ORACLE: OPTIMIZER-RELATIVE ALIGNMENT FOR CONSTRAINED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Constraint-handling methods typically intervene before the optimizer acts, by modifying the objective or the gradient. Yet momentum, adaptive scaling, and structured preconditioning can substantially reshape that signal before it becomes a parameter update. We formulate *optimizer relative constrained learning*, where constraint compatibility is assessed on the post-optimizer update. Building on this view, we introduce ORACLE, which evaluates the native optimizer's realized step through a joint endpoint linearization of heterogeneous constraint families, constructs the resulting alignment in the optimizer's own geometry, bounds its authority, and commits it only after validation. We evaluate ORACLE across eight Partial Differential Equation benchmarks and four optimizers spanning Euclidean, diagonal adaptive, and structured preconditioned geometries, where it improves or matches native optimizer in 94% of configurations. Cross-model analysis shows the same behavior in 92% of configurations, while matched comparisons show improvements over alternative constraint-handling methods acting at the objective, gradient, and post-optimizer levels.

1 INTRODUCTION

Modern learning systems increasingly operate under requirements that extend beyond minimizing a single task loss. Physics-informed learning must reconcile data with governing equations and structural constraints Wang et al. (2026); Moro & Chamon (2025); safety-critical robotics must optimize behavior while satisfying safety requirements Nguyen et al. (2024); Zhang et al. (2026); and decision-focused learning couples prediction with constrained downstream optimization, where predictive accuracy alone may not yield feasible or high-quality decisions Mandi et al. (2024; 2026); Kong et al. (2023). In each case the learner must improve task performance while remaining compatible with additional physical, safety, or decision-level requirements.

The most direct way to impose such requirements is to write them into the quantity being minimized. Penalty terms, constraint multipliers, primal-dual formulations, and augmented-Lagrangian methods encode constraints in the objective itself Moro & Chamon (2025); Nandwani et al. (2019); Dener et al. (2020); Fioretto et al. (2020), and adaptive weighting schemes adjust their relative influence as training proceeds Bischof & Kraus (2025). These methods control how much each requirement contributes to a scalar trade-off, but they cannot express which *directions* of parameter movement respect it: competing terms may remain directionally incompatible even after their magnitudes are balanced Wang et al. (2026).

A second family therefore acts on directions, the gradient itself. Multi-objective and conflict-resolution methods remove or reconcile components of competing gradients before a step is taken Yu et al. (2020); Liu et al. (2021; 2025); Bu et al. (2026), and projection or SQP-type methods correct a proposed movement using local constraint geometry Rosen (1961); Byrd et al. (2008). This resolves directional conflict that reweighting cannot. Both families, however, hand their corrected signal to an optimizer, and treat that handover as though the optimizer will apply what it receives.

Each optimizer defines its own geometry by scaling and prioritizing parameter magnitude and directions differently. Momentum, adaptive coordinate statistics, and structured preconditioning transform the incoming gradient using information accumulated across training, so the displacement ultimately applied to the parameters can differ from that gradient in both scale and direction Kingma & Ba (2014); Gupta et al. (2018); Vyas et al. (2025); Si et al. (2025); Du & Su (2026). Adaptive

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

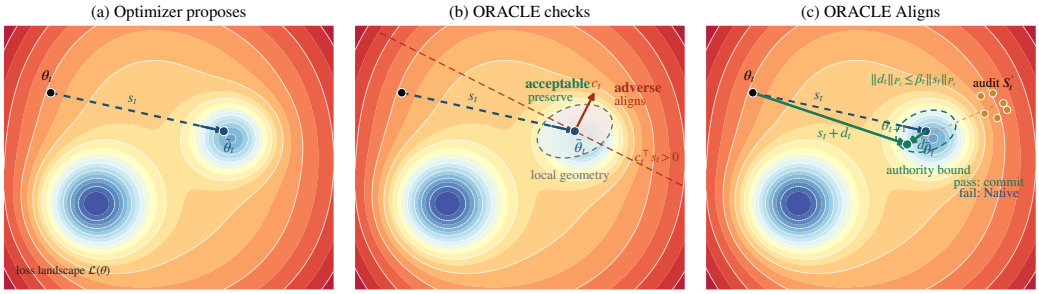

Figure 1: **Toy illustration of ORACLE.** (a) From θ_t, the native optimizer proposes the endpoint $\bar{\theta}_t$. (b) ORACLE checks the proposal against the local constraint geometry. (c) A validated alignment produces θ_{t+1}; otherwise the native proposal is retained.

optimizers descend in a geometry that is not Euclidean Xie et al. (2026), and recent work names the resulting gradient-update mismatch directly in physics-informed training Xiao et al. (2026). A constraint enforced before this transformation is therefore enforced in a geometry that is not the geometry of the step actually taken, and this gap widens precisely for the stateful, preconditioned optimizers on which modern training relies. This leaves one position in the pipeline unoccupied: the realized update, after the optimizer has committed to how the parameters should move.

We introduce ORACLE (*Optimizer-Relative Alignment for Constrained LEarning*), illustrated in Figure 1, which occupies that position. From θ_t, the native optimizer uses its internal state to produce the realized step s_t and proposed endpoint $\bar{\theta}_t = \theta_t + s_t$. At this endpoint, ORACLE jointly linearizes the sampled constraint families so that their local responses to the same parameter movement are considered together. The resulting local model is then used to construct an alignment in the geometry of the optimizer that produced s_t. The quantity $c_t^\top s_t$ identifies locally adverse realized movements; when it is positive, ORACLE constructs a bounded optimizer-relative alignment and commits it only after nonlinear validation and an independent audit. Thus, $\theta_{t+1} = \theta_t + s_t + d_t$ when the alignment is accepted and $\theta_{t+1} = \bar{\theta}_t$ otherwise.

We study this in physics-informed Partial Differential Equation learning, where the gap is most exposed. PINNs Raissi et al. (2019) combine PDE residuals with initial and boundary conditions whose gradients differ sharply in scale, conditioning, and direction, and modern PINN training increasingly depends on adaptive and structured optimizers Wang et al. (2026); Rathore et al. (2024). Heterogeneous constraint sensitivities therefore meet stateful preconditioned updates in the same training loop. We ask five questions of this setting: whether optimizer-relative alignment holds across native optimizer geometries; whether it persists when the model formulation changes; what's the additional computation costs and what it buys; whether intervening on the realized update outperforms intervening earlier under a matched protocol; and which of ORACLE's mechanisms are responsible.

Our contributions are as follows. **(i)** We formulate *optimizer-relative constrained learning*, in which constraint compatibility is defined on the realized update rather than the gradient preceding it. **(ii)** We instantiate it as ORACLE, which jointly linearizes heterogeneous constraint families at the proposed endpoint and constructs a single alignment in the certified geometry of the native optimizer. A matrix-free residual-space solve computes this alignment without forming the parameter-space Jacobian, while native-relative authority and held-out validation control whether it is committed. **(iii)** We evaluate it across optimizer geometries, model formulations, computational cost, intervention points, and internal mechanisms, improving or matching the native optimizer in 94% valid PDE optimizer configurations and 92% model optimizer configurations.

2 Related Work

Modern optimizers separate the raw gradient from the parameter displacement ultimately applied to the model. AdaGrad, RMSProp, Adam, and Adafactor use gradient history for coordinate-wise adaptation Ward et al. (2020); Kingma & Ba (2014); Shazeer & Stern (2018), while natural gradient, K-FAC, Shampoo, SOAP, and Newton-type methods introduce metric, structured-preconditioning, or curvature information Kakade (2001); Grosse & Martens (2016); Gupta et al. (2018); Vyas et al.

(2025); Botev et al. (2017). The realized update can therefore differ from the raw gradient in both direction and scale, making the optimizer's induced geometry consequential for where and how constraint information is introduced.

Penalty, Lagrangian, primal-dual, and augmented-Lagrangian methods encode constraints directly in the optimization problem Nandwani et al. (2019); Dener et al. (2020); Lavado et al. (2023); Fioretto et al. (2020), while multi-objective methods reconcile competing gradients Yu et al. (2020); Liu et al. (2021); Navon et al. (2022) and projection or SQP-type methods correct candidate movement using local constraint geometry Rosen (1961); Byrd et al. (2008). These approaches span objective, gradient, and movement-level intervention, but generally do not make the native optimizer's realized geometry the reference geometry for constraint correction.

Recent methods enforce nonlinear requirements through constrained formulations and feasibility mechanisms Moro & Chamon (2025); Lastrucci & Schweidtmann (2025); Iftakher et al. (2025), or by modifying task gradients Liu et al. (2025); Bu et al. (2026). Yet in PINNs, heterogeneous constraint sensitivities already induce difficult optimization geometry Wang et al. (2021a); Yao et al. (2023); Rathore et al. (2024); Wang et al. (2026), which adaptive and structured optimizers can further reshape Kingma & Ba (2014); Vyas et al. (2025). This creates a missing interface between nonlinear constraint geometry and the geometry of the realized optimizer update, together with the problem of deciding when a locally constructed correction should be trusted.

ORACLE addresses this interface by recovering and certifying the native optimizer's realized metric and expressing sampled residual geometry within it. It applies a bounded correction only when the realized step is locally adverse, then separates candidate construction from acceptance through an independent audit support. Detailed comparisons with post-optimizer alignment, projection methods, and sample-splitting approaches appear in the Extended Related Work A.

3 ORACLE: Optimizer-Relative Alignment for Constrained LEarning

ORACLE wraps a native optimizer. At iteration t the optimizer proposes moving from θ_t to $\bar{\theta}_t = \theta_t + s_t$; ORACLE commits either that proposal or a corrected one,

$$\boxed{\theta_{t+1} = \theta_t + s_t + d_t, \qquad d_t = \mathbf{1}\{\mathrm{Cert}_t\}\,\mathbf{1}\{\mathrm{Trig}_t\}\,\mathbf{1}\{\mathrm{Val}_t\}\;\alpha_t\,\widehat{d}_t.} \tag{1}$$

Each factor of d_t is one mechanism (Fig. 1). The *alignment direction* $\widehat{d}_t$ reduces the local constraint model in the native optimizer's own geometry; the *corrective authority* α_t bounds how far the proposal may be redirected; and three gates decide whether the correction is applied: *geometry certification* Cert_t, the *alignment trigger* Trig_t, and *validation* Val_t. If any gate is closed, $d_t = 0$ and ORACLE returns the native step exactly. Algorithm 1 lists the procedure; the rest of this section defines each factor in turn.

Constraints are organized into K semantic families. In a PINN, for example, these may correspond to the PDE residual, boundary conditions, initial conditions, or observed data. For family k, $r_k(\theta; z)$ measures the violation at evaluation point z, $z \sim \nu_k$ specifies how those points are sampled, and $\omega_{k,t} > 0$ controls the family's relative contribution.

3.1 Native Proposal and Constraint Model

Let g_t be the gradient supplied to the native optimizer, and ξ_t its internal state, such as momentum, running moments, or structured preconditioning statistics. The native optimizer produces

$$s_t = \mathcal{S}(g_t, \xi_t), \qquad \xi_{t+1} = \mathcal{U}(g_t, \xi_t), \qquad \bar{\theta}_t = \theta_t + s_t, \tag{2}$$

where s_t is the realized parameter displacement and $\bar{\theta}_t$ the native proposal. ORACLE leaves both s_t and the state transition untouched. With $d_t = 0$, Eq. 1 is the native update $\theta_{t+1} = \bar{\theta}_t$; this is the default that each mechanism below modifies only under a stated condition.

To evaluate this proposal, we summarize the population-level constraint violation as

$$\Phi_t(\theta) = \sum_{k=1}^{K} \omega_{k,t}\, \mathbb{E}_{z\sim\nu_k}\left[r_k(\theta; z)^2\right]. \tag{3}$$

After $\bar{\theta}_t$ is fixed, ORACLE evaluates its constraint violations on a fresh support: for each family k it draws m_k samples $z_{k,j} \sim \nu_k$, collects them in S_t, stacks the residuals into $r_{S_t}(\theta)$, and measures

$$\Phi_{S_t}(\theta) = \tfrac{1}{2}\, r_{S_t}(\theta)^\top W_t\, r_{S_t}(\theta), \tag{4}$$

where W_t is diagonal and assigns each residual from family k the weight $2\omega_{k,t}/m_k$. Because $\bar{\theta}_t$ is fixed before S_t is sampled, $\Phi_{S_t}(\bar{\theta}_t)$ is an unbiased estimate of the population constraint energy (Appendix B.1). The sampled energy tells us how much violation remains at the proposed endpoint. Since r_{S_t} stacks PDE, boundary, initial-condition, and other residuals into a single endpoint model, ORACLE next asks how these constraint families respond jointly to the same parameter movement. Answering this requires their local sensitivities and the geometry in which the optimizer produced s_t.

3.2 Checking the Proposal in the Native Optimizer Geometry

Different optimizers induce different geometries on parameter updates: Adam rescales coordinates using adaptive statistics, while SOAP and PSGD-Kron additionally apply structured preconditioning. Rather than constructing a new geometry, ORACLE reuses the transformation already encoded by the native optimizer and writes the realized step as

$$s_t = -B_t u_t, \qquad P_t = B_t^{-1} \succ 0, \tag{5}$$

where u_t is the optimizer's effective direction and B_t is the symmetric positive-definite (SPD) scaling or preconditioning operator induced by its update rule and state. For SGD, B_t is scalar learning-rate scaling; for Adam, diagonal adaptive scaling; for structured optimizers, it acts through their existing factors. The inverse P_t defines the optimizer-relative metric $\|v\|_{P_t}^2 = v^\top P_t v$, which reflects the native optimizer's directional preferences. Neither B_t nor P_t is formed explicitly; ORACLE uses them only through optimizer-specific vector products. The geometry is meaningful only if it faithfully describes the step actually produced, so ORACLE certifies it against the realized displacement,

$$\|s_t + B_t u_t\|_2 \le \tau_{\mathrm{rel}} \|s_t\|_2 + \tau_{\mathrm{abs}}, \tag{6}$$

together with optimizer-specific validity checks (Appendix B.2); Cert_t is the event that both hold. On failure ORACLE preserves the native proposal and never replaces an uncertified geometry with $P_t = I$. This supplies the gate $\mathbf{1}\{\mathrm{Cert}_t\}$ and the norm in which every quantity below is measured.

ORACLE now asks whether the movement from θ_t to $\bar{\theta}_t$ is locally adverse to the sampled constraints. At the proposed endpoint, define

$$r_t = r_{S_t}(\bar{\theta}_t), \qquad J_t = \left.\frac{\partial r_{S_t}(\theta)}{\partial \theta}\right|_{\theta=\bar{\theta}_t}, \qquad c_t = J_t^\top W_t r_t = \nabla \Phi_{S_t}(\bar{\theta}_t), \tag{7}$$

where r_t measures the sampled violations at the endpoint, J_t their local sensitivity to parameter movement, and c_t the first-order direction of increasing constraint energy. The trigger is the directional interaction between c_t and the realized step:

$$\mathrm{Trig}_t = \{\, c_t^\top s_t > 0 \,\}. \tag{8}$$

When it fires, the native movement is locally adverse to the sampled constraint model and ORACLE opens the alignment path; otherwise the native proposal is preserved. J_t is never formed explicitly: a Jacobian-vector product $J_t v$ measures how a parameter movement v changes the residuals, and a vector-Jacobian product $J_t^\top q$ maps a residual-space direction back to parameter space, both by automatic differentiation. Formal analysis of the endpoint trigger appears in Appendix B.4. This supplies the gate $\mathbf{1}\{\mathrm{Trig}_t\}$.

