because θ moved under the target-error gradient. There is no scalar utility to design, no exploration–exploitation weighting to tune, and no inner maximization over the design space. Sampling is target-directed as a by-product of learning the inverse map. The queried region is also, by construction, where the surrogate most needs to be accurate. Unlike (Jiang & Fan, 2019), only response evaluations are required.

Periodic offline phases then reuse every accumulated design–response pair. The surrogate is retrained on the observed forward relationship, and each achieved response y_i becomes an input target for inverse training through $\ell(S_\phi(D_\theta(y_i)), y_i)$. The two phases differ in what they can teach. The requested target $y^\star$ has no known solution—that is the problem being solved—so the online phase supplies only a direction of improvement. Every response the simulator returns, by contrast, has a known and valid solution: the design that produced it. Offline reuse therefore trains the inverse map on problems that are already solved and applies it to the one that is not, so a proposal that missed $y^\star$ still contributes a realizable target, reproducible by any suitable design rather than by the original parameter vector. Reinterpreting achieved outcomes as useful goals is conceptually related to hindsight experience replay (Andrychowicz et al., 2017); here it operates through forward-model fitting and inverse response reconstruction in a static design problem. In both phases the surrogate is updated first and then frozen while the inverse network is fitted through it, so each simulation contributes to four gradient updates and offline reuse consumes no additional simulator calls.

In summary, this paper makes the following contributions:

- **A framework in which acquisition is not a separate component.** ORO couples a target-conditioned inverse network with an adaptively corrected forward surrogate. Because the inverse network is conditioned on the target, it serves simultaneously as the design model and as the query rule, eliminating the acquisition criterion and its inner maximization that adaptive-sampling methods require, and requiring no simulator derivatives.
- **Alternating refinement and achieved-response reuse.** The two phases extract different information from the same observation: online updates move the design toward the requested response, while offline reuse treats each achieved response as a valid specification in its own right, so the networks learn the simulator's behavior rather than only the neighborhood of one target.
- **A benchmark suite that separates sources of difficulty.** Gaussian resonances with $K!$ equivalent solutions, a Rosenbrock-style curved valley with a unique solution, and rational pole–zero responses resembling scattering parameters, which combine both. This relates performance to structural properties of the landscape rather than reporting problem by problem.
- **Controlled evaluation under matched budgets.** Every method receives an identical number of simulator calls, all counted, including initialization. A single hyperparameter configuration is used throughout, with no per-landscape tuning. Given sufficient budget ORO achieves lower response error than genetic search on the true simulator, Bayesian optimization, and genetic search on an offline-trained surrogate on all three benchmarks; below a crossover budget direct genetic search remains competitive, and we report where this occurs.
- **An ablation isolating the mechanism.** Removing only the offline reuse phase, holding budget, initialization, and architecture fixed, causes learning to stall, indicating that data reuse rather than architecture drives the observed behavior.
- **Amortization across targets.** Presented with a sequence of specifications, ORO carries its dataset and networks forward and solves later targets more cheaply, reducing per-target cost by $3.2\times$ and $2.6\times$ on two of the three benchmarks relative to an otherwise identical variant that is reset before each target. On the third, the hardest, the effect is absent.
- **Application to full-wave electromagnetic design.** We apply ORO to bandpass filter design with a commercial full-wave solver, obtaining a working design in approximately $1{,}200$ simulations. The framework additionally underlies an internal design workflow in our group covering several classes of radio-frequency components; those results cannot be released.

2 METHOD

2.1 PROBLEM SETUP

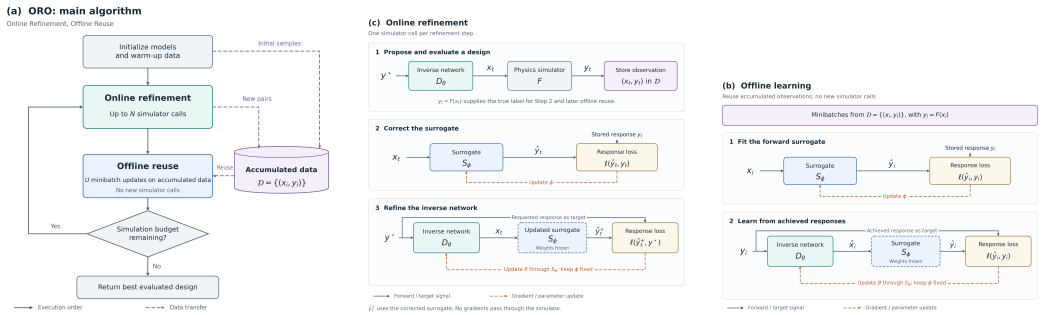
Let $F : \mathcal{X} \to \mathcal{Y}$ denote the simulator, mapping an admissible design $x \in \mathcal{X} \subset \mathbb{R}^d$, where d is the number of design parameters, to a response $y = F(x) \in \mathbb{R}^m$ sampled at m frequency (or spatial) points. Given a target response $y^\star$, the inverse design objective is

$$x^\star \in \arg\min_{x \in \mathcal{X}} \ell(F(x), y^\star), \qquad \text{where } \ell(a, b) = \tfrac{1}{m}\|a - b\|_2^2. \tag{1}$$

We assume the weakest practical form of simulator access: F is a deterministic black box returning only $y = F(x)$. No derivatives, adjoint sensitivities, or solver internals are available, and F dominates cost, so all methods are measured by the number of calls to F rather than by wall-clock time. Two structural properties of equation 1 shape the method. The solution is generally non-unique: distinct designs may satisfy $F(x_1) = F(x_2)$, so supervision in parameter space is ill-posed and all losses below are defined in response space. And $y^\star$ need not be exactly attainable, so the objective is an acceptable design with sufficiently small response error rather than an exact inverse.

