THE IMPROVEMENT GUARANTEE OF LEARNING MPC LEAKS ON NONLINEAR SYSTEMS, AND HOW TO BOUND IT

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning Model Predictive Control (LMPC) improves at a repetitive task by storing executed trajectories and reusing their realized cost-to-go as terminal cost. Its improvement guarantee is proved on the stored states, but a nonlinear closed loop only ever reaches a tolerance-neighbourhood of them, so the guarantee must survive whatever extends the value off the store. We define the *consistency leak*, the amount by which one step of true progress can fail to buy one step of promised value, and prove that the guarantee degrades by at most the leak per step. Measured on stores that a minimum-time LMPC loop generates for itself, the position-indexed constructions the literature runs—convex safe sets, nearest-neighbour lookups, a learned value—leak more as experience accumulates: within nine iterations the improvement bound exceeds the cost it bounds. State-aware constructions keep the leak bounded but expose almost none of the stored improvement. The remedy is to take the nearest neighbour *within* each stored trajectory and then the minimum over trajectories; its leak is bounded by plant and design constants alone. In closed loop the per-trajectory minimum completes all 176 iterations across two plants while the pooled lookup completes fewer than half, at 0.58–$2.2\times$ the cost per MPC solve. Code is available at `https://anonymous.4open.science/r/lmpcleak-54F8`.

1 INTRODUCTION

Many robotic deployments repeat one task under nearly identical conditions, inviting the controller to improve from its own execution history. Learning MPC (LMPC) (Rosolia & Borrelli, 2017a) realizes this: it stores executed trajectories in a *sampled safe set* $\mathcal{SS}$, uses the realized cost-to-go on $\mathcal{SS}$ as terminal cost, and constrains the predicted terminal state to lie in $\mathcal{SS}$. Because every stored state carries a stored tail to the goal, this delivers recursive feasibility and iterative improvement, $J^j \leq J^{j-1}$, with extensions to linear (Rosolia & Borrelli, 2017b), racing (Xue et al., 2024), output-lifted (Nair et al., 2021) and uncertain (Rosolia et al., 2022) settings.

That improvement proof has a hypothesis it never states: at the predicted terminal state it appends the stored input that carries the system one step down its own trajectory—sound because that state lies *in* $\mathcal{SS}$, where the empirical value Q^j is exact and telescopes. A nonlinear closed loop never lands there—steering the predicted terminal state onto a finite set of stored states is a measure-zero event—so implementations require only that it come within a tolerance τ, and the proof must run through whatever extends Q^j off the store, on the τ-neighbourhood $\mathcal{N}_\tau$. We show (Thm. 4) that the argument survives exactly when one quantity is small: the *consistency leak* η_τ, the amount by which one step of true progress can fail to buy one step of promised value. The guarantee degrades to $J^j \leq J^{j-1} + T^j\eta_\tau$: classical at $\eta_\tau = 0$, certifying nothing once $T^j\eta_\tau$ exceeds the cost it bounds.

Consistency is necessary but not sufficient. An evaluation can zero η_τ by promising exactly what the controlled vehicle (the *ego*) is already doing; such a value is inert. A terminal value must also have *drive*, promising strictly less than the trajectory a stored state sits on, and it must be *achievable*, never promising below what some stored trajectory delivered (Sec. 4.3). Every published evaluation family gives up at least one of the three properties. The *convex safe set* (Rosolia & Borrelli, 2017b) is a linear-systems construction: on a nonlinear plant, in both its global and local per-lap forms (Rosolia & Borrelli, 2021), it prices convex mixtures the plant cannot reach and under-states the

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.

cost-to-go at most of the states it must reproduce. *Position-indexed* lookups hop between iterations: later iterations are faster, so one task position carries a different cost-to-go in each, and the pooled lookup's drive is identically zero on the stored set. *State-aware* indexings, such as the full-state K-nearest minimum of i2LQR (Zeng et al., 2023; 2025), keep the leak bounded but expose only a small share of the store's improvement. *LMPC's own accumulation of experience is what destroys its improvement guarantee*—which is why the defect is invisible in the single-iteration studies where these implementations are validated.

The remedy is not a better approximation but the definition: take the nearest neighbour *within* each stored trajectory, then minimize over trajectories. This evaluation, $Q^j_{\min}$, stays on the discrete store, needs no convexity of the dynamics or admissible set, has all three properties by construction (Prop. 8), and its leak is bounded by plant and design constants independent of the iteration count (Prop. 6). The bounded leak buys more than bounded degradation: because the store never discards, an iteration cannot fall behind the *best* one so far by more than $T^j\eta_\tau$ (Thm. 5), and the iteration cost is asymptotically confined to a band of width $\bar{T}\eta_\tau$ above its own infimum (Cor. 7)—a plant constant for the per-trajectory minimum, growing with the store for the pooled lookup (Prop. 6). Convergence to the infimum needs the leak to vanish; over 25 iterations we find it falling, and claim only that.

Contributions.

- *Theory:* the consistency leak η_τ (Def. 3), the bound $J^j \le J^{j-1} + T^j\eta_\tau$ (Thm. 4), its record-relative strengthening (Thm. 5), and convergence into a band of width $\bar{T}\eta_\tau$ (Cor. 7).
- *Analysis:* leak bounds per evaluation family—iteration-independent for the per-trajectory minimum, growing with the store for the pooled lookup (Prop. 6)—and which of the three properties each keeps (Props. 1–3, 8).
- *Measurement:* faithful representatives of every published construction, measured for all three properties on stores LMPC generated for itself: position-indexed and convexified leaks grow with experience as the bounds predict; state-aware alternatives stay bounded at the price of drive (Table 1).
- *Closed loop:* $Q^j_{\min}$ completes all 176 bicycle and planar-quadrotor iterations and converges into the predicted band while the published evaluations lose iterations and cost; on a 13-state MuJoCo quadrotor both arms complete every lap and the defect costs the best lap found (Table 2). Appendix F maps every claim to its evidence.

2 Related Work

LMPC builds terminal constraints and costs from executed trajectories (Rosolia & Borrelli, 2017a;b); variants add error-dynamics regression, output lifting, robustness, multiple objectives, multi-modality and exploration (Xue et al., 2024; Nair et al., 2021; Nair & Borrelli, 2023; Rosolia et al., 2022; Petrenz et al., 2025; Nair et al., 2024; Kopp & Borrelli, 2024; Hashimoto & Hashimoto, 2025; Zang et al., 2026); see Hewing et al. (2020) for a survey. The exact empirical value is a minimum over *discrete stored indices* (Def. 1), which makes (4) mixed-integer, so every practical method substitutes something evaluable off the stored states. The linear-systems line retains the *convex safe set* (Rosolia & Borrelli, 2017b; Nair & Borrelli, 2023; Nair et al., 2024; Petrenz et al., 2025), whose guarantees are valid within that system class and not at issue here. Nonlinear implementations run its *local* variant, which convexifies the K nearest stored states of each stored lap per query (Rosolia & Borrelli, 2021), or select K candidates by a weighted full-state metric and take the minimal cost-to-go among them (Zeng et al., 2023; 2025); *learned approximators* regress the stored cost-to-go (Bao et al., 2024; Hashimoto et al., 2025). The position-indexed 1-NN *pooled lookup* Q^j_{pool} is the one-line simplification of those K-nearest rules; Sec. 5.2 measures it alongside faithful representatives of each construction, separating the simplification's failures from the family's. None of these evaluations is analyzed for what it does to the improvement guarantee that motivates the method; Sec. 4.2 identifies the quantity that decides it. Appendix L extends the survey.

The upper-bound half of our theory belongs to the lineage of telescoped one-step Bellman residuals, and we claim the instantiation, not the technique: rollout's sequential-improvement condition (Bertsekas et al., 1997; Bertsekas, 2022) is our $\eta_\tau = 0$; parallel rollout (Chang et al., 2004) minimizes over per-policy values as $Q^j_{\min}$ does over per-trajectory values; and relaxed DP (Lincoln & Rantzer,

2006; Grüne & Rantzer, 2008), approximate-DP error propagation (Bertsekas & Tsitsiklis, 1996; Munos & Szepesvári, 2008), economic and suboptimal MPC bounds (Grüne, 2013; Angeli et al., 2012; Pannocchia et al., 2011; Allan et al., 2017), ILC (Amann et al., 1996; Owens & Hätönen, 2005; Bristow et al., 2006) and learned terminal values for online planning (Lowrey et al., 2019; Bhardwaj et al., 2021; Karnchanachari et al., 2020) are of the same shape. What we add is specific to LMPC's data structure. First, the τ-neighbourhood framing: a sampled safe set has measure zero, so certificates must survive on the tube the constraint enforces. Second, *lower* bounds per evaluation family, in particular the cross-iteration disparity G_j of Prop. 6(ii), which grows *because the method succeeds*. Third, the separation of consistency from drive and achievability, which answers why an evaluation can satisfy every residual condition and still not improve.

3 Problem Formulation

3.1 System, task, and safe set

Consider the discrete-time nonlinear system

$$x_{t+1} = f(x_t, u_t), \qquad t \in \mathbb{N}, \tag{1}$$

with f locally Lipschitz, subject to $x_t \in \mathcal{X}$ and $u_t \in \mathcal{U}$ for all t, with $\mathcal{X} \subset \mathbb{R}^n, \mathcal{U} \subset \mathbb{R}^m$ compact.

Assumption 1 (Iterative task)**.** *The system repeatedly steers from $x_S \in \mathcal{X}$ to a terminal equilibrium $x_F \in \mathcal{X}$, iteration j completing in finite time T^j. An iteration is* successful *if it reaches x_F with all constraints satisfied; only successful iterations are stored.*

Definition 1 (Sampled safe set and empirical value)**.** $\mathcal{SS}^j = \bigcup_{i\le j}\bigcup_{t=0}^{T^i}\{x_t^i\}$ *over successful iterations, and for $x \in \mathcal{SS}^j$, $Q^j(x) = \min\{\sum_{k=t}^{T^i} \ell(x_k^i, u_k^i) : i \le j,\ x_t^i = x\}$, where $\ell \ge 0$ vanishes only at $(x_F, 0)$.*

Write $g_t^i = \sum_{k=t}^{T^i} \ell(x_k^i, u_k^i)$ for the realized cost-to-go of stored state x_t^i, strictly decreasing in t along each stored trajectory. Q^j is defined only on $\mathcal{SS}^j$, whereas (4) below evaluates a terminal cost at a predicted state not generally stored, so every implementation commits to an extension of Q^j off $\mathcal{SS}^j$. The two we compare follow, with $\lambda > 0$ a distance-to-cost weight and $p(\cdot)$ the position components.

Definition 2 (Pooled and per-trajectory evaluation)**.** *Let $(\hat{\imath}(x), \hat{t}(x)) \in \arg\min_{i\le j,\, t} \left\|p(x) - p(x_t^i)\right\|$ index the nearest stored state over the* pooled *point cloud, and let $t_i(x) \in \arg\min_t \left\|p(x) - p(x_t^i)\right\|$ index the nearest stored state* within *trajectory i alone. Define*

$$Q^j_{\mathrm{pool}}(x) = g^{\hat{\imath}(x)}_{\hat{t}(x)} + \lambda \left\|p(x) - p(x^{\hat{\imath}(x)}_{\hat{t}(x)})\right\|, \qquad Q^j_{\min}(x) = \min_{i\le j}\Big[\, g^i_{t_i(x)} + \lambda \left\|p(x) - p(x^i_{t_i(x)})\right\| \Big]. \tag{2}$$

The distinction is order: $Q^j_{\min}$ takes the minimum Definition 1 asks for, restricted to be well posed off $\mathcal{SS}^j$; Q^j_{pool} takes the nearest neighbour first and inherits whichever iteration it came from. The third evaluation we measure is the classical *convex safe set* (Rosolia & Borrelli, 2017b), the linear program

$$Q^j_{\mathrm{hull}}(x) \;=\; \min_{\alpha \ge 0}\Big\{\; \textstyle\sum_{i,t} \alpha_t^i\, g_t^i \;:\; \sum_{i,t} \alpha_t^i = 1,\ \sum_{i,t} \alpha_t^i\, p(x_t^i) = p(x) \;\Big\}, \tag{3}$$

defined wherever the program is feasible, i.e. on $\mathrm{conv}(p(\mathcal{SS}^j))$. Nonlinear implementations restrict the program per query to the K nearest stored states of each stored lap (the *local* convex safe set, Rosolia & Borrelli, 2021); we measure both, completed off the hull by the nearest-neighbour fallback implementations use.

3.2 The optimal control problem

At iteration j and time t, with terminal tolerance $\tau \ge 0$, the controller solves

$$\min_{U_t}\ \textstyle\sum_{k=0}^{N-1} \ell(x_{k|t}, u_{k|t}) + \widehat{Q}^{\,j-1}(x_{N|t}) \tag{4a}$$

$$\text{s.t. } x_{k+1|t} = f(x_{k|t}, u_{k|t}), \quad x_{0|t} = x_t, \tag{4b}$$

$$x_{k|t} \in \mathcal{X},\ u_{k|t} \in \mathcal{U},\ \ k = 0, \dots, N-1, \qquad x_{N|t} \in \mathcal{X}, \quad d(x_{N|t}, \mathcal{SS}^{j-1}) \le \tau, \tag{4c}$$

with control law $\pi(x_t) = u^*_{0|t}$ and $d(x, \mathcal{SS}) = \min_{s\in\mathcal{SS}} \|p(x) - p(s)\|$.