3.3 Optimizer-Relative Alignment and Validated Commitment

Once the trigger fires, the certified metric determines how the proposal is corrected. At $\bar{\theta}_t$, let $b_t = W_t^{1/2} r_t$ and $C_t = W_t^{1/2} J_t$. ORACLE computes the alignment direction by solving

$$\widehat{d}_t = \arg\min_{d \in \mathbb{R}^p} \ \tfrac{1}{2}\|b_t + C_t d\|_2^2 + \tfrac{\lambda_t}{2}\|d\|_{P_t}^2, \tag{9}$$

with damping $\lambda_t > 0$. The first term jointly reduces the local linearized violations of the stacked constraint families; the second keeps the adjustment close to the native proposal in the optimizer's own geometry. Rather than solving Eq. 9 in parameter space, ORACLE uses

$$G_t = C_t P_t^{-1} C_t^\top = W_t^{1/2} J_t P_t^{-1} J_t^\top W_t^{1/2}, \qquad (G_t + \lambda_t I)\, z_t = b_t, \qquad \widehat{d}_t = -P_t^{-1} C_t^\top z_t. \tag{10}$$

Algorithm 1 ORACLE: Optimizer-Relative Alignment for Constrained Learning

Require: Parameters θ_t, optimizer state ξ_t, task gradient g_t;
Require: constraint sampling laws and ORACLE hyperparameters
1: Form the native proposal $(s_t, \xi_{t+1}) \leftarrow \mathrm{NativeStep}(g_t, \xi_t)$, $\bar{\theta}_t \leftarrow \theta_t + s_t$
2: Recover and certify the optimizer geometry (B_t, P_t, u_t) via Eq. 5- Eq. 6; **return** $(\bar{\theta}_t, \xi_{t+1})$ if certification fails
3: Draw construction support S_t and evaluate the local constraint model at $\bar{\theta}_t$ to obtain (r_t, c_t) and Jacobian products with J_t
4: **return** $(\bar{\theta}_t, \xi_{t+1})$ if $c_t^\top s_t \le 0$ {trigger Eq. 8}
5: Construct the optimizer-relative alignment direction $\widehat{d}_t$ using Eq. 10
6: **return** $(\bar{\theta}_t, \xi_{t+1})$ if $c_t^\top \widehat{d}_t \ge 0$ {finite-precision guard}
7: Apply the native-relative authority bound Eq. 11 to obtain $\widetilde{d}_t$
8: Validate $\widetilde{d}_t$ on S_t using Eq. 12; **return** $(\bar{\theta}_t, \xi_{t+1})$ if validation fails
9: Fix $\widetilde{d}_t$, draw independent audit support S'_t, and **return** $(\bar{\theta}_t, \xi_{t+1})$ if the held-out audit Eq. 13 fails
10: **return** $(\bar{\theta}_t + \widetilde{d}_t, \xi_{t+1})$

Neither J_t nor G_t is materialized: a product with G_t uses one VJP, one application of P_t^{-1}, and one JVP. The resulting direction is a strict first-order constraint-descent direction; Appendix B.5 gives the exact and finite-CG results. This supplies the direction of d_t.

ORACLE next limits how far the constraints may redirect the native proposal, scaling $\widehat{d}_t$ by

$$\alpha_t = \min\left\{ \frac{c_t^\top s_t}{-c_t^\top \widehat{d}_t + \varepsilon_{\mathrm{auth}}},\ \frac{\beta_t \|s_t\|_{P_t}}{\|\widehat{d}_t\|_{P_t}} \right\}, \qquad \widetilde{d}_t = \alpha_t \widehat{d}_t, \tag{11}$$

where $\beta_t \in [0, 1)$ controls the maximum corrective authority relative to the native step and $\varepsilon_{\mathrm{auth}} > 0$ is a small numerical stabilizer. The first term is the scale needed to neutralize the detected first-order adverse signal; the second caps the correction in the optimizer-relative metric. Consequently $\|\widetilde{d}_t\|_{P_t} \le \beta_t \|s_t\|_{P_t}$, so the aligned update remains bounded relative to the movement selected by the native optimizer. This supplies the magnitude of d_t: $d_t \in \{0, \widetilde{d}_t\}$.

Because the alignment is derived from a local linear model, ORACLE verifies that its predicted improvement persists under the nonlinear constraints,

$$\mathrm{Pred}_t = \Phi_{S_t}(\bar{\theta}_t) - \frac{1}{2}\|b_t + C_t \widetilde{d}_t\|_2^2, \qquad \mathrm{Ared}_t = \Phi_{S_t}(\bar{\theta}_t) - \Phi_{S_t}(\bar{\theta}_t + \widetilde{d}_t). \tag{12}$$

and the candidate proceeds only if $\mathrm{Pred}_t > 0$, $\mathrm{Ared}_t > 0$, and $\mathrm{Ared}_t / \mathrm{Pred}_t \ge \rho_{\min}$. Because this check reuses S_t, it tests local-model agreement rather than independent improvement. ORACLE therefore fixes the candidate and evaluates it on an independent audit support S'_t, accepting only if

$$\Phi_{S'_t}(\bar{\theta}_t + \widetilde{d}_t) \le \Phi_{S'_t}(\bar{\theta}_t). \tag{13}$$

Val_t is the event that both checks pass. Finally,

$$d_t = \begin{cases} \widetilde{d}_t, & \text{if all checks pass,} \\ 0, & \text{otherwise,} \end{cases} \qquad \theta_{t+1} = \theta_t + s_t + d_t, \tag{14}$$

which is equation 1. Any rejection recovers the exact native update while leaving ξ_{t+1} unchanged. In summary, the native optimizer *proposes*, the constraints *check*, ORACLE *aligns* in the certified native geometry, and validation determines whether the aligned proposal is committed.

4 Experiment

The central claim of ORACLE is that constraint alignment should be defined relative to the movement produced by the native optimizer. We test this claim from breadth to mechanism, and structure our experiments around the following research questions: Q1 varies optimizer geometry; Q2 varies model formulation; Q3 quantifies the cost and solution-level benefit of alignment; Q4 compares

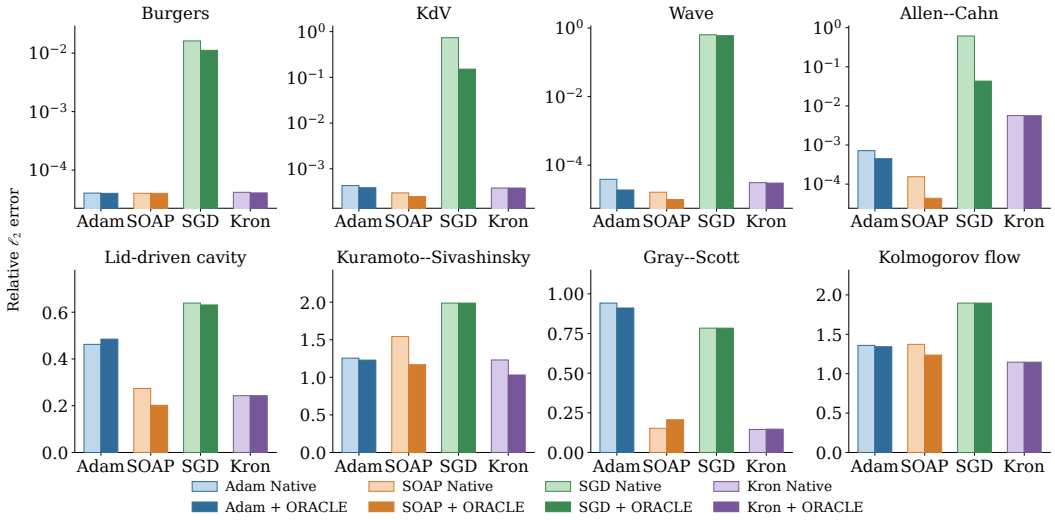

Figure 2: **Performance across optimizer geometries.** Endpoint rel-L_2 error across eight PDE benchmarks for Adam, SOAP, SGD, and PSGD-Kron, comparing each Native VS Native+ORACLE. Bars show means; the standard deviation across three independent runs is $< 5\%$ for all reported entries.

objective, gradient, and post-optimizer intervention; and Q5 isolates ORACLE's geometry, authority, and validation mechanisms.

Within every Native ORACLE pair, we hold the architecture, native objective, optimizer configuration, learning-rate schedule, sampling pipeline, and training budget fixed; ORACLE acts only after the native optimizer forms its realized update. Our primary metric is relative ℓ_2 solution error (rel-L2), with lower values indicating greater solution accuracy. Complete architectures, optimizer settings, sampling protocols, and ORACLE hyperparameters appear in Appendix C.

4.1 Q1: Does ORACLE Work Across Optimizer Geometries?

ORACLE defines alignment relative to the geometry of the native optimizer. Its first test is therefore whether the same alignment principle remains effective when that geometry changes. We pair ORACLE with SGD Robbins & Monro (1951), Adam Kingma & Ba (2014), SOAP Vyas et al. (2025), and PSGD-Kron Li (2017) across eight PDE benchmarks, fixing PirateNet Wang et al. (2024) throughout. These optimizers span Euclidean, diagonal-adaptive, and structured preconditioned update geometries.

Results. Figure 2 shows that ORACLE improves or matches Native in $30/32$ valid PDE optimizer pairs: 23 improve, 7 match, and 2 degrade. Improvements occur under every optimizer family, spanning Euclidean SGD, diagonal-adaptive Adam, and the structured geometries of SOAP and PSGD-Kron. ORACLE therefore does not depend on a single optimizer or a shared update geometry.

Q1 establishes breadth across native optimizer geometries, but all runs use PirateNet. Q2 tests whether the same behavior persists when the model formulation changes. Complete results appear in Appendix D.1.

4.2 Q2: Does Alignment Persist Across Model Formulations?

Q1 varies the optimizer while fixing PirateNet, leaving open whether the effect depends on that model. We therefore evaluate a standard MLP, a conventional PINN Raissi et al. (2019), and PirateNet Wang et al. (2024) on Wave, Burgers, Allen-Cahn, and KdV under Adam and SOAP. The MLP and PINN share the same backbone but differ in whether the PDE residual enters the native objective, while PirateNet changes both the architecture and its physics-specific inductive structure.

Results. Figure 3 shows that ORACLE improves or matches Native in $22/24$ configurations. The effect spans all three model formulations and both native optimizers, with gains reaching 60.6% for PINN+Adam on Burgers and 90.6% for PirateNet+SOAP on KdV.

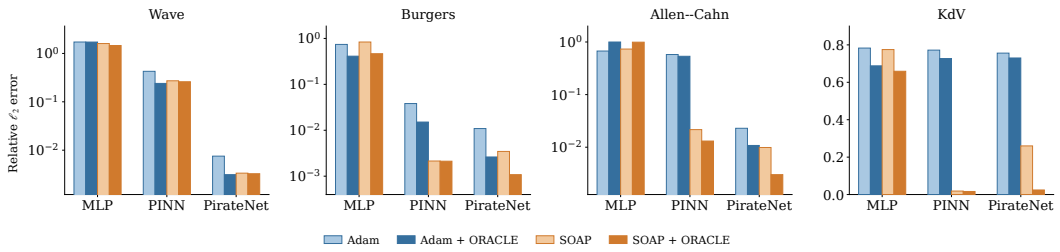

Figure 3: **Performance across model formulations.** Endpoint rel-L_2 error on Wave, Burgers, Allen-Cahn, and KdV for MLP, PINN, and PirateNet under Adam and SOAP, VS ORACLE. Lower is better. Each entry is mean, the std-dev across three independent runs < 7% for reported entries.

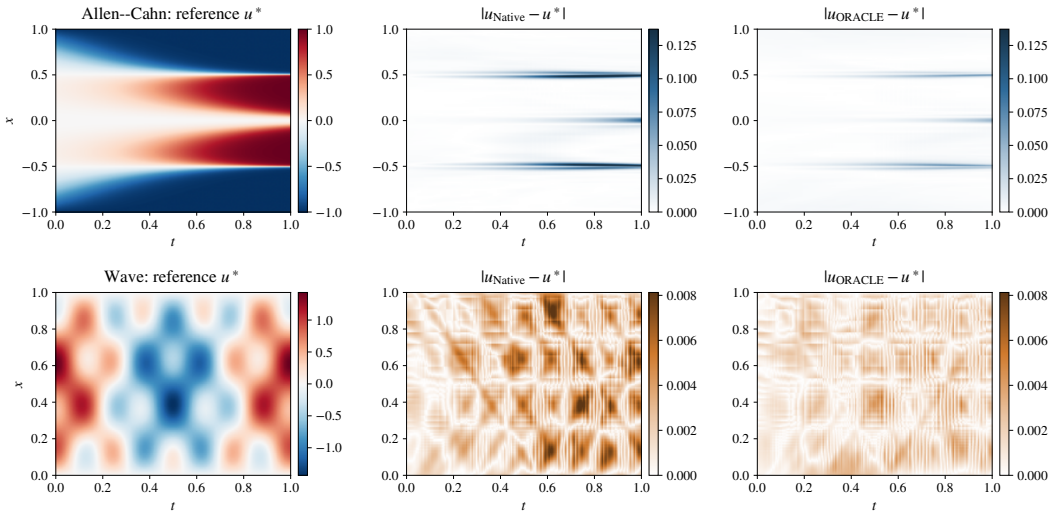

Figure 4: **Representative solution and error fields.** Reference solution (left), Native pointwise error (middle), and ORACLE pointwise error (right) for Allen–Cahn and Wave. ORACLE reduces interface localized error on Allen–Cahn and broader space time error on Wave. Native and ORACLE use the same error scale within each PDE.

The two degradations occur for Pure MLP on Allen-Cahn under Adam and SOAP, showing that ORACLE does not override limitations of the underlying learning formulation. Nevertheless, Q2 extends the Q1 result beyond PirateNet and shows that optimizer-relative alignment persists across substantially different model formulations.

Where do the gains appear? Aggregate rel-L_2 obscures where solution error is reduced. Figure 4 compares the reference solution with matched Native and ORACLE pointwise error fields. On Allen–Cahn, Native error is concentrated along the phase interfaces, where ORACLE substantially attenuates the error bands. On Wave, the reduction is distributed more broadly across space time. Thus, the endpoint gains reflect meaningful changes in the learned solution rather than only improvements in an aggregate metric. Complete results appear in Appendix D.2.

4.3 Q3: What Does ORACLE Cost, and What Does That Cost Buy?

Q1 and Q2 establish that ORACLE remains effective across optimizer geometries and model formulations. We next ask what these gains cost, whether they persist beyond endpoint error, and whether they amortize under a fixed compute budget.

Results. Figure 5(a) shows that ORACLE reduces normalized endpoint error to $0.90\times$, $0.75\times$, and $0.38\times$ Native for MLP, PINN, and PirateNet, respectively. These gains come at a cost: ORACLE requires approximately 4.0–$5.8\times$ more time per optimization step.(Fig. 5b).

Panel 5(c) asks whether the improvement persists through the physical evolution of the PDE rather than appearing only in the final aggregate error. At each physical time slice, we compare Native and

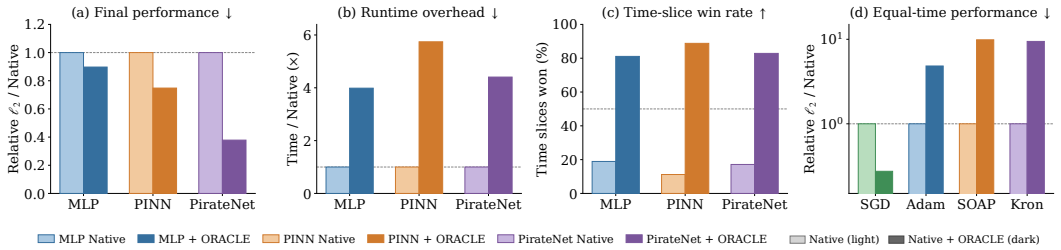

Figure 5: **Cost-benefit of ORACLE.** Endpoint accuracy, per-step runtime, temporal win rate, and equal-wall-clock performance across model formulations and optimizer geometries.