2.2 NETWORKS

ORO maintains two networks. The **forward surrogate** $S_\phi : \mathbb{R}^d \to \mathbb{R}^m$ approximates F; being differentiable, it supplies the gradients that F cannot. The **inverse design network** $D_\theta : \mathbb{R}^m \to \mathcal{X}$



(a) Main Algorithm (b) ORO online refinement. (c) ORO offline learning

Figure 1: ORO alternates online refinement with offline reuse of accumulated design–response pairs. In online refinement (b), the inverse network is trained toward the requested response y^*, whose solution is unknown. In offline reuse (c), it is trained toward achieved responses y_i drawn from $\mathcal{D}$, each of which is attainable by construction since the design that produced it was simulated. In both phases the surrogate is fitted first and then frozen while the inverse network is fitted through it; gradients never pass through the simulator. Offline reuse consumes no simulator calls.

maps a desired response to design parameters. Because it is conditioned on the response, a design is obtained in a single forward pass and no search is performed at inference. Admissibility is enforced in the output layer rather than by constrained optimization: each parameter group passes through a bounded or positive activation matched to its physical range, so every proposal satisfies $D_\theta(\cdot) \in \mathcal{X}$ by construction. Both networks are multilayer perceptrons; architectures are given in Appendix C.1, and the framework is agnostic to these choices.

Both phases below follow the same update pattern: the surrogate is fitted first, then frozen while the inverse network is fitted through it. Gradients pass through S_ϕ but never modify it during an inverse-network update.

2.3 Online refinement

Online refinement operates on the current target $y^\star$ and consumes one simulator call per step (Fig. 1b). At step t the design network proposes

$$\hat{x}_t = D_\theta(y^\star)\,, \qquad y_t = F(\hat{x}_t), \tag{2}$$

where equation 2 contains the only evaluation of F in the step. That single observation is then used twice. First the surrogate is corrected at exactly the point the design network currently proposes,

$$\mathcal{L}_S^{\text{on}}(\phi) = \ell(S_\phi(\hat{x}_t),\, y_t)\,, \tag{3}$$

so the correction lands where surrogate error most affects the next proposal. Then, with ϕ held fixed, the design network is updated by differentiating the target error through the freshly corrected surrogate,

$$\mathcal{L}_D^{\text{on}}(\theta) = \ell(S_\phi(D_\theta(y^\star))\,,\, y^\star)\,. \tag{4}$$

The simulator supplies ground-truth responses; the surrogate supplies the approximate gradients that move the proposal. The pair $(\hat{x}_t, y_t)$ is appended to $\mathcal{D}$ regardless of its error, and the best design observed so far is retained as x_{best}.

No acquisition function. Equations 2–4 contain no acquisition criterion. The next query is the forward pass of the network being trained, and it moves between steps only because θ moved under the target-error gradient. There is no scalar utility to specify, no exploration–exploitation weighting to tune, and no inner maximization over $\mathcal{X}$ at any round. Sampling is target-directed as a consequence of conditioning D_θ on $y^\star$, not through a criterion placed beside the learner.

2.4 Offline reuse

Every N online steps, both networks are retrained on the whole accumulated dataset $\mathcal{D}$ (Fig. 1c). This phase makes *no* calls to F: it revisits observations already paid for. The surrogate is fitted to the

Algorithm 1 ORO: Online Refinement, Offline Reuse

Require: target $y^\star$; simulator F; budget B; online burst N; offline updates U; batch size n_b

1: $\mathcal{D} \leftarrow \{(x_i, F(x_i))\}_{i=1}^{n_0}$; $b \leftarrow n_0$ ▷ warm-up, charged to B
2: pre-train S_ϕ with equation 5, then D_θ with equation 6
3: **while** $b < B$ **do**
4: **for** $t = 1$ **to** $\min(N,\ B - b)$ **do** ▷ **online**
5: $\hat{x} \leftarrow D_\theta(y^\star)$; $y \leftarrow F(\hat{x})$; $b \leftarrow b + 1$ ▷ only call to F
6: update ϕ with equation 3
7: freeze ϕ; update θ with equation 4
8: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\hat{x}, y)\}$; update x_{best}
9: **end for**
10: **for** $u = 1$ **to** U **do** ▷ **offline**, 0 calls to F
11: sample $\mathcal{B} \sim \mathcal{D}$, $|\mathcal{B}| = n_b$; update ϕ with equation 5
12: **end for**
13: freeze ϕ
14: **for** $u = 1$ **to** U **do**
15: sample $\mathcal{B} \sim \mathcal{D}$, $|\mathcal{B}| = n_b$; update θ with equation 6
16: **end for**
17: **end while**
18: **return** x_{best}

observed forward relationship,

$$\mathcal{L}_S^{\text{off}}(\phi) = \frac{1}{|\mathcal{B}|} \sum_{(x_i, y_i) \in \mathcal{B}} \ell(S_\phi(x_i),\, y_i)\,, \qquad \mathcal{B} \subset \mathcal{D}, \tag{5}$$

and is then frozen while the design network is trained using each *achieved* response y_i as its own input target,

$$\mathcal{L}_D^{\text{off}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{(x_i, y_i) \in \mathcal{B}} \ell(S_\phi(D_\theta(y_i))\,,\, y_i)\,. \tag{6}$$

The inputs to D_θ differ between the two phases, and this is the central distinction: online it is the requested target $y^\star$ equation 4, offline it is a response y_i the simulator has actually produced equation 6. The requested target does not appear in equation 6 at all.

That difference determines what each phase can teach. The requested target has no known solution—that is the problem being solved—so the online phase supplies only a direction of improvement. Every $y_i \in \mathcal{D}$, by contrast, has a known and valid solution: the design x_i that produced it. Offline reuse therefore trains the inverse map on problems that are already solved and applies it to the one that is not. A design that missed $y^\star$ still realises *some* response y_i and is a correct answer to the specification y_i; because the loss is in response space, D_θ may reproduce y_i with any admissible design and need not recover x_i, which is the appropriate supervision under non-uniqueness. Reinterpreting achieved outcomes as valid goals is conceptually related to hindsight experience replay (Andrychowicz et al., 2017), realised here through forward-model fitting and inverse response reconstruction in a static design problem.