The terminal cost $\widehat{Q}^{j-1}$ is where this paper's result lives: an extension of Definition 1 off the stored set, instantiated as one of the evaluations above; the terminal *constraint* uses the ordinary geometric distance under every choice, so the arms differ in the cost alone. The outer loop (Algorithm 1, Appendix M) is standard LMPC: a backup input u_b is applied—recorded and reported—when the solver returns nothing, and every successful trajectory is appended to $\mathcal{SS}$. The implementation relaxes (4c) by a bounded, exactly penalized slack that coincides with the hard constraint whenever the hard problem is solvable (Appendix M); the obstacle scenarios of Appendix I add a control-barrier-function layer (supplementary material), and the results here concern obstacle-free operation.

4 METHOD

4.1 THE COST-TO-GO MUST BE A MINIMUM OVER TRAJECTORIES

What separates the two evaluations in (2) is the *size* of their discontinuities: both are piecewise smooth with jumps where the relevant nearest-neighbour index changes, and only one can bound the jump. Fix the stored data and let $h_s = \max_{i,t} \left\|p(x^i_{t+1}) - p(x^i_t)\right\|$ be its spatial sampling step and $\ell_{\max} = \max_{i,t} \ell(x^i_t, u^i_t)$ its largest stage cost.

Proposition 1 (Hop-freeness). *Along any path, $Q^j_{\min}$ is λ-Lipschitz between jumps, and at any point where a single index map $t_i(\cdot)$ changes to an* adjacent *index its jump is at most $\ell_{\max}$. Both bounds are independent of j. For Q^j_{pool} the jump at a boundary between stored states x^i_t and $x^{i'}_{t'}$ of different iterations is $\left|g^i_t - g^{i'}_{t'}\right|$, which no stored quantity bounds and which grows as later iterations improve.*

The proof is two lines (Appendix A); the adjacency hypothesis fails where a stored trajectory approaches itself (Remark 2), and Appendix D finds the residual it predicts.

Proposition 2 (Monotone in accumulated experience). *$Q^{j+1}_{\min}(x) \le Q^j_{\min}(x)$ for every x and every admissible trajectory appended. No such statement holds for Q_{pool}: there are stored configurations and states x with $Q^{j+1}_{\text{pool}}(x) > Q^j_{\text{pool}}(x)$, and the gap can be made arbitrarily large.*

The positive half is immediate: more experience helps. The negative half is a two-state construction (Appendix A), realized whenever a later, faster lap passes near a query at a position it reaches *earlier*: on a circuit, along the whole track.

Proposition 3 (Dominance and placebo). *$Q^j_{\min} \le Q^j_{\text{pool}}$ pointwise, with equality where the trajectory of the pooled nearest neighbour attains the minimum; at $j = 1$ the two are identical.*

Dominance says the pooled lookup systematically *over-states* the cost-to-go (at full scale on real stores, Sec. 5.2); the $j = 1$ half fixes in advance a condition of zero effect, used throughout as a pre-registered *placebo* check. Computationally, $Q^j_{\min}$ needs one k-d query per stored trajectory instead of one globally, a bounded constant factor per MPC solve (Appendix M).

4.2 WHAT THE IMPROVEMENT GUARANTEE NEEDS OFF THE STORED SET

The classical argument (Rosolia & Borrelli, 2017a) steps the terminal state down its own stored trajectory, exact only *on* $\mathcal{SS}^{j-1}$; the closed loop reaches only the τ-neighbourhood $\mathcal{N}_\tau = \{x \in \mathcal{X} : d(x, \mathcal{SS}^{j-1}) \le \tau\}$ that (4c) enforces, so the argument must run through the extension $\widehat{Q}^{j-1}$ there; one quantity captures what it needs.

Definition 3 (Consistency leak). *For an extension $\widehat{Q}$ of Q^j and a tolerance $\tau \ge 0$,*

$$\eta_\tau(\widehat{Q}) \;=\; \sup_{x \in \mathcal{N}_\tau} \; \inf_{u \in \mathcal{U} \,:\, f(x,u) \in \mathcal{N}_\tau} \Big[\, \ell(x,u) + \widehat{Q}(f(x,u)) - \widehat{Q}(x) \,\Big]_+, \tag{5}$$

the amount by which one step of progress can fail to buy one step of value, with the convention $\inf_\emptyset = +\infty$, so a tube state from which no admissible input returns to the tube makes η_τ infinite; finiteness is itself a hypothesis.

Theorem 4 (Iterative improvement up to the leak). *Let $\widehat{Q}^{j-1} \ge 0$, let $\Delta_{j-1} = \sup_{x \in \mathcal{SS}^{j-1}} \left[\widehat{Q}^{j-1}(x) - Q^{j-1}(x)\right]_+$, and let $\eta_\tau(\widehat{Q}^{j-1}) \le \eta < \infty$. If iteration j of the closed loop under* (4) *completes, it satisfies*

$$J^j(x_S) \;\le\; J^{j-1}(x_S) \;+\; T^j \eta \;+\; \Delta_{j-1}. \tag{6}$$

At $\eta = \Delta_{j-1} = 0$ this recovers the classical guarantee. The leak enters multiplied by the iteration length, so the bound is informative only while $T^j\eta$ stays below the cost it bounds. The overshoot Δ_{j-1} enters once, not per step: zero for every lookup whose candidate set contains a zero-distance stored state, nonzero for the pooled lookup at co-located positions and for least-squares regressors, which interpolate *through* the stored values (measured in Appendix J).

Remark 1 (What an iteration actually pays)**.** *Step 2 of the proof telescopes the one-step residual of* (5) *at the predicted terminal states the loop actually visits, so* (6) *holds with $T^j\eta$ replaced by the sum of those realized residuals; the supremum in Def. 3 bounds that sum* before *the loop runs. No sampled statistic bounds η_τ from either side—finitely many inputs over-estimate the inner infimum, finitely many states under-estimate the supremum—so Sec. 5.2 reports sampled statistics with both bias directions stated, every arm at the same samples and inputs.*

Bound (6) compares iteration j only with its immediate predecessor, permitting a slow upward walk; because the store never discards, the comparison can be made against the best iteration so far at no cost.

Theorem 5 (Record-relative improvement)**.** *Under the hypotheses of Thm. 4, every iteration $j \geq 1$ that completes satisfies*

$$J^j(x_S) \;\leq\; \min_{0\leq k<j} J^k(x_S) \;+\; T^j\eta \;+\; \Delta_{j-1}. \tag{7}$$

Only Step 3 changes: any earlier iteration's stored inputs serve as well as iteration $j-1$'s, since $\mathcal{SS}^k \subseteq \mathcal{SS}^{j-1}$; the cost cannot walk away from its own record, and no monotonicity of $\widehat{Q}$ is assumed.

It remains to bound the leak of each evaluation; one assumption carries the positive result.

Assumption 2 (Transferability and index locality)**.** *Distances between states are position distances, $\mathrm{d}(x,y) = \|p(x) - p(y)\|$—the metric that the evaluations* (2) *and the constraint* (4c) *already use. There are $\kappa \geq 1$ and $\kappa_u \geq 0$ such that for every $x \in \mathcal{N}_\tau$ and every stored x_t^i with $\mathrm{d}(x, x_t^i) \leq D$:*

(a) (transfer that returns to the tube) *some admissible u gives $y = f(x,u) \in \mathcal{N}_\tau$ with $\mathrm{d}(y, x_{t+1}^i) \leq \kappa\,\mathrm{d}(x, x_t^i)$ and $\left\|u - u_t^i\right\| \leq \kappa_u\,\mathrm{d}(x, x_t^i)$;*

(b) (index locality) *whenever $\mathrm{d}(y, x_{t+1}^i) \leq \kappa D$, either $t_i(y) \geq t+1$, or $g^i_{t_i(y)} \leq g_t^i - \ell(x_t^i, u_t^i)\,\nu^{-1}$, where $\nu = \max_{i,i'} \delta_i/\delta_{i'}$ compares the stored sampling steps δ_i of different iterations.*

Every clause is measurable rather than postulated: in (a), κ is a one-step incremental-reachability constant, bounded analytically by $1 + \Delta t(1+\bar{v}) = 1.4$ for the bicycle and measured below that bound (Appendix H). The return-to-tube clause asks of the transfer input only what (4c) asks of the closed loop; input proximity compares a Lipschitz stage cost across the transfer, vacuous for minimum time ($\Lambda_\ell = 0$). Clause (b) says a stored trajectory does not approach itself—the geometry Prop. 1 also needs—or that the stored value still falls by $\ell\nu^{-1}$ when the index slips.

Remark 2 (On closed circuits the constant-leak claim is per-arc)**.** *Clause (b) fails, visibly, at a closed circuit's start/finish line, where a stored lap approaches itself and the within-trajectory index can jump from the end of the lap back to its beginning. On circuits the constant bound of Prop. 6(i) therefore holds per arc between self-approach points rather than globally, for* every *position-indexed arm, $Q^j_{\min}$ included; Appendix C measures the whole-lap residuals this predicts. Phase-indexed candidate selection removes the self-approach by construction (Rosolia & Borrelli, 2021); our racing loops' curvilinear indexing (Sec. 5.3) is one instance.*

Proposition 6 (The leak of each evaluation)**.** *Let Λ_ℓ be a Lipschitz constant for ℓ and let D bound the distance from $\mathcal{N}_\tau$ to a stored trajectory attaining the minimum in* (2)*. Under Assumption 2,*

(i) $\eta_\tau(Q^j_{\min}) \leq \lambda(\kappa-1)D + \Lambda_\ell(1+\kappa_u)D + \ell_{\max}(1-\nu^{-1})$*. No factor depends on j: λ, κ, κ_u are design and plant constants, D is bounded by the extent of the admissible set, and $\nu \leq \bar{v}/v_{\min}$;*

(ii) Q^j_{pool} *admits no matching bound by design constants. Along any realized transition $x \to y = f(x,u)$ within $\mathcal{N}_\tau$ whose pooled indices come from different iterations $i \neq i'$, the realized one-step residual of Remark 1 is at least $g^{i'}_{t'} - g^i_t - \lambda\tau - \ell_{\max}$: of order the cross-iteration disparity $G_j = \max\{|g^i_t - g^{i'}_{t'}| : i \neq i',\ \left\|p(x^i_t) - p(x^{i'}_{t'})\right\| \leq \tau\}$ in the worst orientation. No*

design constant fixed before the loop runs bounds G_j, which grows exactly when later iterations improve;

(iii) Q^j_{hull} *is undefined on* $\mathcal{N}_\tau \setminus \mathrm{conv}(p(\mathcal{SS}^j))$*; under the convention of Def. 3, one tube state off the hull makes* $\eta_\tau(Q^j_{\mathrm{hull}}) = +\infty$*. Implementations complete it with a nearest-neighbour fallback, and it is that completed arm we measure.*

The contrast between (i) and (ii) is the paper's mechanism: the per-trajectory minimum leaks with the distance from the query to the experience pricing it, which the track bounds; the pooled lookup pays, per crossing, how much faster later iterations became—exactly what accumulating experience increases. Statement (ii) bounds the realized residuals of Remark 1's path sum, not η_τ itself; whether any admissible input avoids the crossing is empirical, answered per arm by the input-searched statistic of Sec. 5.2.

Corollary 7 (Convergence into the leak band)**.** *Suppose every iteration completes with, for all j, $\widehat{Q}^j \geq 0$, $\eta_\tau(\widehat{Q}^j) \leq \eta$, $\Delta_j \leq \Delta$, and $T^j \leq \bar{T}$. Then $m_j = \min_{k \leq j} J^k$ converges to $m^\star = \inf_j J^j$, and*

$$m^\star \;\leq\; \liminf_j J^j \;\leq\; \limsup_j J^j \;\leq\; m^\star + \bar{T}\eta + \Delta, \tag{8}$$

so $\{J^j\}$ is asymptotically confined to a band of width $\bar{T}\eta + \Delta$ above its infimum. If moreover $\eta_\tau(\widehat{Q}^j) \to 0$ and $\Delta_j \to 0$, then $J^j \to \inf_j J^j$. If in addition $\widehat{Q}^j \leq \widehat{Q}^{j-1}$ pointwise, the value itself converges: $\widehat{Q}^j \downarrow \widehat{Q}^\infty$ pointwise.

Prop. 6 settles the width (for the lookup arms, $\Delta = 0$): part (i) gives $Q^j_{\min}$ an η free of j, so $\bar{T}\eta$ is a constant of plant and design; part (ii) puts a floor of order G_j, non-decreasing in j, on the pooled loop's per-step price wherever a transition crosses iterations, so no j-uniform η exists and (8) certifies nothing.

Three remarks bound what the corollary says. First, what stops the drift is a finite leak, not value monotonicity: (7) never uses $\widehat{Q}^j \leq \widehat{Q}^{j-1}$. Second, reaching the infimum needs $\eta_\tau \to 0$, which Prop. 6(i) reduces to $D \to 0$ and $\nu \to 1$; Sec. 5.3 measures the trend, not the limit. Third, neither statement asserts that iteration j *completes*; completion is reported empirically.

4.3 Consistency is necessary and not sufficient

Theorem 4 bounds degradation and is silent on improvement: a value that promises exactly what the ego is doing zeroes η_τ and is inert; two further requirements follow. Write $F(x) = \min_{i \leq j} g^i_{t_i(x)}$ for the lowest cost-to-go any stored trajectory offers near x.

Definition 4 (Drive and achievability)**.** $\widehat{Q}$ *has* drive $\zeta(x^i_t) = \left[g^i_t - \widehat{Q}(x^i_t)\right]_+$ *at a stored state x^i_t—how far it undercuts the trajectory that state sits on; and $\widehat{Q}$ is* achievable *if $\widehat{Q} \geq F$ everywhere.*

The floor F is built from the stored data alone—the lowest promise any stored trajectory can back near x—so it is a yardstick for every evaluation, not a feature of one; that it is $Q^j_{\min}$ stripped of its distance term is definitional, not circular. Drive is what improvement runs on; achievability bounds it, a value below F being priced for motion the system has never performed. Tables report drive as the share $\sum_{i,t} \zeta(x^i_t) / \sum_{i,t} \left[g^i_t - F(x^i_t)\right]$ over stored states with positive denominator: 100% exposes everything the store contains, 0% hides it, above 100% is achievability failing. The three properties do not replace the leak theory: consistency is what Thm. 4 needs; drive and achievability predict which consistent evaluations improve.