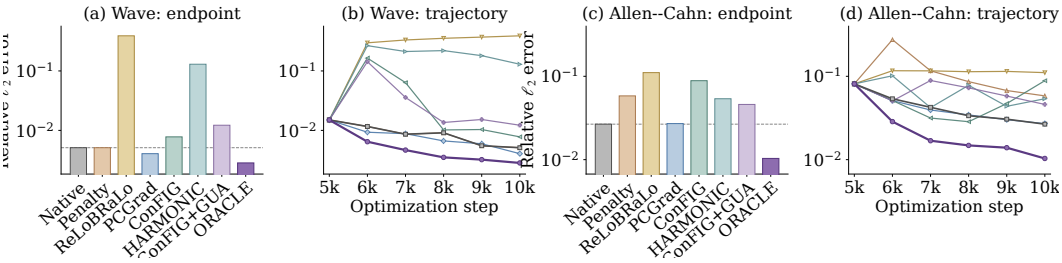

Figure 6: **Matched Q4 intervention comparison on Wave and Allen-Cahn.** Panels (a,c) show endpoint rel-L_2 error; panels (b,d) show trajectories from the shared 5k Native checkpoint to 10k. ORACLE achieves the lowest endpoint error on both PDEs. Each entry is mean, the std-dev across three independent runs < 7% for all reported entries.

ORACLE using the relative ℓ_2 error over the complete spatial field; ORACLE wins a slice only when its full spatial prediction is more accurate. Across matched configurations, ORACLE wins 81.1%, 88.8%, and 82.9% of evaluated time slices for MLP, PINN, and PirateNet, respectively. Thus, the endpoint gains reflect persistent improvements throughout the predicted dynamics rather than a small set of favorable spatial locations or times.

Panel 5(d) imposes the stricter equal-wall-clock test, giving Native and ORACLE the same elapsed time rather than the same number of optimization steps. This comparison combines update quality with update throughput. ORACLE still reduces aggregate error by approximately 73% with SGD, showing that its alignment benefit can exceed its correction cost. For Adam, SOAP, and PSGD-Kron, the additional correction does not yet recover its runtime overhead within the same budget.

This sharpens Q1: optimizer-relative alignment remains effective across all tested geometries, while its compute-normalized benefit depends on how quickly that geometry-aware correction can be realized. Together, the results separate *alignment effectiveness* from *compute efficiency*. For detailed analysis see Appendix D.3.

4.4 Q4: Where Should Constraint Intervention Occur?

Q1, Q2 and Q3 establish breadth, but do not determine whether the gains arise from optimizer-relative alignment or merely from additional constraint handling. We therefore compare objective, gradient, and post-optimizer interventions on Allen-Cahn and Wave. Every method branches from the same $5{,}000$-step Native and receives an identical additional $5{,}000$ optimization steps, controlling initialization, optimizer state, and training horizon.

Results. Figure 6 shows that ORACLE achieves the lowest endpoint rel-L_2 error on both PDEs, improving over Native by 61.2% on Allen–Cahn and 44.6% on Wave. The post-fork trajectories strengthen this result: from the shared 5k Native, ORACLE remains below Native at every recorded milestone and decreases monotonically to the 10k endpoint on both problems. Objective- and gradient-level interventions do not show the same consistency. Post-optimizer intervention alone does not explain which part of ORACLE produces the gain. Q5 therefore fixes *where* constraints act and examines *how* the resulting update is aligned.

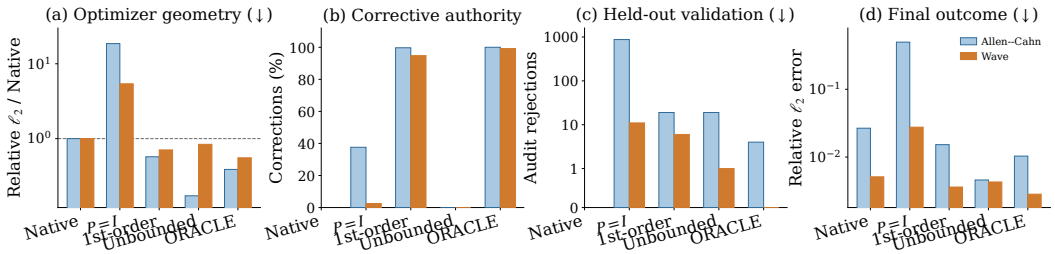

Figure 7: **Mechanism analysis of ORACLE on Allen–Cahn and Wave.** (a) Native optimizer geometry is essential; (b) the authority bound is active for nearly all accepted alignments; (c) held-out validation rejects unreliable candidates; and (d) full ORACLE improves both PDEs more consistently

4.5 Q5: What Makes Optimizer-Relative Alignment Work?

Q5 therefore examines three mechanisms: optimizer-relative geometry, corrective authority, and validation. We compare five settings on Allen-Cahn and Wave: *Native* applies no post-optimizer correction; $P_t = I$ replaces the optimizer metric with Euclidean geometry; *First-order* retains the native metric but uses the direct correction $d \propto -P_t^{-1} c_t$; *Unbounded* removes the authority cap; and *ORACLE* combines the full optimizer-relative, bounded, and validated alignment. Figure 7 traces these choices from alignment behavior to final solution error.

To compare how each mechanism affects the final solution on a common scale, we normalize endpoint error by the matched Native optimizer, $R_{\text{err}} = \frac{\text{relL2(method)}}{\text{relL2(Native)}}$, where $R_{\text{err}} < 1$ denotes improvement. Replacing the optimizer-relative metric with $P_t = I$ increases error to $18.6\times$ Native on Allen–Cahn and $5.4\times$ on Wave (Fig. 7a). In contrast, the metric-aware first-order variant and full ORACLE remain below Native on both PDEs. Thus, successful alignment depends strongly on measuring movement in the geometry induced by the native optimizer rather than in a common Euclidean metric. ORACLE bounds the aligned adjustment in the same geometry, $\|\widetilde{d}_t\|_{P_t} \le \beta_t \|s_t\|_{P_t}$. This bound is active on $100\%/99\%$ of accepted ORACLE alignments on Allen-Cahn/Wave and $100\%/95\%$ for the metric-aware first-order variant, compared with $38\%/2.5\%$ under $P_t = I$ (Fig. 7b). The optimizer metric therefore affects not only the alignment direction but also what constitutes a permissible adjustment relative to the native step. ORACLE evaluates each fixed candidate on independent support before commitment. Over 5k active steps, the audit rejects 875 candidates under $P_t = I$ versus only 4 under ORACLE on Allen–Cahn, and 11 versus 0 on Wave (Fig. 7c). The Euclidean ablation therefore produces substantially more candidates whose apparent local improvement does not persist on held-out constraints. This diagnostic links the geometry used to construct the alignment with its off-support reliability. Figure 7(d) shows the resulting endpoint error. Removing the authority bound gives the lowest Allen-Cahn error, but provides a smaller gain on Wave; full ORACLE improves both PDEs substantially. We therefore do not interpret every mechanism as independently necessary. Rather, ORACLE combines three roles: the native optimizer geometry determines *how* alignment is measured, the authority bound limits *how far* the proposal may be redirected, and held-out validation determines *whether* the aligned proposal is committed. Extended mechanism analysis appears in Appendix E.

5 Conclusion and Limitations

We introduced ORACLE, which changes both where and how constraints interact with optimization: it acts on the realized native update, constructs a joint local model of heterogeneous constraints at the proposed endpoint, and aligns the resulting update in the geometry of the optimizer that produced the step. Across five experimental questions, ORACLE improves or matches Native in 94% of optimizer-PDE settings and 92% of model-formulation settings, with endpoint error reductions of up to 90.6% and improvements that persist across the solution domain and physical time. These improvements come with a clear trade-off: ORACLE incurs additional computation and depends on the fidelity of its local correction model. Future work will target cheaper, selective interventions through adaptive correction budgets and reuse of native optimizer geometry, together with richer local models for strongly nonlinear regimes. Extending the current single-step analysis to trajectory-level guarantees and testing optimizer-relative alignment beyond PDE learning are natural next steps.

AI USE

In this work, we used generative AI tools for language editing, improving clarity and organization, drafting and debugging code, and LaTeX formatting. We did not use generative AI to generate experimental measurements, fabricate data, determine reported results, or make final scientific conclusions.

ETHICS STATEMENT

ORACLE studies optimizer-relative constrained learning and introduces a general mechanism for aligning realized optimizer updates with additional constraints. Our experiments are limited to physics-informed learning on numerical PDE benchmarks and involve no human subjects, personal data, or sensitive datasets. The method itself is domain general and could in principle be applied in safety-critical or decision-making systems, where the choice and validity of the imposed constraints would require domain-specific evaluation. ORACLE does not guarantee satisfaction of arbitrary real-world safety constraints, and its local alignment and validation guarantees should not be interpreted as such.

REPRODUCIBILITY STATEMENT

We provide the complete ORACLE algorithm and its mathematical formulation in the main paper, with supporting derivations, proofs, optimizer-specific geometry adapters, and implementation details in the appendices. We report the model architectures, optimizer configurations, ORACLE hyperparameters, PDE definitions, sampling procedures, training schedules, reference-solution settings, evaluation protocols, and complete experimental results required to reproduce the study. Native and ORACLE comparisons use matched architectures, objectives, optimizer settings, sampling pipelines, and training budgets unless stated otherwise. We will release the code and configuration files required to reproduce the reported experiments, metrics, and figures.

REFERENCES

Albert S Berahas, Jorge Nocedal, and Martin Takác. A multi-batch l-bfgs method for machine learning. *Advances in Neural Information Processing Systems*, 29, 2016.

Rafael Bischof and Michael A Kraus. Multi-objective loss balancing for physics-informed deep learning. *Computer Methods in Applied Mechanics and Engineering*, 439:117914, 2025.

Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical gauss-newton optimisation for deep learning. In *International Conference on Machine Learning*, pp. 557–565. PMLR, 2017.

Dohyun Bu, Yujung Byun, and JONGSEOK LEE. Harmonized cone for feasible and non-conflict directions in training physics-informed neural networks. In *International Conference on Learning Representations*, volume 2026, pp. 112050–112093, 2026.

Richard H Byrd, Frank E Curtis, and Jorge Nocedal. An inexact sqp method for equality constrained optimization. *SIAM Journal on Optimization*, 19(1):351–369, 2008.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018.

Alp Dener, Marco Andres Miller, Randy Michael Churchill, Todd Munson, and Choong-Seock Chang. Training neural networks under physical constraints using a stochastic augmented lagrangian approach. *arXiv preprint arXiv:2009.07330*, 2020.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus. Mathématique*, 350(5-6):313–318, 2012.

Zhehang Du and Weijie Su. The newton-muon optimizer. *arXiv preprint arXiv:2604.01472*, 2026.

Ferdinando Fioretto, Pascal Van Hentenryck, Terrence WK Mak, Cuong Tran, Federico Baldo, and Michele Lombardi. Lagrangian duality for constrained deep learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 118–135. Springer, 2020.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *International Conference on Machine Learning*, pp. 573–582. PMLR, 2016.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

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

Dou Huang, Haoran Zhang, Xuan Song, and Ryosuke Shibasaki. Differentiable projection for constrained deep learning. *arXiv preprint arXiv:2111.10785*, 2021.

Ashfaq Iftakher, Rahul Golder, Bimol Nath Roy, and MM Faruque Hasan. Physics-informed neural networks with hard nonlinear equality and inequality constraints. *Computers & Chemical Engineering*, pp. 109418, 2025.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

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

Lingkai Kong, Wenhao Mu, Jiaming Cui, Yuchen Zhuang, B Aditya Prakash, Bo Dai, and Chao Zhang. Df2: Distribution-free decision-focused learning. *arXiv preprint arXiv:2308.05889*, 2023.

Giacomo Lastrucci and Artur M Schweidtmann. Enforce: Nonlinear constrained learning with adaptive-depth neural projection. *arXiv preprint arXiv:2502.06774*, 2025.

Diogo Lavado, Cláudia Soares, and Alessandra Micheletti. Achieving constraints in neural networks: A stochastic augmented lagrangian approach. *arXiv preprint arXiv:2310.16647*, 2023.

Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE transactions on neural networks and learning systems*, 29(5):1454–1466, 2017.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in neural information processing systems*, 34:18878–18890, 2021.

Qiang Liu, Mengyu Chu, and Nils Thuerey. Config: Towards conflict-free training of physics informed neural networks. In *International Conference on Learning Representations*, volume 2025, pp. 59531–59566, 2025.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 80:1623–1701, 2024.

Jayanta Mandi, Marianne Defresne, Senne Berden, and Tias Guns. Feasibility-aware decision-focused learning for predicting parameters in the constraints. *Advances in Neural Information Processing Systems*, 38:163285–163309, 2026.

Viggo Moro and Luiz Chamon. Solving differential equations with constrained learning. In *International Conference on Learning Representations*, volume 2025, pp. 44906–44948, 2025.

Yatin Nandwani, Abhishek Pathak, and Parag Singla. A primal dual formulation for deep learning with constraints. *Advances in neural information processing systems*, 32, 2019.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*, 2022.

Duy P Nguyen, Kai-Chieh Hsu, Wenhao Yu, Jie Tan, and Jaime F Fisac. Gameplay filters: Robust zero-shot safety through adversarial imagination. *arXiv preprint arXiv:2405.00846*, 2024.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Pratik Rathore, Weimu Lei, Zachary Frangella, Lu Lu, and Madeleine Udell. Challenges in training pinns: A loss landscape perspective. *arXiv preprint arXiv:2402.01868*, 2024.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

J Ben Rosen. The gradient projection method for nonlinear programming. part ii. nonlinear constraints. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):514–532, 1961.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International conference on machine learning*, pp. 4596–4604. PMLR, 2018.

Chongjie Si, Debing Zhang, and Wei Shen. Adamuon: Adaptive muon optimizer. *arXiv preprint arXiv:2507.11005*, 2025.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam for language modeling. In *International Conference on Learning Representations*, volume 2025, pp. 93423–93444, 2025.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a.

Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021b.

Sifan Wang, Bowen Li, Yuhan Chen, and Paris Perdikaris. Piratenets: Physics-informed deep learning with residual adaptive networks. *Journal of Machine Learning Research*, 25(402):1–51, 2024.

Sifan Wang, Bowen Li, Paris Perdikaris, et al. Gradient alignment in physics-informed neural networks: A second-order optimization perspective. *Advances in Neural Information Processing Systems*, 38:168482–168532, 2026.

Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. *Journal of Machine Learning Research*, 21(219):1–30, 2020.

Jing Xiao, Xinhai Chen, Qinglin Wang, Menghan Jia, Zhiquan Lai, Dongsheng Li, Jie Liu, and Tiejun Li. Gradient-update mismatch: Rethinking conflict-free training of physics-informed neural networks. *arXiv preprint arXiv:2609.01558*, 2026.

Shuo Xie, Tianhao Wang, Beining Wu, and Zhiyuan Li. A tale of two geometries: Adaptive optimizers and non-euclidean descent. In *International Conference on Learning Representations*, volume 2026, pp. 76867–76913, 2026.

Jiachen Yao, Chang Su, Zhongkai Hao, Songming Liu, Hang Su, and Jun Zhu. Multiadam: Parameter-wise scale-invariant optimizer for multiscale training of physics-informed neural networks. In *International conference on machine learning*, pp. 39702–39721. PMLR, 2023.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Borong Zhang, Yuhao Zhang, Jiaming Ji, Yingshan Lei, Juntao Dai, Yuanpei Chen, and Yaodong Yang. Safevla: Towards safety alignment of vision-language-action model via constrained learning. *Advances in Neural Information Processing Systems*, 38:153335–153373, 2026.

A Extended Related Work

The main text positions ORACLE relative to the principal families of constraint-handling methods. Here we expand that discussion around the interface central to this work: the distinction between the gradient supplied to an optimizer and the parameter displacement that the optimizer ultimately produces.

A.1 Optimizer Geometry and the Realized Update

Modern optimizers transform the instantaneous gradient using information accumulated across training. AdaGrad adapts coordinate scales from gradient history Ward et al. (2020); Adam combines first- and second-moment statistics Kingma & Ba (2014); and Adafactor factorizes adaptive statistics to reduce memory Shazeer & Stern (2018). More structured methods alter the update through non-diagonal geometry: natural gradient uses the Fisher metric Kakade (2001), K-FAC approximates Fisher blocks with Kronecker factors Grosse & Martens (2016), Shampoo maintains tensor-structured preconditioners Gupta et al. (2018), and SOAP combines structured preconditioning with Adam-style adaptation in an evolving basis Vyas et al. (2025). Newton, Gauss Newton, and quasi-Newton methods likewise reshape parameter motion using curvature information Berahas et al. (2016).