Across the two phases each simulated pair therefore contributes to four gradient updates–two online and two offline–at the cost of a single call to F.

2.5 The full procedure

Fig. 1a shows the complete cycle, and Algorithm 1 states it precisely. Starting from a warm-up set of n_0 simulated designs, ORO alternates a burst of N online steps with one offline phase until the budget B is exhausted, returning x_{best}.

Budget accounting. Line 5 is the only call to F in the loop, so the total simulator cost is exactly n_0 plus the number of online steps. The offline phase contributes nothing, and N, U, n_b and all learning rates affect wall-clock but never simulator count. All comparisons in Section 3 equalise this total, including the warm-up n_0.

Offline compute. The offline phase performs a fixed number U of minibatch updates per network rather than a fixed number of epochs. Its cost is therefore independent of $|\mathcal{D}|$, which grows by N every cycle: the effective epoch count $Un_b/|\mathcal{D}|$ falls as the dataset grows, so later cycles make fewer passes over a larger and more diverse dataset at constant cost. Under an epoch-based schedule the same run would grow super-linearly in wall-clock while the simulator budget stayed fixed. We set U to match a setting that performs well at small $|\mathcal{D}|$ and hold it constant thereafter (Appendix C).

Optimizer state. Each network uses a single optimizer shared across both phases, so momentum and step counts are continuous across phase switches. Two optimizers over the same parameters would maintain moment estimates that go stale against each other's updates.

2.6 Sequences of targets

Nothing in Algorithm 1 ties $\mathcal{D}$, ϕ or θ to a single specification. When several targets are to be designed, ORO carries all three forward: the dataset accumulates across targets and both networks resume from their current weights. Because D_θ is conditioned on the response, a pair (x_i, y_i) collected while pursuing one target constrains the inverse map in a neighbourhood of y_i that a later target may occupy, so it remains useful after the target that produced it has been solved. Methods that search per target have no equivalent mechanism: each run begins from nothing and its evaluations are discarded when it ends. Section 3.5 measures the resulting effect against an otherwise identical variant that is reset before each target.

3 Experiments

3.1 Benchmark simulators

Evaluating an inverse-design method on a single physical problem confounds the method with the problem. We therefore use three simulators chosen so that the two principal difficulties of inverse design, non-uniqueness and ill-conditioning, vary independently across them. All three map a design $x \in \mathbb{R}^{15}$ ($K = 5$ sections, three parameters each) to a response sampled at $m = 200$ points and share the same admissible box $\mathcal{X}$, so they are interchangeable without altering any model, baseline, or hyperparameter. Full definitions are given in Appendix A.

Resonance is a sum of K Gaussian resonances. Permuting the section labels leaves the response unchanged, so each response has $K! = 120$ distinct parameter preimages. **Rosenbrock** applies the curved-valley warp $z_0 = u_0$, $z_i = 10(u_i - u_{i-1}^2)$ to the normalised parameters and maps the result through a fixed orthonormal basis, so the searched landscape is exactly a Rosenbrock valley centred on the target's own parameters. The warp is recursively invertible, making this simulator a *bijection*: it presents ill-conditioning without non-uniqueness and serves as a control. **Rational** is a cascade of K second-order sections, each a complex pole pair with a transmission zero, which is the form taken by a filter or matching-network response; quality factors span $Q_i \in [3, 100]$ and the bounded output mimics $|S_{11}|^2$. Near a high-Q pole a small parameter change produces a large response change, so conditioning varies sharply across $\mathcal{X}$. It retains the $K!$ section symmetry, which is physically real.

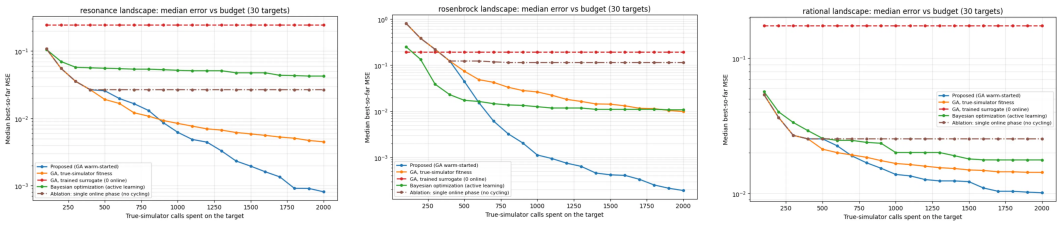
Table 1 confirms that the benchmarks exhibit the difficulties the method addresses, and that those difficulties are separable. All three are non-convex at every sampled point, with conditioning measured as the ratio of largest to smallest absolute Hessian eigenvalue at 25 random designs. A single inverse network trained with the parameter-space loss $\mathrm{MSE}(x, D(y))$ diverges on validation on the two simulators with $K!$ equivalent solutions ($1.71\times$ and $1.56\times$) but not on the bijective control ($1.11\times$), under identical architecture, data and schedule. The cause is structural: two designs with identical responses present the same input with different labels, and the squared-loss minimiser is their average, generally not a valid design. Doubling the training set increases the divergence rather than reducing it, and supervision in response space avoids the problem by construction, lowering true response error by $1.9\times$, $3.3\times$ and $1.4\times$.

3.2 Baselines and protocol

We compare ORO against three baselines. **GA (true simulator)** evaluates every candidate with F, which is standard practice in electromagnetic design. **GA (offline surrogate)** spends its entire budget

Table 1: Structural properties of the three benchmark simulators. Divergence is the ratio of final to minimum validation loss when a single inverse network is trained with a parameter-space loss (Appendix B); values near 1 indicate no divergence. Response error is the mean true-simulator error of the designs produced on held-out targets.

