CONSTRAINED NEURAL NETWORK TRAINING VIA LANDING DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

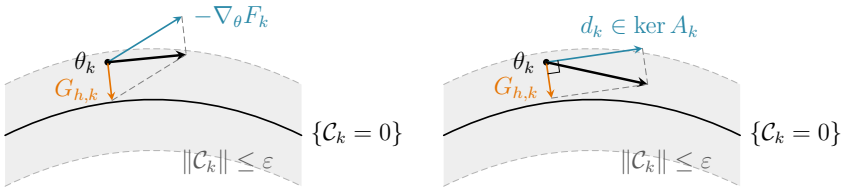
Equality-constrained optimization of neural networks is a common situation, where the network output is required to satisfy conserved quantities, boundary conditions of an equation, or other identities. Existing approaches, however, either lack sound theoretical properties or do not generalize. We propose Landing with Multiplier Tracking (LandTrack), a method that opens another route, in which the constraint acts on the *training dynamics* instead. Specifically, each update is split into two orthogonal components, a tangential one that decreases the objective along the constraint set and a normal one that pulls the iterate back toward feasibility. Training with this update drives the network to constrained optima, in the sense that the fixed points of the update are the Karush-Kuhn-Tucker (KKT) points of the sampled constrained problem. Building on the theoretical foundation of the classical landing method, we obtain the two components by solving a linear system in a Lagrange multiplier, whose every inner iteration costs a Jacobian–vector and a vector–Jacobian product (JVP and VJP). We further reduce this overhead with a Richardson iteration that *tracks* the slowly drifting multiplier across training steps at one correction per step, whose fixed point remains the exact projection. Furthermore, we stabilize the recursion with a warm-started spectral estimate, a trust-region cap, and a damped initialization. In experiments spanning the prediction of physical system dynamics and partial differential equations (PDEs), with constraints ranging from conserved quantities to boundary conditions, LandTrack improves the entire accuracy-conservation trade-off curve over other baselines.

1 INTRODUCTION

Optimizing a neural network under equality constraints is a common situation. Beyond fitting the training data, the network output is often required to satisfy identities that the task prescribes exactly. The requirement arises across Scientific Machine Learning and beyond. For instance, surrogates for dynamical systems are expected to conserve mass, energy, or momentum, as in Hamiltonian mechanics, interacting particle systems, and chaotic flows (Greydanus et al., 2019; Cranmer et al., 2020; Hu et al., 2026). In addition, neural solvers for PDEs are expected to satisfy the boundary conditions of the underlying equation (Raissi et al., 2019; Li et al., 2021; Wang et al., 2024; Hu et al., 2025), and climate and fluid emulators are expected to obey physical closure relations (Beucler et al., 2021). Despite the diversity of these applications, the mathematical form is the same: an equality constraint on the network output, coupled to the predictive objective through shared parameters. A model that violates it produces predictions that are not physically meaningful, and the violation compounds in use, as when a rolled-out surrogate lets a conserved quantity drift step by step. Training such a model is therefore a constrained optimization problem, minimizing the predictive objective subject to equality constraints evaluated on the training distribution.

The prevailing treatment folds the constraint into the loss as a weighted quadratic penalty (Raissi et al., 2019; Wang et al., 2021; 2022; McClenny & Braga-Neto, 2023). Yet there is a structural weakness that the stationary point of the penalized loss is a KKT point of the constrained problem only in the limit of a diverging weight (Nocedal & Wright, 2006). Therefore, at any finite weight, the constraint is traded against accuracy rather than enforced. A second family, the augmented Lagrangian (Bertsekas, 2014; Platt & Barr, 1987; Basir & Senocak, 2022), removes the diverging weight by maintaining a multiplier estimate, but pins its penalty parameter between two opposing requirements. The classical analysis requires the penalty parameter to exceed an unknown, problem-

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.



(a) (b)

Figure 1: Illustration of the geometry around the feasible region $\{\theta : \mathcal{C}_k(\theta) = 0\}$. (a) The penalty method blends the objective gradient with the feasibility direction. (b) The landing update decomposes into two orthogonal directions, independently responsible for optimality and feasibility, respectively.

dependent threshold for the multiplier iteration to contract, while the conditioning of the subproblem degrades linearly in the same parameter. As a result, first-order training becomes unstable as the parameter grows, unless the step size is reduced, which in turn slows it down (Nocedal & Wright, 2006). A third family builds the constraint into the model itself, as in Hamiltonian and Lagrangian neural networks and related architectural encodings (Greydanus et al., 2019; Cranmer et al., 2020; Beucler et al., 2021). The encoded structure is guaranteed exactly, but each architecture covers only one particular constraint class, ties the constraint to that architecture choice, and offers no recipe for a generic equality constraint outside its class.

In this paper, we propose Landing with Multiplier Tracking (LandTrack), a method that takes a fundamentally different route from these families above. The constraint acts on the *training dynamics*, reshaping the update direction itself so that the network is driven to satisfy the constraint while minimizing the objective. The idea is to split every update into two orthogonal components with separate responsibilities, a tangential component that decreases the objective along the constraint set and a normal component that pulls the iterate back toward it (Figure 1). At a fixed point both components vanish separately, which is exactly the KKT system, attained at a fixed feasibility step with no weight tending to infinity. To realize this idea, we build on the theoretical foundation of the classical landing method (Ablin & Peyré, 2022; Ablin et al., 2024). The decomposition is computed by solving a linear system in a Lagrange multiplier, whose solution yields the tangential projection of the optimizer update. Solving this system anew at every training step, however, is a substantial overhead, since each inner iteration of the solver costs a JVP and a VJP through the network. To remove this cost, we exploit the fact that the solution drifts slowly across steps and *track* it with a Richardson iteration, correcting a maintained multiplier estimate once per step, while its fixed point remains the exact projection. Moreover, we propose stabilization techniques that carry the idealized recursion into real training, namely a warm-started spectral estimate that keeps the correction step contractive, a trust-region cap on the tracked multiplier, and a damped initialization.

Our contributions are as follows. **First**, we propose LandTrack, a method for general equality-constrained neural network training that opens another route, enforcing the constraint through the training dynamics. The update has a transparent geometric meaning, decomposing into a tangential component that decreases the objective along the constraint set and an orthogonal normal component that pulls the iterate back toward feasibility. **Second**, to reduce the computational cost, we amortize the tangential projection by letting a scale-free multiplier estimate chase the slowly drifting solution of the projection system through one Richardson correction per step, whose residual is a by-product of the update itself. The recursion provably contracts with a closed-form step size, and its fixed point remains the exact projection. **Third**, we stabilize the tracking recursion in real training with three safeguards, a warm-started spectral estimate that keeps the correction step contractive, a trust-region cap on the tracked multiplier derived from a projection identity, and a damped initialization that places the recursion inside its contraction region. **Finally**, in five experimental settings spanning the prediction of physical system dynamics and PDEs, with constraints ranging from conserved quantities to boundary conditions, comparisons of Pareto curves show that the update improves the entire accuracy-conservation trade-off curve rather than a single tuned point.

2 Related Work

Manifold optimization and the landing family. Among methods that exploit the geometry of the constraint set, Riemannian optimization is the classical framework, which maintains feasibility at

every iterate by mapping each update back onto the manifold through a retraction (Absil et al., 2008). The landing method (Ablin & Peyré, 2022; Ablin et al., 2024) removes the retraction for orthogonal and Stiefel manifolds via an orthogonal decomposition of the update, and related retraction-free schemes replace the manifold structure with penalized formulations of orthogonality (Gao et al., 2019; Xiao et al., 2022). All of these assume a fixed matrix manifold whose feasibility direction and tangent projection are available in closed form. We extend the landing decomposition to general equality constraints in network training, which is not straightforward due to the computational cost and training stability.

Constraint enforcement in neural network training. The default practice adds weighted constraint penalties to the loss (Stewart & Ermon, 2017; Karpatne et al., 2017; Xu et al., 2018; Raissi et al., 2019; Wang et al., 2021). A stationary point is in general a KKT point only in the limit of a diverging weight (Nocedal & Wright, 2006). Multiplier methods (Arrow et al., 1958; Platt & Barr, 1987; Bertsekas, 2014; Cotter et al., 2019; Stooke et al., 2020; Gallego-Posada et al., 2022; Basir & Senocak, 2022) remove the diverging weight by maintaining a multiplier estimate, but the augmented Lagrangian needs a penalty parameter above an unknown threshold, and pure primal–dual schemes tend to oscillate. Our update instead enforces feasibility by a fixed-step geometric pull-back, with neither a diverging weight nor such a threshold.