Across these methods, the gradient is an input to the optimizer rather than the parameter movement itself. Momentum, adaptive scaling, and structured preconditioning can change both the scale and direction of the resulting step. ORACLE refers to this optimizer-transformed displacement as the *realized optimizer update*. This distinction becomes consequential when constraint information must interact with an optimizer whose state already defines a nontrivial geometry.

A.2 Constraint Handling and the Intervention Point

Classical constrained optimization typically introduces constraint information before or during construction of the optimization trajectory. Penalty methods add violation terms to the objective; primal dual methods jointly evolve parameter and multiplier variables Nandwani et al. (2019); and augmented-Lagrangian approaches combine multiplier dynamics with explicit feasibility penalties Dener et al. (2020); Lavado et al. (2023). Related formulations extend these mechanisms to constrained neural networks Fioretto et al. (2020). In each case, constraint information affects the signal from which the optimizer subsequently constructs its step.

Multi-objective methods intervene more directly on gradients. GradNorm balances objectives using gradient magnitudes Chen et al. (2018); MGDA constructs a common descent direction Désidéri (2012); PCGrad removes pairwise conflicting components Yu et al. (2020); CAGrad trades average descent against worst-objective conflict Liu et al. (2021); and Nash-MTL derives a shared direction through bargaining Navon et al. (2022). These methods address imbalance or directional conflict before the native optimizer applies its state-dependent transformation.

Projection and local constrained-optimization methods act closer to the parameter movement itself. Projected-gradient methods map candidate steps toward feasible sets, SQP constructs locally constrained subproblems, and trust-region mechanisms restrict movement to regions where local models remain reliable. Related learning methods include GEM, which projects gradients that would interfere with previously learned tasks Lopez-Paz & Ranzato (2017), and differentiable projection layers that explicitly map predictions or parameters toward feasibility Huang et al. (2021). These methods separate a proposed direction from a constraint-induced modification, but the proposal is often defined before a stateful optimizer has produced its realized update.

This ordering matters when the optimizer itself carries useful structure. For Adam, SOAP, Shampoo, or other stateful optimizers, modifying the raw gradient does not directly specify the parameter displacement that will follow. ORACLE instead takes the realized native step as the reference movement and asks whether that movement should be aligned with the constraints.

A.3 Constraint Geometry in Physics-Informed Learning

PINNs provide a demanding setting for this distinction because their training objectives combine PDE residuals, initial conditions, boundary conditions, and possibly observations Raissi et al. (2019). These components can differ substantially in scale, conditioning, spatial support, and gradient direction.

Several methods address this difficulty before the optimizer forms its step. Gradient-statistics-based annealing adapts loss weights from constituent gradient magnitudes Wang et al. (2021a); NTK-based weighting accounts for unequal convergence rates Wang et al. (2021b); ReLoBRaLo adapts weights using relative loss progress Bischof & Kraus (2025); and MultiAdam maintains separate adaptive statistics for different PINN loss components Yao et al. (2023). Other work instead strengthens the optimizer geometry itself through curvature-aware or structured methods, including NysNewton-CG Rathore et al. (2024), Shampoo Gupta et al. (2018), and SOAP Vyas et al. (2025). Recent analyses further show that PINN difficulty is not explained by gradient magnitude alone: constituent gradients can remain directionally incompatible even after loss balancing Wang et al. (2026), while benchmark studies show substantial variation in optimizer behavior across PDE families Hao et al. (2024).

These developments create the setting studied by ORACLE. The native optimizer may already encode useful task-dependent scaling or preconditioning, while the physical constraints provide a separate collection of local sensitivities. Combining the two before optimization changes the signal from which the native update is constructed; operating only on the raw gradient acts before the optimizer geometry has been expressed. ORACLE instead preserves the realized native update as the reference movement.

A.4 Positioning ORACLE

ORACLE operates after the native optimizer has formed its realized step. It first recovers and certifies the geometry associated with that step, then evaluates whether the movement is locally adverse to a sampled constraint model. When alignment is required, ORACLE constructs the adjustment in the same optimizer-relative metric, bounds its authority relative to the native step, and commits it only after nonlinear validation and evaluation on independent support.

The distinction is therefore not merely *when* constraint information is introduced. ORACLE separates two forms of structure: *optimizer geometry*, which determines how the native optimizer prefers to move, and *constraint geometry*, which determines how that realized movement affects local constraint violation. Matrix-free Jacobian vector and vector Jacobian products allow the latter to be represented without materializing the parameter-space Jacobian, while native-relative authority control and held-out validation regulate whether the resulting alignment is committed.

ORACLE therefore complements rather than replaces the underlying optimizer or constraint model. Its contribution is the interface between them: constraints are evaluated against the movement the optimizer actually produced, and any alignment is expressed relative to the geometry of that same movement.

B Supporting Theory and Derivations

This appendix provides the technical details supporting Section 3. We first formalize the finite-support constraint estimator, then derive the optimizer-specific geometry adapters used to recover P_t, characterize when the PSGD–Kron adapter must abstain, and justify the endpoint signal used to open the ORACLE alignment path. Subsequent subsections derive the optimizer-relative alignment, authority bound, and validation mechanisms.

B.1 Finite-Support Constraint Estimator

Fix iteration t and let $\mathcal{F}_t$ contain all randomness determined before the construction support S_t is drawn. In particular, θ_t, the realized native step s_t, the proposed endpoint $\bar{\theta}_t = \theta_t + s_t$, the family weights $\omega_{k,t}$, and the sample counts m_k are $\mathcal{F}_t$-measurable. For each constraint family k, draw

$$z_{k,1}, \dots, z_{k,m_k} \overset{\text{i.i.d.}}{\sim} \nu_k.$$

The sampled constraint energy is

$$\Phi_{S_t}(\theta) = \sum_{k=1}^{K} \omega_{k,t} \frac{1}{m_k} \sum_{j=1}^{m_k} r_k(\theta; z_{k,j})^2. \tag{15}$$

This is equivalent to the weighted residual form $\frac{1}{2} r_{S_t}(\theta)^\top W_t r_{S_t}(\theta)$ used in the main text, with each residual from family k assigned weight $2\omega_{k,t}/m_k$.

For any $\mathcal{F}_t$-measurable evaluation point Θ_t,

$$\mathbb{E}\left[\Phi_{S_t}(\Theta_t) \mid \mathcal{F}_t\right] = \sum_{k=1}^{K} \omega_{k,t} \mathbb{E}_{z\sim\nu_k}\left[r_k(\Theta_t; z)^2\right] \tag{16}$$

$$= \Phi_t(\Theta_t). \tag{17}$$

Hence

$$\boxed{\mathbb{E}\left[\Phi_{S_t}(\Theta_t) \mid \mathcal{F}_t\right] = \Phi_t(\Theta_t)} \tag{18}$$

and, whenever differentiation and expectation may be interchanged,

$$\mathbb{E}\left[\nabla\Phi_{S_t}(\Theta_t) \mid \mathcal{F}_t\right] = \nabla\Phi_t(\Theta_t). \tag{19}$$

The ordering used by ORACLE satisfies this condition because $\bar{\theta}_t$ is fixed before S_t is sampled. For finite stored constraint sets, ν_k denotes the corresponding empirical distribution.

B.2 Certified Optimizer Geometry

Let u_t denote the optimizer's effective direction after its stateful transformation but before application of its local mobility. ORACLE uses an optimizer-relative geometry only when it can identify an SPD operator B_t satisfying

$$s_t = -B_t u_t, \qquad P_t = B_t^{-1} \succ 0, \tag{20}$$

and this representation reconstructs the realized deployed step.

Certification. Define the reconstruction defect

$$e_t = s_t + B_t u_t.$$

ORACLE accepts the recovered geometry only if

$$\boxed{\|e_t\|_2 \le \tau_{\mathrm{rel}}\|s_t\|_2 + \tau_{\mathrm{abs}}, \qquad \tau_{\mathrm{rel}} = 10^{-4}, \quad \tau_{\mathrm{abs}} = 10^{-9}.} \tag{21}$$

If this test or an optimizer-specific validity condition fails, ORACLE abstains and preserves the Native proposal; it never substitutes $P_t = I$.

For nonzero native steps, the corresponding relative reconstruction residual is

$$\varepsilon_{\mathrm{cert},t} = \frac{\|s_t + B_t u_t\|_2}{\|s_t\|_2}. \tag{22}$$

Runtime certification is evaluated on the deployed numerical path; exact algebraic identities alone are not treated as production certification.

SGD. For SGD without momentum,

$$s_t = -\eta_t g_t.$$

Thus

$$u_t = g_t, \qquad B_t = \eta_t I, \qquad P_t = \eta_t^{-1} I. \tag{23}$$

For $\eta_t > 0$, both operators are SPD and equation 20 holds exactly.

Adam. Let $\widehat{\mu}_t$ and $\widehat{\nu}_t$ denote the bias-corrected moments used by the deployed Adam update, and define

$$D_t = \operatorname{diag}\left(\sqrt{\widehat{\nu}_t + \varepsilon_{\text{root}}} + \varepsilon\right) \succ 0.$$

Then

$$s_t = -\eta_t D_t^{-1} \widehat{\mu}_t,$$

so

$$u_t = \widehat{\mu}_t, \qquad B_t = \eta_t D_t^{-1}, \qquad P_t = \eta_t^{-1} D_t. \tag{24}$$

The adapter follows the numerical stabilizers and state timing of the deployed optimizer exactly.

SOAP. Let Q_t denote the orthogonal basis that generated the realized SOAP step, $D_t \succ 0$ its adaptive denominator, $h_t > 0$ the scalar step factor, and u_t^{soap} the effective momentum in parameter coordinates. The realized step is

$$s_t = -h_t Q_t^\top D_t^{-1} Q_t u_t^{\text{soap}},$$

which gives

$$u_t = u_t^{\text{soap}}, \qquad B_t = h_t Q_t^\top D_t^{-1} Q_t, \qquad P_t = h_t^{-1} Q_t^\top D_t Q_t. \tag{25}$$

Because Q_t is orthogonal and $D_t \succ 0$, both B_t and P_t are SPD.

The basis timing is essential. If SOAP refreshes its basis after constructing the current step, certification must use the basis that actually generated s_t; using the refreshed basis describes a different operator and may fail the reconstruction test.

PSGD–Kron. In the unclipped linear regime, let

$$K_t^{(\ell)} = \bigotimes_i A_{i,t}^{(\ell)}, \qquad A_{i,t}^{(\ell)} \succ 0,$$

and

$$K_t = \bigoplus_\ell K_t^{(\ell)}.$$

Kronecker products and direct sums of SPD operators remain SPD. If

$$s_t = -\eta_t K_t u_t^{\text{kron}},$$

then

$$u_t = u_t^{\text{kron}}, \qquad B_t = \eta_t K_t, \qquad P_t = \eta_t^{-1} K_t^{-1}. \tag{26}$$

This representation is valid only while the deployed update remains linear in the preconditioned direction.

B.3 PSGD–Kron Clipping and Abstention

Let

$$p = K_t u_t^{\text{kron}} \in \mathbb{R}^n$$

denote the linear preconditioned direction before clipping. The deployed implementation first applies the global normalization

$$\gamma(p) = p \, \frac{\sqrt{n}}{\max(\sqrt{n}, \|p\|_2)} \tag{27}$$

and then the elementwise map

$$h(y)_j = \min\{1, \max\{-1, y_j\}\}. \tag{28}$$

The realized step is therefore

$$s_t = -\eta_t h(\gamma(p)).$$

A global norm rescaling alone would preserve an SPD linear mobility. In the deployed rule, however,

$$\|p\|_2 > \sqrt{n} \quad \Longrightarrow \quad \|p\|_\infty \geq \frac{\|p\|_2}{\sqrt{n}} > 1,$$

and hence

$$\boxed{\text{global clipping active} \quad \Longrightarrow \quad \text{elementwise clipping active.}} \tag{29}$$

Once the elementwise clip binds, the optimizer response is coordinatewise nonlinear. The linear Kronecker operator can no longer represent the mobility used to evaluate arbitrary alignment directions. ORACLE therefore rejects the Kron adapter on such steps rather than replacing it with an unverified geometry.

B.4 Endpoint Constraint Signal

Suppress the iteration index and write

$$\theta = \theta_t, \qquad s = s_t, \qquad \bar{\theta} = \theta + s, \qquad \Phi = \Phi_{S_t}, \qquad c = \nabla\Phi(\bar{\theta}), \qquad P = P_t.$$

Assume that $\nabla\Phi$ is Lipschitz along the realized native segment $\{\theta + \tau s : \tau \in [0,1]\}$ in the optimizer metric:

$$\|\nabla\Phi(x) - \nabla\Phi(y)\|_{P^{-1}} \le L_\Phi^P \|x - y\|_P. \tag{30}$$

By the fundamental theorem of calculus,

$$\Phi(\bar{\theta}) - \Phi(\theta) - c^\top s = \int_0^1 \left[\nabla\Phi(\theta + \tau s) - \nabla\Phi(\bar{\theta})\right]^\top s \, d\tau. \tag{31}$$

Applying dual Cauchy–Schwarz and equation 30 yields

$$\boxed{\left|\Phi_{S_t}(\bar{\theta}_t) - \Phi_{S_t}(\theta_t) - c_t^\top s_t\right| \le \frac{L_{\Phi,t}^P}{2} \|s_t\|_{P_t}^2.} \tag{32}$$

Thus $c_t^\top s_t$ is the first-order term associated with the sampled constraint-energy change along the realized native movement. Because ORACLE does not estimate $L_{\Phi,t}^P$ at runtime, the deployed rule

$$c_t^\top s_t > 0$$

is a first-order trigger, not an unconditional finite-step sign certificate.

Why evaluate at the endpoint? Define

$$h(\tau) = \Phi(\theta + \tau s).$$

If Φ is convex along the realized native segment, then h' is nondecreasing and

$$h'(0) \le h(1) - h(0) \le h'(1).$$

Equivalently,

$$\boxed{\nabla\Phi(\theta)^\top s \le \Phi(\theta + s) - \Phi(\theta) \le \nabla\Phi(\theta + s)^\top s.} \tag{33}$$

Therefore,

$$\nabla\Phi(\theta + s)^\top s \le 0 \quad \Longrightarrow \quad \Phi(\theta + s) \le \Phi(\theta) \tag{34}$$

under segment convexity.

The endpoint signal is therefore asymmetric. A non-positive value provides a one-sided condition for preserving the Native proposal under the stated assumption, whereas a positive value only opens the alignment path. The resulting candidate must still satisfy the alignment-descent, authority, nonlinear-agreement, and held-out validation conditions before commitment.

B.5 Optimizer-Relative Alignment

This subsection derives the optimizer-relative alignment used in Section 3.3. We establish the equivalence between its parameter- and residual-space forms, show that both exact and finite-CG solutions produce first-order constraint-descent directions on triggered steps, and give the matrix-free operator used in implementation.

Parameter- and residual-space formulations. At the proposed endpoint $\bar{\theta}_t$, define

$$C_t = W_t^{1/2} J_t, \qquad b_t = W_t^{1/2} r_t.$$

Before authority control, ORACLE computes

$$\widehat{d}_t = \arg\min_d \left\{ \frac{1}{2}\|b_t + C_t d\|_2^2 + \frac{\lambda_t}{2}\|d\|_{P_t}^2 \right\}, \qquad \lambda_t > 0. \tag{35}$$

Because $P_t \succ 0$, the objective is strictly convex. Its optimality condition is

$$\left(C_t^\top C_t + \lambda_t P_t\right) \widehat{d}_t = -C_t^\top b_t = -c_t. \tag{36}$$

Define the optimizer-relative residual Gram operator

$$G_t = C_t P_t^{-1} C_t^\top = W_t^{1/2} J_t P_t^{-1} J_t^\top W_t^{1/2} \succeq 0. \tag{37}$$

If

$$(G_t + \lambda_t I) z_t = b_t, \tag{38}$$

then

$$\boxed{\widehat{d}_t = -P_t^{-1} C_t^\top z_t = -P_t^{-1} J_t^\top W_t^{1/2} z_t.} \tag{39}$$

To verify the equivalence,

$$\begin{aligned} &\left(C_t^\top C_t + \lambda_t P_t\right)\left(-P_t^{-1} C_t^\top z_t\right) \\ &\quad = -C_t^\top \left(C_t P_t^{-1} C_t^\top + \lambda_t I\right) z_t \qquad (40) \\ &\quad = -C_t^\top (G_t + \lambda_t I) z_t = -C_t^\top b_t = -c_t. \qquad (41) \end{aligned}$$

Hence equation 39 satisfies equation 36.