	Resonance	Rosenbrock	Rational
Global optima	$K! = 120$	1	$K! = 120$
Non-convex (of 25 pts)	25	25	25
Hessian cond., median	1.4×10^3	3.4×10^3	2.9×10^4
Hessian cond., max	5.8×10^4	1.6×10^4	5.6×10^5
Direct-learning divergence	$1.71\times$	$1.11\times$	$1.56\times$
Response error, direct	0.603	0.0277	0.144
Response error, tandem	0.311	0.0083	0.102



(a) Resonance (b) Rosenbrock (c) Rational

Figure 2: Median best-so-far response error over 30 held-out targets against true-simulator calls. All methods receive identical budgets and every call is counted. ORO, the ablation and GA share a warm-up search and coincide up to call 400 by construction. ORO attains the lowest error at the full budget on all three benchmarks; GA leads below the crossover, and the ablation flattens as soon as offline reuse is removed.

generating a random dataset, trains a surrogate on it, and searches that surrogate at negligible online cost. **Bayesian optimization** uses a Gaussian process with an ARD kernel and expected improvement; responses are standardised before fitting, as a zero-mean GP prior requires. Implementation details, architectures and hyperparameters are given in Appendix C; a single configuration is used for all three benchmarks, with none tuned per landscape.

Every method receives an identical number of calls to F per target, and *all* calls are counted, including warm-up and surrogate training data. Performance is best-so-far response error against calls spent, over 30 targets drawn independently of the training data. Because the error distribution across targets is heavy-tailed, we report medians here and means with standard errors in Appendix D, together with paired win rates and sign tests.

3.3 Budget-matched comparison

Figure 2 reports median best-so-far error against calls to F. ORO shares its warm-up with GA on the true simulator: a single genetic search is run per target and its first 400 evaluations serve both as ORO's warm-up dataset and as the opening segment of the GA baseline, so ORO, the ablation and GA coincide up to call 400 by construction. Table 2 gives the error at the full budget of 2,000 calls.

At the full budget ORO attains the lowest error on all three benchmarks, with median improvements over GA on the true simulator of $5.5\times$, $51.4\times$ and $1.4\times$ and over Bayesian optimization of $52.4\times$, $56.2\times$ and $1.7\times$ on resonance, Rosenbrock and rational respectively. All six win rates are significant under a two-sided sign test.

The advantage depends on both budget and landscape. Below a crossover of roughly 400 to 900 calls, visible in Fig. 2, GA on the true simulator is the better choice: it is productive immediately, whereas ORO must first correct its surrogate in the region the inverse network explores. We do not tabulate the crossovers, since they depend on how the budget is divided between warm-up and refinement

Table 2: Median response error over 30 targets at 2,000 calls per target, with paired win rates against the two strongest baselines.

	Resonance	Rosenbrock	Rational
ORO (proposed)	**0.000811**	**0.000191**	**0.010104**
GA, true simulator	0.004483	0.009823	0.014300
Bayesian optimization	0.042507	0.010742	0.017681
GA, offline surrogate	0.246320	0.194249	0.174991
Ablation (no offline reuse)	0.026667	0.113936	0.025327
Win rate vs. GA	28/30 (p=8.7×10^{-7})	30/30 (p=1.9×10^{-9})	24/30 (p=1.4×10^{-3})
Win rate vs. BO	30/30 (p=1.9×10^{-9})	30/30 (p=1.9×10^{-9})	23/30 (p=5.2×10^{-3})

rather than on the method alone. Across landscapes the margin narrows with difficulty, from $51.4\times$ on Rosenbrock to $5.5\times$ on resonance to $1.4\times$ on rational. This tracks Table 1: Rosenbrock is ill-conditioned but has a unique solution, resonance adds $K!$ equivalent solutions, and rational combines those with the most severe conditioning measured.

Of the remaining baselines, the offline-surrogate GA plateaus immediately, since its budget is spent before the target is known, and is the weakest method on resonance and Rosenbrock by two to three orders of magnitude. Bayesian optimization is landscape-dependent: it leads all methods on Rosenbrock below roughly 550 calls but is an order of magnitude weaker than GA on resonance. We have no confirmed explanation for the resonance case—permutation symmetry is ruled out by the rational benchmark, which shares it and on which BO performs well—and report the observation without a causal claim.

3.4 Ablation: the role of offline reuse

Removing only the offline reuse phase, holding budget, warm-up, architecture, optimizer and learning rates fixed, causes ORO to stall on all three benchmarks. The ablated variant coincides with the full method through the shared warm-up and then flattens almost immediately, ending $32.9\times$, $596.5\times$ and $2.5\times$ above it on resonance, Rosenbrock and rational. Without the offline phase the network is never trained on responses whose solutions are known (Section 2.4), and the surrogate, corrected at a single point per step, becomes accurate only in a shrinking neighbourhood of the current proposal, so the gradient reaching the inverse network carries little information. Because the offline phase issues no calls to F, both variants consume identical simulator budgets, and the gap is attributable entirely to how collected data is used rather than how much of it is collected.

3.5 Designing many targets with few simulations

Surrogate-based inverse design replaces the simulator with a model trained once on a large dataset, so that many targets can later be designed cheaply. We test whether ORO makes that up-front investment unnecessary. Twenty targets are presented in sequence, with ORO carrying its accumulated dataset and networks forward (Section 2.6) and stopping each target once it matches the accuracy that a GA achieves on a surrogate trained offline on $30{,}000$ random designs. A control restores the warm-up state before every target, isolating the effect of accumulation.

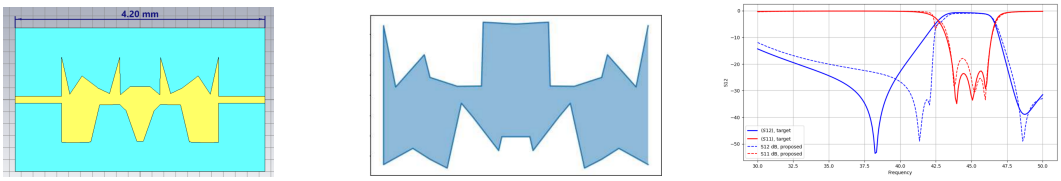
Including warm-up, ORO matches the reference on all twenty targets with $2{,}856$ simulations on resonance and $2{,}118$ on Rosenbrock—9.5% and 7.1% of what the surrogate costs to build before it designs anything. Accumulation contributes directly: per-target cost is $3.2\times$ and $2.6\times$ lower than with the reset control. On rational, accumulation gives no measurable benefit, and ORO fails to match the reference within budget on 8 of 60 instances; the remaining instances are matched at about a fifth of the surrogate's cost. This landscape also shows the smallest margin in Section 3.3.