Proposition 8 (What each evaluation has)**.** *(i) $Q^j_{\min} \geq F$ everywhere, under no hypothesis; and $Q^j_{\min}(x^i_t) \leq g^i_t$ with equality exactly where no other stored trajectory offers less nearby. It is achievable by construction and has drive wherever improvement is available.*

(ii) $Q^j_{\mathrm{pool}}(x^i_t) = g^i_t$ *at every stored state whose position no other iteration shares (at a position stored by several iterations—the start state is one every lap—the zero-distance tie resolves to one of the co-located costs-to-go), so its drive vanishes wherever experience is not literally co-located.*

Table 1: **The consistency leak, its certificate estimate, and the three properties.** Measured on the store one minimum-time loop of Table 2 generated for itself ($38.0 \to 12.4$ s over nine iterations), $\tau = 0.05$, same store and samples per row (protocol: Appendix H). Leak: mean sampled residual of Def. 3 at $j = 1/3/9$; at $j = 9$ also the input-searched mean $\eta^{\inf}$ and sampled maximum. The *certificate estimate* $T^9\eta_\tau/J^9$ is a sampled statistic, not a bound (Remark 1); verdicts rest on order-of-magnitude contrasts, not thresholds. Drive: share of the stored improvement exposed (Def. 4); 'bel. F': share of tube states priced under the floor F. Hull rows include the nearest-neighbour fallback used where the program is infeasible (rates and split leaks: Appendix G).

	leak η_τ, mean			j=9		cert. est. $T^9\eta/J^9$		properties, j=9	
Terminal value $\widehat{Q}$	j=1	j=3	j=9	$\eta^{\inf}$	max	mean	max	drive	bel. F
Pooled lookup Q^j_{pool}	0.003	2.707	2.254	2.254	76.8	7.5×	256×	1.5%	0.0%
Global hull Q^j_{hull}	0.128	2.099	1.884	1.884	86.8	6.3×	289×	13.7%	17.8%
Local convex safe set	0.081	2.350	1.692	1.692	78.0	5.6×	260×	15.0%	19.2%
MLP, position input	0.088	2.679	3.390	3.390	373	11.3×	1244×	22.6%	33.2%
State-KNN min (i2LQR)	0.231	0.252	0.190	0.095	72.3	0.63×	241×	16.1%	0.5%
MLP, full state	9.406	1.446	0.914	0.660	200	3.0×	668×	2.0%	14.2%
Arc length *(unlearned)*	0.327	0.304	0.346	0.346	16.7	1.2×	56×	242%	100%
k-nearest minimum	0.402	0.290	0.066	0.066	1.15	0.22×	3.8×	94.0%	18.5%
Best iteration only	0.003	0.178	0.140	0.140	37.1	0.47×	124×	93.7%	0.0%
Per-trajectory min $Q^j_{\min}$	0.003	0.128	**0.063**	**0.063**	**0.511**	**0.21**×	**1.7**×	**94.5**%	**0.0**%

(iii) Q^j_{hull} can fall below F; arc length does wherever its reference speed exceeds a stored iteration's.

All three are immediate: (i) is $\lambda, d \geq 0$ in (2); (ii): the pooled nearest neighbour *of a stored state is that state* (co-located ties aside), so the lookup returns its own cost-to-go; (iii): a convex combination is priced by states it need not be reachable from.

5 Experiments

5.1 Setup

A bicycle kinematic model, $x = [p_x, p_y, \theta, v]^\top$, $u = [a, \delta]^\top$, $\Delta t = 0.1$ s, $v \in [0, 3]$ m/s, runs on an oval track (perimeter 41.1 m, width 2 m). Problem (4) is solved with SLSQP, shifted warm start, one retry and a backup on failure (solver success is reported separately from task completion, since the theorems concern exact optimization); per-study horizons, tolerances and penalties are in Appendix M, and all arms within a table share one pipeline and one scenario list. An *iteration* is one task execution (a *lap* on circuits, a *flight* for the quadrotors). Failure produces a long-tailed cost distribution (Fig. 2), so inference uses paired exact sign and McNemar tests on discordant pairs rather than means with standard deviations; significance statements appear without family-wise correction, and the load-bearing separations sit at $p \leq 0.005$ and survive any standard correction (Appendix F).

5.2 Experience degrades the pooled value; the minimum repairs it

The constructions the literature runs leak, and the leak grows with experience. Table 1 measures what the theory contains, on the store the LMPC loop built for itself; each family enters through the construction the cited papers run (protocol: Appendix H). Of the ten evaluations, exactly one has all three properties. The pooled lookup, both convex safe sets and the position-input MLP leak at full scale by three stored iterations and never recover. The state-aware members of the same families buy consistency and pay in drive: the i2LQR K-nearest minimum keeps its leak bounded but exposes only 16% of the stored improvement, the full-state MLP 2%, and in closed loop both barely improve on their seed lap. The per-trajectory minimum's leak instead *falls* as the store fills in, with 94% of available improvement exposed.

What the certificate estimate shows, under either statistic. The certificate columns of Table 1 convert each leak into $T^9\eta_\tau$ as a multiple of the iteration cost it bounds (what each statistic estimates: Remark 1). On this plant the next position depends on the current state alone, so for position-indexed evaluations the sampled residual *is* the inner infimum of (5); for the state-aware rows the input matters, hence $\eta^{\inf}$. Under the mean residual every published construction's certificate estimate exceeds the iteration cost it bounds (5.6–11×) against 0.21× for the per-trajectory minimum (Table 1). The sampled maximum is loose for every arm, the per-trajectory minimum's included (1.7×), so it cannot certify. What it does show is the scaling Prop. 6 predicts: the per-trajectory maximum is flat in j while the pooled maximum grows from 0.32 to 76.8, tracking G_j of part (ii); the pooled mean jumps three orders of magnitude at the *second* stored iteration—the first moment there is another iteration

Table 2: **Minimum-time LMPC loops on three plants**, arms paired by seed. 'done' counts iterations reaching the goal inside the step budget; 'final', the mean cost of the last completed iteration, is truncated at early deaths, so descriptive only. Inference uses the paired columns, exact sign tests against $Q^j_{\min}$: iterations survived, and cost at the last iteration *both* arms completed (final costs outright would flatter whichever arm died first). In the MuJoCo block 'final' is the mean *best* (record) lap and the right-hand test compares records, the quantity Thm. 5 bounds (Appendix N).

	iterations		cost		survived		last common	
Terminal value $\widehat{Q}$	done	of	$j{=}1$	final	W–L	p	W–L	p
Kinematic bicycle, oval circuit, 10 *loops, lap time in s, iteration* $0 = 38.26$								
Per-trajectory min $Q^j_{\min}$	**80**	80	13.15	12.38		—		—
Pooled lookup Q^j_{pool}	32	80	13.15	21.50	9–0	**0.0039**	8–0	**0.0078**
Global hull Q^j_{hull}	54	80	13.15	14.82	5–0	0.063	6–0	**0.031**
Local convex safe set	69	80	13.14	12.51	2–0	0.50	5–0	0.063
State-KNN min (i2LQR-like)	80	80	37.84	37.29	0–0	1	10–0	**0.0020**
Planar quadrotor, point to point, 12 *loops, task cost, iteration* $0 = 32.77$								
Per-trajectory min $Q^j_{\min}$	**96**	96	15.70	16.40		—		—
Pooled lookup Q^j_{pool}	46	96	15.70	33.15	12–0	$\mathbf{4.9{\times}10^{-4}}$	9–1	**0.021**
MuJoCo quadrotor, 13 *states, closed circuit,* 12 *loops, best lap in s (*$Q^j_{\min}$ */* Q^j_{pool}*; record comparison)*								
Both arms	**96/96**	96	—	29.88/31.00	0–0	1	8–0	**0.0078**

to hop to. The remaining scalings the bound predicts are confirmed in Appendix C: the contrast persists to the loop's own $\tau = 0.30$, where every position-indexed maximum is dominated by the self-approach of Remark 2, and the leak is affine in λ as Prop. 6(i) predicts ($R^2 = 0.995$).

Each published choice gives up a property, and the three-way tension is visible. The pooled lookup keeps achievability but has a drive of 1.5%—Prop. 8(ii) says exactly zero on the stored set—so whatever improvement it obtains comes from the stage cost, not the learned terminal value. Dominance (Prop. 3) is visible at full scale: over 20 000 sampled states the pooled value never falls below the per-trajectory minimum and matches it at only 19.4%. Both convex safe sets and the position-input MLP give up all three properties. The unlearned arc-length value is the degenerate corner, all drive and no floor; the state-aware pair is consistent and too inert to exploit the store. The k-nearest ablation isolates the operative property: it leaks at a *single* stored iteration where every nearest-neighbour arm barely does (Table 1), so what matters is one candidate per stored iteration, not geometric proximity. Two design changes the frame rules out (a speed-aware state metric; discarding stale iterations), the second plant and three further track geometries are in Appendix B.

Does it change the closed loop? We run the minimum-time form (Rosolia & Borrelli, 2021): stage cost constant, the horizon sum dropping out, the objective *is* the terminal value—one slow initial feasible trajectory given, every completed iteration appended. The refuter was fixed in advance: the advantage must grow with stored iterations and be *exactly* zero at one (Prop. 3); every arm gets the same seed trajectory and differs in (2) alone.

The placebo holds exactly: at one stored iteration the pooled and best-iteration arms reproduce $Q^j_{\min}$'s trajectory on all 22 seeds (and on all 52 paired studies in the paper). From the second iteration on the arms separate: under $Q^j_{\min}$ every one of 176 iterations completes and lap time falls monotonically, $38.26 \to 12.38$ s; under the pooled lookup most laps and flights never finish and the survivors get *worse*; the global hull sits between. The literature-faithful arms fail on the two sides Table 1 predicts: the local convex safe set is competitive until it stops completing (69 of 80), and the state-KNN minimum completes everything while improving almost nothing ($37.84 \to 37.29$ s against 12.38; 10–0, $p = 0.0020$)—consistency without drive, in closed loop.

What failure looks like, and why it is the value's fault. At two stored laps the pooled arm drives 3.5 m into a 41.1 m lap and holds still for 570 steps, with *one* solver failure in 601: the optimizer was working, minimising a value that priced the position at 105.4 where a stored lap had delivered 37.0—just ahead the pooled neighbour switches to the slow initial lap, so moving forward raises the objective. Prop. 6(ii) predicts a spurious barrier of exactly this height; the closed loop stalls against it (Fig. 1, right).

Which minimum? Three consistency-keeping minima exist: the per-trajectory minimum, the k-nearest minimum, and the best stored iteration alone. Run through the same minimum-time loops, each alternative pays where its missing property predicts: the k-nearest minimum converges 0.34 s

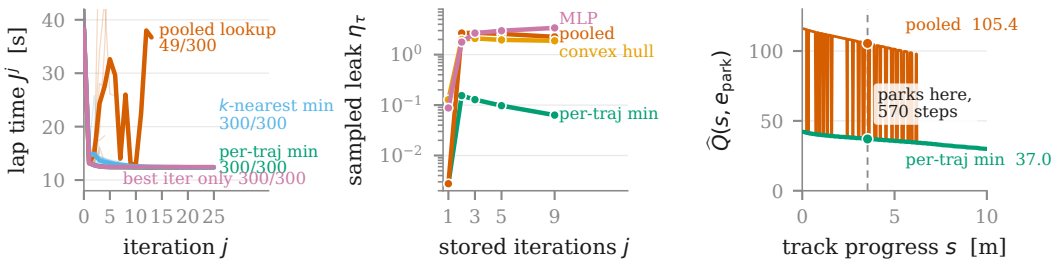


Figure 1: **Convergence, the leak that predicts it, and the failure it prices. Left:** lap time over the 25-iteration loops (12 per arm; faint: individual, heavy: mean over live loops); the pooled lookup completes 49 of 300 and wanders, the minimum-reducing evaluations all 300. **Middle:** the leak ($\tau = 0.05$, log scale) on each iteration's own store: pooled, hull and MLP grow with experience, the per-trajectory minimum *falls*. **Right:** the pooled value's spurious barrier (Prop. 6(ii)) at the stalling failure of Sec. 5.2.

worse on the bicycle (8–1, $p = 0.039$), best-iteration-only loses 18 of 96 quadrotor flights, and the per-trajectory minimum is never significantly beaten (Appendix J). Two controls bound the effect: under a tracking stage cost the ordering holds at one tenth the size (Appendix I), and a softmin that halves the pooled field's curvature is closed-loop inert—monotonicity, not conditioning, is the mechanism (Appendix E).

The third plant moves the defect to the record. On a 13-state MuJoCo quadrotor whose horizon rollout is a call to the simulator, both arms complete every one of 96 laps (Table 2); the loop wanders under either arm, so completion and last-common cost stop discriminating. What still discriminates is the record, exactly the quantity Thm. 5 bounds: the per-trajectory minimum's best lap beats the pooled lookup's on all 8 discordant seeds (8–0, $p = 0.0078$). The full study, with a point-to-point task where a velocity-aware index makes the pooled lookup safe, is Appendix N.

5.3 Does the loop converge, and into the band the leak predicts?

Eight iterations say nothing about a limit, so we ran 12 loops per arm to 25 iterations, measuring at each iteration the leak of the store it actually used (Fig. 1; Table 8 in Appendix J).

It converges, and only one arm does. The per-trajectory minimum completes every iteration and settles at 12.36 s with zero spread across loops; the pooled lookup completes 49 of 300 (12–0, $p = 4.9 \times 10^{-4}$) and wanders. Measured, the per-trajectory minimum's leak *falls* with experience ($0.129 \to 0.039$ from three to 25 stored iterations) while the pooled lookup's rises and stays high (Fig. 1, middle). We claim only that it is not increasing.