Hard-constraint methods. Another line builds the constraint into the model. Architectures encoding Hamiltonian or Lagrangian structure (Greydanus et al., 2019; Cranmer et al., 2020) or analytically eliminable output constraints (Beucler et al., 2021) are exact but cover one constraint class each. Projection and solver layers embed an optimization problem in the forward pass, exactly via differentiable optimization layers (Amos & Kolter, 2017; Agrawal et al., 2019), approximately via learned completion and correction (Donti et al., 2021; Négiar et al., 2023), or by projecting a generative trajectory onto a conservation manifold (Hu et al., 2026). Our method constrains the training dynamics instead, leaving the architecture and the inference unchanged, and the two routes are complementary.

3 METHOD

In this section, we develop the LandTrack method in four steps. Section 3.1 formulates the sampled constrained problem and recalls the landing decomposition together with its optimality certificate. Section 3.2 instantiates the decomposition in parameter space and identifies the exact tangential projection as its computational bottleneck. Section 3.3 removes this bottleneck by tracking the multiplier of the projection system across training steps. Finally, Section 3.4 equips the tracker with the safeguards that make it reliable in real training.

3.1 CONSTRAINED TRAINING AND THE LANDING UPDATE

We train a network $\mathcal{N}_\theta$ with parameters $\theta \in \mathbb{R}^P$ to minimize a predictive objective while satisfying the constraint $h_z\left(\mathcal{N}_\theta(z)\right) = 0$ prescribed by the task, such as a conservation law or a boundary condition of a PDE. At step k, a mini-batch $\mathcal{B}_k = \{z_i\}_{i=1}^b$ induces the objective $F_k(\theta) := b^{-1}\sum_{i=1}^b f_{z_i}\left(\mathcal{N}_\theta(z_i)\right)$ and the stacked residual $\mathcal{C}_k(\theta) := \operatorname{stack}\left(h_{z_1}(\mathcal{N}_\theta(z_1)), \ldots, h_{z_b}(\mathcal{N}_\theta(z_b))\right) \in \mathbb{R}^b$, giving the sampled constrained problem

$$\min_\theta \; F_k(\theta) \qquad \text{s. t.} \qquad \mathcal{C}_k(\theta) = 0. \tag{1}$$

We write $A_k := \mathrm{D}_\theta \mathcal{C}_k(\theta_k)$ for the constraint Jacobian at the current iterate. Given any first-order optimizer without modification, including steepest descent (Robbins & Monro, 1951), momentum methods (Polyak, 1964), and preconditioned ones such as AdaGrad (Duchi et al., 2011) and Adam (Kingma & Ba, 2014), our method can compose with it. All it needs from the optimizer is the update proposal δ_k, the step that the base optimizer with learning rate η_k would apply to the parameters at step k. Everything below treats δ_k as given and never looks inside the optimizer.

We enforce (1) through the *training dynamics*, reshaping the update direction itself. The starting point is to split every update into two orthogonal components, each carrying a single responsibility,

$$\theta_{k+1} = \theta_k + d_k + \alpha\, G_{h,k}, \qquad G_{h,k} := -A_k^*\left[\mathcal{C}_k(\theta_k)\right], \qquad d_k \in \ker A_k. \tag{2}$$

The tangential component d_k lies along the constraint set and answers for the objective alone. Ideally it is the orthogonal projection $P_k\delta_k$ of the optimizer proposal onto $\ker A_k$, which decreases the

objective without disturbing feasibility to first order. The normal component $G_{h,k}$, the negative gradient of the feasibility potential $\frac{1}{2}\|\mathcal{C}_k\|^2$, points back toward the feasible set and answers for feasibility alone, contracting the constraint residual at a fixed step $\alpha > 0$. Repeating the update therefore *lands* the parameters on the constraint set, where the objective is stationary along it.

The landing method (Ablin & Peyré, 2022; Ablin et al., 2024) is a classical method built on exactly this decomposition. It was designed for orthogonal and Stiefel manifolds, where both ingredients of (2) are available in closed form. The constraint set is fixed, the feasibility direction has an explicit expression, and projecting onto the tangent space costs a few small matrix products, so the update can be run as written, with stochastic extensions and convergence guarantees. The decomposition also gives the method its optimality guarantee. Because $d_k \in \ker A_k$ and $G_{h,k} \in \mathrm{range}(A_k^*)$ are orthogonal, a fixed point of (2) forces both to vanish, so the objective is stationary along the constraint set and, when A_k has full row rank, $\mathcal{C}_k = 0$. These are the KKT conditions of (1), reached at a fixed feasibility step without any weight tending to infinity, which Figure 1 contrasts with penalty-type training. With a general optimizer proposal, a fixed point gives $P_k\delta_k = 0$, which is exactly this tangential stationarity when δ_k is proportional to $\nabla_\theta F_k$, as for steepest descent and momentum methods. For adaptive optimizers such as Adam, δ_k is rescaled by a diagonal preconditioner that does not commute with P_k, so reaching the same KKT conclusion requires a separate convergence argument, for which the analysis of Yang et al. (2026) and the convergence theory of Adam (Zhang et al., 2022) provide the relevant tools.

The closed-form structure that makes landing practical, however, does not carry over to network training. The feasibility direction remains cheap, but no closed-form projection exists, and computing $P_k\delta_k$ becomes an iterative solve whose every iteration costs a full forward–backward pass through the network. Realizing this decomposition in network training is therefore not a routine transfer of the landing idea, but a design problem in its own right.

3.2 Landing in Parameter Space: the Exact Projection and Its Cost

Both components of (2) are built from the constraint Jacobian A_k, the normal direction as A_k^* applied to the residual and the tangential direction as a projection onto $\ker A_k$, so realizing the update reduces to working with this single operator. The matrix itself, however, cannot be assembled. Since the residual composes the constraint map with the network, A_k inherits the network Jacobian and therefore has no closed-form expression. Furthermore, its size is $b \times P$ with P the parameter count, so storing it is out of reach for a network of realistic size. What remains available are its actions,

$$A_k[v] = \mathrm{JVP}\left(\mathcal{C}_k; v\right), \qquad A_k^*[\Lambda] = \mathrm{VJP}\left(\mathcal{C}_k; \Lambda\right), \tag{3}$$

each costing one forward or one backward pass.

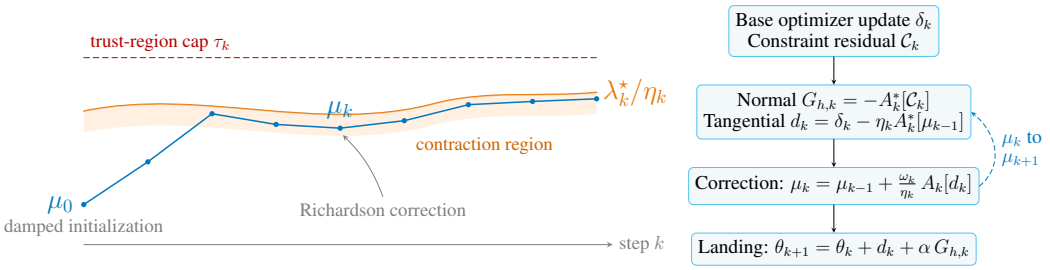
The feasibility component fits this oracle immediately, as $G_{h,k} = -A_k^*[\mathcal{C}_k]$ is a single VJP of a residual that has already been computed. The tangential component is the expensive one. The projection admits the dual representation

$$P_k\delta_k = \delta_k - A_k^*[\lambda_k^\star], \qquad \text{where} \qquad A_k A_k^*[\lambda_k^\star] = A_k[\delta_k] \tag{4}$$

and $\lambda_k^\star \in \mathbb{R}^b$ is the least-norm solution of a $b \times b$ normal system. When the operator is accessible only through its actions, the standard choice is a Krylov method such as conjugate gradients (Hestenes et al., 1952) acting on the Gram operator $A_k A_k^*$, and every Krylov iteration costs one JVP–VJP pair. A budget of m iterations per step therefore multiplies the cost of training by roughly m. Exact projection is thus the bottleneck of the parameter-space landing update.