3.6 Band-pass filter design with a full-wave solver

To test whether the benchmark conclusions transfer to production, we apply ORO to millimetre-wave band-pass filter design in CST, adopting unchanged the geometry and parameterisation of Elbadrawy et al. (2026): a single-layer irregular vector-graphics structure defined by controllable vertices under a

Table 3: Simulator calls needed to match a GA searching a 30,000-sample offline surrogate, over 20 sequential targets (3 seeds, 60 instances per row). Totals exclude the 2,000-call warm-up. Entries marked $*$ include instances that never matched the reference within budget and are held at the cap, so they are lower bounds.

Landscape	Variant	Missed	Per target	Total
Resonance	Accumulating	0/60	**42.8**	**856**
	Reset	0/60	136.0	2,721
Rosenbrock	Accumulating	0/60	**5.9**	**118**
	Reset	0/60	15.6	313
Rational	Accumulating	8/60	195.9*	3,918*
	Reset	3/60	191.7*	3,834*



(a) Geometry produced by ORO (b) Reference geometry (c) Simulated responses

Figure 3: Band-pass filter design with a commercial full-wave solver. (a) Geometry produced by ORO. (b) Geometry of Elbadrawy et al. (2026), whose simulated response defines the target. (c) Simulated S_{11} and S_{21} of both designs. ORO meets the specification in approximately 1,200 simulations against 16,300, and sharpens the upper band edge through an additional transmission zero.

symmetry constraint, with S_{11} and S_{21} simulated over 30–50 GHz. That work generates its dataset in advance by Latin hypercube sampling and searches a convolutional surrogate with a genetic algorithm, using 16,300 full-wave simulations; it is accepted at the European Microwave Conference in October 2026 but not yet online, so we state its simulation count explicitly. ORO instead queries the solver only along the trajectory its inverse network induces toward the requested response, and reaches a compliant design in approximately 1,200 simulations with the same geometry and solver settings.

Both designs meet the passband specification, but ORO's geometry realises two transmission zeros in the upper stopband rather than one, sharpening the S_{21} roll-off. This was not requested: the target was the reference response itself, and the additional zero emerged from the network's own exploration of the vertex space, since the response-space loss admits any geometry that matches the specification rather than one resembling the design used to define it.

4 Conclusion

We introduced ORO, a framework for inverse design with expensive black-box simulators in which a target-conditioned inverse network serves as both the design model and the query rule, so no acquisition function is designed or maximized. Each simulation is used twice: online, to correct the surrogate and move the proposal toward the requested response, and offline, as a solved instance of the response it actually achieved. Under strictly matched budgets ORO attains lower final error than genetic search, Bayesian optimization, and surrogate-assisted search on three benchmarks that separate non-uniqueness from ill-conditioning; removing offline reuse causes it to stall; and on two of the three benchmarks an accumulated dataset makes later targets substantially cheaper. On a full-wave filter design it reaches a compliant design in roughly 1,200 simulations against 16,300 for a surrogate-based genetic approach.

AI use statement

In this work, we used generative AI tools for polishing the text, discovering related works, and assisting with code generation. We have not used generative AI tools for generating core research

ideas, primary claims, or experimental results, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for basic syntax formatting and proofreading. We have reviewed all AI-assisted work. Specifically, all LLM-generated code was thoroughly revised, tested, and validated for correctness by the authors. AI-suggested related literature was manually verified against primary sources, and polished writing was reviewed to ensure scientific accuracy and preservation of the original intent. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

Abdelrahman Elbadrawy, Chandan Roy, and Ming Jian. Vector graphics driven deep learning framework for mmWave bandpass filter design. In *European Microwave Conference (EuMC)*, 2026. To appear; accepted for presentation, October 2026.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Luka Grbcic, Juliane Müller, and Wibe Albert de Jong. Autotandemml: Active learning enhanced tandem neural networks for inverse design problems. *Applied Soft Computing*, 184:113828, 2025. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2025.113828. URL https://www.sciencedirect.com/science/article/pii/S156849462501141X.

Jiaqi Jiang and Jonathan A Fan. Global optimization of dielectric metasurfaces using a physics-driven neural network. *Nano letters*, 19(8):5366–5372, 2019.

Christopher M. Lalau-Keraly, Samarth Bhargava, Owen D. Miller, and Eli Yablonovitch. Adjoint shape optimization applied to electromagnetic design. *Opt. Express*, 21(18):21693–21701, Sep 2013. doi: 10.1364/OE.21.021693. URL https://opg.optica.org/oe/abstract.cfm?URI=oe-21-18-21693.

Dianjing Liu, Yixuan Tan, Erfan Khoram, and Zongfu Yu. Training deep neural networks for the inverse design of nanophotonic structures. *Acs Photonics*, 5(4):1365–1369, 2018.

Simiao Ren, Willie Padilla, and Jordan Malof. Benchmarking deep inverse models over time, and the neural-adjoint method. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 38–48. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/007ff380ee5ac49ffc34442f5c2a2b86-Paper.pdf.

Jose Sanchez-Dehesa, Andreas Hakansson, and Lorenzo Sanchis. Inverse design of photonic devices by using a genetic algorithm. In Richard M. De La Rue, Pierre Viktorovitch, Clivia M. Sotomayor Torres, and Michele Midrio (eds.), *Photonic Crystal Materials and Nanostructures*, volume 5450, pp. 200 – 206. International Society for Optics and Photonics, SPIE, 2004. doi: 10.1117/12.545902. URL https://doi.org/10.1117/12.545902.

Jasper Snoek, Hugo Larochelle, and Ryan Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine*

Learning Research, pp. 10358–10368. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/trabucco21a.html.