What the bands certify. The tail-window instantiation brackets the per-trajectory plateau at $0.41\times$ under the mean residual; the uniform-in-j instantiation Cor. 7 hypothesizes is wider, 1.65–$6.1\times$ the plateau, so the a priori band exceeds the cost it brackets under every statistic but the most favourable, and we report all (Table 8). What survives is the contrast Prop. 6 buys—the pooled band is two to three orders of magnitude wider than the task—and the record bound, held at all 349 completed iterations. Counting is the wrong statistic for that bound: by how *often* iterations fall behind the record the arms do not separate (4–6, $p = 0.75$); by how *far* they do—regressions beyond one control step occur in none of the per-trajectory loops and 9 of 12 pooled loops (9–0, $p = 0.0039$; Table 8; Appendix D shows the same reversal from another instrument).

6 Discussion, Limitations and Outlook

- Every plant is simulated; hardware may expose failure modes that simulation does not.
- Every closed loop rests on the solver mostly executing solutions, not backups; once backups reach a few percent, survival stops separating while cost still does (Appendix K).
- The band of Cor. 7 is bounded, not claimed tight (1.65–$6.1\times$ the plateau under its own uniform-in-j hypothesis); the cross-arm contrast and the leak's j-scaling, not the band's width, carry the comparison.
- The leak is measured up to $\tau = 0.30$ only, at $\tau = 0.05$ for Table 8's band, never for the MuJoCo store, and on Cartesian indexings where the racing loop is curvilinear: extrapolations in τ, statistics of the instrument.
- The safe set records *states*, not conditions; indexing by condition is the natural next step.

Reproducibility Statement

System models, track geometries, costs and controller parameters are given in Sec. 5 and Appendix M; every theorem is proved in Appendix A, and every experiment runs from fixed seeds on a laptop CPU. The complete code is in the supplementary material and at `https://anonymous.4open.science/r/lmpcleak-54F8`, one script per reported table or figure, with a `README` mapping each claim to the script and stored result it is recomputed from. Verification scripts re-derive the reported numbers from stored results and, when this source is present, check that each appears verbatim in it; the `README` states which appendix claims are reproducible from single scripts but not machine-checked.

AI Use Statement

In this work, we used generative AI tools for editing the manuscript for readability and for reviewing code, both tasks with recommended disclosure under the ICLR 2027 AI Policy for Authors. We have not used generative AI tools for any task with required disclosure: no synthetic data was generated, and the mathematical claims, proofs, methodology, experimental design, implementation of the methods and interpretation of the results are the authors' own. Every AI-suggested edit was reviewed by the authors before acceptance, and every code-review comment was checked against the code and its outputs by the authors; the reported numbers are recomputed from stored results by the verification scripts described above. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Douglas A Allan, Cuyler N Bates, Michael J Risbeck, and James B Rawlings. On the inherent robustness of optimal and suboptimal nonlinear MPC. *Systems & Control Letters*, 106:68–78, 2017.

Notker Amann, David H Owens, and Eric Rogers. Iterative learning control for discrete-time systems with exponential rate of convergence. *IEE Proceedings—Control Theory and Applications*, 143(2): 217–224, 1996.

David Angeli, Rishi Amrit, and James B Rawlings. On average performance and stability of economic model predictive control. *IEEE Transactions on Automatic Control*, 57(7):1615–1626, 2012.

Yuxuan Bao, Yu Kang, Dawei Shi, Zhenyi Zhou, Junjie An, and Yusuf Al-Turki. Value approximator-based learning model predictive control. *IEEE Transactions on Automatic Control*, 69(10): 7020–7027, 2024.

Dimitri P Bertsekas. *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Athena Scientific, 2022.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

Dimitri P Bertsekas, John N Tsitsiklis, and Cynara Wu. Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, 3(3):245–262, 1997.

Mohak Bhardwaj, Sanjiban Choudhury, and Byron Boots. Blending MPC & value function approximation for efficient reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Douglas A Bristow, Marina Tharayil, and Andrew G Alleyne. A survey of iterative learning control. *IEEE control systems magazine*, 26(3):96–114, 2006.

Hyeong Soo Chang, Robert Givan, and Edwin KP Chong. Parallel rollout for online solution of partially observable Markov decision processes. *Discrete Event Dynamic Systems*, 14(3):309–341, 2004.

Lars Grüne. Economic receding horizon control without terminal constraints. *Automatica*, 49(3): 725–734, 2013.

Lars Grüne and Anders Rantzer. On the infinite horizon performance of receding horizon controllers. *IEEE Transactions on Automatic Control*, 53(9):2100–2111, 2008.

Shih-Ping Han and Olvi L Mangasarian. Exact penalty functions in nonlinear programming. *Mathematical Programming*, 17(1):251–269, 1979.

Wataru Hashimoto and Kazumune Hashimoto. Multi-modal learning model predictive control. *arXiv preprint arXiv:2510.00410*, 2025.

Wataru Hashimoto, Kazumune Hashimoto, Masako Kishida, and Shigemasa Takai. Reference-free iterative learning model predictive control with neural certificates. *IET Control Theory & Applications*, 2025.

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296, 2020.

Napat Karnchanachari, Miguel de la Iglesia Valls, David Hoeller, and Marco Hutter. Practical reinforcement learning for MPC: Learning from sparse objectives in under an hour on a real robot. In *Learning for Dynamics and Control (L4DC)*, pp. 211–224, 2020.

Eric C Kerrigan and Jan M Maciejowski. Soft constraints and exact penalty functions in model predictive control. In *Control 2000 Conference, Cambridge*, pp. 2319–2327, 2000.

Fionna B. Kopp and Francesco Borrelli. Data-driven multi-modal learning model predictive control. In *IEEE Conference on Decision and Control (CDC)*, 2024.

Guanrui Li, Alex Tunchez, and Giuseppe Loianno. Learning model predictive control for quadrotors. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5872–5878, 2022.

Bo Lincoln and Anders Rantzer. Relaxing dynamic programming. *IEEE Transactions on Automatic Control*, 51(8):1249–1260, 2006.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations (ICLR)*, 2019.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9:815–857, 2008.

Siddharth H. Nair and Francesco Borrelli. Robust output-lifted learning model predictive control. *arXiv preprint arXiv:2303.12127*, 2023.

Siddharth H Nair, Ugo Rosolia, and Francesco Borrelli. Output-lifted learning model predictive control. *IFAC-PapersOnLine*, 54(6):365–370, 2021.

Siddharth H. Nair, Charlott Vallon, and Francesco Borrelli. Multi-objective learning model predictive control. *arXiv preprint arXiv:2405.11698*, 2024.

David H Owens and Jari Hätönen. Iterative learning control—an optimization paradigm. *Annual Reviews in Control*, 29(1):57–70, 2005.

Gabriele Pannocchia, James B Rawlings, and Stephen J Wright. Conditions under which suboptimal nonlinear MPC is inherently robust. *Systems & Control Letters*, 60(9):747–755, 2011.

Hannes Petrenz, Johannes Köhler, and Francesco Borrelli. Robust adaptive learning model predictive control for linear uncertain systems. In *IEEE Conference on Decision and Control (CDC)*, 2025.

Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017a.

Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks: A computationally efficient approach for linear system. *IFAC-PapersOnLine*, 50(1):3142–3147, 2017b.

Ugo Rosolia and Francesco Borrelli. Minimum time learning model predictive control. *International Journal of Robust and Nonlinear Control*, 31(18):8830–8854, 2021.

Ugo Rosolia, Xiaojing Zhang, and Francesco Borrelli. Robust learning model-predictive control for linear systems performing iterative tasks. *IEEE Transactions on Automatic Control*, 67(2): 856–869, 2022.

Pierre OM Scokaert, David Q Mayne, and James B Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999.

Haoru Xue, Edward L Zhu, John M Dolan, and Francesco Borrelli. Learning model predictive control with error dynamics regression for autonomous racing. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13250–13256. IEEE, 2024.

Zirui Zang, Ahmad Amine, Nick-Marios T Kokolakis, Truong X Nghiem, Ugo Rosolia, and Rahul Mangharam. SIT-LMPC: Safe information-theoretic learning model predictive control for iterative tasks. *IEEE Robotics and Automation Letters*, 11(1):986–993, 2026.

Yifan Zeng, Suiyi He, Hai Nguyen, Yihan Li, Zhongyu Li, Koushil Sreenath, and Jun Zeng. i2lqr: Iterative lqr for iterative tasks in dynamic environments. In *IEEE Conference on Decision and Control (CDC)*, 2023.

Yifan Zeng, Yihan Li, Suiyi He, Koushil Sreenath, and Jun Zeng. Iteraoptiracing: An iterative optimal trajectory planner for competitive autonomous racing. *arXiv preprint arXiv:2507.09714*, 2025.

APPENDIX (SUPPLEMENTARY MATERIAL)

Reviewers are not required to read this material; every main-text claim cites its appendix evidence in place.

A PROOFS

Throughout, $\ell \geq 0$ and $\widehat{Q}^{\,j-1} \geq 0$ (a normalization consistent with an approximation of a non-negative cost-to-go).

A.1 PROOF OF PROPOSITION 1 (HOP-FREENESS)

Write $q_i(x) = g^i_{t_i(x)} + \lambda\, d_i(x)$ with $d_i(x) = \min_t \left\|p(x) - p(x^i_t)\right\|$, so that $Q^j_{\min} = \min_{i \leq j} q_i$ by (2).

Lipschitz part. Each d_i is a minimum of finitely many 1-Lipschitz functions of $p(x)$, hence 1-Lipschitz. On any connected region where the index maps $t_1, \ldots, t_j$ are all constant, each q_i differs from λd_i by a constant and is therefore λ-Lipschitz. Since $|\min_i a_i - \min_i b_i| \leq \max_i |a_i - b_i|$ for finite families, $Q^j_{\min}$ is λ-Lipschitz there. The bound involves neither j nor $|\mathcal{SS}^j|$.

Jump part. Let $x^\star$ be a point where exactly one index map changes, say t_i from t to an adjacent index $t \pm 1$, the others being locally constant. At such a point d_i is continuous—it is the minimum over trajectory i's states, and both candidates realize it at $x^\star$—so the only discontinuity in q_i comes from the stored cost-to-go, and

$$\left|g^i_t - g^i_{t\pm1}\right| \;=\; \ell\big(x^i_{t\wedge(t\pm1)},\, u^i_{t\wedge(t\pm1)}\big) \;\leq\; \ell_{\max},$$

by the telescoping definition $g^i_t = \sum_{k=t}^{T^i} \ell(x^i_k, u^i_k)$. Applying the same $\max_i$ inequality, $Q^j_{\min}$ jumps by at most $\ell_{\max}$, again independently of j.

The pooled lookup. At a point on the Voronoi boundary between pooled states x^i_t and $x^{i'}_{t'}$ the distance term of Q^j_{pool} is likewise continuous, and its jump equals $\left|g^i_t - g^{i'}_{t'}\right|$ exactly. When $i \neq i'$ this is the difference between two iterations' realized costs-to-go at neighbouring positions, which is bounded by no quantity appearing in ℓ, λ or h_s: it is determined by how much iteration i' improved on iteration i, and grows as the method succeeds. □

Remark (where the hypothesis fails). The adjacency hypothesis holds wherever the nearest-neighbour index within a trajectory advances one step at a time, which is the generic situation for a trajectory that does not approach itself. It fails at self-approach: on a closed circuit, a query near the start/finish line has candidates from both the beginning and the end of the same stored lap, and t_i can jump from $\approx T^i$ to ≈ 0, moving $g^i_{t_i}$ by nearly the whole lap cost. The bound is then $\max_t g^i_t$ rather than $\ell_{\max}$. This is not an artifact of the analysis, and it is visible when the store exercises it: on the independently built store the same walk has $Q^j_{\min}$ rising at 5.4–6.9% of steps with a largest jump of 54.84, localized at that point. On the self-generated store of Table 4 the tie at the line resolves the other way and the largest rise along the walk is 0.12; yet once the sampling ball is wide enough to span the start/finish gap, the same store exercises the bound too—at $\tau = 0.30$ the sampled maximum of Sec. 5.2's protocol reaches roughly a full stored-lap cost for every arm, per-trajectory minimum included. The failure mode is real, and whether a given store and tolerance exercise it is exactly the geometry this remark names. A task whose stored trajectories are simple arcs from x_S to x_F has no such point; a repeated circuit does, and reporting it is more useful than assuming it away.

A.2 PROOF OF PROPOSITION 2 (MONOTONE IN ACCUMULATED EXPERIENCE)

Appending a trajectory enlarges the index set of the minimum in (2) from $\{1, \ldots, j\}$ to $\{1, \ldots, j{+}1\}$ while leaving every q_i, $i \leq j$, unchanged—each depends only on trajectory i's own stored states. A minimum over a superset cannot exceed a minimum over a subset, so $Q^{j+1}_{\min} \leq Q^j_{\min}$ pointwise. □

For the pooled lookup, take $\lambda > 0$, $\varepsilon > 0$ and $G > 1 + \lambda\varepsilon$. Let iteration 1 store a single state at position p_0 with $g^1 = 1$, and let x satisfy $\|p(x) - p_0\| = \varepsilon$, so $Q^1_{\text{pool}}(x) = 1 + \lambda\varepsilon$. Let iteration 2 store a single state at position $p(x)$ with $g^2 = G$. The pooled nearest neighbour of x is now that state at distance 0, so $Q^2_{\text{pool}}(x) = G > Q^1_{\text{pool}}(x)$, and $G - Q^1_{\text{pool}}(x)$ is unbounded above. By contrast $Q^2_{\min}(x) = \min\{1 + \lambda\varepsilon,\, G\} = Q^1_{\min}(x)$. □

The configuration is not contrived. It is realized whenever a later, faster lap passes close to a query point at a task position that lap reaches *early*—so its cost-to-go there is large—while an earlier lap passes slightly farther away at a position it reaches late. On a repeated circuit with laps of differing length this occurs along the whole track, which is why the effect in Table 4 is not confined to isolated states.