Exact-solve descent. Since $c_t = C_t^\top b_t$,

$$-c_t^\top \widehat{d}_t = b_t^\top G_t (G_t + \lambda_t I)^{-1} b_t. \tag{42}$$

Let

$$G_t = Q \,\mathrm{diag}(\sigma_i) Q^\top, \qquad \beta_i = q_i^\top b_t, \qquad \sigma_i \geq 0.$$

Then

$$-c_t^\top \widehat{d}_t = \sum_i \frac{\sigma_i}{\sigma_i + \lambda_t} \beta_i^2. \tag{43}$$

Moreover,

$$b_t^\top G_t b_t = c_t^\top P_t^{-1} c_t = \|c_t\|_{P_t^{-1}}^2. \tag{44}$$

On a triggered step, $c_t^\top s_t > 0$, so $c_t \neq 0$. Therefore $b_t^\top G_t b_t > 0$, implying that b_t has nonzero mass on at least one eigenspace with $\sigma_i > 0$. Equation 43 is therefore strictly positive, and

$$\boxed{c_t^\top \widehat{d}_t < 0.} \tag{45}$$

Thus the exact optimizer-relative solution is a strict first-order constraint-descent direction whenever the alignment path is triggered.

Finite-CG descent. The deployed method solves the residual system with conjugate gradients rather than forming $(G_t + \lambda_t I)^{-1}$. Let

$$A_t = G_t + \lambda_t I \succ 0,$$

and let $z_{t,k}$ be the k-th CG iterate for $A_t z = b_t$, initialized at $z_{t,0} = 0$. Define

$$\widehat{d}_{t,k} = -P_t^{-1} C_t^\top z_{t,k}. \tag{46}$$

Let

$$\delta_{t,k} = b_t - A_t z_{t,k}$$

denote the CG residual. For $k \geq 1$, CG orthogonality gives

$$\delta_{t,k} \perp \mathcal{K}_k(A_t, b_t),$$

while both b_t and $z_{t,k}$ belong to $\mathcal{K}_k(A_t, b_t)$. Hence

$$b_t^\top A_t z_{t,k} = b_t^\top (b_t - \delta_{t,k}) = \|b_t\|_2^2, \tag{47}$$

$$b_t^\top z_{t,k} = (A_t z_{t,k} + \delta_{t,k})^\top z_{t,k} = \|z_{t,k}\|_{A_t}^2. \tag{48}$$

Using $G_t = A_t - \lambda_t I$,

$$-c_t^\top \widehat{d}_{t,k} = b_t^\top G_t z_{t,k} \tag{49}$$

$$= \|b_t\|_2^2 - \lambda_t \|z_{t,k}\|_{A_t}^2. \tag{50}$$

From equation 48,

$$\|z_{t,k}\|_{A_t}^2 = \left\langle A_t^{-1} b_t,\, z_{t,k} \right\rangle_{A_t}.$$

Cauchy–Schwarz therefore gives

$$\|z_{t,k}\|_{A_t}^2 \leq b_t^\top A_t^{-1} b_t. \tag{51}$$

In the eigenbasis of G_t,

$$\lambda_t b_t^\top A_t^{-1} b_t = \sum_i \frac{\lambda_t}{\sigma_i + \lambda_t} \beta_i^2. \tag{52}$$

The coefficient equals one only when $\sigma_i = 0$ and is strictly smaller than one when $\sigma_i > 0$. By equation 44, a triggered step has nonzero β_i for at least one $\sigma_i > 0$. Hence

$$\lambda_t b_t^\top A_t^{-1} b_t < \|b_t\|_2^2. \tag{53}$$

Combining equation 50, equation 51, and equation 53 yields

$$\boxed{c_t^\top \widehat{d}_{t,k} < 0, \qquad k \geq 1.} \tag{54}$$

Thus every nonzero finite-CG iterate is a strict first-order constraint-descent direction on a triggered step in exact arithmetic. The runtime sign test in Algorithm 1 therefore serves as a finite-precision safeguard rather than an additional modeling assumption.

Relative damping. ORACLE scales the absolute damping to the residual geometry:

$$\lambda_t = \lambda_{\mathrm{rel},t} \varrho_t, \qquad \varrho_t = \frac{b_t^\top G_t b_t}{b_t^\top b_t}. \tag{55}$$

On a triggered step, $c_t \neq 0$, and therefore $b_t \neq 0$, so the quotient is well defined. Using equation 44 and

$$b_t^\top b_t = 2\Phi_{S_t}(\bar{\theta}_t),$$

we obtain

$$\boxed{\varrho_t = \frac{\|c_t\|_{P_t^{-1}}^2}{2\Phi_{S_t}(\bar{\theta}_t)}.} \tag{56}$$

Equivalently, ϱ_t is the Rayleigh quotient of G_t along the current weighted-residual direction. It therefore provides a residual-dependent scale for the regularization rather than using a fixed absolute damping across problems and optimizer geometries. The deployed controller for $\lambda_{\mathrm{rel},t}$ is specified in Section 3.

Matrix-free application. Neither J_t nor G_t is materialized. For any residual-space vector v,

$$\boxed{G_t v = W_t^{1/2} J_t \left[P_t^{-1} \left(J_t^\top W_t^{1/2} v \right) \right].} \tag{57}$$

Each application of G_t therefore requires one VJP through $J_t^\top$, one application of the certified mobility P_t^{-1}, and one JVP through J_t. CG can consequently solve equation 38 without forming either the residual Jacobian or a parameter-space matrix.

B.6 AUTHORITY, VALIDATION, AND EXACT NATIVE REDUCTION

This subsection supports the authority and validation mechanisms in Section 3. We establish the native-relative authority bound, derive its geometric interpretation, formalize the nonlinear-agreement and held-out validation tests, and show that any rejected alignment recovers the exact Native update from the current optimizer state.

Native-relative alignment authority. On a triggered step, define

$$h_t = c_t^\top s_t > 0, \qquad \kappa_t = -c_t^\top \widehat{d}_t > 0.$$

Here h_t measures the detected first-order constraint increase along the Native step, while κ_t measures the first-order decrease provided by the optimizer-relative alignment direction.

ORACLE defines

$$\alpha_{\mathrm{need},t} = \frac{h_t}{\kappa_t + \varepsilon_{\mathrm{auth}}}, \qquad \alpha_{\mathrm{safe},t} = \frac{\beta_t \|s_t\|_{P_t}}{\|\widehat{d}_t\|_{P_t}}, \tag{58}$$

with $\varepsilon_{\mathrm{auth}} = 10^{-30}$, and sets

$$\alpha_t = \min\{\alpha_{\mathrm{need},t}, \alpha_{\mathrm{safe},t}\}, \qquad \widetilde{d}_t = \alpha_t \widehat{d}_t. \tag{59}$$

Without the numerical guard, $\alpha_{\mathrm{need},t} = h_t/\kappa_t$ is exactly the scale required to neutralize the detected first-order constraint signal:

$$c_t^\top \left(s_t + \alpha_{\mathrm{need},t} \widehat{d}_t \right) = h_t - \alpha_{\mathrm{need},t} \kappa_t = 0.$$

The numerical guard keeps the deployed scale infinitesimally below this exact cancellation point.

Since $\alpha_t \le \alpha_{\mathrm{safe},t}$,

$$\|\widetilde{d}_t\|_{P_t} = \alpha_t \|\widehat{d}_t\|_{P_t} \tag{60}$$

$$\le \beta_t \|s_t\|_{P_t}. \tag{61}$$

Therefore,

$$\boxed{\|\widetilde{d}_t\|_{P_t} \le \beta_t \|s_t\|_{P_t}.} \tag{62}$$

The denominator of $\alpha_{\mathrm{safe},t}$ is nonzero on the triggered alignment path because $c_t^\top \widehat{d}_t < 0$ implies $\widehat{d}_t \neq 0$.

Geometric interpretation of the authority bound. Let

$$x = P_t^{1/2} s_t, \qquad y = P_t^{1/2} \widetilde{d}_t.$$

Equation 62 implies

$$\|y\|_2 \le \beta_t \|x\|_2.$$

Moreover, the P_t-angle between s_t and $s_t + \widetilde{d}_t$ equals the Euclidean angle between x and $x + y$.

Normalize $\|x\|_2 = 1$ and decompose

$$y = ax + bq, \qquad q^\top x = 0, \qquad \|q\|_2 = 1, \qquad a^2 + b^2 \le \beta_t^2.$$

If

$$\vartheta = \angle(x, x + y),$$

then

$$\sin^2 \vartheta = \frac{b^2}{(1+a)^2 + b^2}. \tag{63}$$

For fixed a, this quantity increases with b^2, so its maximum lies on $a^2 + b^2 = \beta_t^2$. Substituting $b^2 = \beta_t^2 - a^2$ gives

$$f(a) = \frac{\beta_t^2 - a^2}{1 + 2a + \beta_t^2}.$$

The interior maximizer is $a = -\beta_t^2$, for which

$$f(-\beta_t^2) = \beta_t^2.$$

Hence, for $0 \le \beta_t < 1$,

$$\boxed{\angle_{P_t}\left(s_t,\, s_t + \widetilde{d}_t\right) \le \arcsin(\beta_t).} \tag{64}$$

The bound is sharp. Equality is attained in transformed coordinates by

$$y = -\beta_t^2 x + \beta_t \sqrt{1-\beta_t^2}\, q, \qquad q \perp x.$$

Thus β_t has a direct geometric interpretation: it limits how far the aligned movement may rotate away from the direction selected by the Native optimizer, measured in that optimizer's own metric.

Nonlinear agreement. The alignment direction is constructed from a local linearization, so ORACLE checks whether the predicted reduction agrees with the true nonlinear constraint energy before proceeding.

On the same construction support S_t, define

$$\mathrm{Pred}_t = \Phi_{S_t}(\bar{\theta}_t) - \frac{1}{2}\left\| b_t + C_t \widetilde{d}_t \right\|_2^2, \tag{65}$$

$$\mathrm{Ared}_t = \Phi_{S_t}(\bar{\theta}_t) - \Phi_{S_t}\left(\bar{\theta}_t + \widetilde{d}_t\right). \tag{66}$$

Recall that $C_t = W_t^{1/2} J_t$ and $b_t = W_t^{1/2} r_t$.

The candidate proceeds only if

$$\mathrm{Pred}_t > 0, \qquad \mathrm{Ared}_t > 0, \qquad \rho_t = \frac{\mathrm{Ared}_t}{\mathrm{Pred}_t} \ge \rho_{\min}. \tag{67}$$

Because S_t is used both to construct and evaluate the candidate, this test measures agreement between the local model and the nonlinear constraints; it does not provide independent evidence that the alignment improves unseen constraint samples.

Independent held-out validation. ORACLE therefore fixes $\widetilde{d}_t$ before drawing an independent audit support S_t'. Let $\mathcal{G}_t$ contain all information available after the candidate is fixed but before S_t' is sampled. Conditional on $\mathcal{G}_t$, the audit samples are independent draws from their prescribed constraint-family distributions.

For family k, define

$$X_{k,t}(z) = \omega_{k,t}\left[r_k(\bar{\theta}_t + \widetilde{d}_t; z)^2 - r_k(\bar{\theta}_t; z)^2 \right]. \tag{68}$$

With m_k' audit samples from family k, the empirical change is

$$\widehat{\Delta}_t^{\mathrm{audit}} = \sum_{k=1}^{K} \frac{1}{m_k'} \sum_{j=1}^{m_k'} X_{k,t}(z_{k,j}'). \tag{69}$$

The corresponding population change is

$$\begin{aligned} \Delta_t^{\mathrm{pop}} &= \sum_{k=1}^{K} \mathbb{E}\left[X_{k,t}(z) \mid \mathcal{G}_t \right] \\ &= \Phi_t\left(\bar{\theta}_t + \widetilde{d}_t\right) - \Phi_t(\bar{\theta}_t). \end{aligned} \tag{70}$$

Therefore,

$$\boxed{\mathbb{E}\left[\widehat{\Delta}_t^{\mathrm{audit}} \mid \mathcal{G}_t \right] = \Delta_t^{\mathrm{pop}}.} \tag{71}$$

Assume that, conditional on $\mathcal{G}_t$, each $X_{k,t}(z)$ lies in an interval of width $R_{k,t}$. Hoeffding's inequality then gives, for any $u > 0$,

$$\Pr\left[\widehat{\Delta}_t^{\mathrm{audit}} - \Delta_t^{\mathrm{pop}} \le -u \,\middle|\, \mathcal{G}_t\right] \le \exp\left(-\frac{2u^2}{\sum_{k=1}^{K} R_{k,t}^2/m_k'}\right). \tag{72}$$

The deployed zero-margin audit accepts when

$$\widehat{\Delta}_t^{\mathrm{audit}} \le 0.$$

Suppose instead that the true population constraint energy increases by at least u, so that $\Delta_t^{\mathrm{pop}} \ge u$. If the audit nevertheless accepts, then

$$\widehat{\Delta}_t^{\mathrm{audit}} - \Delta_t^{\mathrm{pop}} \le -u.$$

Hence

$$\boxed{\Delta_t^{\mathrm{pop}} \ge u \Longrightarrow \Pr\left[\widehat{\Delta}_t^{\mathrm{audit}} \le 0 \,\middle|\, \mathcal{G}_t\right] \le \exp\left(-\frac{2u^2}{\sum_{k=1}^{K} R_{k,t}^2/m_k'}\right).} \tag{73}$$

Setting the right-hand side to δ gives the finite-sample resolution

$$\boxed{u_\delta = \sqrt{\frac{\log(1/\delta)}{2}\sum_{k=1}^{K}\frac{R_{k,t}^2}{m_k'}}.} \tag{74}$$

Thus, conditional on the fixed candidate, a population increase of at least u_δ passes the zero-margin audit with probability at most δ. For comparable range widths and balanced family counts,

$$u_\delta = O\left(\sqrt{\frac{\log(1/\delta)}{m_{\mathrm{audit}}}}\right).$$

This is a finite-sample statement rather than an exact sign guarantee: population changes below this resolution may not be distinguished reliably by the zero-margin audit.

Scope of the held-out guarantee. The preceding result applies to one fixed alignment candidate relative to its current proposed endpoint. After an accepted alignment, the parameters change, and so do subsequent gradients, Native proposals, sampled supports, and potentially the constraint-family weights. The single-step audit guarantee therefore does not telescope into a terminal guarantee comparing an entire ORACLE trajectory with an independently trained Native trajectory.

Exact one-step reduction to Native. Let Cert_t, Trig_t, Desc_t, NL_t, and Audit_t denote successful geometry certification, trigger, alignment-descent, nonlinear-agreement, and held-out validation events, respectively. Define

$$\mathrm{Acc}_t = \mathrm{Cert}_t \wedge \mathrm{Trig}_t \wedge \mathrm{Desc}_t \wedge \mathrm{NL}_t \wedge \mathrm{Audit}_t. \tag{75}$$

ORACLE commits

$$d_t = \widetilde{d}_t \mathbf{1}\{\mathrm{Acc}_t\}. \tag{76}$$

If any gate fails, $d_t = 0$, and therefore

$$\theta_{t+1} = \theta_t + s_t.$$

Table 1: Parameter settings and numerical configurations used to generate the reference solutions for the eight PDE benchmarks. Resolutions are those of the shipped reference arrays; the evaluation pipeline discards the final temporal row for the windowed systems (Kuramoto–Sivashinsky, Gray–Scott), so the evaluated horizons are 250 and 100 respectively. *Segmentation* gives the training decomposition: single-segment problems are trained over one contiguous horizon, windowed problems over ten sequential time windows, and the lid-driven cavity over a five-stage Reynolds-number curriculum.