3.3 Amortizing the Projection via Multiplier Tracking

In this section, we remove the computational bottleneck of Section 3.2, the Krylov solve of (4) that the exact projection requires at every step. The remedy is to spread that solve across training instead of completing it within each step. One training step changes the parameters by a single small update, and re-sampling the mini-batch perturbs the residual statistics around a slowly moving population value, so the exact multiplier $\lambda_k^\star$ drifts slowly across steps. We therefore maintain a multiplier estimate across steps and refine it with one iterative correction per step, letting it chase the moving solution. Figure 2(a) illustrates this tracking dynamic.



(a) (b)

Figure 2: (a) Multiplier tracking. The exact multiplier (orange) drifts slowly across training steps, and the tracked estimate μ_k (blue) chases it with one Richardson correction per step. The damped initialization, spectral refresh, and trust-region cap (red) keep the recursion within the contraction region. (b) One training step of LandTrack, from the optimizer proposal δ_k to the landing update, with μ_k carried to the next step.

We maintain an estimate μ_k of the scale-free multiplier $\lambda_k^\star/\eta_k$. Factoring the learning rate out of the tracked quantity keeps its target stationary under learning-rate schedules. Replacing $\lambda_k^\star$ in (4) by the tracked value $\eta_k\mu_{k-1}$, we obtain the tangent direction

$$d_k = \delta_k - \eta_k\, A_k^*[\mu_{k-1}]. \tag{5}$$

The tracked value is then corrected by a Richardson step (Golub & Van Loan, 2013, §11.2) for the normal system in (4), which moves λ by a multiple of the current residual. At the tracked value $\lambda = \eta_k\mu_{k-1}$ the residual is

$$A_k[\delta_k] - A_k A_k^*\left[\eta_k\mu_{k-1}\right] = A_k\left[\delta_k - \eta_k A_k^*[\mu_{k-1}]\right] = A_k[d_k]. \tag{6}$$

Written in the variable $\mu = \lambda/\eta_k$, the correction reads

$$\mu_k = \mu_{k-1} + \frac{\omega_k}{\eta_k}\, A_k[d_k], \tag{7}$$

with a relaxation parameter $\omega_k > 0$ fixed below. Two properties make this amortization worthwhile. First, it is cheap, as the exact route runs m solver iterations at every training step, while the pair (5)–(7) needs only one. Second, it is unbiased, as a fixed point of the pair satisfies $d_k^* = P_k\delta_k$ exactly, so the tracker converges to the projection. The following proposition quantifies the convergence.

Proposition 1. *Fix $A_k \equiv A$, $\delta_k \equiv \delta$, and $\eta_k \equiv \eta$, let $\sigma_1 \geq \cdots \geq \sigma_q > 0$ be the nonzero singular values of A, and let $\mu_0 \in \mathrm{range}(A)$. If $\omega\sigma_1^2 < 2$, the iterates of (5)–(7) satisfy*

$$\|d_k - P\delta\| \leq \Big(\max_{1\leq i\leq q} \left|1 - \omega\sigma_i^2\right|\Big)^{k-1} \|d_1 - P\delta\|\,. \tag{8}$$

Proof. See Appendix C.1. □

The contraction condition prescribes the relaxation parameter rather than leaving it as a hyperparameter, and we set $\omega_k = 1/\hat{\sigma}_{\max}^2$, where $\hat{\sigma}_{\max}^2$ estimates the largest eigenvalue of $A_k A_k^*$.

3.4 Stabilizing the Tracker

To keep the tracking recursion stable in real training, we introduce three techniques. They are not heuristics but come from Proposition 1, which states what the recursion needs in order to contract, namely a relaxation parameter matched to the leading singular value, an operator and a proposal that hold still between corrections, and a starting point in $\mathrm{range}(A)$.

Spectral refresh. The contraction condition $\omega\sigma_1^2 < 2$ involves the leading singular value of an operator that drifts with θ_k and the mini-batch. We refresh $\hat{\sigma}_{\max}^2$ by a warm-started power iteration (Golub & Van Loan, 2013) every N_ω steps and reset $\omega = 1/\hat{\sigma}_{\max}^2$, keeping the relaxation parameter inside the contraction regime throughout training at negligible amortized cost.

Trust-region scaling. The exact correction in (4) obeys the identity $\|A_k^*[\lambda_k^\star]\| = \|\delta_k - P_k\delta_k\| \leq \|\delta_k\|$, so a tracked correction that greatly exceeds the norm of the proposal must be transient tracking

error rather than signal. In the spirit of a trust-region safeguard (Conn et al., 2000), we scale the applied correction by $s_k := \min\{1,\ c\,\|\delta_k\| \,/\, (\eta_k\,\|A_k^*[\mu_{k-1}]\|)\}$, which leaves accurate corrections untouched and caps inaccurate ones at c times the bound satisfied by the exact solution.

Damped initialization. Proposition 1 guarantees geometric convergence from any starting point in $\mathrm{range}(A)$, but a cold start $\mu = 0$ can make the transient long when the initial residual is large. During the first N_{init} updates, we refine μ by a damped conjugate-gradient solve of $\left(A_k A_k^* + \epsilon_0 \hat{\sigma}_{\max}^2 I\right) \lambda_k = A_k[\delta_k]$ within m_0 iterations (Nocedal & Wright, 2006), warm-started at the tracked value $\eta_k \mu$. The damping handles the near-singular Gram operators that arise before the residual statistics settle.

Algorithm 1 in Appendix A collects the whole method into a single training loop, and Figure 2(b) depicts one iteration of it as a flow of the four moves. One iteration evaluates the residual $\mathcal{C}_k$ and the optimizer proposal δ_k, assembles the normal direction $G_{h,k}$, forms the tangential direction d_k against the tracked multiplier, advances that multiplier by the Richardson correction (7), and applies the landing update (2). The three techniques of this section enter as the tagged lines, the spectral refresh every N_ω steps, the damped initialization during the first N_{init} updates, and the trust-region scaling of the applied correction at every step.

4 Experiments

In this section, we conduct experiments to test the claim that the geometric update improves the accuracy–conservation trade-off relative to other methods, in the strong sense that the whole sampled trade-off curve moves rather than a single tuned weight. To probe this claim broadly rather than on one favourable configuration, the benchmark suite deliberately crosses different system classes with different constraint types. The systems range from interacting-particle dynamics to spatiotemporal PDEs, covering planar 15-body and full-state 3D 24-body systems, the Korteweg–de Vries (KdV) equation, and the reaction–diffusion equation. The constraints cover distinct functional forms of the network output, including a quadratic conserved vector angular momentum, a linear mass invariant, a nonlinear Hamiltonian invariant, and pointwise boundary conditions of the PDE. Beyond the main experiments, Section 4.5 reports ablations on long-horizon autoregressive stability, the effect of stability techniques, and the time cost. The code are available here.

Baselines and metrics. We compare against three baselines. The central baseline is *weighted-penalty training*, the prevailing treatment of constraints in network training (Raissi et al., 2019; Nocedal & Wright, 2006). We also report an *Original* baseline, which trains only the predictive objective and does not use the constraint term. The third baseline is a tuned *augmented Lagrangian (AL) method* (Bertsekas, 2014; Platt & Barr, 1987; Nocedal & Wright, 2006), which maintains a multiplier estimate alongside a quadratic penalty of strength ρ. Implementation details for the baselines are given in Appendices B and B.6. We report the test next-state MSE E_p and test constraint MSE E_c for the dynamical systems, and test MSE E_p and test boundary-condition RMSE E_c for the PDE task.

Pareto view. Our evaluation follows the Pareto view of constrained training, which compares entire trade-off curves rather than a single tuned operating point. The constraint-loss weight β trades predictive accuracy against constraint satisfaction, so sweeping it traces, for each method, a curve of attainable pairs of predictive error E_p and constraint error E_c. To place LandTrack on the same axis, its training objective also carries the quadratic penalty term of the baseline. This leaves the theory of Section 3.1 intact, because the gradient of that term, $\beta A_k^*[\mathcal{C}_k]$, vanishes wherever the constraint is satisfied, so the fixed points are the same KKT points at every β. We therefore compare LandTrack with a penalty model trained at the same β, while AL, whose operating point is set by its own penalty strength ρ, is compared at the curve level. Both β and ρ are swept on a logarithmic grid, and the two endpoints of every sweep are chosen so that pushing the weight further outward degrades both errors at once. Therefore, the reported range brackets the whole useful trade-off. One method improves on another when its curve lies to the lower left of the other's, which enlarges the set of jointly attainable outcomes rather than re-balancing a single trade-off.