Tailin Wu, Takashi Maruyama, Long Wei, Tao Zhang, Yilun Du, Gianluca Iaccarino, and Jure Leskovec. Compositional generative inverse design. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 29766–29795, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/7f8b8bc8ebac661c442c4dafd5d98c08-Paper-Conference.pdf.

A BENCHMARK SIMULATORS

All three simulators map a design $x \in \mathbb{R}^{15}$ to a response sampled at $m = 200$ uniformly spaced points $f \in [0, 1]$. The design is organised as $K = 5$ sections with three parameters each, $x = (a_{1:K},\, c_{1:K},\, w_{1:K})$. Targets and warm-up designs are drawn uniformly from the box

$$a_i \in [0.5,\, 1.5], \qquad c_i \in [0.1,\, 0.9], \qquad w_i \in [0.03,\, 0.12], \tag{7}$$

and, where a simulator requires it, parameters are normalised to the unit cube, $u = (x - x_{\min})/(x_{\max} - x_{\min})$, using the bounds of equation 7. The same box and parameter layout are shared by all three, so every model, baseline and hyperparameter is used unchanged across them.

Resonance. A sum of K Gaussian resonances with amplitudes a_i, centres c_i and widths w_i,

$$R(f) = \sum_{i=1}^{K} a_i \exp\!\left(-\frac{(f - c_i)^2}{2w_i^2}\right). \tag{8}$$

The sum is invariant to permutation of the section index, so any response has $K! = 120$ parameter preimages that differ only in labelling.

Rosenbrock. The normalised parameters are passed through the curved-valley warp

$$z_0 = u_0, \qquad z_i = 10\left(u_i - u_{i-1}^2\right), \quad i = 1, \ldots, 14, \tag{9}$$

and mapped to a response by $R = Bz$, where $B \in \mathbb{R}^{200\times 15}$ has orthonormal columns, obtained once from the QR decomposition of a fixed Gaussian matrix and held constant throughout. Because $B^\top B = I$,

$$\ell(R(x), R(x^\star)) = \tfrac{1}{m}\,\|z(u) - z(u^\star)\|_2^2\,, \tag{10}$$

so the response-error landscape searched by every method is exactly a Rosenbrock valley centred on the target's own parameters, with value zero at $x = x^\star$ (verified numerically to within 2.4×10^{-7}). The warp is recursively invertible, $u_i = z_i/10 + u_{i-1}^2$, and B has full column rank, so the simulator is a bijection onto its range: it presents ill-conditioning without non-uniqueness.

Rational. A cascade of K second-order sections, each with a complex pole pair and a transmission zero. The three parameter groups map, through their normalised values (u_a, u_c, u_w), to a quality factor, a pole frequency and a zero frequency,

$$Q_i = Q_{\min}\!\left(\tfrac{Q_{\max}}{Q_{\min}}\right)^{u_{a,i}}, \qquad f_{p,i} = 0.1 + 0.8\,u_{c,i}, \qquad f_{z,i} = f_{p,i} + 0.25\left(u_{w,i} - \tfrac{1}{2}\right), \tag{11}$$

with $Q_{\min} = 3$, $Q_{\max} = 100$, and $f_{z,i}$ clipped to $[0.02,\, 1.2]$. The response is

$$H(f) = \prod_{i=1}^{K} \frac{f^2 - f_{z,i}^2}{\left(f^2 - f_{p,i}^2\right) + \jmath\, f\, f_{p,i}/Q_i}, \qquad R(f) = \frac{|H(f)|^2}{1 + |H(f)|^2}. \tag{12}$$

The product is evaluated as a sum of logarithms and clamped before exponentiation, so high-Q poles cannot overflow. The bounded output mimics $|S_{11}|^2$ and lies in $[0, 1]$. Over 500 random designs the response spans the full interval with no degenerate curves, and only 4% and 9% of samples saturate near 1 and 0 respectively. Like the resonance simulator, the product is invariant to permutation of the section index.

Conditioning measurement. For each simulator a single target $y^\star = F(x^\star)$ is fixed, and at 25 further designs drawn from equation 7 we compute the Hessian of $\ell(F(x), y^\star)$ with respect to x by automatic differentiation. A point is counted as non-convex if the smallest eigenvalue is below -10^{-8}, and the condition number is the ratio of largest to smallest absolute eigenvalue. These are the quantities reported in Table 1. Automatic differentiation is used only for this characterisation; no method under comparison has access to simulator gradients.

B DIRECT INVERSE LEARNING

This appendix gives the full experiment summarised in Section 3.1. A single inverse network D_θ, with the architecture used throughout the paper, is trained to regress design parameters from responses with the parameter-space loss

$$\mathcal{L}_{\text{direct}}(\theta) = \text{MSE}(x,\, D_\theta(y))\,. \tag{13}$$

For each simulator we draw 8,000 training and 1,500 validation designs from equation 7, standardise the responses using training-set statistics, and train for 420 epochs with Adam at learning rate 10^{-3} and batch size 128. Architecture, data size, optimizer and schedule are identical across the three simulators, so they differ only in the structure of the mapping. As a control on the same data we also train the tandem construction of Liu et al. (2018): a forward surrogate is fitted first, then frozen while an inverse network is trained through it with the response-space loss $\text{MSE}(y,\, S_\phi(D_\theta(y)))$. Both are finally scored by the true-simulator response error of the designs they produce on the validation targets, which is the quantity that matters for design.

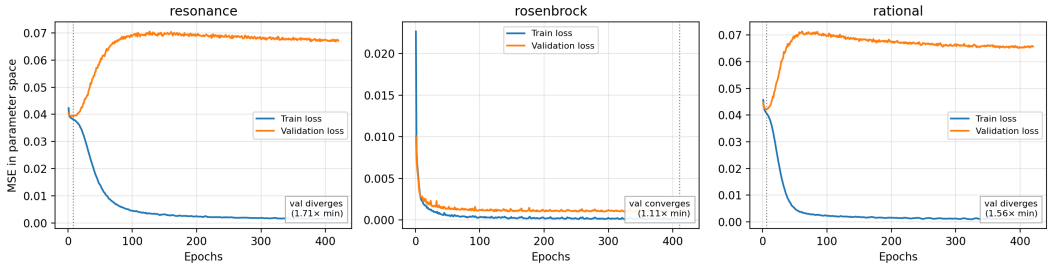


Figure 4: Training and validation loss of a single inverse network trained with the parameter-space loss equation 13 on 8,000 designs. Validation loss reaches its minimum within the first ten epochs and then rises on the two simulators with $K!$ equivalent solutions; it does not on the bijective Rosenbrock control. The dotted line marks the epoch of minimum validation loss.