PDE	**Parameters**	**Solver / package**	**Reference resolution**	**Segmentation**
Wave	$c = 2$ (residual uses $c^2 = 4$)	Analytic	200×128	$1 \times 100{,}000$
Burgers	$\nu = 0.01/\pi$	Chebfun	201×512	$1 \times 100{,}000$
Allen–Cahn	$\epsilon = 10^{-4},\ a = 5$	Chebfun	201×512	$1 \times 100{,}000$
KdV	$\eta = 1,\ \mu = 0.022$	Chebfun	201×512	$1 \times 100{,}000$
Kuramoto–Sivashinsky	$\alpha = 100/16,\ \beta = 100/16^2,\ \gamma = 100/16^4$	Chebfun	251×512	$10 \times 10{,}000$
Gray–Scott	$\epsilon_1 = 0.2,\ \epsilon_2 = 0.1,\ b_1 = 40,\ b_2 = 100,\ c_1 = c_2 = 1000$	Chebfun	$101 \times 200 \times 200$	$10 \times 10{,}000$
Lid-driven cavity	$\mathrm{Re} \in \{100, 400, 1000, 1600, 3200\}$	Incompressible Navier–Stokes	256×256 per stage	$5 \times 20{,}000$
Kolmogorov flow	$\mathrm{Re} = 10^4$ ($\nu = 10^{-4}$)	Incompressible Navier–Stokes	$50 \times 256 \times 256$	$10 \times 10{,}000$

Because ORACLE never writes its alignment adjustment into the Native optimizer state,

$$\xi_{t+1} = \mathcal{U}(g_t, \xi_t).$$

Consequently,

$$\boxed{\neg \mathrm{Acc}_t \Longrightarrow (\theta_{t+1}, \xi_{t+1}) = (\theta_t + s_t,\, \mathcal{U}(g_t, \xi_t))\,.} \tag{77}$$

Thus, from the current state (θ_t, ξ_t), rejection of any ORACLE gate recovers exactly the update that the Native optimizer had already proposed. This is a one-step identity. It does not imply equality with an independently run Native trajectory after an earlier accepted ORACLE alignment, because an accepted alignment changes the parameters from which subsequent gradients and Native updates are generated.

C Experimental Details

This appendix provides the experimental specification underlying Section 4. The central claim of ORACLE is that constraint alignment should be defined relative to the movement produced by the native optimizer. We test this claim from breadth to mechanism. Q1 varies optimizer geometry; Q2 varies model formulation; Q3 quantifies the cost and solution-level benefit of alignment; Q4 compares objective-, gradient-, and post-optimizer intervention; and Q5 isolates ORACLE's geometry, authority, and validation mechanisms. We report architectures, optimizer settings, training and sampling protocols, ORACLE hyperparameters, controlled baselines, and compute-accounting conventions below.

C.1 Reference Solution Generation

Parameter settings and numerical configurations used to generate the reference solutions for the PDE benchmarks as given in Table 1.

Table 2: Overview of the experimental design.

	Question	What changes	Scale
Q1	Does ORACLE work across optimizer geometries?	SGD, Adam, SOAP, PSGD–Kron	8 PDEs $\times$ 4 optimizers
Q2	Does alignment persist across model formulations?	Pure MLP, Simple PINN, PirateNet	4 PDEs $\times$ 3 models $\times$ 2 optimizers
Q3	What does ORACLE cost, what does it buy	Native VS ORACLE	3 Models
Q4	Where should constraint intervention occur?	objective-, gradient-, and post-optimizer methods	Allen–Cahn and Wave
Q5	What makes optimizer-relative alignment work?	geometry, alignment rule, and authority	Allen–Cahn and Wave

Table 3: Native optimizer families used in Q1.

Optimizer	Native geometry	Update structure
SGD	Euclidean	isotropic
Adam	diagonal adaptive	coordinate-wise
SOAP	structured preconditioned	matrix-structured
PSGD–Kron	Kronecker preconditioned	structured

C.2 Common Evaluation Protocol

ORACLE is evaluated as a wrapper around a Native optimizer rather than as a replacement optimizer. Within each paired Native–ORACLE comparison, we hold the architecture, native objective, optimizer configuration, learning-rate schedule, sampling pipeline, initialization or shared checkpoint, and optimization budget fixed. ORACLE acts only after the Native optimizer has formed its realized parameter update.

Unless otherwise stated, comparisons are matched by optimizer steps rather than wall-clock time. The primary accuracy metric is relative solution error,

$$\mathrm{relL2} = \frac{\|u_\theta - u^\star\|_2}{\|u^\star\|_2}, \tag{78}$$

where $u^\star$ denotes the reference solution. For paired comparisons we also report

$$\Delta_{\mathrm{rel}} = 100 \left(1 - \frac{\mathrm{relL2}_{\mathrm{method}}}{\mathrm{relL2}_{\mathrm{Native}}}\right) \%, \tag{79}$$

so that positive values indicate lower error than the corresponding Native configuration.

C.3 Q1: Variation in Native Optimizer Geometry

Q1 fixes the model to PirateNet and varies the Native optimizer among SGD, Adam, SOAP, and PSGD–Kron. These optimizers span isotropic Euclidean, diagonal-adaptive, and structured preconditioned update geometries.

Each optimizer is evaluated on Burgers, KdV, Wave, Allen–Cahn, lid-driven cavity, Kuramoto–Sivashinsky, Gray–Scott, and Kolmogorov flow, giving a nominal $8 \times 4 = 32$ PDE–optimizer grid. Every configuration uses a 100,000-step training horizon. ORACLE becomes active at the fork listed in Table 7; the trajectory before that point is Native.

C.4 Q2: Variation in Model Formulation

Q2 changes the model formulation while retaining Adam and SOAP as two qualitatively different Native optimizer geometries. We evaluate Pure MLP, Simple PINN, and PirateNet on Wave, Burgers, Allen–Cahn, and KdV, producing

$$4 \text{ PDEs} \times 3 \text{ model formulations} \times 2 \text{ optimizers} = 24$$

Table 4: Architecture configurations used in Q2. All models use hidden width 256.

Architecture	Structure	Activation	Initialization	Parameters	Architecture-specific configuration
MLP	4 hidden layers + linear output	$\tanh$	Glorot normal; zero bias	198,401[a] / 198,657[b]	Same backbone as Simple PINN, but trained only on IC/BC terms; no PDE residual is included in the Native objective. Fourier features and RWF are disabled.
PINN	4 hidden layers + linear output	$\tanh$	Glorot normal; zero bias	198,401[a] / 198,657[b]	Standard physics-informed objective containing PDE-residual, initial-condition, and boundary-condition terms.
PirateNet	3 PI-modified bottleneck blocks (9 dense layers) + 2 gate encoders + linear output	$\tanh$	Glorot normal with RWF; $g = \exp(\mathcal{N}(1, 0.1))$, $v = w/g$; zero bias	727,045[a] / 727,173[b]	Random Fourier features (dimension 256) with scales 10 (Wave), 1 (Burgers), 2 (Allen–Cahn), and 1 (KdV). RWF uses mean 1 and std. 0.1. Adaptive gating uses $\alpha_0 = 0$; the learnable skip is initialized to identity.

[a] Wave and Burgers use 2-D inputs. [b] Allen–Cahn and KdV use a non-trainable periodic embedding that lifts the input to 3-D without introducing additional parameters.

paired configurations, each with a fixed $10{,}000$-step training horizon.

Pure MLP and Simple PINN deliberately share the same 4×256 $\tanh$ backbone. Pure MLP optimizes only the prescribed initial- and boundary-condition terms, whereas Simple PINN additionally includes the PDE residual in the Native objective. PirateNet changes both the representation and its physics-specific inductive structure through Fourier features, random weight factorization, and gated residual blocks.

C.5 Network Architecture Details

Table 4 specifies the three model formulations used in Q2. The same PirateNet configuration is used throughout Q1.

The 8-PDE $\times$ 4-optimizer Q1 study uses PirateNet exclusively. For the executed PirateNet configurations, the experiment path does not apply the optional `use_pi_init` initialization flag.

C.6 Native Optimizer Configuration

Native and Native+ORACLE use identical learning rates, schedules, momentum parameters, numerical stabilizers, and preconditioner-update rules. Table 5 reports the Q1 configurations.

C.6.1 Learning-Rate Schedule

Within each Q1 PDE, all four optimizers use the same learning-rate schedule. The learning rate increases linearly from zero to its configured initial value during warmup and then follows exponential

Table 5: Native optimizer configurations used in Q1.

Optimizer	Initial LR	LR schedule	β_1	β_2	Weight decay	Preconditioner update	Other settings
SGD	10^{-3}	shared	—	—	none	—	`optax.sgd`; no momentum or Nesterov acceleration.
Adam	10^{-3}	shared	0.9	0.999	none	—	`optax.adam`; $\epsilon = 10^{-8}$.
SOAP	10^{-3}	shared	0.9	0.999	0.0	every 2 steps	`soap_jax`; $\epsilon = 10^{-8}$; `shampoo_beta`$= -1$ (reuses β_2); bias correction enabled; `max_precond_dim`$= 10^4$.
PSGD–Kron	$10^{-3}/10^{-4}$[a]	shared	0.9[b]	—	0.0	stochastic	`psgd_jax` 0.2.3; `max_size_triangular`$=$ 8192. Global-norm and elementwise clipping precede geometry certification.

[a] 10^{-4} for Burgers and lid-driven cavity; 10^{-3} for all remaining PDEs. [b] Momentum coefficient; PSGD–Kron does not use an Adam-style second-moment coefficient.

Table 6: Learning-rate schedule parameters in Q1.

PDE	Warmup steps	Decay steps
Burgers	5,000	1,000
KdV	5,000	2,000
Wave	5,000	2,000
Allen–Cahn	5,000	5,000
Lid-driven cavity	5,000	10,000
Kuramoto–Sivashinsky	5,000	2,000
Gray–Scott	5,000	2,000
Kolmogorov flow	2,000	2,000

decay,

$$\eta_t = \mathrm{ExpDecay}\left(\eta_{\mathrm{base}}, \text{rate} = 0.9, \text{decay steps}\right), \tag{80}$$

with no learning-rate floor. The schedule is indexed by the global training step and does not restart at curriculum or time-window boundaries.

C.6.2 Geometry Certification for PSGD–Kron

ORACLE uses an optimizer-relative metric only when the realized Native update admits the certified linear representation derived in Appendix B.2. The deployed PSGD–Kron update applies global-norm and elementwise clipping to the preconditioned direction. When elementwise clipping is active, the resulting update is no longer represented by the linear Kronecker mobility used by ORACLE. The adapter is therefore treated as uncertified and ORACLE preserves the Native proposal rather than falling back to $P_t = I$.

The algebraic reason for this abstention rule is given in Appendix B.3. Empirical certification statistics are reported with the corresponding Q1 results rather than repeated here.

C.7 PDE Training, Sampling, and Evaluation

Table 7 gives the Q1 training and evaluation settings. Windowed PDEs use a time-marching curriculum with a fresh model for each window; the lid-driven cavity uses a Reynolds-number curriculum while retaining state. "Interior" denotes residual collocation points sampled per optimization step.

Table 7: **Q1 training and evaluation settings across PDE benchmarks.**

PDE	Training	Interior	Boundary	Initial condition	Sampling	Evaluation	Fork
Burgers	100k	8,192	402 fixed	512 fixed	$[0,1]\times[-1,1]$	201×512	5k
KdV	100k	4,096	periodic	512 fixed	$[0,1]\times[-1,1]$	201×512	5k
Wave	100k	8,192	400 fixed	$128(u)+128(u_t)$	$[0,1]^2$	200×128	5k
Allen–Cahn	100k	8,192	periodic	512 fixed	$[0,1]\times[-1,1]$	201×512	5k
Lid-driven cavity	5×20k	4,096	1,024 fixed	steady	$[0,1]^2$	256×256 / Re	5k
Kuramoto–Sivashinsky	10×10k	4,096	periodic	512 fixed	window	251×512	5k
Gray–Scott	10×10k	4,096	periodic	40,000/species	window	$101\times200\times200$	5k
Kolmogorov flow	10×10k	8,192	periodic	16,384 sampled[a]	window	$50\times256\times256$	2k

Note. Interior residual points are resampled uniformly at each step. For windowed problems, sampling is restricted to the active space–time window. Boundary and initial-condition data are fixed where applicable, except for Kolmogorov flow. [a]Sampled with replacement from the 65,536-point reference grid.

Table 8: Constraint-handling methods used in Q4.

Method	Intervention point	Modification
Native	none	unmodified optimizer
Tuned penalty	objective	fixed constraint-loss reweighting
ReLoBRaLo	objective	adaptive objective weighting
PCGrad	gradient	projects conflicting gradients
ConFIG	gradient	constructs a conflict-aware gradient
HARMONIC	gradient	reconciles gradients before optimization
ConFIG+GUA	post-optimizer	modifies the realized optimizer update
ORACLE	post-optimizer	optimizer-relative residual alignment

C.8 Q4: Matched Intervention Comparison

Q4 trades breadth for control. For Allen–Cahn and Wave, every method begins from the same exact 5,000-step Native checkpoint, including model parameters and optimizer state. Each arm then receives exactly 5,000 additional method-active optimization steps and is evaluated at the common global endpoint of step 10,000.

The comparison is therefore matched in initialization, optimizer state, sampling protocol, and optimization horizon. Methods differ only in how constraint information enters the optimization pipeline.

Objective-level methods alter the objective before differentiation; gradient-level methods alter the signal supplied to the Native optimizer; and post-optimizer methods act after the optimizer has formed a realized parameter displacement.

C.9 ORACLE Experimental Configuration

The mathematical construction of ORACLE is given in Appendix B. Here we report only the experimental settings used by the reference configuration. The alignment system is solved matrix-free with 16 CG iterations using a construction support of 1024 residuals. A candidate that passes the nonlinear-agreement test is evaluated on an independent 1024-point audit support before commitment.

Archived experiment directories may refer to this reference configuration as `H2-S`; throughout the paper we use the final name ORACLE.

C.10 Q5: Controlled ORACLE Ablations

Q5 isolates the mechanisms used by the reference ORACLE configuration. Each ablation inherits the same Native checkpoint and experimental protocol and changes only the component listed in Table 10.

Table 9: Reference ORACLE hyperparameters.

Quantity	Symbol	Value
Residual-space alignment		
Construction support size	m	1024
Held-out audit size	m_{audit}	1024
CG iterations	K	16
Relative LM damping, initial	$\lambda_{\text{rel}}^{(0)}$	10^{-2}
Relative LM damping bounds	$[\lambda_{\min}, \lambda_{\max}]$	$[10^{-4}, 10^{-1}]$
Damping grow / shrink	–	4.0 / 0.5
Alignment authority		
Initial authority fraction	β_0	0.10
Authority bounds	$[\beta_{\min}, \beta_{\max}]$	$[0.01, 0.25]$
Agreement threshold	$\rho_{\min}$	0.10
Shrink threshold / factor	ρ_{shrink}	0.25 / $\times 0.5$
Expand threshold / factor	ρ_{expand}	0.75 / $\times 1.25$
Numerical floor	$\varepsilon_{\text{auth}}$	10^{-30}

Table 10: Controlled ORACLE ablations used in Q5.

Variant	Modification
First-order	Replace the residual-space alignment with $d \propto -P_t^{-1} c_t$ while retaining the same certified optimizer geometry and authority mechanism.
Unbounded	Retain the optimizer-relative alignment direction but remove the native-relative authority bound, using $\alpha_t = \alpha_{\text{need}}$.
$P_t = I$	Replace the optimizer-relative metric by the identity in both alignment construction and authority measurement.

C.11 Mechanism Diagnostics

The Q5 diagnostics characterize the internal behavior of ORACLE but are not used as substitutes for endpoint solution accuracy.