4.1 Planar 15-body dynamics with angular-momentum conservation

We first consider a planar 15-body system, in which fifteen interacting particles move in a plane, and an MLP predicts the next state of the whole system from the current one. Angular momentum is a classical conserved quantity of this system, and we impose its conservation as the constraint. Under

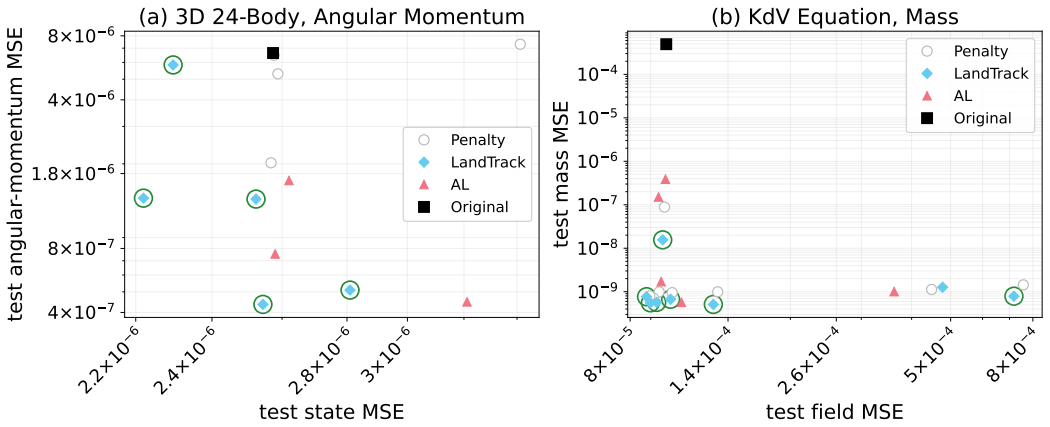


Figure 3: Accuracy–conservation results for (a) the 3D 24-body system with angular-momentum conservation and (b) the KdV equation with mass conservation. Each point corresponds to one constraint-loss weight (β for LandTrack and penalty, ρ for AL). Green rings mark LandTrack configurations that improve both predictive and constraint error relative to the same-weight penalty result. AL points lying far outside the plotted region are omitted for readability, and they are marked † in the result tables. Lower-left is better.

Table 1: Results of the 15-body system with angular-momentum conservation. The left block reports the matched LandTrack–penalty comparison at shared constraint-loss weights β. Bold marks the lower value between LandTrack and penalty at the same weight. The middle block reports the independent AL sweep over its own penalty strength ρ. The right block reports the Original baseline.

	LandTrack		Penalty			AL			Original	
β	E_p	E_c	E_p	E_c	ρ	E_p	E_c	E_p	E_c
10^{-5}	**9.32430**	**0.96897**	9.32697	0.99639	$1^\dagger$	10.65520	16.54864	9.32697	0.99638
10^{-4}	**9.32429**	**0.96899**	9.32696	0.99643	$10^\dagger$	10.39163	11.81476		
10^{-3}	**9.32429**	**0.96908**	9.32684	0.99684	$10^{2\dagger}$	9.97560	8.60774		
10^{-2}	**9.32444**	**0.96958**	9.32605	0.99995	10^3	9.37617	2.26107		
10^{-1}	**9.32587**	**1.00147**	9.32879	1.00583	10^4	9.33563	1.02898		
					10^5	9.34768	1.26143		
					10^6	9.35538	1.33195		

the shared MLP architecture, we sweep the constraint-loss weight over $\beta \in \{10^{-5}, \ldots, 10^{-1}\}$ for penalty and LandTrack, and the penalty strength over $\rho \in \{1, \ldots, 10^6\}$ for AL. Table 1 collects the results. LandTrack improves on the matched penalty run at every sampled weight, and on both metrics, so the sampled LandTrack curve lies to the lower left of the penalty curve. Because the Original model never uses the constraint term, its constraint error is relatively high. But its accuracy is also worse than LandTrack's, which suggests that enforcing the invariant may help prediction rather than costing it. As for the AL baseline, it attains its best trade-off at $\rho = 10^4$, where it minimizes both of its metrics at once, but every one of the five LandTrack points dominates that best AL point on both axes. Taken together, LandTrack dominates every baseline on this system, on both axes and at every point of its weight grid. The corresponding Pareto panel is Figure 5 in the appendix.

4.2 3D 24-BODY DYNAMICS WITH ANGULAR-MOMENTUM CONSERVATION

We next take the same physics into three dimensions with a 24-body system, in which twenty-four interacting particles move in space, and the MLP predicts the full next state. Angular momentum is again the conserved quantity, now a three-component vector rather than a scalar. We sweep $\beta \in \{10^{-7}, \ldots, 10^{-3}\}$ for penalty and LandTrack and $\rho \in \{10^{-5}, \ldots, 10\}$ for AL. Figure 3(a) plots the resulting points, and the complete per-weight numbers are in Table 5 in Appendix B.1. LandTrack again improves on the matched penalty run at every sampled weight and on both metrics, and the margin is far larger than in the planar case. The LandTrack point also beats the Original model on both axes. Against AL, the picture is cleaner still, because that single configuration dominates all nine sampled ρ: the two best AL points, $\rho = 3{\times}10^{-5}$ and $\rho = 10^{-4}$, are both above and to the right of $\beta = 10^{-7}$. LandTrack therefore dominates every baseline here as well.

Table 2: Results of the reaction–diffusion equation with boundary-condition constraints. The table layout and notation are the same as in Table 1.

	LandTrack		Penalty		AL			Original	
β	E_p	E_c	E_p	E_c	ρ	E_p	E_c	E_p	E_c
10	**9.758e−7**	**1.466e−4**	8.533e−3	7.591e−3	1	5.292e−2	4.913e−4	1.129e−6	2.411e−3
10^2	**6.988e−7**	**5.701e−5**	7.360e−7	7.590e−5	10	7.164e−2	8.879e−4		
10^3	**1.135e−6**	**2.994e−5**	8.956e−6	3.238e−5	10^2	7.613e−2	9.049e−4		
10^4	1.019e−6	1.534e−5	**8.670e−7**	**1.395e−5**	10^3	7.530e−2	6.976e−4		
10^5	**5.988e−7**	**6.162e−6**	8.822e−7	1.369e−5	10^4	3.295e−2	5.034e−5		
					10^5	6.482e−2	1.384e−3		
					$10^{6\,\dagger}$	1.079e−1	4.796e−3		
					$10^{7\,\dagger}$	2.647e−1	2.026e−2		

4.3 KdV equation with mass and Hamiltonian conservation

The third system is the KdV equation (Korteweg & De Vries, 1895), a nonlinear dispersive PDE whose solutions carry travelling solitons, discretized to a 64-dimensional field that the network advances one step at a time. KdV admits a hierarchy of invariants, and we take the two most classical, the mass, which is linear in the field, and the Hamiltonian, which is not. They give two independent sweeps on the same data and architecture, mass over $\beta \in \{0.1, \ldots, 1000\}$ and the Hamiltonian over $\beta \in \{1, \ldots, 10^4\}$. Figure 3(b) plots the first of the two sweeps, while the complete numbers and the Pareto curves for both invariants are in Tables 6 and 7 and Figure 6 in the appendix. For mass, LandTrack improves both metrics at seven of the eight weights. The exception is $\beta = 500$, and even there the penalty is itself dominated by LandTrack at $\beta = 100$, so it never reaches the joint front. The Original model shows what the invariant costs when nothing enforces it, at 4.934×10^{-4} against 5.551×10^{-10} for LandTrack at $\beta = 5$, almost six orders of magnitude, with a better field MSE as well. That same $\beta = 5$ configuration dominates the entire AL sweep, including AL's own best trade-offs at $\rho = 10^2$ and $\rho = 10^3$. The Hamiltonian sweep is the more uniform of the two, with a joint improvement at every weight and both errors reduced by roughly 1–13% relative to the matched penalty run. The Original model drifts less on the Hamiltonian than on mass, yet the matched runs at the first three weights all dominate it on both axes, and LandTrack at $\beta = 10^4$ cuts its Hamiltonian error by a factor of 11. Read as curves rather than as isolated points, the comparison with AL is again unambiguous, because the AL polyline lies entirely above and to the right of the LandTrack one.