A.3 Proof of Proposition 3 (dominance)

Let $(\hat{\imath}, \hat{t})$ index the pooled nearest neighbour of x, so that $\left\|p(x) - p(x_{\hat{t}}^{\hat{\imath}})\right\| \le \left\|p(x) - p(x_t^i)\right\|$ for every stored (i,t). Restricting that inequality to $i = \hat{\imath}$ shows $x_{\hat{t}}^{\hat{\imath}}$ is also the nearest stored state within trajectory $\hat{\imath}$ alone, i.e. $t_{\hat{\imath}}(x) = \hat{t}$ (up to ties, which both evaluations break with the same query on the same tree). Hence the $\hat{\imath}$ term of the minimum in (2) is

$$g_{t_{\hat{\imath}}(x)}^{\hat{\imath}} + \lambda \left\|p(x) - p(x_{t_{\hat{\imath}}(x)}^{\hat{\imath}})\right\| \;=\; g_{\hat{t}}^{\hat{\imath}} + \lambda \left\|p(x) - p(x_{\hat{t}}^{\hat{\imath}})\right\| \;=\; Q_{\mathrm{pool}}^{j}(x),$$

and $Q_{\min}^{j}(x) = \min_{i \le j}[\cdot] \le Q_{\mathrm{pool}}^{j}(x)$. Equality holds iff the minimum is attained at $i = \hat{\imath}$. □

Coincidence at one iteration. For $j = 1$ the pooled point cloud is trajectory 1's stored states and nothing else, so $\hat{\imath}(x) = 1$ and $\hat{t}(x) \in \arg\min_t \left\|p(x) - p(x_t^1)\right\| = t_1(x)$ for every x; the minimum in $Q_{\min}^{j}$ is over the singleton $\{1\}$. Both expressions in (2) reduce to $g_{t_1(x)}^{1} + \lambda \left\|p(x) - p(x_{t_1(x)}^{1})\right\|$. Ties in either $\arg\min$ are broken by the same rule in both, since it is the same query on the same tree. □

A.4 Proof of Theorem 4 (iterative improvement up to the leak)

Throughout, J_t^* denotes the optimal value of (4) at time t of iteration j, U_t^* a minimizer, $x_{k|t}^*$ its predicted states, and $u_t = u_{0|t}^*$ the applied input, so $x_{t+1} = x_{1|t}^*$.

Step 1: one step of the closed loop. Fix $\varepsilon > 0$. The terminal constraint places $x_{N|t}^* \in \mathcal{N}_\tau$. Since $\eta_\tau(\widehat{Q}^{j-1}) \le \eta < \infty$, the inner feasible set of (5) at $x_{N|t}^*$ is nonempty (else its infimum would be $+\infty$ by convention), and since the infimum need not be attained, pick an admissible u with $f(x_{N|t}^*, u) \in \mathcal{N}_\tau$ and

$$\ell(x_{N|t}^*, u) + \widehat{Q}(f(x_{N|t}^*, u)) - \widehat{Q}(x_{N|t}^*) \;\le\; \eta + \varepsilon. \tag{9}$$

Consider the shifted candidate at $t+1$, $\tilde{U}_{t+1} = (u_{1|t}^*, \ldots, u_{N-1|t}^*, u)$. Its first $N-1$ predicted states are $x_{2|t}^*, \ldots, x_{N|t}^*$, which satisfy the state and input constraints because U_t^* did, and its terminal state is $f(x_{N|t}^*, u) \in \mathcal{N}_\tau \subseteq \mathcal{X}$, so $\tilde{U}_{t+1}$ is feasible. Its cost is

$$J(\tilde{U}_{t+1}) \;=\; J_t^* - \ell(x_t, u_t) - \widehat{Q}(x_{N|t}^*) + \ell(x_{N|t}^*, u) + \widehat{Q}(f(x_{N|t}^*, u)) \;\le\; J_t^* - \ell(x_t, u_t) + \eta + \varepsilon$$

by (9). Optimality at $t+1$ gives

$$J_{t+1}^* \;\le\; J_t^* - \ell(x_t, u_t) + \eta + \varepsilon. \tag{10}$$

Step 2: telescoping. Summing (10) over $t = 0, \ldots, T^j - 1$ leaves $J_{T^j}^* \le J_0^* - \sum_{t=0}^{T^j-1} \ell(x_t, u_t) + T^j(\eta + \varepsilon)$. Since $\ell \ge 0$ and $\widehat{Q} \ge 0$ we have $J_{T^j}^* \ge 0$, so the realized iteration cost obeys $J^j(x_S) = \sum_{t<T^j} \ell(x_t, u_t) \le J_0^* + T^j(\eta + \varepsilon)$, and letting $\varepsilon \downarrow 0$, $J^j(x_S) \le J_0^* + T^j\eta$.

Step 3: initialization. At $t = 0$, $x_0 = x_S = x_0^{j-1}$. The first N stored inputs of iteration $j-1$ reproduce $x_1^{j-1}, \ldots, x_N^{j-1}$, which satisfy every constraint by Assumption 1, and $x_N^{j-1} \in \mathcal{SS}^{j-1} \subseteq \mathcal{N}_\tau$, so they are feasible for (4) at $t = 0$. (If $T^{j-1} < N$, pad with the equilibrium input at x_F, which is admissible and adds stage cost zero since ℓ vanishes at $(x_F, 0)$; the padded terminal state is $x_F \in \mathcal{SS}^{j-1}$.) Hence

$$J_0^* \le \sum_{k=0}^{N-1} \ell(x_k^{j-1}, u_k^{j-1}) + \widehat{Q}^{j-1}(x_N^{j-1}) \le \sum_{k=0}^{N-1} \ell(x_k^{j-1}, u_k^{j-1}) + g_N^{j-1} + \Delta_{j-1} = J^{j-1}(x_S) + \Delta_{j-1},$$

the middle inequality because $x_N^{j-1} \in \mathcal{SS}^{j-1}$ gives $\widehat{Q}^{j-1}(x_N^{j-1}) \le Q^{j-1}(x_N^{j-1}) + \Delta_{j-1}$ by the definition of Δ_{j-1}, together with $Q^{j-1}(x_N^{j-1}) \le g_N^{j-1}$. Combining with Step 2 gives (6). □

A.5 Proof of Theorem 5 (record-relative improvement)

Steps 1 and 2 of the proof of Theorem 4 are unchanged and give $J^j(x_S) \le J_0^* + T^j\eta$, with J_0^* the optimal value of (4) at $t = 0$ of iteration j. It remains to bound J_0^* by *every* earlier realized cost rather than only by J^{j-1}.

Fix k with $0 \le k < j$. Iteration k was successful, so by Assumption 1 it starts at $x_0^k = x_S$ and its stored states and inputs satisfy every state and input constraint. Its first N inputs $(u_0^k, \dots, u_{N-1}^k)$ therefore reproduce $x_1^k, \dots, x_N^k$ admissibly, and $x_N^k \in \mathcal{SS}^k \subseteq \mathcal{SS}^{j-1} \subseteq \mathcal{N}_\tau$, the inclusion because Definition 1 takes a union over $i \le j$ and the store never discards. That input sequence is thus feasible for (4) at $t = 0$ of iteration j, whence

$$J_0^* \;\le\; \sum_{n=0}^{N-1} \ell(x_n^k, u_n^k) + \widehat{Q}^{j-1}(x_N^k) \;\le\; \sum_{n=0}^{N-1} \ell(x_n^k, u_n^k) + g_N^k + \Delta_{j-1} \;=\; J^k(x_S) + \Delta_{j-1},$$

the middle inequality because $x_N^k \in \mathcal{SS}^{j-1}$ gives $\widehat{Q}^{j-1}(x_N^k) \le Q^{j-1}(x_N^k) + \Delta_{j-1}$, together with $Q^{j-1}(x_N^k) \le g_N^k$, which holds because x_N^k occurs in the store with realized cost-to-go g_N^k and Definition 1 minimizes over occurrences. (Padding as in Step 3 covers $T^k < N$.) Since $k < j$ was arbitrary, $J_0^* \le \min_{k<j} J^k(x_S) + \Delta_{j-1}$. □

A.6 Proof of Corollary 7 (convergence into the leak band)

$m_j = \min_{k \le j} J^k$ is non-increasing by construction and bounded below by 0, so it converges, necessarily to $m^\star = \inf_j J^j$. Theorem 5 with $T^j \le \bar{T}$ and $\Delta_{j-1} \le \Delta$ gives $J^j \le m_{j-1} + \bar{T}\eta + \Delta$; taking $\limsup$ and using $m_{j-1} \to m^\star$ yields $\limsup_j J^j \le m^\star + \bar{T}\eta + \Delta$, while $J^j \ge \inf_k J^k = m^\star$ for every j gives $\liminf_j J^j \ge m^\star$. Together these are (8), and subtracting them bounds the width of the band by $\bar{T}\eta + \Delta$.

If $\eta_j := \eta_\tau(\widehat{Q}^{j-1}) \to 0$ and $\Delta_j \to 0$, the same inequality reads $J^j \le m_{j-1} + \bar{T}\eta_j + \Delta_{j-1}$, so $\limsup_j J^j \le m^\star$, which with $\liminf_j J^j \ge m^\star$ forces $J^j \to m^\star = \inf_j J^j$.

Finally, if $\widehat{Q}^j \le \widehat{Q}^{j-1}$ pointwise then for each fixed x the sequence $\widehat{Q}^j(x)$ is non-increasing and, being bounded below by 0, converges; its pointwise limit is $\widehat{Q}^\infty = \inf_j \widehat{Q}^j$. For $Q^j_{\min}$ both hypotheses hold—monotonicity by Prop. 2 and non-negativity by $Q^j_{\min} \ge F \ge 0$ (Prop. 8(i))—and for Q^j_{pool} the first fails by Prop. 2. □

A.7 Proof of Proposition 6 (the leak of each evaluation)

(i) Per-trajectory minimum. Fix $x \in \mathcal{N}_\tau$ and let (i^*, t^*) attain the minimum in (2), so $Q^j_{\min}(x) = g_{t^*}^{i^*} + \lambda d^*$ with $d^* = \mathrm{d}(x, x_{t^*}^{i^*}) \le D$. Assumption 2(a) supplies an admissible u with $y = f(x,u) \in \mathcal{N}_\tau$, $\mathrm{d}(y, x_{t^*+1}^{i^*}) \le \kappa d^*$ and $\|u - u_{t^*}^{i^*}\| \le \kappa_u d^*$; membership $y \in \mathcal{N}_\tau$ makes u feasible for the inner infimum of (5), so the residual at u upper-bounds that infimum. Since $\kappa d^* \le \kappa D$, Assumption 2(b) applies to y, and in *either* branch of its disjunction

$$g_{t_{i^*}(y)}^{i^*} \;\le\; g_{t^*}^{i^*} - \ell(x_{t^*}^{i^*}, u_{t^*}^{i^*})\,\nu^{-1} :$$

if $t_{i^*}(y) \ge t^* + 1$ this holds because g^{i^*} is decreasing in its index and $g_{t^*+1}^{i^*} = g_{t^*}^{i^*} - \ell(x_{t^*}^{i^*}, u_{t^*}^{i^*}) \le g_{t^*}^{i^*} - \ell(x_{t^*}^{i^*}, u_{t^*}^{i^*})\nu^{-1}$ (as $\nu \ge 1$); otherwise it is the second branch verbatim. Evaluating $Q^j_{\min}$ at y, keeping only the term of trajectory i^*, and using that $t_{i^*}(y)$ *minimizes* the within-trajectory distance (so $\mathrm{d}(y, x_{t_{i^*}(y)}^{i^*}) \le \mathrm{d}(y, x_{t^*+1}^{i^*}) \le \kappa d^*$),

$$Q^j_{\min}(y) \;\le\; g_{t_{i^*}(y)}^{i^*} + \lambda\,\mathrm{d}(y, x_{t_{i^*}(y)}^{i^*}) \;\le\; g_{t^*}^{i^*} - \ell(x_{t^*}^{i^*}, u_{t^*}^{i^*})\nu^{-1} + \lambda\kappa d^*.$$

Hence

$$\ell(x,u) + Q^j_{\min}(y) - Q^j_{\min}(x) \;\le\; \big[\ell(x,u) - \ell(x_{t^*}^{i^*}, u_{t^*}^{i^*})\big] + \ell_{\max}(1 - \nu^{-1}) + \lambda(\kappa - 1)d^*.$$

The bracket is at most $\Lambda_\ell(\mathrm{d}(x, x^{i^*}_{t^*}) + \|u - u^{i^*}_{t^*}\|) \le \Lambda_\ell(1+\kappa_u)d^*$ by Assumption 2(a), and vanishes identically when ℓ is constant, as for a minimum-time objective. Taking the supremum over $x \in \mathcal{N}_\tau$ and using $d^* \le D$ gives the bound. The ν term is the price of pricing one iteration's step against another's: a faster stored iteration covers the task in fewer, longer steps, so its cost-to-go falls more slowly per unit of distance, and a step along a slower one genuinely buys less than a full stage cost of its value. It is bounded by the plant, since $\nu \le \bar{v}/v_{\min}$, and does not grow with j.