Authority-bound activation. For each accepted alignment, we record whether the native-relative authority limit determines the applied scale,

$$I_{\text{auth},t} = \mathbf{1}\left[\alpha_{\text{safe},t} \le \alpha_{\text{need},t}\right]. \tag{81}$$

The reported authority-bound rate is the fraction of accepted alignments for which $I_{\text{auth},t} = 1$.

Held-out rejection. The number of fixed alignment candidates rejected by the independent audit is

$$N_{\text{audit}} = \sum_t \mathbf{1}[\text{Audit}_t = \text{reject}]. \tag{82}$$

These quantities measure different mechanisms: authority activation describes how often the candidate is limited relative to the Native movement, whereas audit rejection records how often an already constructed candidate fails to generalize to fresh constraint support.

D Complete Experimental Results

D.1 Q1: Optimizer Geometry

Table 11 reports the complete Q1 endpoint results. PirateNet is fixed throughout, while the Native optimizer varies among SGD, Adam, SOAP, and PSGD–Kron. Within each Native–ORACLE pair, the architecture, objective, optimizer configuration, learning-rate schedule, sampling pipeline, and training budget are unchanged.

Of the 32 nominal PDE–optimizer configurations, 32 produce valid scientific endpoints.

Table 11: **Complete Q1 results across optimizer geometries.** Endpoint relative ℓ_2 error across eight PDE benchmarks for four Native optimizers, with and without ORACLE; lower is better. **Bold** marks the better endpoint for configurations classified as an improvement or degradation, while unbolded pairs denote matched outcomes under the Q1 reporting criterion. Each entry is mean, the standard deviation across three independent runs < 5% for all reported entries.

	Adam		SOAP		SGD		PSGD–Kron	
PDE	**Native**	**+ORACLE**	**Native**	**+ORACLE**	**Native**	**+ORACLE**	**Native**	**+ORACLE**
Burgers	4.039×10^{-5}	4.000×10^{-5}	4.009×10^{-5}	4.001×10^{-5}	1.618×10^{-2}	1.118×10^{-2}	4.166×10^{-5}	4.067×10^{-5}
KdV	4.302×10^{-4}	3.847×10^{-4}	2.973×10^{-4}	2.469×10^{-4}	7.373×10^{-1}	1.512×10^{-1}	3.800×10^{-4}	3.800×10^{-4}
Wave	3.877×10^{-5}	1.888×10^{-5}	1.627×10^{-5}	9.971×10^{-6}	6.379×10^{-1}	6.009×10^{-1}	3.095×10^{-5}	3.005×10^{-5}
Allen–Cahn	7.179×10^{-4}	4.462×10^{-4}	1.547×10^{-4}	4.349×10^{-5}	6.130×10^{-1}	4.307×10^{-2}	5.648×10^{-3}	5.606×10^{-3}
Lid-driven cavity	4.625×10^{-1}	4.848×10^{-1}	2.737×10^{-1}	2.019×10^{-1}	6.391×10^{-1}	6.311×10^{-1}	2.429×10^{-1}	2.429×10^{-1}
Kuramoto–Sivashinsky	1.255	1.229	1.542	1.168	1.988	1.988	1.231	1.031
Gray–Scott	0.9420	0.9110	0.1525	0.2063	0.784	0.784	0.145	0.145
Kolmogorov flow	1.359	1.342	1.372	**1.236**	1.896	1.896	1.146	1.146

PirateNet is used for every Q1 PDE–optimizer pair. Full architecture, optimizer, and PDE configurations are reported in Tables 4, 5, and 7.

Table 12: Q1 outcome summary by Native optimizer.

Optimizer	Improve	Match	Degrade	Valid
Adam	7	0	1	8
SOAP	7	0	1	8
SGD	5	3	0	8
PSGD–Kron	4	4	0	8
Total	**23**	**7**	**2**	**32**

Outcome summary. Under the Q1 reporting criterion, ORACLE improves 23 configurations, matches Native in 7, and degrades 2, corresponding to improvement or matching behavior in $30/32$ valid PDE-optimizer pairs. Table 12 gives the breakdown by Native optimizer.

The complete results show that the effect is not confined to one update geometry. Improvements occur under isotropic SGD, diagonal-adaptive Adam, and the structured preconditioned geometries of SOAP and PSGD–Kron. The two degradations are retained as measured outcomes rather than excluded.

D.2 Q2: Model Formulation

Table 13 reports the complete Q2 results for Pure MLP, Simple PINN, and PirateNet under Adam and SOAP. Pure MLP and Simple PINN share the same 4×256 $\tanh$ backbone but differ in whether the PDE residual participates in the Native objective; PirateNet additionally changes the architecture and physics-specific inductive structure. The study contains

$$4 \text{ PDEs} \times 3 \text{ model formulations} \times 2 \text{ optimizers} = 24$$

paired Native–ORACLE configurations.

ORACLE lowers endpoint rel-L2 across the large majority of configurations, with improvements spanning all three model formulations and both Native optimizers. The largest gains include 60.6% for Simple PINN+Adam on Burgers, 76.1% for PirateNet+Adam on Burgers, and 90.6% for PirateNet+SOAP on KdV.

The strongest degradations occur for Pure MLP on Allen–Cahn under both Adam and SOAP. Because Pure MLP and Simple PINN share the same backbone, whereas Simple PINN additionally includes the PDE residual in its Native objective, this comparison shows that ORACLE does not override the underlying learning formulation.

Trigger and acceptance rates are reported as mechanism diagnostics rather than performance metrics. A high trigger rate does not imply a large endpoint gain, and triggering does not imply automatic

Table 13: **Complete Q2 results across model formulations.** Endpoint relative ℓ_2 error for Pure MLP, Simple PINN, and PirateNet under Adam and SOAP; lower is better. Improvement is $100(1 - \mathrm{relL2}_{\mathrm{ORACLE}}/\mathrm{relL2}_{\mathrm{Native}})$. *Trigger* is the fraction of optimization steps on which the realized Native movement opens the ORACLE alignment path; *Accept* is the fraction of steps on which an aligned candidate is ultimately committed. Each entry is mean, the std-dev across three independent runs < 7% for all reported entries.

PDE	Model	Opt.	Native	+ORACLE	Improvement	Trigger	Accept
Wave	Pure MLP	Adam	1.723	**1.719**	+0.2%	64%	64%
Wave	Pure MLP	SOAP	1.602	**1.459**	+8.9%	74%	74%
Wave	Simple PINN	Adam	4.27×10^{-1}	$\mathbf{2.40\times10^{-1}}$	+43.7%	100%	100%
Wave	Simple PINN	SOAP	2.72×10^{-1}	$\mathbf{2.60\times10^{-1}}$	+4.4%	100%	41%
Wave	PirateNet	Adam	7.53×10^{-3}	$\mathbf{3.12\times10^{-3}}$	+58.6%	83%	80%
Wave	PirateNet	SOAP	3.35×10^{-3}	$\mathbf{3.26\times10^{-3}}$	+2.8%	100%	100%
Burgers	Pure MLP	Adam	7.41×10^{-1}	$\mathbf{4.11\times10^{-1}}$	+44.6%	55%	55%
Burgers	Pure MLP	SOAP	8.36×10^{-1}	$\mathbf{4.65\times10^{-1}}$	+44.3%	74%	73%
Burgers	Simple PINN	Adam	3.82×10^{-2}	$\mathbf{1.51\times10^{-2}}$	+60.6%	98%	94%
Burgers	Simple PINN	SOAP	2.14×10^{-3}	$\mathbf{2.12\times10^{-3}}$	+0.9%	97%	32%
Burgers	PirateNet	Adam	1.09×10^{-2}	$\mathbf{2.62\times10^{-3}}$	+76.1%	93%	91%
Burgers	PirateNet	SOAP	3.46×10^{-3}	$\mathbf{1.08\times10^{-3}}$	+68.9%	98%	78%
Allen–Cahn	Pure MLP	Adam	$\mathbf{6.73\times10^{-1}}$	9.99×10^{-1}	−48.5%	62%	60%
Allen–Cahn	Pure MLP	SOAP	$\mathbf{7.34\times10^{-1}}$	9.92×10^{-1}	−35.2%	55%	49%
Allen–Cahn	Simple PINN	Adam	5.78×10^{-1}	$\mathbf{5.35\times10^{-1}}$	+7.5%	100%	98%
Allen–Cahn	Simple PINN	SOAP	2.16×10^{-2}	$\mathbf{1.31\times10^{-2}}$	+39.5%	100%	89%
Allen–Cahn	PirateNet	Adam	2.29×10^{-2}	$\mathbf{1.08\times10^{-2}}$	+52.6%	100%	100%
Allen–Cahn	PirateNet	SOAP	9.88×10^{-3}	$\mathbf{3.01\times10^{-3}}$	+69.5%	100%	100%
KdV	Pure MLP	Adam	7.83×10^{-1}	$\mathbf{6.88\times10^{-1}}$	+12.2%	52%	51%
KdV	Pure MLP	SOAP	7.75×10^{-1}	$\mathbf{6.59\times10^{-1}}$	+15.0%	56%	50%
KdV	Simple PINN	Adam	7.72×10^{-1}	$\mathbf{7.27\times10^{-1}}$	+5.8%	99%	94%
KdV	Simple PINN	SOAP	1.86×10^{-2}	$\mathbf{1.64\times10^{-2}}$	+12.1%	100%	57%
KdV	PirateNet	Adam	7.60×10^{-1}	$\mathbf{7.36\times10^{-1}}$	+3.6%	94%	83%
KdV	PirateNet	SOAP	2.60×10^{-1}	$\mathbf{2.44\times10^{-2}}$	+90.6%	100%	100%

commitment. For example, Simple PINN+SOAP on Burgers opens the alignment path on 97% of steps but commits an aligned candidate on only 32%.

D.3 Detailed Cost–Benefit Analysis

Figure 5 summarizes the cost and solution-level benefit of ORACLE across the model-formulation study. Here we define each quantity used in that figure and report how the configuration-level measurements are aggregated.

Evaluation setting. Let

$$c = (a, p, o)$$

denote one experimental configuration, where a is the model formulation, p is the PDE, and o is the native optimizer. We evaluate three model formulations,

$$a \in \{\text{Pure MLP}, \text{Simple PINN}, \text{PirateNet}\},$$

four PDEs,

$$p \in \{\text{Wave}, \text{Burgers}, \text{Allen–Cahn}, \text{KdV}\},$$

and two native optimizers,

$$o \in \{\text{Adam}, \text{SOAP}\}.$$

Thus, each architecture contains

$$|\mathcal{C}_a| = 4 \times 2 = 8$$

matched Native–ORACLE comparisons, for 24 configurations overall. Native and ORACLE use the same model formulation, PDE, optimizer, training budget, and evaluation grid within each configuration; the only difference is the ORACLE intervention.

Let $\hat{u}_c^N$ and $\hat{u}_c^O$ denote the solutions produced by Native and ORACLE, respectively, and let $u_c^\star$ denote the reference solution.

D.3.1 Normalized Endpoint Performance

For each configuration, we first compute the relative ℓ_2 solution error at the final checkpoint. For method $m \in \{N, O\}$,

$$E_c^m = \frac{\|\hat{u}_c^m - u_c^\star\|_2}{\|u_c^\star\|_2}. \tag{83}$$

For a discrete evaluation grid $\mathcal{Z}_c = \{z_q\}_{q=1}^{N_c}$, this is

$$E_c^m = \sqrt{\frac{\sum_{q=1}^{N_c} \|\hat{u}_c^m(z_q) - u_c^\star(z_q)\|_2^2}{\sum_{q=1}^{N_c} \|u_c^\star(z_q)\|_2^2}}. \tag{84}$$

Because the absolute error scale differs substantially across PDEs and model formulations, we normalize ORACLE by its matched Native baseline:

$$\rho_c = \frac{E_c^O}{E_c^N}. \tag{85}$$

Hence,

$$\rho_c < 1$$

indicates that ORACLE improves over Native, $\rho_c = 1$ denotes equal endpoint error, and $\rho_c > 1$ indicates degradation.

For each architecture a, panel (a) of Figure 5 reports the geometric mean of the eight matched error ratios:

$$\bar{\rho}_a = \left(\prod_{c \in \mathcal{C}_a} \rho_c \right)^{1/|\mathcal{C}_a|} = \exp\left[\frac{1}{|\mathcal{C}_a|} \sum_{c \in \mathcal{C}_a} \log \rho_c \right]. \tag{86}$$

We use a geometric rather than arithmetic mean because the quantity being aggregated is a multiplicative ratio. This treats proportional improvements consistently across problems with substantially different absolute error scales.

Native is therefore represented by the normalized reference

$$\bar{\rho}_a^N = 1,$$

while the ORACLE bar shows $\bar{\rho}_a$.

The corresponding architecture-level values are

$$\bar{\rho}_{\mathrm{MLP}} = 0.897, \qquad \bar{\rho}_{\mathrm{PINN}} = 0.748, \qquad \bar{\rho}_{\mathrm{PirateNet}} = 0.379,$$

reported in the main figure as $0.90\times$, $0.75\times$, and $0.38\times$ Native, respectively.

Equivalently, the relative endpoint-error reduction can be written as

$$\Delta_a^{\mathrm{err}} = 100\,(1 - \bar{\rho}_a)\,\%. \tag{87}$$

This corresponds to approximately 10.3%, 25.2%, and 62.1% lower normalized endpoint error for Pure MLP, Simple PINN, and PirateNet, respectively.

D.3.2 Per-Step Runtime Overhead

We measure runtime relative to the matched Native configuration so that differences in architecture, PDE, and optimizer cost do not confound the comparison. Let

$$\tau_{c,k}^m$$

denote the wall-clock duration of optimization step k for method $m \in \{N, O\}$. Over the K_c timed steps, the mean per-step runtime is

$$\bar{\tau}_c^m = \frac{1}{K_c} \sum_{k=1}^{K_c} \tau_{c,k}^m. \tag{88}$$

The matched runtime ratio is then

$$r_c = \frac{\bar{\tau}_c^O}{\bar{\tau}_c^N}. \tag{89}$$

Thus, $r_c = 1$ corresponds to Native runtime, while $r_c = 4$ means that an ORACLE step requires four times the wall-clock time of its matched Native step.

Panel (b) reports the unweighted architecture-level mean

$$\bar{r}_a = \frac{1}{|\mathcal{C}_a|} \sum_{c \in \mathcal{C}_a} r_c. \tag{90}$$

Every PDE–optimizer configuration therefore contributes equally rather than allowing a more expensive PDE or model to dominate the aggregate.

The resulting runtime ratios are

$$\bar{r}_{\mathrm{MLP}} = 3.99, \qquad \bar{r}_{\mathrm{PINN}} = 5.74, \qquad \bar{r}_{\mathrm{PirateNet}} = 4.41,$$

reported as approximately $4.0\times$, $5.8\times$, and $4.4\times$ Native runtime per step.

If expressed as percentage overhead rather than a multiplicative ratio, the same quantity is

$$\Delta_a^{\mathrm{time}} = 100\,(\bar{r}_a - 1)\,\%. \tag{91}$$

We use the multiplicative ratio in the main paper because it is easier to compare directly against the normalized performance ratio.

D.3.3 Pointwise Domain Win Rate

Endpoint relative ℓ_2 error compresses the full solution into one scalar. To determine whether ORACLE's advantage is distributed across the solution domain, we additionally compare Native and ORACLE point by point.

For evaluation point $z_q \in \mathcal{Z}_c$, define the local prediction error for method m as

$$d_{c,q}^m = \|\hat{u}_c^m(z_q) - u_c^\star(z_q)\|_2\,. \tag{92}$$

For scalar PDEs this reduces to the absolute error

$$d_{c,q}^m = |\hat{u}_c^m(z_q) - u_c^\star(z_q)|\,.$$

ORACLE wins at point z_q when

$$d_{c,q}^O < d_{c,q}^N. \tag{93}$$

Using the indicator function $1[\cdot]$, the configuration-level pointwise win rate is

$$S_c = \frac{100}{N_c} \sum_{q=1}^{N_c} 1\left[d_{c,q}^O < d_{c,q}^N\right]. \tag{94}$$

A value of $S_c = 50\%$ indicates equal pointwise coverage, while $S_c > 50\%$ means that ORACLE is more accurate than Native on the majority of evaluated points.