4.4 Reaction–diffusion equation with boundary-condition constraints

The last system is a reaction–diffusion equation (Fisher, 1937), and the constraint is no longer a conserved quantity but the boundary condition of the PDE, imposed pointwise on the predicted field. The data and the experimental setup follow PDEBench (Takamoto et al., 2022). The model is the Fourier Neural Operator (FNO) (Li et al., 2021), and predictions and constraint residuals act on the discretized field. We sweep $\beta \in \{10, \ldots, 10^5\}$ for penalty and LandTrack and $\rho \in \{1, \ldots, 10^7\}$ for AL. Table 2 and Figure 7 collect the results, where LandTrack improves both metrics at four of the five weights. The exception is $\beta = 10^4$, where again the penalty at this weight is dominated by LandTrack at $\beta = 10^5$. This means that this point of the penalty never reaches the joint front. The Original model sits at 2.411×10^{-3} on the constraint axis against 6.162×10^{-6} for LandTrack at $\beta = 10^5$, a factor of 391, and is worse on accuracy as well. AL is not competitive here, as its best point $\rho = 10^4$ is still dominated by that same LandTrack configuration. LandTrack thus performs the best on this system too, and the gain carries over from conserved quantities to boundary conditions.

4.5 Ablation Studies

Long-term prediction. To test on the long-horizon setting, we roll out the checkpoints on the KdV equation with mass conservation for 20 steps and report the rollout field MSE E_p^{roll} and the normalized mass drift E_c^{roll}, whose normalization and reference point are defined in Appendix B.5. In Figure 4(a), the LandTrack points are concentrated further toward the lower left than those of the other methods, so the corresponding Pareto curve lies to the lower left and LandTrack is both more accurate and more conservative under rollout. Figure 4(b) further shows how the conservation error of each method evolves with the number of rollout steps. The LandTrack curves lie lower overall, indicating a smaller error along the whole horizon. The penalty run at $\beta = 10$ is close to LandTrack

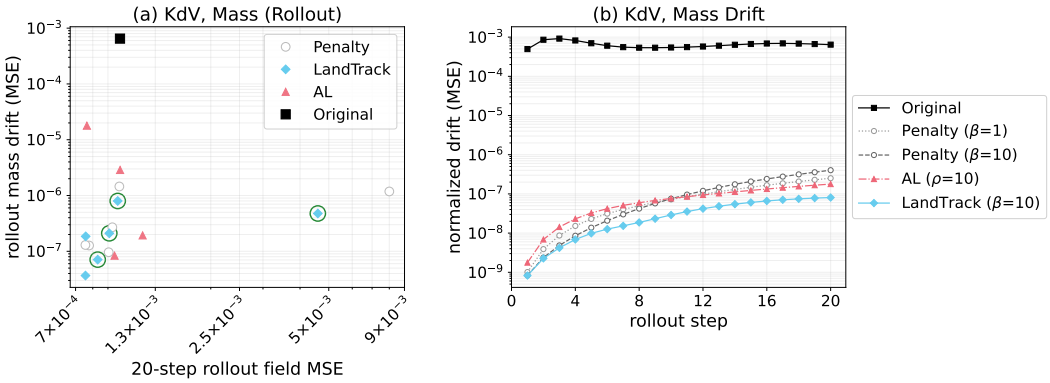


Figure 4: Long-term prediction on the KdV equation. (a) Pareto plot of 20-step rollout field MSE versus rollout mass drift. Points far outside the plotted region are omitted for readability and marked † in Table 8. Green rings mark LandTrack points that jointly improve on the penalty run of the same weight. (b) Mean normalized mass drift versus rollout step for representative checkpoints.

Table 3: Leave-one-out stabilizer ablation on the 3D 24-body system with angular-momentum conservation. "-" marks the run whose training loss becomes NaN.

Variant	E_p	E_c
Full LandTrack	2.22e−6	1.30e−6
No spectral refresh	2.22e−6	1.80e−6
No damped initialization	2.26e−6	1.34e−6
No trust region	-	-
All three disabled	2.26e−6	1.82e−6

Table 4: Per-epoch training cost on the reaction–diffusion equation with boundary-condition constraints.

	LandTrack			
	init-CG	tracking	Penalty	AL
s/epoch	29.2	21.4	16.7	17.5

over the first few steps, but its error grows faster as the rollout proceeds and is about five times larger by the final step. LandTrack is therefore more stable in long-horizon autoregressive prediction, with an error that grows more slowly. Complete numbers are reported in Table 8 in Appendix B.1.

Stabilizer ablation. Here, we verify the effectiveness of safeguards with a leave-one-out ablation on the 3D 24-body system with angular-momentum conservation at $\beta = 10^{-4}$. Table 3 reports the outcome. We find that the trust-region cap is the load-bearing safeguard. Without it, training reaches a NaN loss, as the uncapped correction $\eta_k A_k^*[\mu]$ transiently exceeds the proposal and the blow-up feeds back through the recursion. The remaining variants finish training, but degrade both metrics jointly. Each safeguard therefore makes its own contribution to stable and accurate training.

Time cost. We report the per-epoch cost on the reaction–diffusion equation in Table 4. During the first N_{init} updates, each step solves a damped projection system by conjugate gradients (the *init-CG phase*). Afterwards, the multiplier follows a cheap recursive update (the *tracking phase*). An init-CG epoch costs about 36% more than a tracking-phase epoch, but init-CG covers only the first few steps, so its surcharge is a one-off cost of a few seconds. The tracking phase adds $4.7s$ and $3.9s$ per epoch over the penalty and AL baselines. The amortization thus keeps the overhead in the same cost regime.

5 Conclusion

We introduce LandTrack, a method for equality-constrained neural network training that enforces the constraint through the training dynamics. Every update splits into a tangential component that decreases the objective along the constraint set and an orthogonal normal component that contracts the residual, so its fixed points are the KKT points of the sampled problem. Rather than pay a Krylov solve per step for the exact tangential projection, we track its multiplier across training steps with one Richardson correction per step, computed from a residual the update already provides and retaining the exact projection as its fixed point. Across particle dynamics and PDEs, with constraints ranging from conserved quantities to boundary conditions, LandTrack improves the entire accuracy–conservation trade-off curve over baselines, suggesting that constraining the training dynamics is a broadly applicable principle for learned models that must respect exact structural requirements.

AI USE STATEMENT

In this work, we used generative AI tools to assist with creating and modifying scientific figures, suggesting candidate experimental parameters, editing software code, summarizing and analysing existing literature, brainstorming, sourcing and searching for information, identifying relevant literature, and editing the manuscript to improve readability. We did not use generative AI tools to develop the theoretical model or conceptual framework, to formulate the mathematical claims, to provide critical ingredients for or assist in the writing of the proofs, to propose or refine the hypotheses, to design the research methodology or experiments, to implement the method, to generate synthetic data sets, or to interpret the results. The remaining categories of AI use listed in the ICLR policy are not applicable to this work.

We have reviewed all AI-assisted work. All AI-suggested experimental parameters were validated by running the corresponding experiments, and all reported numbers come from those runs rather than from any AI tool. AI-edited code was read and tested by the authors, and all figures were checked against the underlying result files. AI-identified references were verified by the authors against the primary sources. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All datasets in this work are collected from numerical solvers, without involving sensitive information. Caution that models trained in this work should not be directly applied in safety-critical scenarios without further validation.

REPRODUCIBILITY STATEMENT

The code and data are available at `https://anonymous.4open.science/r/LandTrack`, with full documentation to ensure transparency and reproducibility.

REFERENCES

Pierre Ablin and Gabriel Peyré. Fast and accurate optimization on the orthogonal manifold without retraction. In *International Conference on Artificial Intelligence and Statistics*, pp. 5636–5657. PMLR, 2022.

Pierre Ablin, Simon Vary, Bin Gao, and Pierre-Antoine Absil. Infeasible deterministic, stochastic, and variance-reduction algorithms for optimization under orthogonality constraints. *Journal of Machine Learning Research*, 25(389):1–38, 2024.

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. volume 32, 2019.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pp. 136–145. PMLR, 2017.