(ii) Pooled lookup. Let $x \to y = f(x,u)$ be any realized transition with $x, y \in \mathcal{N}_\tau$ whose pooled nearest neighbours come from different iterations: $\widehat{Q}(x) = g^i_t + \lambda\,\mathrm{d}(x, x^i_t)$ and $\widehat{Q}(y) = g^{i'}_{t'} + \lambda\,\mathrm{d}(y, x^{i'}_{t'}) \ge g^{i'}_{t'}$ with $i \ne i'$, both distances at most τ by tube membership. Then

$$\ell(x,u) + \widehat{Q}(y) - \widehat{Q}(x) \;\ge\; g^{i'}_{t'} - g^i_t - \lambda\tau - \ell_{\max}$$

after using $\ell \ge 0$; when the crossed pair's disparity is near G_j in the unfavourable orientation ($g^{i'}_{t'} - g^i_t \approx G_j$), the realized residual is at least $G_j - \lambda\tau - \ell_{\max}$, which is the claim. This bounds the summands of Remark 1's path sum, not η_τ: at a given x another admissible successor might stay in iteration i's cells. No quantity in Definition 1 fixed before the loop runs constrains G_j: it is a difference of costs-to-go of *distinct* iterations, and by Assumption 1 later iterations are admitted precisely when they improve, so the set over which the maximum is taken only grows with j. Whether the loop encounters such crossings is empirical: pairs from different iterations within τ of each other rise from 7.4% to 26.1% of $\mathcal{SS}^j$ between two and nine stored iterations here, and the input-searched statistic of Sec. 5.2 measures, per arm, whether any candidate input avoids the price.

(iii) Convex safe set. $Q^j_{\mathrm{hull}}(x)$ is the value of a linear program whose feasible set is $\{\alpha \ge 0 : \sum_i \alpha_i = 1,\ \sum_i \alpha_i p(x^i_t) = p(x)\}$, which is empty exactly when $p(x) \notin \mathrm{conv}(p(\mathcal{SS}^j))$. A tube state off the hull therefore has $\widehat{Q}$ undefined; under the convention $\inf_\emptyset = +\infty$ of Definition 3, taking the supremum over such states gives $\eta_\tau(Q^j_{\mathrm{hull}}) = +\infty$, and any nearest-neighbour completion is a different evaluation, which we measure as its own arm. □

B Ablations and the Second Plant

The three studies summarized in Sec. 5.2, in full. The first two were predicted from the three-property frame before they were run and are reported whichever way they came out; the third asks whether anything in the racing setting is doing the work instead of the argument.

Two design changes the frame rules out. Indexing by position is a second silent approximation, since Definition 1 defines $\mathcal{SS}$ as a set of *states* and two states at one position with $v = 1.0$ and $v = 2.8$ are very differently far from the goal. A metric that also sees speed—weights $(1, 1.25, 0.55)$ on position, speed and heading, derived as marginal transfer costs—should stop the pooled lookup pairing the ego with an iteration it cannot execute, and it does, cutting the leak $2.254 \to 0.061$. It buys that entirely by going inert, drive $1.5\% \to 0.1\%$: consistency alone is available at the price of learning nothing. (The same metric under the per-trajectory minimum leaks 0.233 at 89.6% drive with 0.9% of $\mathcal{N}_\tau$ below F—no better than the plain-position form on any property, so the state metric is not what $Q^j_{\min}$ needs.) Second, since the leak's residual term scales with the spread of speeds among stored iterations, discarding stale ones looks free. Trimming nine iterations to the last five, three and two leaves the leak flat ($0.065 \to 0.115$) while the drive collapses ($94.5\% \to 0.0\%$), because the improvement a terminal value can expose is exactly the spread that trimming removes.

A plant that shares no structure with the first. Two features of the racing setting could be doing the work instead of the argument, and a 6-state planar quadrotor flying point to point replaces both: it is underactuated, so Assumption 2's tracking input is a projection rather than an inversion, and its trajectories never approach themselves, so Prop. 1's adjacency hypothesis holds cleanly. Table 2 reports its LMPC loops; the three properties on the store those loops build return the verdict the bicycle does, with the pooled leak $13\times$ the per-trajectory minimum's here against $36\times$ there.

A store that is not exposed, and a refuted prediction. Before building that loop we filled the quadrotor's store from a tracking controller at rising speed limits ($111 \to 46$ steps over nine flights, at the racing store's spread); the three properties reproduce on it exactly as on an LMPC store, which rules out having tuned the evaluation and the data to each other. Because that store's flights get

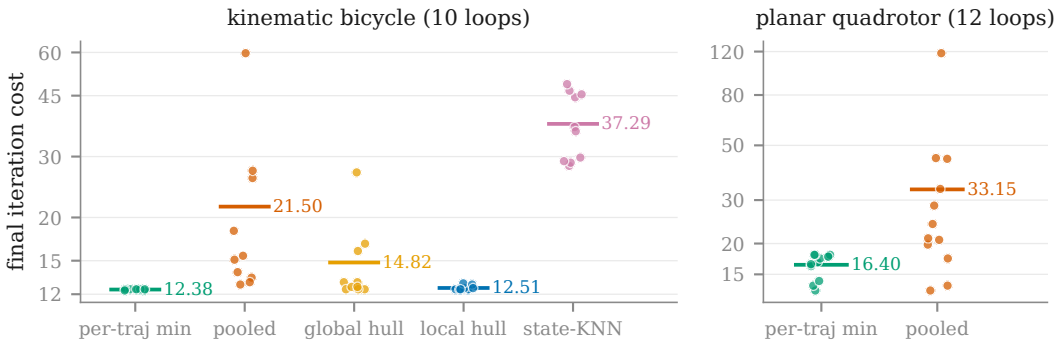


Figure 2: **Per-seed final costs behind Table 2.** One dot per seed—the cost of that arm's last completed iteration, log scale; bars are the table's means. For an arm that dies early the dot is the cost where it died, the truncation Table 2's caption flags.

faster along the *same* path, its iterations 0.8 mm apart, we predicted a closed loop on it would show no difference between evaluations—and the prediction was *wrong*: the pooled arm degrades from 13.2 to 42.6 and then completes nothing, while $Q^j_{\min}$ holds 13.5. What decides exposure is not the separation between stored iterations but the leverage the formulation gives the terminal value; we record the refuted prediction because the diagnostic it suggested is one a reader might otherwise reach for.

Three track geometries the study never trained on. The racing loops above run on one stadium oval; seeds vary the initial lap, never the topology. Three further closed tracks vary it, each star-shaped by construction so no trajectory approaches itself, keeping Prop. 1's adjacency hypothesis clean; the tightest asks hairpins 2.5× sharper than anything the oval does. The identical minimum-time loop reproduces the ordering on every geometry, reported per geometry rather than pooled (Table 3); the convex safe set's completions are worst on the hairpin track, where the curvature makes the cheaper convex mixture most unreachable.

Table 3: **Three unseen track geometries, same minimum-time loop.** 6 seeds each, 8 iterations, the same seed-lap distribution; 'done' counts completed iterations of 48, and the pooled column adds its mean final cost against $Q^j_{\min}$'s. The pooled lookup loses iterations survived 6–0 on every geometry ($p = 0.031$ each); the placebo is exact on all 18 seeds. (`param_tracks.py`, `exp_geometry.py`.)

			done, of 48		
Geometry	perimeter	min. radius	$Q^j_{\min}$	Q^j_{pool} (final vs. $Q^j_{\min}$)	Q^j_{hull}
Gentler ellipse	41.9 m	3.38 m	**48**	11 (+6.2 s)	27
Repeated chicane	40.2 m	3.01 m	**48**	14 (+6.5 s)	34
Hairpins	38.9 m	1.59 m (oval: 4.0)	**48**	18 (+9.1 s)	20

The dispersion behind Table 2. The paired tests compress each arm to one comparison per seed; Fig. 2 shows the seeds themselves. Under $Q^j_{\min}$ the ten bicycle loops finish within 0.1 s of one another; under the pooled lookup the final costs span 12.8 to 59.7 s on the bicycle and 12.9 to 118.2 on the quadrotor. The means of Table 2 summarize a failure distribution with a long tail, which is why the tests are paired.

C The Leak as a Function of τ and λ

The τ sweep, up to the tolerance the loop runs at. Across $\tau \in \{0, 0.05, 0.15, 0.30\}$—the last being the tolerance the minimum-time loops of Table 2 run at—the per-trajectory mean leak rises only from 0.061 to 0.183, while the pooled leak is a step function of whether τ exceeds the spacing between stored iterations: 0.119 on the stored states themselves, 2.0–2.5 beyond (2.254, 2.009, 2.522)—spacing the loop's own progress drives from 0.218 down to 0.064 m while τ is fixed by the plant. In certificate-estimate terms at $\tau = 0.30$, the mean-residual statistic is 0.61× the iteration cost for the per-trajectory minimum against 8.4× pooled, 7.1×/7.2× for the global/local safe sets, and 0.84× for the state-KNN minimum (0.27× after its input search). The faithful constructions track their simplifications through the sweep, not $Q^j_{\min}$. The sampled maximum at $\tau = 0.30$ is dominated,

Table 4: **The pooled lookup degrades with experience; smoothing it does not help; the per-trajectory minimum repairs it.** All three fields are evaluated on the *same* self-generated LMPC store over the same 204-point walk; only the reduction over stored states differs. "incr." is the fraction of steps on which the terminal cost rises, which for a valid cost-to-go should be zero; $|\Delta^2 Q|$ is the second difference a finite-difference Jacobian is most sensitive to—the quantity the softmin was built to reduce and the one a "bad conditioning" account appeals to. Stored lap lengths are $402, 268, \ldots, 151$ steps ($|\mathcal{SS}^j| = 402/670/898/1284/2235$): later laps are faster, which is what makes a stored state's cost-to-go iteration-dependent.

	Pooled lookup Q^j_{pool}			Pooled softmin *(control)*			Per-trajectory min $Q^j_{\min}$		
Laps	incr.	$\lvert\Delta Q\rvert$	$\lvert\Delta^2 Q\rvert$	incr.	$\lvert\Delta Q\rvert$	$\lvert\Delta^2 Q\rvert$	incr.	$\lvert\Delta Q\rvert$	$\lvert\Delta^2 Q\rvert$
1	0.0%	0.551	2.7	1.0%	0.775	5.3	0.0%	0.551	2.7
2	4.9%	5.002	219.6	19.1%	1.548	49.3	12.3%	0.200	5.8
3	5.9%	5.020	220.3	21.1%	1.371	40.2	13.7%	0.196	5.7
5	5.9%	5.020	220.3	20.1%	1.790	61.8	10.3%	0.189	4.6
9	5.9%	**5.028**	**220.5**	21.6%	2.837	111.5	8.3%	**0.184**	**4.1**
	closed loop: baseline			inert, $p = 0.45$			$p = 7.4{\times}10^{-6}$		

for every position-indexed arm, by the start/finish self-approach jump Remark 2 names: roughly a stored lap's whole cost (36.8 for $Q^j_{\min}$; 113 for the pooled lookup, which self-approaches the slow initial lap).

The λ sweep. Proposition 6(i) bounds $\eta_\tau(Q^j_{\min})$ by $\lambda(\kappa-1)D + \Lambda_\ell(1+\kappa_u)D + \ell_{\max}(1-\nu^{-1})$. For a minimum-time objective ℓ is constant, so $\Lambda_\ell = 0$ and the bound is *affine in* λ with slope $(\kappa-1)D$ and intercept $\ell_{\max}(1-\nu^{-1})$. That is a functional form, not merely a ceiling, and it is falsifiable by sweeping the one free parameter.

Sweeping $\lambda \in \{0, \frac{1}{2}, 1, 2, 4, 8, 16\}$ at nine stored iterations and $\tau = 0.05$ gives

$$\eta = 0.0038\,\lambda + 0.0566, \qquad R^2 = 0.995.$$

Both coefficients are predictions rather than fits. The intercept, 0.0566, must lie under $\ell_{\max}(1-\nu^{-1}) = 0.202$, and does. The slope must equal $(\kappa-1)D$; at the measured $\kappa = 1.28$ this implies $D = 0.014$ m, which is the right order for the separation between stored iterations, whose median is 0.064 m—though the two are different quantities and we do not claim they agree. Every λ in the swept range leaves the leak more than an order of magnitude below the pooled lookup's 2.254 at the same coverage.

A coarser instrument agrees. Walking the centreline and accumulating the total variation of each value, hops between stored iterations carry 90.2% of the pooled value's variation while being a minority of the steps: the defect is the hop, not roughness. Appendix D gives that study in full.

D The Centreline Walk in Full

The centreline walk summarized at the end of Appendix C evaluates each terminal cost at 204 probes spaced 0.20 m along the track centreline—one MPC horizon covers 1.50 m at 2.5 m/s—and counts the steps on which the value rises. It is a coarser instrument than the leak of Table 1: it walks a fixed geometric path rather than the neighbourhood the terminal constraint actually selects, and it takes no step of the plant, so it cannot see whether a step of genuine progress buys a step of value. It is correspondingly free of the tracking assumption, and it reaches the same ordering.

Three features matter more than the headline, and the first is a warning about the instrument. *The share of rising steps misleads.* By that share the pooled lookup (5.9% at nine laps) looks *better* than $Q^j_{\min}$ (8.3%), because a rise of 0.01 counts the same as a rise of 76. By size it is not close: mean variation 5.028 against 0.184 and mean curvature 220.5 against 4.1, a factor of 27 and 54. We report the reversal because the share is the statistic a reader would reach for first. *The direction:* both fields coincide exactly at one stored lap and separate from the second onward, which is Prop. 2 in measurement and explains why the defect is not folklore—it is invisible in exactly the single-iteration setting where a terminal cost is usually sanity-checked. *The attribution:* at three stored laps mean $|\Delta Q|$ is 0.549 where the pooled nearest neighbour stays within one iteration and 43.985 where it hops, so hops carry 90.2% of the variation while being a minority of steps. The defect is the hop, not roughness. The control column we return to after the closed-loop result.