We first compute S_c separately for every matched configuration and then average the eight configuration-level rates within each architecture:

$$\bar{S}_a = \frac{1}{|\mathcal{C}_a|} \sum_{c \in \mathcal{C}_a} S_c. \tag{95}$$

This configuration-first aggregation prevents a PDE with a denser evaluation grid from receiving greater weight solely because it contains more points.

$$\bar{S}_{\mathrm{MLP}} = 67.4\%, \qquad \bar{S}_{\mathrm{PINN}} = 61.6\%, \qquad \bar{S}_{\mathrm{PirateNet}} = 70.8\%.$$

Thus, ORACLE is more accurate than Native on a majority of evaluated domain points for all three model formulations.

For visualization, the Native share is shown as the complementary quantity

$$S_a^N = 100 - \bar{S}_a, \tag{96}$$

under the negligible-tie convention used in the aggregate plot.

D.3.4 Physical Time-Slice Win Rate

The pointwise metric above measures where ORACLE improves the solution. We separately ask whether this advantage persists across the physical time dimension of the PDE.

Let

$$\{t_j\}_{j=1}^{T_c}$$

denote the physical time slices in the evaluation grid. At each time t_j, we compute the spatial relative ℓ_2 error

$$e_{c,j}^m = \frac{\|\hat{u}_c^m(\cdot, t_j) - u_c^\star(\cdot, t_j)\|_2}{\|u_c^\star(\cdot, t_j)\|_2}. \tag{97}$$

For a discrete spatial grid $\mathcal{X}_{c,j} = \{x_i\}_{i=1}^{N_{x,j}}$,

$$e_{c,j}^m = \sqrt{\frac{\sum_{i=1}^{N_{x,j}} \|\hat{u}_c^m(x_i, t_j) - u_c^\star(x_i, t_j)\|_2^2}{\sum_{i=1}^{N_{x,j}} \|u_c^\star(x_i, t_j)\|_2^2}}. \tag{98}$$

ORACLE wins time slice t_j when

$$e_{c,j}^O < e_{c,j}^N. \tag{99}$$

The configuration-level time-slice win rate is therefore

$$T_c = \frac{100}{T_c^{\mathrm{eval}}} \sum_{j=1}^{T_c^{\mathrm{eval}}} 1\left[e_{c,j}^O < e_{c,j}^N\right], \tag{100}$$

where T_c^{eval} is the number of evaluated physical time slices.

Importantly, this metric refers to *physical PDE time*, not optimization steps or training checkpoints. It therefore measures whether ORACLE's solution-quality advantage persists throughout the predicted dynamics.

As with the pointwise metric, we first compute the win rate independently for each matched configuration and then average across the eight configurations within an architecture:

$$\bar{T}_a = \frac{1}{|\mathcal{C}_a|} \sum_{c \in \mathcal{C}_a} T_c. \tag{101}$$

Panel (c) reports

$$\bar{T}_{\mathrm{MLP}} = 81.1\%, \qquad \bar{T}_{\mathrm{PINN}} = 88.8\%, \qquad \bar{T}_{\mathrm{PirateNet}} = 82.9\%.$$

Hence, ORACLE produces lower spatial solution error than Native on more than 80% of evaluated physical time slices for every model formulation.

Analogously to the spatial plot, the Native share is displayed as

$$T_a^N = 100 - \bar{T}_a \tag{102}$$

under the negligible-tie convention.

Table 14: **Architecture-level quantities used in the cost–benefit analysis.** Endpoint performance is the geometric mean of matched ORACLE/Native relative-ℓ_2 ratios across four PDEs and two optimizers. Runtime, spatial win rate, and time-slice win rate are arithmetic means of the corresponding eight configuration-level quantities. Lower is better for endpoint ratio and runtime; higher is better for the two win rates.

Model	Endpoint / Native ↓	Runtime / Native ↓	Spatial win ↑	Time-slice win ↑
MLP	0.897	3.986×	67.4%	81.1%
PINN	0.748	5.742×	61.6%	88.8%
PirateNet	0.379	4.406×	70.8%	82.9%

D.4 Matched Wall-Clock Evaluation

Panel (d) reports The primary experiments compare Native and ORACLE after the same number of optimization steps, isolating the effect of modifying the realized update. Because ORACLE performs additional constraint evaluation, linear solves, and validation, we additionally ask whether these gains amortize when both methods receive the same elapsed-time budget.

For each optimizer–PDE configuration c, Native and ORACLE are therefore compared at a common wall-clock budget T_c. Let $E_c^{\mathrm{N}}(T_c)$ and $E_c^{\mathrm{O}}(T_c)$ denote their relative ℓ_2 solution errors at that budget. We report the normalized equal-time ratio

$$\rho_c^{\mathrm{time}} = \frac{E_c^{\mathrm{O}}(T_c)}{E_c^{\mathrm{N}}(T_c)}, \tag{103}$$

where $\rho_c^{\mathrm{time}} < 1$ means that ORACLE achieves lower error within the same elapsed time. Unlike the step-matched comparison, ORACLE generally completes fewer optimization steps because each step includes the additional alignment procedure.

To summarize across PDEs, we compute the geometric mean separately for each native optimizer,

$$\bar{\rho}_o^{\mathrm{time}} = \exp\left(\frac{1}{|\mathcal{C}_o|}\sum_{c\in\mathcal{C}_o}\log\rho_c^{\mathrm{time}}\right), \tag{104}$$

where $\mathcal{C}_o$ contains the available PDE configurations for optimizer o. The geometric mean is appropriate because the comparison is expressed as a multiplicative error ratio and the underlying errors span several orders of magnitude.

D.4.1 Summary of Aggregated Quantities

Table 14 reports the architecture-level values used directly in Figure 5.

Taken together, these metrics separate four distinct questions. The endpoint ratio measures *how much* final solution quality changes; the runtime ratio measures *what that improvement costs*; the pointwise win rate measures *how broadly* the improvement is distributed over the solution domain; and the time-slice win rate measures *how persistently* it holds across physical time. The main-text conclusion therefore does not rely on the endpoint metric alone: ORACLE incurs additional computation, but the resulting accuracy gains extend across both the domain and the temporal evolution of the solution.

Table 15: **Endpoint behavior under controlled ORACLE ablations.** Relative ℓ_2 error and error normalized by the matched Native endpoint on Allen–Cahn and Wave. Lower is better.

	Allen–Cahn		Wave	
Method	rel-L2	/ Native	rel-L2	/ Native
Native	0.026653	1.000	0.005114	1.000
$P_t = I$	0.495820	18.60	0.027621	5.40
First-order	0.015248	0.572	0.003614	0.707
Unbounded	0.004590	0.172	0.004292	0.839
ORACLE	0.010333	0.388	0.002835	0.554

E Extended Mechanism Analysis

The main paper isolates three mechanisms within ORACLE: the geometry used to construct alignment, the authority permitted relative to the Native step, and the held-out validation used before commitment. This appendix reports the corresponding exact values and provides additional analysis of the Native optimizer geometry. We first examine endpoint behavior under controlled ablations, then quantify the anisotropy of the optimizer-induced metric, and finally relate that geometry to authority activation and held-out rejection.

E.1 Optimizer Geometry and Endpoint Behavior

Because Allen–Cahn and Wave have different absolute error scales, we compare mechanism ablations using endpoint error normalized by the matched Native result,

$$R_{\mathrm{err}} = \frac{\mathrm{relL2(method)}}{\mathrm{relL2(Native)}}, \tag{105}$$

where $R_{\mathrm{err}} < 1$ denotes lower endpoint error than Native.

Table 15 reports the exact values underlying Fig. 7(a,d). Replacing the optimizer-relative metric with $P_t = I$ increases endpoint error to $18.60\times$ Native on Allen–Cahn and $5.40\times$ on Wave. By contrast, the metric-aware First-order variant reaches $0.572\times$ and $0.707\times$ Native, while full ORACLE reaches $0.388\times$ and $0.554\times$, respectively.

The Euclidean ablation therefore changes the outcome substantially rather than acting as a benign simplification of the ORACLE metric. At the same time, geometry alone does not determine the final endpoint: the Unbounded variant achieves the lowest Allen–Cahn error but provides a smaller gain on Wave. This motivates separating the geometry used to construct alignment from the authority assigned to that alignment.

E.2 Anisotropy of the Native Optimizer Geometry

The strong degradation under $P_t = I$ raises a natural question: how different is the Native optimizer geometry from an isotropic Euclidean metric in practice? For Adam, the optimizer-relative metric is determined by its coordinate-wise adaptive denominator. Writing

$$D_t = \sqrt{\widehat{v}_t} + \epsilon,$$

we quantify its anisotropy using both the full coordinate condition number,

$$\kappa(D_t) = \frac{\max_i (D_t)_i}{\min_i (D_t)_i}, \tag{106}$$

and the robust percentile ratio

$$\rho_{95/05}(D_t) = \frac{p_{95}(D_t)}{p_{05}(D_t)}. \tag{107}$$

Table 16 shows substantial anisotropy on both PDEs. The full coordinate spread exceeds 10^6 at every reported checkpoint and reaches 1.94×10^7 on Allen–Cahn at step 10k. Even after reducing sensitivity to extreme coordinates, the 95th-to-5th percentile ratio ranges from 16.7 to 33.8.

Table 16: **Anisotropy of the Adam-induced geometry.** The full condition number $\kappa(D_t)$ measures the total coordinate-wise spread, while $p_{95}(D_t)/p_{05}(D_t)$ provides a robust measure less sensitive to extreme coordinates.

PDE	Step	$\kappa(D_t)$	$p_{95}(D_t)/p_{05}(D_t)$
Allen–Cahn	5k	8.21×10^6	21.9
Allen–Cahn	10k	1.94×10^7	33.8
Wave	5k	1.13×10^6	16.7
Wave	10k	1.39×10^6	21.3

Table 17: **Activation of the native-relative authority boundary.** Percentage of accepted alignments for which Eq. equation 108 is active in the matched mechanism study. Native constructs no alignment candidate, while Unbounded removes the authority boundary by definition.

Method	Allen–Cahn	Wave
Native	–	–
$P_t = I$	37.6%	2.5%
First-order	99.7%	94.8%
Unbounded	0.0%	0.0%
ORACLE	100.0%	99.2%

Thus, replacing the Native Adam geometry with $P_t = I$ discards a large amount of coordinate-dependent scaling. This does not by itself prove that anisotropy causes the endpoint differences in Table 15, but it shows that the Euclidean ablation constitutes a substantial change in the geometry rather than a small approximation to the Native metric.

The same geometry enters ORACLE twice: it determines the alignment direction through P_t^{-1} and determines the scale of admissible alignment through the P_t-norm. We therefore next examine how often the native-relative authority bound becomes active.

E.3 Geometry and Alignment Authority

ORACLE bounds the aligned adjustment relative to the Native movement in the same optimizer-induced metric,

$$\|\widetilde{d}_t\|_{P_t} \le \beta_t \|s_t\|_{P_t}, \qquad \|v\|_{P_t}^2 = v^\top P_t v. \tag{108}$$

Table 17 reports how often this authority boundary is active among accepted alignments. For full ORACLE, the boundary binds on 100.0% of accepted Allen–Cahn alignments and 99.2% on Wave. The metric-aware First-order variant behaves similarly, with rates of 99.7% and 94.8%. Under the $P_t = I$ ablation, the corresponding rates fall to 37.6% and 2.5%.

The authority mechanism therefore depends materially on the geometry in which the Native and aligned movements are measured. Changing P_t affects not only the direction produced by the alignment objective but also whether a proposed adjustment reaches the native-relative authority boundary.

The Unbounded variant provides the complementary case: its authority boundary is absent by construction. Its strong Allen–Cahn endpoint shows that greater alignment authority can help in some regimes, whereas its smaller improvement on Wave shows that unrestricted authority is not uniformly preferable across problems. We therefore treat the authority bound as a mechanism for regulating alignment magnitude rather than claim that it independently improves every endpoint.

E.4 Held-Out Validation and Commitment

The authority bound controls the magnitude of an alignment candidate but does not establish that its locally predicted improvement persists on unseen constraint samples. ORACLE therefore separates candidate construction from commitment by evaluating the fixed candidate on an independent support S_t'.

Table 18: **Held-out validation rejections in the matched mechanism study.** Number of fixed alignment candidates rejected on independent support over 5k active optimization steps. Native constructs no alignment candidate and therefore has no audit decision.

Method	Allen–Cahn	Wave
Native	–	–
$P_t = I$	875	11
First-order	19	6
Unbounded	19	1
ORACLE	4	0

We count candidates rejected at this stage as

$$N_{\text{audit}} = \sum_{t=1}^{T} \mathbf{1}\left[\text{Audit}_t = \text{reject}\right]. \tag{109}$$

Table 18 reports the resulting counts over the matched 5k-step mechanism study. On Allen–Cahn, the $P_t = I$ variant produces 875 held-out rejections, compared with 19 for First-order, 19 for Unbounded, and only 4 for ORACLE. On Wave, the corresponding counts are 11, 6, 1, and 0.

These counts provide information not captured by endpoint rel-L2. In particular, the Euclidean ablation not only ends with substantially larger solution error but also constructs many more candidates whose local improvement does not persist on independent support. This diagnostic does not establish that audit rejection itself causes the endpoint difference; rather, it shows that the geometry used to construct alignment is associated with substantially different off-support behavior.

E.5 Mechanism Summary

The extended diagnostics support a sequential interpretation of ORACLE. The Native optimizer first determines the metric in which alignment is constructed. The same metric defines the scale of the aligned adjustment relative to the Native proposal. Finally, held-out validation determines whether the fixed candidate is committed:

$$\boxed{P_t \longrightarrow \text{alignment} \longrightarrow \text{authority} \longrightarrow \text{held-out validation} \longrightarrow \text{commitment}.}$$

This sequence clarifies the roles of the individual mechanisms. Replacing P_t by the identity changes both alignment behavior and authority activation; removing the authority bound changes the allowable intervention scale; and independent validation can reject candidates that appear locally acceptable on the construction support.

We therefore do not interpret every component as independently necessary for every problem. Instead, the complete ORACLE design assigns distinct roles to the three mechanisms: optimizer geometry determines *how* alignment is measured, the authority bound limits *how far* the Native movement may be redirected, and held-out validation determines *whether* the aligned candidate is ultimately committed.

E.6 Optimizer-Level Gate Statistics

Table 19 reports how frequently each stage of ORACLE is reached across the eight Q1 PDE configurations for each optimizer. *Certified* denotes steps for which the recovered optimizer geometry passes certification; *Trigger* denotes steps satisfying $c_t^\top s_t > 0$; *Proposal* denotes steps for which a valid alignment candidate is constructed; and *Accept* denotes steps for which the candidate passes all validation gates and is committed.

Geometry certification succeeds throughout these evaluated configurations, showing that ORACLE can recover the realized optimizer geometry reliably for all four optimizer families. The subsequent behavior is optimizer dependent: the alignment trigger fires on 59.56% of SGD steps but on 88.46–94.73% of Adam, SOAP, and PSGD–Kron steps. Once triggered, a valid alignment proposal is constructed almost always, while the lower final acceptance rates reflect the nonlinear-agreement

Table 19: **ORACLE gate statistics across optimizer geometries.** Rates are aggregated across the eight PDE configurations associated with each optimizer. Certification checks whether the recovered geometry faithfully represents the realized native update; trigger marks locally adverse native steps; proposal records successfully constructed alignment candidates; and accept records corrections ultimately committed after validation and held-out audit.

Optimizer	Cells	Certified	Trigger	Proposal	Accept
SGD	8	100.00%	59.56%	59.55%	44.78%
Adam	8	100.00%	88.46%	88.46%	84.40%
SOAP	8	100.00%	94.73%	94.73%	90.46%
PSGD–Kron	8	100.00%	90.24%	90.24%	87.24%

and held-out validation gates. Thus, ORACLE's selectivity arises primarily from detecting whether alignment is needed and from validating the resulting candidate, rather than from failures to construct the optimizer-relative correction.