Kenneth J Arrow, Leonid Hurwicz, and Hirofumi Uzawa. Studies in linear and non-linear programming. 1958.

Shamsulhaq Basir and Inanc Senocak. Physics and equality constrained artificial neural networks: Application to forward and inverse problems with multi-fidelity data fusion. *Journal of Computational Physics*, 463:111301, 2022.

Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

Tom Beucler, Michael Pritchard, Stephan Rasp, Jordan Ott, Pierre Baldi, and Pierre Gentine. Enforcing analytic constraints in neural networks emulating physical systems. *Physical review letters*, 126(9):098302, 2021.

Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*. SIAM, 2000.

Andrew Cotter, Heinrich Jiang, Maya Gupta, Serena Wang, Taman Narayan, Seungil You, and Karthik Sridharan. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research*, 20(172):1–59, 2019.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. 2020.

Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. 2021.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Ronald Aylmer Fisher. The wave of advance of advantageous genes. *Annals of eugenics*, 7(4): 355–369, 1937.

Jose Gallego-Posada, Juan Ramirez, Akram Erraqabi, Yoshua Bengio, and Simon Lacoste-Julien. Controlled sparsity via constrained optimization or: How i learned to stop tuning penalties and love constraints. *Advances in Neural Information Processing Systems*, 35:1253–1266, 2022.

Bin Gao, Xin Liu, and Ya-xiang Yuan. Parallelizable algorithms for optimization problems with orthogonality constraints. *SIAM Journal on Scientific Computing*, 41(3):A1949–A1983, 2019.

Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2013.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. volume 32, 2019.

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

Peiyan Hu, Rui Wang, Xiang Zheng, Tao Zhang, Haodong Feng, Ruiqi Feng, Long Wei, Yue Wang, Zhi-Ming Ma, and Tailin Wu. Wavelet diffusion neural operator. In *International Conference on Learning Representations*, volume 2025, pp. 12291–12333, 2025.

Peiyan Hu, Jian Zhang, Jiashu Pan, Ruiqi Feng, Tao Zhang, Zhi-Ming Ma, Yuan-Sen Ting, Gongjie Li, and Tailin Wu. Solving inverse problems of chaotic systems with bidirectional conditional flow matching, 2026. URL https://arxiv.org/abs/2606.24824.

Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2 (10.48550), 2017.

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

Diederik Johannes Korteweg and Gustav De Vries. Xli. on the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 39(240):422–443, 1895.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Levi D McClenny and Ulisses M Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722, 2023.

Geoffrey Négiar, Michael W. Mahoney, and Aditi Krishnapriyan. Learning differentiable solvers for systems with hard constraints. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vdv6CmGksr0.

J Nocedal and S Wright. Numerical optimization, 2006.

John Platt and Alan Barr. Constrained differential optimization. In *Neural information processing systems*, 1987.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

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.

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

Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International conference on machine learning*, pp. 9133–9143. PMLR, 2020.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in neural information processing systems*, 35:1596–1611, 2022.

Haixin Wang, Yadi Cao, Zijie Huang, Yuxuan Liu, Peiyan Hu, Xiao Luo, Zezheng Song, Wanjia Zhao, Jilin Liu, Jinan Sun, et al. Recent advances on machine learning for computational fluid dynamics: A survey. *arXiv preprint arXiv:2408.12171*, 2024.

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

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Nachuan Xiao, Xin Liu, and Ya-xiang Yuan. A class of smooth exact penalty function methods for optimization problems with orthogonality constraints. *Optimization Methods and Software*, 37(4): 1205–1241, 2022.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International conference on machine learning*, pp. 5502–5511. PMLR, 2018.

Yan Yang, Bin Gao, and Ya-xiang Yuan. Optimization over the intersection of manifolds. *arXiv preprint arXiv:2605.22736*, 2026.

Yushun Zhang, Congliang Chen, Naichen Shi, Ruoyu Sun, and Zhi-Quan Luo. Adam can converge without any modification on update rules. *Advances in neural information processing systems*, 35: 28386–28399, 2022.

A ALGORITHM

Algorithm 1 Landing with Multiplier Tracking

Require: Initial parameters θ_0, iterations K, feasibility step α, base optimizer Opt with learning rates $\{\eta_k\}$, trust factor c, refresh period N_ω, initialization steps N_{init} with CG budget m_0, and damping ϵ_0.

$\mu \leftarrow \mathbf{0}$
for $k = 0, 1, \ldots, K-1$ **do**
  Sample $\mathcal{B}_k$; $X_i \leftarrow \mathcal{N}_{\theta_k}(z_i)$, $C_i \leftarrow h_{z_i}(X_i)$ for $z_i \in \mathcal{B}_k$
  $\mathcal{C}_k \leftarrow \mathrm{stack}\,(C_1, \ldots, C_b)$; $\delta_k \leftarrow \mathrm{Opt}\,(\nabla_\theta F_k; \eta_k)$
  Define $A_k[\cdot]$, $A_k^*[\cdot]$ by JVP / VJP of $\mathcal{C}_k$ at θ_k
  $G_{h,k} \leftarrow -A_k^*[\mathcal{C}_k]$
  if $k \equiv 0 \pmod{N_\omega}$ **then**
    refresh $\hat{\sigma}^2_{\max}$ by warm power iteration; $\omega \leftarrow 1/\hat{\sigma}^2_{\max}$ [Spectral refresh]
  end if
  if $k < N_{\mathrm{init}}$ **then**
    solve $\left(A_k A_k^* + \epsilon_0 \hat{\sigma}^2_{\max} I\right) \lambda_k = A_k[\delta_k]$ within m_0 CG iterations, warm-started at $\eta_k \mu$;
    $\mu \leftarrow \lambda_k / \eta_k$ [Damped initialization]
  end if
  $s_k \leftarrow \min\{1,\, c\,\|\delta_k\| \,/\, (\eta_k\, \|A_k^*[\mu]\|)\}$ [Trust-region scaling]
  $d_k \leftarrow \delta_k - s_k \eta_k A_k^*[\mu]$; $\mu \leftarrow \mu + (\omega/\eta_k)\, A_k[d_k]$ [Multiplier tracking (7)]
  $\theta_{k+1} \leftarrow \theta_k + \alpha\, G_{h,k} + d_k$
end for
return θ_K

For completeness, Algorithm 1 states the full LandTrack training loop described in Section 3, with the three stabilization techniques of Section 3.4 entering as the tagged lines.

B ADDITIONAL EXPERIMENTAL DETAILS AND RESULTS

B.1 COMPLETE RESULT TABLES

In this section, we collect the complete per-weight result tables referenced from the main text, covering the 3D 24-body system with angular-momentum conservation (Table 5), the KdV equation with mass and with Hamiltonian conservation (Tables 6 and 7), and the 20-step rollout on the KdV equation with mass conservation (Table 8).

Table 5: Results of the 3D 24-body system with angular-momentum conservation. Both metrics are test MSE. The left block reports the matched LandTrack–penalty comparison at shared constraint-loss weights β. Bold marks the lower value between LandTrack and penalty at the same weight. The middle block reports the independent AL sweep over its own penalty strength ρ. The right block reports the Original baseline.

	LandTrack		Penalty		AL			Original	
β	E_p	E_c	E_p	E_c	ρ	E_p	E_c	E_p	E_c
10^{-7}	**2.544e−6**	**4.364e−7**	2.574e−6	6.645e−6	10^{-5}	2.621e−6	1.669e−6	2.573e−6	6.641e−6
10^{-6}	**2.296e−6**	**5.837e−6**	2.577e−6	6.488e−6	$3{\times}10^{-5}$	2.580e−6	7.532e−7		
10^{-5}	**2.524e−6**	**1.365e−6**	2.588e−6	5.300e−6	10^{-4}	3.211e−6	4.484e−7		
10^{-4}	**2.220e−6**	**1.379e−6**	2.567e−6	2.023e−6	$3{\times}10^{-4\,\dagger}$	7.098e−6	8.607e−5		
10^{-3}	**2.810e−6**	**5.096e−7**	3.412e−6	7.306e−6	$10^{-3\,\dagger}$	4.055e−5	1.706e−3		
					$10^{-2\,\dagger}$	4.382e−4	1.515e−2		
					$10^{-1\,\dagger}$	2.373e−3	5.617e−2		
					$1^{\dagger}$	1.360e−2	1.292e−1		
					$10^{\dagger}$	2.525e−1	1.673e−1		

Table 6: Results of the KdV equation with mass conservation. E_p is test field MSE and E_c is test mass MSE. The left block reports the matched LandTrack–penalty comparison at shared constraint-loss weights β. Bold marks the lower value between LandTrack and penalty at the same weight. The middle block reports the independent AL sweep over its own penalty strength ρ. The right block reports the Original baseline.