E SMOOTHING THE TERMINAL MEASURE

Since margin is lost only at backup steps and those are exactly solver failures, the quantity to attack is the failure rate. A plausible cause: $d(\cdot, \mathcal{SS}^{j-1})$ is a hard minimum over a finite point set, nondifferentiable on that set's Voronoi boundaries, and the sampled cost-to-go is worse—piecewise *constant* in the nearest-neighbour index. SLSQP finite-differences both, so near a boundary the difference quotient reports a change of nearest neighbour rather than a derivative.

This is the premise the smoothing below is meant to address. Here we keep the metric and stored states exactly as they are and smooth only the reduction over them,

$$d_\beta(x, \mathcal{SS}) = -\beta \log \textstyle\sum_{i \in \mathcal{N}_k(x)} \exp\big(-\|p(x) - p(s_i)\| / \beta\big) \;+\; \beta \log k, \tag{11}$$

over the k nearest stored states, with the cost-to-go averaged under the same softmin weights. Equation (11) is C^∞, converges to $d(\cdot, \mathcal{SS})$ as $\beta \to 0$, and costs a k-nearest query in place of a 1-nearest query on the same KD-tree. At $\beta = 0.10$, $k = 8$ the mean second difference of the distance field falls $4.4\times$ ($1.98\times10^3 \to 4.52\times10^2$) and of the cost-to-go field $3.7\times$, with mean deviation from the hard distance of 0.036 m: the conditioning premise is confirmed a second time, on a measure that leaves the terminal constraint's geometry untouched.

The conclusion still does not follow. Over 64 paired runs the softmin is statistically indistinguishable from the hard minimum on completion ($51/64$ vs. $55/64$; 6 wins, 10 losses, $p = 0.45$) and its fallback rate is *higher*, not lower (5.18% vs. 4.38%). A better-conditioned terminal field is measurably smoother and closed-loop inert. So the premise holds and the remedy does not pay—evidence that finite-differencing across Voronoi boundaries is not what drives SLSQP's failures here.

An artifact of the parameterization. Because $d_\beta \geq d(\cdot, \mathcal{SS})$ by up to $\beta \log k$, the softmin constraint silently shifts the admissible true distance relative to τ; two early sweeps at settings where that admissible region was empty, or tighter than the hard constraint's, produced collapses that looked like findings about smoothing and were artifacts of this one parameter interaction. Only the uncorrected form, whose allowance brackets the hard constraint, is a fair test, and it is the one reported above.

F CLAIM ACCOUNTING

Multiplicity is stated in Sec. 5: no family-wise correction, load-bearing separations at $p \leq 0.005$ (iterations survived 12–0; step-sized regressions 9–0; the MuJoCo record 8–0; the paired which-minimum comparisons) surviving any standard correction over the full set of reported tests; results quoted with $0.01 < p < 0.05$ should be read as exploratory.

G THE CONVEX SAFE SET: IMPLEMENTATION CHECKS

The paper's largest closed-loop margin is against the classical convex safe set, and one of its cells is a collapse to a single completion of 48. That invites the obvious objection—that the baseline is misimplemented—so we state the checks that answer it. Each is a property the barycentric program $Q(x) = \min_\alpha \sum_i \alpha_i g_i$ s.t. $\sum_i \alpha_i p_i = p(x)$, $\alpha \geq 0$, $\sum_i \alpha_i = 1$ must have independently of anything we claim, and all of them hold (`test_exact_hull.py`).

- The returned multipliers lie on the simplex and reproduce the query position to LP tolerance, so the value is a genuine barycentric one and not the output of a relaxation. We stress this because a penalised relaxation of the equality—which is what a first implementation of ours did—degenerates to selecting a single vertex and is a different estimator entirely.
- At a stored state the value never exceeds that state's own cost-to-go, as $\alpha = e_i$ is always feasible.
- Feasibility agrees with an independent hull test (`scipy`'s Delaunay triangulation) on 600 probes at each coverage level.
- The k-neighbourhood program used inside the solver never reports feasible where the program over the *entire* store is infeasible, so the fallback we count is a genuine fallback and never a fabricated value.
- The infeasible set shrinks monotonically in j ($257 \to 243 \to 238 \to 230 \to 172$ of 400 probes at $1, 2, 3, 5, 11$ stored iterations), as adding states to a hull must.

Table 5: Status of each theoretical and measured claim.

Property	Status
Iterative improvement degrades by at most $T^j\eta_\tau$ under an inexact terminal value	Thm. 4; obstacle-free, exact optimization
An iteration never falls behind the *best* one so far by more than $T^j\eta_\tau$	Thm. 5; no drift ($J^j \le J^0 + \bar{T}\eta$) follows as the $k = 0$ case
The iteration cost converges into a band of width $\bar{T}\eta_\tau$ above its infimum	Cor. 7; needs a leak bounded uniformly in j: Prop. 6(i) gives this, (ii) denies it; $\eta_\tau \to 0$ is measured, not proved
$\eta_\tau(Q^j_{\min})$ is bounded independently of j, and the pooled per-crossing price is not	Prop. 6(i)–(ii) under Assumption 2, whose constants are measured ($\kappa = 1.28$, 3.6% saturation), not posited; on circuits the constant is per-arc (Remark 2); Table 1
The published evaluations leave Thm. 4 vacuous at nine iterations	Established: certificate estimates 6–11$\times$ the iteration cost by the mean residual against $0.21\times$ for $Q^j_{\min}$; the sampled maximum is loose for every arm and the arms separate by their scaling in j (Table 1)
Hop-freeness, monotonicity in experience, coincidence with Q^j_{pool} at one iteration	Props. 1–3; hop-freeness holds where a stored trajectory does not approach itself, and Table 4 shows the predicted residual where it does
$Q^j_{\min}$ improves closed-loop completion and cost	Established under a purely terminal objective: $176/176$ iterations against $78/176$, paired 21–0 over two plants (Table 2); an order of magnitude smaller under a tracking cost (Table 6)
The mechanism is monotonicity, not conditioning	Established by dissociation: a 4–10$\times$ smoother softmin leaves non-monotonicity unchanged and is closed-loop inert, $p = 0.45$ (Appendix E)
$Q^j_{\min}$ improves safety in the obstacle scenarios	*Not* established, and not expected: $6/288$ vs. $5/288$ runs with a barrier violation, all at fallback steps
The defect afflicts the convex safe set, global and local	Established for both forms: leaks growing to $1.884/1.692$ with 26/11 of 80 LMPC laps lost (Tables 1, 2)
State-aware indexings keep consistency and surrender drive	Established: the full-state K-NN minimum's leak is bounded with 16% drive, completing 80 of 80 while improving $37.84 \to 37.29$ s (10–0, $p = 0.0020$); the full-state MLP matches the pattern
The convex safe set under-states the cost-to-go at states it must reproduce	Established: strictly cheaper than the stored value at $0\%/75.8\%/69.5\%$ of stored states at 1/3/11 iterations; implementation checked against an independent hull test (Appendix G)
The defect afflicts *position-indexed* learned evaluation	Established with the scoping measured: the position-input MLP fails all three properties and completes 23 of 80 laps; a capacity control blames the data, not the optimizer; the full-state fit trades the leak for near-zero drive (Appendix J)
Among consistency-keeping minima, each alternative pays where its missing property predicts	Established per plant and horizon: 8–1 ($p = 0.039$), plateau 7–0 ($p = 0.016$), record 8–0 ($p = 0.0078$); $Q^j_{\min}$ never significantly beaten (Appendices J, N)
The separation needs the exact-optimization premise	Established by sweep: survival separation dies as the backup rate enters the circuit's 3–18% regime; cost separation persists throughout (Appendix K)
The leak is affine in λ with the predicted slope and intercept	Established: $R^2 = 0.995$ over $\lambda \in [0, 16]$; intercept under $\ell_{\max}(1 - \nu^{-1})$ (Appendix C)
A better-conditioned terminal field reduces solver failure	*Refuted*: the softmin is $4.4\times$ smoother, $p = 0.45$ on completion, fallback rate no lower (Appendix E)

What the checks also establish is the mechanism of Sec. 5.2. Equality at a stored state is *not* expected and its absence is not a defect: wherever a cheaper convex combination reaches the same position, the program is obliged to return it. The construction is working as specified. On a linear plant the combination is reached by the corresponding combination of stored inputs and the value is honest; on a nonlinear plant it is not reachable and the value is an optimism no stored experience backs. Measured at stored states, that optimism is absent with one stored iteration (0%) and present at 75.8% and 69.5% of states with three and eleven, by a mean of 8.05 and 7.78.

Where the hull rows' leak lives. The arms of Table 1 complete the program with a nearest-neighbour fallback where it is infeasible, as implementations do (23.8%/22.6% of queries, global/local, at $j = 9$). Splitting the $j = 9$, $\tau = 0.05$ sample by whether either endpoint's query touched the fallback: over the 74–76$\%$ of samples resolved entirely by the barycentric program the mean residual is 0.133 (global) and 0.150 (local), with sampled maxima of 29.3; over all samples it is 1.884 and 1.692. Most of the mean leak therefore lives at the seam between the program and its completion—a pooled-style hop the fallback inherits—while the failure that belongs to the hull itself is achievability: the below-F optimism above is measured *inside* the feasible region. Reporting the all-samples number in Table 1 is the honest choice because the closed loop crosses the seam whenever the predicted terminal state leaves the hull, which is exactly where its iterations die.

H Measurement Protocol for the Three Properties

We measure what the theory contains rather than proxies, and on the store the LMPC loop built for itself rather than one assembled to look like one.

- *Sampling.* Draw 4000 states from $\mathcal{N}_\tau$ by perturbing stored states within τ; step the true dynamics under the input a tracker would apply toward the stored successor; record the residual of (5). The stored costs-to-go come from $\ell \equiv 0.3$, so a positive residual means the value failed to fall by one step's worth after genuine progress.
- *Input search.* Every positive residual is refined: for each stored lap, track one step toward the successor of that lap's nearest stored state, and take the smallest residual over these candidates and the original input (reported as $\eta^{\inf}$). The candidate set contains the original, so refinement can only shrink; the result remains an over-estimate of the inner infimum of (5), and Remark 1 states both bias directions. The minimizing successor stays within τ of the store at 90–99$\%$ of searched samples across arms and tolerances, so the refined statistic is attained essentially without leaving the tube the theorem reasons about.
- *Tracking constants.* Tracking is measured, not assumed: $\kappa = 1.28$ against the 1.4 the model bounds analytically, with the input saturating at 3.6% of samples at $\tau = 0.05$.
- *Disclosed slices.* The sampler perturbs heading and speed only within $\pm\tau$ of a stored state, whereas $\mathcal{N}_\tau$ as defined admits arbitrary heading and speed at position distance τ, so the sample is a tube-aligned slice of the set the supremum ranges over. States are drawn with probability proportional to stored-lap length, which weights slow laps more heavily than the closed loop's own visitation does.
- *Drive and achievability* are computed on the stored states and on the same $\mathcal{N}_\tau$ samples, normalized as in Def. 4.

Replication on an independent second store. Table 1 is measured on a single store; to separate the profile from that store's idiosyncrasies we built a second one the same way—a fresh minimum-time $Q^j_{\min}$ loop from a seed outside Table 2's ten, $46.2 \to 12.4$ s over nine iterations—and re-measured the five headline arms with the identical instrument at $\tau = 0.05$. The profile reproduces on a store no number in Table 1 touched: at nine stored iterations the mean residual is 2.818 for the pooled lookup (certificate estimate $9.4\times$ the iteration cost), 2.212/2.170 for the global/local convex safe sets ($7.4\times$/$7.2\times$), 0.220, bounded throughout, for the state-KNN minimum ($0.73\times$), and 0.075, falling with j, for the per-trajectory minimum ($0.25\times$). The position-indexed maxima are *larger* here than on the first store, which is not noise but the mechanism: this store's initial lap is slower (46.2 against 38.0 s), and the cross-iteration disparity G_j of Prop. 6(ii) scales with exactly that spread.

I THE TWO REGIMES, AND THE CONDITIONING CONTROL

The same comparison under a tracking stage cost. Table 2 gives the terminal value the whole objective. Here it is instead a correction to $\ell(x,u) = 3.0\, d_{\text{lat}}^2 + 0.1\, a^2 + 0.3\, \delta^2 - 0.8\, v$ around a fixed target speed, on a store built by a speed schedule rather than by the loop—so nothing about the improvement across iterations comes from $\widehat{Q}$, and the terminal value acts only through the terminal cost of a single solve. The ordering survives with the effect an order of magnitude smaller, which is what a defect in $\widehat{Q}$ should do as $\widehat{Q}$'s leverage falls. Two further readings from it. The convex safe set completes *one* of 48 runs at three stored iterations and not for want of a solution: the solver succeeds at 99.8% of steps and its program is feasible at 98.8% of evaluations, yet those runs consume 591 of 600 permitted steps against 218. And the minimum-based arms are *not* separated from each other here at four times the data—the k-nearest minimum loses at one stored iteration (7–13) and wins at eleven (10–7), the crossover the account predicts, but Fisher's exact test gives $p = 0.19$, and the best-iteration arm is 11–9. Violating a property costs completions; the count does not resolve finer in this regime. Under the minimum-time objective it does (Appendix J): each alternative minimum pays on the plant its missing property predicts, which is the leverage account working at one scale finer than Table 1 alone could establish.