Figure 4 shows the result. On resonance and rational, validation loss is minimised at epochs 7 and 5 respectively and then rises steadily, ending $1.71\times$ and $1.56\times$ above that minimum while training loss falls by more than an order of magnitude. On Rosenbrock the minimum occurs at epoch 409 and the final value is within $1.11\times$ of it, with validation tracking training throughout.

The difference is structural. On a simulator with K interchangeable sections, two designs $x_1 \neq x_2$ with $F(x_1) = F(x_2)$ present the network with the same input and different labels. The minimiser of equation 13 over such a pair is their average, which in general realises neither response and need not be a valid design at all. The network fits these conflicting labels by memorisation on the training set, which does not transfer. On the bijective control no such conflict exists, and the same network generalises.

Table 4 confirms that this is not a small-sample effect. Doubling the training set increases the divergence on both non-unique simulators, from $1.55\times$ to $1.71\times$ on resonance and from $1.43\times$ to $1.56\times$ on rational, while the bijective control is essentially unchanged. This is the expected direction: a larger sample contains more pairs of distinct designs with near-identical responses, so the conflicting supervision becomes more prevalent rather than less.

In response space the tandem construction improves on direct learning by $1.9\times$, $3.3\times$ and $1.4\times$ at 8,000 designs. At 4,000 designs it is marginally worse than direct learning on the rational simulator (0.163 against 0.151) and better elsewhere. The tandem network can only be as good as the surrogate it optimises through, and the rational simulator is the hardest to fit, having the most severe local

Table 4: Direct parameter-space learning versus the tandem response-space construction, at two training-set sizes. Divergence is the ratio of final to minimum validation loss under equation 13. Response error is the mean true-simulator error of the designs produced on the validation targets.

Training designs	Quantity	Resonance	Rosenbrock	Rational
4,000	Divergence, direct	1.55×	1.09×	1.43×
	Response error, direct	0.641	0.0392	0.151
	Response error, tandem	0.320	0.0087	0.163
8,000	Divergence, direct	1.71×	1.11×	1.56×
	Response error, direct	0.603	0.0277	0.144
	Response error, tandem	0.311	0.0083	0.102

conditioning in Table 1; with more data the surrogate improves and the tandem advantage appears. This dependence on surrogate quality is one motivation for correcting the surrogate online, where the inverse network is actually proposing, rather than fitting it once to a fixed sample.

C Implementation details

C.1 Networks for the benchmark simulators

Both networks are fully connected with ReLU activations. The forward surrogate S_ϕ maps the 15 design parameters to the 200-point response through hidden layers of width 128, 256 and 256. The inverse design network D_θ maps the 200-point response to the 15 design parameters through four hidden layers of width 256, followed by a linear head.

Admissibility is enforced in the output layer of D_θ rather than by constrained optimization. The three parameter groups pass through activations matched to their physical ranges,

$$p_i = 0.1 + \mathrm{softplus}(h_{p,i}), \qquad q_i = \sigma(h_{q,i}), \qquad r_i = 0.01 + 0.2\,\sigma(h_{r,i}), \tag{14}$$

where h denotes the pre-activation output and σ the logistic sigmoid. This admissible set is deliberately wider than the sampling box equation 7, so the inverse network may propose designs outside the region from which targets and warm-up designs are drawn.

C.2 ORO hyperparameters

A single configuration is used for all three benchmarks, with no per-landscape tuning. Table 5 lists the settings for the two experimental protocols. All networks are trained with Adam, and each network uses one optimizer shared across the online and offline phases (Section 2.5). Gaussian noise with standard deviation 0.002 is added to responses when pre-training on the warm-up set; no noise is added to the target during refinement. Offline minibatches have size $\min(128, |\mathcal{D}|)$.

Table 5: ORO hyperparameters for the budget-matched comparison (Section 3.3) and the sequential experiment (Section 3.5).

	Budget-matched	Sequential
Simulator calls per target	2,000	700
Warm-up designs n_0	400 (shared GA search)	2,000 (random, once)
Warm-up pre-training epochs (S, D)	120, 150	200, 250
Warm-up learning rate	10^{-3}	10^{-3}
Refinement learning rate (S, D)	5×10^{-4}	5×10^{-4}
Online burst N (steps per cycle)	3	3
Offline training per cycle	15 epochs	$U = 175$ minibatch updates
Offline minibatch size	128	128

The two protocols differ in how offline training is scheduled. The budget-matched comparison performs 15 epochs over $\mathcal{D}$ per cycle; there $|\mathcal{D}|$ grows only from 400 to 2,000, so the cost stays bounded. The sequential experiment accumulates up to 16,000 designs, where an epoch-based

schedule would grow super-linearly in wall-clock, so it instead performs a fixed $U = 175$ minibatch updates per network, chosen to match 15 epochs at $|\mathcal{D}| = 1{,}500$ ($15 \times 1{,}500/128 \approx 175$). The effective epoch count $Un_b/|\mathcal{D}|$ then falls as the dataset grows, at constant cost.

C.3 Baselines

GA on the true simulator. Population 20, elite fraction 0.2, uniform crossover, and Gaussian mutation applied per parameter with probability 0.2 and standard deviation 0.08 of that parameter's range, with offspring clipped to the admissible box. The search runs until the simulator budget is exhausted. In the budget-matched comparison a single GA search is run per target and its first 400 evaluations also form ORO's warm-up set, so the two coincide over that range by construction.