	LandTrack		Penalty		AL			Original	
β	E_p	E_c	E_p	E_c	ρ	E_p	E_c	E_p	E_c
10^{-1}	**9.638e−5**	**1.557e−8**	9.737e−5	8.851e−8	1	9.790e−5	3.861e−7	9.836e−5	4.934e−4
1	**9.313e−5**	**5.676e−10**	9.441e−5	9.939e−10	10	9.554e−5	1.699e−9		
5	**8.982e−5**	**5.551e−10**	9.145e−5	7.108e−10	10^2	9.409e−5	1.505e−7		
10	**8.788e−5**	**7.668e−10**	8.951e−5	7.994e−10	10^3	1.073e−4	5.680e−10		
50	**1.008e−4**	**6.706e−10**	1.017e−4	9.528e−10	10^4	3.619e−4	1.004e−9		
100	**1.287e−4**	**5.101e−10**	1.320e−4	9.943e−10	$10^{5\dagger}$	1.873e−3	1.402e−9		
500	4.776e−4	1.268e−9	**4.484e−4**	**1.126e−9**	$10^{6\dagger}$	2.634e−3	1.535e−9		
1000	**7.180e−4**	**7.881e−10**	7.575e−4	1.437e−9					

Table 7: Results of the KdV equation with Hamiltonian conservation. E_p is test field MSE and E_c is test Hamiltonian MSE. The left block reports the matched LandTrack–penalty comparison at shared constraint-loss weights β. Bold marks the lower value between LandTrack and penalty at the same weight. The middle block reports the independent AL sweep over its own penalty strength ρ. The right block reports the Original baseline, which is shared with the mass sweep and evaluated on the Hamiltonian here.

	LandTrack		Penalty		AL			Original	
β	E_p	E_c	E_p	E_c	ρ	E_p	E_c	E_p	E_c
1	**8.642e−5**	**8.698e−5**	9.269e−5	9.495e−5	1	9.373e−5	9.801e−5	9.836e−5	1.438e−4
10	**9.069e−5**	**1.002e−4**	9.172e−5	1.020e−4	10	9.281e−5	9.071e−5		
10^2	**8.507e−5**	**9.642e−5**	8.588e−5	1.007e−4	10^2	9.246e−5	9.458e−5		
10^3	**2.160e−4**	**1.948e−5**	2.249e−4	1.995e−5	10^3	8.841e−5	1.099e−4		
10^4	**1.009e−3**	**1.334e−5**	1.155e−3	1.502e−5	10^4	1.817e−4	4.050e−5		
					10^5	1.056e−3	2.810e−5		
					$10^{6\dagger}$	1.718e−3	3.465e−5		

B.2 Additional Pareto plots

The main text shows the Pareto panels for the 3D 24-body system with angular-momentum conservation and the KdV equation with mass conservation in Figure 3. This appendix gives the panels for the remaining settings, namely the 15-body system in Figure 5, both KdV invariants in Figure 6, and the reaction–diffusion equation in Figure 7. They are the graphical counterparts of Tables 1, 2, 6 and 7, and they are read in the same way as Figure 3. In every panel the LandTrack points lie toward the

Table 8: Results of the 20-step rollout on the KdV equation with mass conservation. E_p^{roll} is rollout field MSE and E_c^{roll} is rollout mass drift MSE, both averaged over 20 steps and 181 test start states. The left block reports the matched LandTrack–penalty comparison at shared constraint-loss weights β. Bold marks the lower value between LandTrack and penalty at the same weight. The middle block reports the independent AL sweep over its own penalty strength ρ. The right block reports the Original baseline. † marks points lying far outside the region plotted in Figure 4(a).

	LandTrack		Penalty		AL			Original	
β	E_p^{roll}	E_c^{roll}	E_p^{roll}	E_c^{roll}	ρ	E_p^{roll}	E_c^{roll}	E_p^{roll}	E_c^{roll}
0.1	**9.712e−4**	**7.982e−7**	9.835e−4	1.453e−6	1	9.892e−4	2.872e−6	9.884e−4	6.495e−4
1	**8.313e−4**	**7.095e−8**	9.052e−4	9.616e−8	10	9.476e−4	8.359e−8		
5	**7.573e−4**	1.853e−7	7.774e−4	**1.271e−7**	10^2	7.643e−4	1.787e−5		
10	7.560e−4	**3.676e−8**	**7.556e−4**	1.297e−7	10^3	1.178e−3	1.929e−7		
50	**9.095e−4**	**2.102e−7**	9.313e−4	2.696e−7	$10^{4\dagger}$	7.308e−1	1.343e−2		
10^2	**4.593e−3**	**4.764e−7**	8.002e−3	1.182e−6	$10^{5\dagger}$	7.027e0	2.783e−1		
$500^\dagger$	1.385e0	3.176e−2	**1.036e0**	**2.971e−2**	$10^{6\dagger}$	9.357e0	6.168e−1		
$10^{3\dagger}$	2.973e0	**8.009e−2**	**2.656e0**	1.075e−1					

lower left of the penalty and AL points, so the improvement discussed in Section 4 appears as a shift of the whole curve rather than of a single operating point.

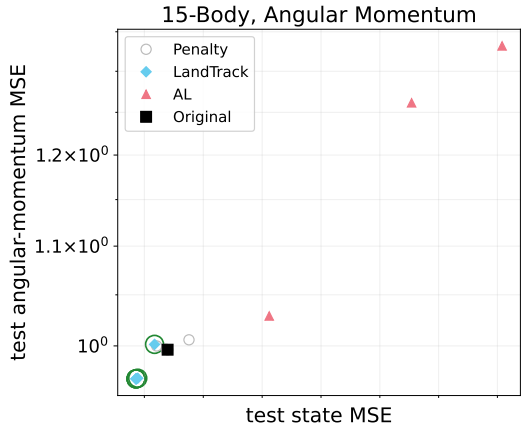


Figure 5: Accuracy–conservation results for the 15-body system with angular-momentum conservation. Each point corresponds to one constraint-loss weight (β for LandTrack and penalty, ρ for AL). Green rings mark LandTrack configurations that improve both predictive and constraint error relative to the same-weight penalty result. AL points lying far outside the plotted region are omitted for readability, and they are marked † in the result tables. Lower-left is better.

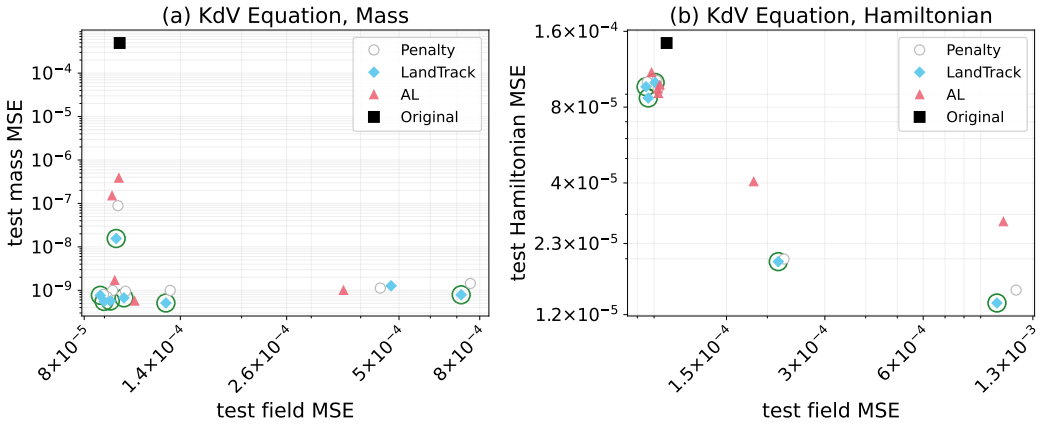


Figure 6: Accuracy–conservation results for the KdV equation with mass conservation (a) and with Hamiltonian conservation (b). Each point corresponds to one constraint-loss weight (β for LandTrack and penalty, ρ for AL). Green rings mark LandTrack configurations that improve both predictive and constraint error relative to the same-weight penalty result. AL points lying far outside the plotted region are omitted for readability, and they are marked † in the result tables. Lower-left is better.

B.3 Hyperparameters

In Table 9, we record the model parameters used for the matched penalty and LandTrack comparisons. Every setting additionally includes an Original run with no constraint term. Additionally, the two KdV equation sweeps share one such run. The MLP rows use Adam and share the listed hidden width and batch size between penalty and LandTrack.

Time Cost. The per-epoch times in Table 4 are wall-clock measurements on a single H20 GPU, taken from the epoch timer of the shared training loop. Each entry therefore covers one complete pass over the training split, plus the per-epoch validation forward pass; the validation cost is identical across methods and cancels in the relative comparison. All four columns run the same pipeline with the

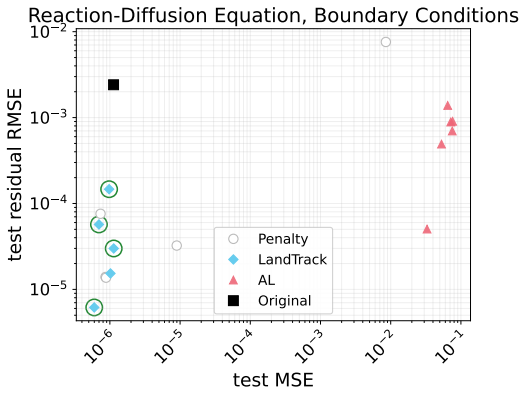


Figure 7: Accuracy–conservation results for the reaction–diffusion equation with the boundary-condition constraint. Each point corresponds to one constraint-loss weight (β for LandTrack and penalty, ρ for AL). Green rings mark LandTrack configurations that improve both predictive and constraint error relative to the same-weight penalty result. AL points lying far outside the plotted region are omitted for readability, and they are marked † in the result tables. Lower-left is better.

Table 9: Model hyperparameters used for the matched penalty and LandTrack comparisons.

Experiment	Model	Update Steps	Hidden Width	Batch Size	Learning Rate	Weight Decay
15-body	MLP	185 steps	200	32	3×10^{-5}	10^{-5}
3D 24-body	MLP	58,603 steps	200	32	1.2×10^{-4}	10^{-5}
KdV	MLP	29,800 steps	200	32	5×10^{-4}	10^{-5}
Reaction–diffusion	FNO	90 epochs	–	5	10^{-3}	10^{-5}

same data split, the same batch size of 5 from the shared configuration, and hence the same number of optimizer updates per epoch; the launchers override only the method and the constraint weight. The differences in Table 4 thus isolate the per-update cost of each method. Because the production runs use $N_{\text{init}} = 25$ updates, a small fraction of the 180 updates in one epoch, no production epoch consists purely of init-CG steps. The init-CG column is therefore measured from a dedicated timing run that uses the same tuned configuration but extends N_{init} to cover three full epochs, and reports their mean. The tracking column of the same run reproduces the value obtained from the production logs.

B.4 Weight grids

The finalized positive-weight points shown in the figures use the following weight sets. Every setting additionally shows the standalone Original point:

- 15-body system: angular momentum $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.
- 3D 24-body system: angular momentum $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$.
- KdV equation: mass $\{0.1, 1, 5, 10, 50, 100, 500, 1000\}$ and Hamiltonian $\{1, 10, 10^2, 10^3, 10^4\}$. The two sweeps share a single β-free Original run, and each reports its own invariant error evaluated on that shared checkpoint.
- Reaction–diffusion equation: boundary condition $\{10, 10^2, 10^3, 10^4, 10^5\}$.

B.5 Constraint normalization and the rollout drift metric

In the three conservation-law settings, 15-body, 24-body, and KdV, the constrained quantity is a conserved invariant I of the system: energy, linear and angular momentum for the many-body suites, and mass or the Hamiltonian for KdV. The constraint errors we report, both the training-time E_c and

the rollout drift E_c^{roll} of Section 4.5, are expressed in units of a fixed scale

$$s_I \;=\; \max\Big(\sqrt{\tfrac{1}{N}\textstyle\sum_{n=1}^{N} I(u^{(n)})^2},\; 1\Big), \tag{9}$$

estimated once on the training states before training begins. It is one constant per invariant, so it cannot change the ordering of any two runs, and it is the form in which the constraint enters training: the penalty term, the AL multiplier update, and LandTrack's residual $\mathcal{C}_k$ all act on $I(\cdot)/s_I$. The reported errors are therefore exactly the quantity that training optimizes. The floor at 1 leaves invariants of very small magnitude unscaled. For KdV, it is inactive for mass ($s_{\mathrm{mass}} = \sqrt{3}$) and active for the Hamiltonian ($s_{\mathrm{ham}} = 1$).

The reaction–diffusion setting is the one exception. Its constraint is the periodicity residual of the spatial boundary, which is not a conserved quantity, so no scale (9) is defined and E_c is reported in the raw residual units in which it is trained. Absolute constraint errors should thus not be compared across settings.

The two reported errors differ in their reference point. The training-time E_c averages the one-step deviation $\left(I(\hat{u}_{k+1}) - I(u_{k+1})\right)/s_I$ from the true invariant, whereas the rollout metric averages the departure from each trajectory's own initial value,

$$E_c^{\mathrm{roll}} \;=\; \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\Big[\big(\left(I(u_t) - I(u_0)\right)/s_I\big)^2\Big], \qquad u_t = f_\theta^t(u_0), \tag{10}$$

over $T = 20$ steps and the 181 held-out start states. A model can be accurate at every single step and still accumulate visible drift under (10), which is what Figure 4 probes. The two are not directly comparable in magnitude. Full definitions are in the released training and evaluation code.

B.6 Baseline implementation and tuning

Penalty and Original. The penalty baseline minimizes the predictive objective plus β times the constraint loss and shares the architecture, data split, seed, optimizer, and training budget of Table 9 with the matched LandTrack run at the same β. The Original baseline is the β-free member of the same protocol, training only the predictive objective and never using the constraint term.

AL baseline. The AL baseline maintains its own multiplier estimate ν, unrelated to the tracked multiplier μ of Section 3.3, alongside a quadratic penalty of strength ρ, and applies the classical update $\nu \leftarrow \nu + \rho\, C$ every T_d optimizer steps, where C is the running mean constraint residual since the previous update; the multiplier is zero-initialized and updated with the classical step size (dual learning rate equal to ρ). It shares each setting's training protocol from Table 9, and every run is trained to convergence with the same early-stopping rule as the recorded sweeps.

C Deferred Proofs

C.1 Proof of Proposition 1

Proof. Let $\mu^\star := \lambda^\star/\eta$, where $\lambda^\star$ is the least-norm solution in (4), and set $e_k := \mu_k - \mu^\star$. Substituting $d_k = \delta - \eta A^*[\mu_{k-1}]$ into the update (7) yields $A[d_k] = \eta A A^*\left[\mu^\star - \mu_{k-1}\right]$, and hence $e_k = \left(I - \omega A A^*\right) e_{k-1}$. Moreover, it holds that $d_k - P\delta = \eta A^*\left[\mu^\star - \mu_{k-1}\right] = -\eta A^*[e_{k-1}]$. The least-norm solution satisfies $\lambda^\star \in \mathrm{range}(A)$, and $\mathrm{range}(A)$ is invariant under $I - \omega A A^*$; combining the two facts with $\mu_0 \in \mathrm{range}(A)$, expanding e_{k-1} along the left singular vectors u_i of A gives $\langle e_{k-1}, u_i\rangle = \left(1 - \omega\sigma_i^2\right)^{k-1}\langle e_0, u_i\rangle$. Therefore,

$$\left\|d_k - P\delta\right\|^2 = \eta^2 \sum_{i=1}^{q} \sigma_i^2 \left(1 - \omega\sigma_i^2\right)^{2(k-1)} \langle e_0, u_i\rangle^2 \le \Big(\max_{1\le i\le q}\left|1 - \omega\sigma_i^2\right|\Big)^{2(k-1)} \left\|d_1 - P\delta\right\|^2,$$

where the inequality uses $\left\|d_1 - P\delta\right\|^2 = \eta^2 \sum_{i=1}^{q} \sigma_i^2 \langle e_0, u_i\rangle^2$, which completes the proof. □