GA on an offline surrogate. A surrogate with the architecture of Appendix C.1 is trained for 200 epochs on designs drawn uniformly from equation 7: 2,000 in the budget-matched comparison, matching the full per-target budget, and 30,000 in the sequential experiment. A GA (population 30, 150 generations in the budget-matched comparison and 200 in the sequential experiment) then searches the surrogate, and only its final design is evaluated with F.

Bayesian optimization. A Gaussian process with a squared-exponential kernel and automatic relevance determination (one lengthscale per design dimension), initialised with 20 uniformly random designs. Three choices are needed for a fair baseline. First, observed errors are standardised to zero mean and unit variance before every fit: a GP has a zero-mean prior, whereas response errors are strictly positive, so without standardisation the posterior mean far from the data falls below the best observed value and expected improvement favours unexplored regions regardless of promise. Second, once more than 300 points have been observed the GP is fitted to a *random* subset of 300, always including the incumbent; retaining only the best points would discard all information about poor regions. Third, the kernel hyperparameters are refitted every 25 evaluations by maximising the marginal likelihood (60 Adam steps, learning rate 0.05, log-lengthscales clipped to $[-3, 3]$). Expected improvement is maximised by scoring 3,000 random candidates and refining the best 8 with 40 steps of gradient ascent, since a candidate pool alone under-optimises the acquisition in 15 dimensions.

C.4 Band-pass filter design

The filter geometry is defined by the planar coordinates of 15 vertices on one side of the structure's symmetry axis, giving 30 design parameters. Each vertex is confined to its own interval along the propagation axis, so that vertex order is preserved, and to a fixed transverse range on its side of the axis; the network output is mapped to these bounds from the unit interval. The target and simulated responses are the magnitudes of S_{11} and S_{21} in decibels over 30–50 GHz, concatenated.

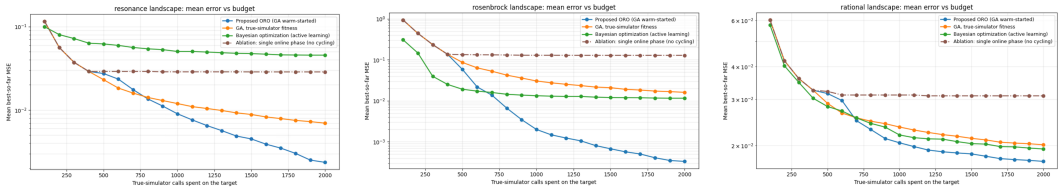
Both networks are fully connected with ReLU activations and hidden layers of width 256, 512 and 512. The inverse network ends in a sigmoid, so every proposal lies in the admissible box by construction. Refinement uses learning rate 10^{-4} for both networks, an online burst of $N = 3$, and 20 offline epochs per network per cycle with minibatch size 128.

D Mean results

The main text reports medians because the distribution of final error across targets is heavy-tailed. For completeness, Figure 5 shows the corresponding mean best-so-far error and Table 6 gives means with standard errors at the full budget. The GA on an offline surrogate is omitted from the figure, since its error lies one to three orders of magnitude above that of ORO and would compress the remaining curves; its values are included in the table.

The means support the same ordering as the medians. ORO attains the lowest mean error at the full budget on all three benchmarks; the ablation flattens as soon as the shared warm-up ends; and GA on the true simulator leads at small budgets before ORO overtakes it. The margin again narrows with landscape difficulty, from $48.3\times$ over GA on Rosenbrock to $2.9\times$ on resonance and $1.2\times$ on rational.

The two statistics differ mainly in magnitude. Among the online methods, ORO's mean exceeds its median by the largest factor on every benchmark, most markedly on resonance ($2.9\times$, against $1.6\times$



(a) Resonance (b) Rosenbrock (c) Rational

Figure 5: Mean best-so-far response error over 30 held-out targets against true-simulator calls, the counterpart of Fig. 2. The ordering of methods at the full budget agrees with the medians.

Table 6: Mean response error $\pm$ one standard error over 30 targets at 2,000 calls per target.

	Resonance	Rosenbrock	Rational
ORO (proposed)	$(\mathbf{2.36} \pm 0.80) \times 10^{-3}$	$(\mathbf{3.31} \pm 0.75) \times 10^{-4}$	$(\mathbf{1.74} \pm 0.40) \times 10^{-2}$
GA, true simulator	$(6.95 \pm 1.30) \times 10^{-3}$	$(1.60 \pm 0.35) \times 10^{-2}$	$(2.01 \pm 0.38) \times 10^{-2}$
Bayesian optimization	$(4.54 \pm 0.31) \times 10^{-2}$	$(1.15 \pm 0.08) \times 10^{-2}$	$(1.94 \pm 0.18) \times 10^{-2}$
GA, offline surrogate	$(3.18 \pm 0.41) \times 10^{-1}$	$(3.63 \pm 0.69) \times 10^{-1}$	$(1.86 \pm 0.14) \times 10^{-1}$
Ablation (no offline reuse)	$(2.86 \pm 0.36) \times 10^{-2}$	$(1.28 \pm 0.12) \times 10^{-1}$	$(3.09 \pm 0.39) \times 10^{-2}$

for GA). Its error distribution therefore has the heaviest upper tail: on most targets ORO reaches very low error, and a small number on which it performs less well raise the mean. This is why we report medians in the main text. The effect narrows ORO's mean advantage relative to its median advantage on every benchmark—over GA on resonance, from $5.5\times$ to $2.9\times$—but does not reverse the ordering on any.

On the rational benchmark the mean intervals of ORO, GA and Bayesian optimization overlap, so the means alone do not separate them. The evidence for ORO's advantage there is the paired comparison in Table 2, which accounts for per-target difficulty and gives win rates of $24/30$ against GA and $23/30$ against Bayesian optimization ($p = 1.4 \times 10^{-3}$ and 5.2×10^{-3}). As in the median curves, Bayesian optimization leads all methods on Rosenbrock at small budgets, up to roughly 700 calls, before ORO overtakes it; on resonance it remains well above both GA and ORO throughout.