Table 6: **The closed loop under a tracking stage cost.** Left: 720 runs, 24 scenarios × 3 coverage levels × 2 obstacle geometries × 5 evaluations (obstacle machinery: supplementary material). Middle: 1600 runs at $n = 100$ over the four arms the first could not separate. Right: wins/losses against $Q^j_{\min}$ on discordant pairs at eleven iterations, exact McNemar, from the $n = 100$ study where available. Every arm shares one scenario list, the terminal constraint is hard throughout, and at j=1 the pooled and best-iteration arms are bit-identical to $Q^j_{\min}$ in both studies (Prop. 3).

	n=24, of 48			n=100, of 200		paired at j=11		
Terminal value $\widehat{Q}$	j=1	j=3	j=11	j=1	j=11	W	L	p
Pooled lookup Q^j_{pool}	41	37	40	175	162	30	8	$\mathbf{4.7{\times}10^{-4}}$
Convex safe set Q^j_{hull}	35	**1**	41	—	—	60	6	$\mathbf{2.7{\times}10^{-12}}$
k-nearest minimum	45	44	42	181	181	10	7	0.63
Best iteration only	41	44	43	175	182	11	9	0.82
Per-trajectory min $Q^j_{\min}$	41	44	**46**	175	**184**			—

How much of this is the pure terminal objective? All of the size and none of the direction: across the 2320 closed loops of Table 6 the same ordering holds with the effect cut to a lap or two, and the placebo is again exact. The defect scales with the leverage the formulation gives $\widehat{Q}$, which is what a defect *in* $\widehat{Q}$ should do, and reporting only the regime that flatters us would be reporting half of it. That study also isolates how the convex safe set fails: at a stored state, the one place it must reproduce the empirical value exactly, it returns something strictly cheaper, because a mixture reaches the same position more cheaply on paper and cannot be driven (Appendix G).

Monotonicity, not conditioning. A control run before the mechanism above was known dissociates the two: a softmin over the eight nearest *pooled* states—same states and metric, only the reduction made differentiable—halves the curvature of the pooled field and is closed-loop inert ($p = 0.45$), while $Q^j_{\min}$ cuts the same curvature 53-fold and moves the closed loop (Appendix E). Safety in the obstacle scenarios is unchanged rather than improved (6 of 288 runs with a barrier violation against 5, all at fallback steps).

J WHICH MINIMUM, AND THE LEARNED APPROXIMATOR

Three constructions reduce the stored costs-to-go by a minimum and therefore keep consistency: the per-trajectory minimum, the minimum over the k=16 nearest *pooled* states, and the nearest neighbour within the single best stored iteration. Under a tracking stage cost they were statistically inseparable at four times the data (Table 6, $p = 0.19$), which left the paper's specific prescription resting on Table 1 alone. This appendix runs both alternatives, and the learned approximator of the third family in Sec. 2, through the minimum-time loops of Table 2—the regime where the terminal value is the whole objective—on the same seeds (Table 7). The loops are deterministic given the seed trajectory

Table 7: **Which minimum? The consistency-keeping alternatives and the learned arm, on the loops of Table 2.** Same seeds, same protocol, same columns; the $Q^j_{\min}$ rows are repeated from Table 2 for reference. The k-nearest minimum and the MLP are different functions from the exact lookup already at one stored iteration, hence their differing $j{=}1$ columns; best-iteration-only coincides with $Q^j_{\min}$ there, and does, exactly, on all 22 seeds.

	iterations		cost		survived		last common	
Terminal value $\widehat{Q}$	done	of	$j{=}1$	final	W–L	p	W–L	p
Kinematic bicycle, oval circuit, 10 *loops, lap time in s*								
Per-trajectory min $Q^j_{\min}$	**80**	80	13.15	**12.38**		—		—
k-nearest min Q^j_{knn}	**80**	80	14.52	12.72	0–0	1	8–1	**0.039**
Best iteration only Q^j_{best}	**80**	80	13.15	12.43	0–0	1	4–0	0.125
Learned MLP Q^j_θ	23	80	16.65	30.50	10–0	$\mathbf{2.0{\times}10^{-3}}$	10–0	$\mathbf{2.0{\times}10^{-3}}$
Planar quadrotor, point to point, 12 *loops, task cost*								
Per-trajectory min $Q^j_{\min}$	**96**	96	15.70	**16.40**		—		—
k-nearest min Q^j_{knn}	**96**	96	13.00	16.73	0–0	1	4–6	0.75
Best iteration only Q^j_{best}	78	96	15.70	28.95	3–0	0.25	8–1	**0.039**

(a rerun of seed 0 reproduces its stored lap times exactly), so arms run later pair with the stored arms by seed index, and the verified numbers of the original arms are untouched.

The k-nearest minimum pays where iterations crowd. Its candidate set is geometric, so nothing guarantees one candidate per stored iteration; by nine iterations on the bicycle the stored laps sit 0.064 m apart and the 16 nearest states come from whichever iterations happen to be closest. It completes all 80 laps—reducing by a minimum is what survival needs—but converges 0.34 s worse at the last common iteration (8–1, $p = 0.039$). On the quadrotor, whose flights never approach one another, the comparison is flat (4–6, $p = 0.75$). One reading must be reported against us: on the quadrotor its *best* cost beats $Q^j_{\min}$'s on all 12 seeds (12.97 against 14.35 on average), driven by its first iteration (13.00 against 15.70), where its willingness to price below the stored floor F (Table 1: below F on 18.5% of $\mathcal{N}_\tau$) acts as an aggressive discount this plant happens to tolerate. The loop then never improves on its own first flight (final 16.73): the optimism buys one good iteration, not an improving sequence, and the guarantee under test is about the sequence.

Best-iteration-only pays where coverage matters. A single trajectory cannot hop (Prop. 1), so this arm is trivially consistent; what it discards is every other tube the store has visited. On the bicycle that costs nothing detectable (80 of 80 laps, final 12.43 against 12.38, 4–0, $p = 0.125$). On the underactuated quadrotor it loses 18 of 96 flights and gives up 11.65 in mean cost at the last common iteration (8–1, $p = 0.039$); its worst completed flight runs 347 steps with 25 backups against a median of 55 steps, the closed loop repeatedly failing to reach the one stored tube it is allowed to price.

Table 8: **Twenty-five-iteration minimum-time loops,** 12 **per arm.** 'plateau' averages the last five iterations of the loops that got that far (12 and 3). The band $\bar{T}\eta$ uses the leak sampled at each loop's own store, in two instantiations—*tail window* (last five iterations) and *uniform in* j ($\max_j T^j \times \max_j \eta_j$, the hypothesis of Cor. 7)—each under the mean residual and the sampled maximum (parentheses: band as a multiple of that arm's plateau).

	Per-trajectory min $Q^j_{\min}$	Pooled lookup Q^j_{pool}
iterations completed	**300** of 300	49 of 300
plateau, last five iterations (spread)	12.36 s (0.00)	29.99 s (31.5)
band, tail window (mean / max)	5.03 s (0.41×) / 59.3 s (4.8×)	1714 s (57×) / 40 208 s (1341×)
band, uniform in j (mean / max)	20.4 s (1.65×) / 74.9 s (6.1×)	2821 s (94×) / 52 133 s (1739×)
loops regressing beyond one control step	**0** of 12	9 of 12
worst regression behind the record	0.10 s (one step)	46.7 s

At twenty-five iterations the defects move from survival to the path and the limit. Run through the schedule of Sec. 5.3 on the same 12 loops, both alternatives complete all 300 iterations, as consistency predicts. The k-nearest minimum ends at the same plateau (12.40 against 12.36 s, spread 0.01) but is the last of the two to reach the record on every loop (12–0, $p = 4.9{\times}10^{-4}$), and on the way it falls behind its own record by more than one control step on 2 of 12 loops with a worst single regression of 9.70 s against the per-trajectory minimum's worst of 0.10 s—one control step. Its sampled leak stays bounded but sits above throughout ($0.188 \to 0.056$ against $0.129 \to 0.039$

from three to twenty-five stored iterations), with sampled maxima reaching 57.1 where the geometric candidate set misses the trajectory attaining the minimum. Best-iteration-only converges cleanly—no regression above one step, leak $0.131 \to 0.042$, sampled maxima never above 0.6—to the *wrong limit*: 12.43 against 12.36 s, above the per-trajectory minimum on all 7 discordant loops (7–0, $p = 0.016$). A single stored tube cannot splice the best sectors of different iterations, so its plateau is the best lap it has seen rather than the best lap the store supports.

The learned approximator fails all three properties, one of them against our prediction. The arm is the ordinary choice, not a strawman: an MLP of two tanh layers of 64 on standardized inputs and targets, trained by L-BFGS from a fixed seed on the stored (state, cost-to-go) pairs, refit from scratch at every iteration so that iteration j has seen exactly the data of iterations $\leq j$. We predicted before running that it would fail achievability—a regressor interpolates *through* the data and has no floor—and it does: 33.2% of $\mathcal{N}_\tau$ priced below F at nine iterations, by a mean depth of 3.754. We also expected its leak to be small, consistency being cheap for a smooth fit, and that was *wrong*: the leak grows with experience, $0.088 \to 3.390$ over one to nine stored iterations with the violation rate essentially constant (49.6–52.8% of sampled states), because the pooled cloud it fits becomes multi-valued in position as iterations improve and a least-squares compromise prices no stored iteration correctly—the pooled lookup's disease, inherited through the training set rather than the index map. A capacity control confirms the data and not the optimizer is at fault: quadrupling the width and training $13\times$ longer leaves the nine-iteration training RMSE pinned at 10.6–11.1 and the leak at 3.4–5.8, the irreducible conflict in the targets, while at one stored iteration, where the targets are single-valued, every variant fits to an RMSE of 0.16 or better and barely leaks *by the mean* (at most 0.109); the sampled maximum at one iteration is already 30.4, a heavy tail the mean conceals, in a paper that repeatedly warns against gentle statistics.

The input representation, not the fit, carries the defect. Refitting the same regressor on the derived state metric's coordinates (position, scaled speed and heading; Appendix B) instead of position alone removes the multi-valued conflict at its source: the full-state MLP's leak *falls* with experience (9.41 at one stored lap, where five input dimensions overfit 146 points, to 0.914 at nine, 0.660 after the input search) instead of growing. What it buys in consistency it pays in drive—2.0% of available improvement exposed, against the position fit's 22.6%—because a fit that can tell iterations apart in state space reproduces each iteration's own value rather than undercutting it. The overshoot constant of Thm. 4 makes the same point from the theory side: $\Delta = \sup_{\mathcal{SS}}[\widehat{Q} - Q]_+$, measured on the stored states, is 50.2 for the position fit at nine iterations—the regression averages across co-located fast and slow laps, so the Δ-corrected bound is vacuous—and 0.92 for the full-state fit. So the claim 'the leak afflicts learned-approximator evaluations' is, precisely, a claim about *position-indexed* regressors; richer inputs trade the leak for inertness, and the per-trajectory construction that would combine a per-lap fit with an outer minimum—the learned analogue of $Q^j_{\min}$, which our mechanism predicts would keep both—remains unmeasured.

In the loop the learned arm completes 23 of 80 laps, every seed ending in a timeout, with mean cost at each seed's last completed iteration 30.50 against 12.38 (10–0 on iterations survived and on cost at the last common iteration, both $p = 2.0\times10^{-3}$). Its $j{=}1$ column (16.65) is no placebo violation: with one stored lap the fit is already a different function from the exact lookup, which the caption of Table 7 notes.

K Sensitivity to the Solver Budget

Every closed-loop result in this paper rests on SLSQP being good enough that the loop mostly executes solutions rather than backups, and the closed-circuit experiment of Appendix N shows the arm comparison *vanishing* when that premise fails. Rather than assert where the boundary lies, we measure it: the bicycle minimum-time loop of Table 2 re-run at every combination of `maxiter` $\in \{10, 20, 40, 80\}$ and `ftol` $\in \{10^{-2}, 10^{-3}, 10^{-4}\}$, plus two beyond-grid control settings at `maxiter` $= 5$ (6 seeds each, both arms, 8 iterations), recording at each setting the backup rate and the separation. Three findings, the third against our stated prediction.

First, the defect is also a solver hazard. The per-trajectory minimum completes 48 of 48 iterations at every one of the 12 settings, with a backup rate of at most 0.17% even at an eighth of the default

iteration budget; the pooled arm's backup rate at the same settings runs 0.03% to 2.88%. The non-monotone value is harder to *optimize*, not just wrong to follow.

Second, the failure mode shifts as the premise degrades. With a strong solver the pooled loop stalls against the value's spurious barriers and dies (as few as 13 of 48 complete); with a weak one it completes more, at far worse cost. Survival stops separating the arms at the weakest settings, while cost at the last common iteration separates at *every* setting (mean gaps $+1.1$ to $+22.9$ s): survival is the statistic that fails first.

Third, the boundary is where the circuit said it was. We predicted the separation would collapse as the backup rate approached the circuit's 3–18%, and within the grid the survival statistic decays on schedule; the beyond-grid control ($\texttt{maxiter} = 5$, backup rates 2.09% and 9.91%) reproduces the circuit's null on completions exactly while the cost separation persists (6–0, $p = 0.031$). What the prediction missed: even an $8\times$ cut in iterations and a $100\times$ loosening of tolerance barely reach the regime that broke the circuit, and once it is reached the cost signal still survives at one backup per ten steps—only survival dies.

L Extended Related Work

The compressed account in Sec. 2 omits one point worth stating in full.

Iterative learning MPC, and how it evaluates its own cost-to-go. Sec. 2 lists the LMPC variants; Hewing et al. (2020); Bristow et al. (2006) survey the broader field, and Zang et al. (2026) add information-theoretic exploration. What deserves fuller statement is the status of the convex safe set. It is a *linear*-systems construction and Rosolia & Borrelli (2017b) present it as such—a convex combination of stored states is reached by the convex combination of the stored inputs and remains admissible because the constraint set is convex—with convergence proved under linear dynamics, polytopic constraints and a convex cost. Neither step is available for a nonlinear plant on a non-convex admissible set. The nonlinear implementations therefore run a *local* convex safe set—K nearest stored states per stored lap, convexified per query, often with time-varying (phase-indexed) candidate windows (Rosolia & Borrelli, 2021)—or replace convexification by selection: K candidates under a diagonally weighted full-state metric, minimal stored cost-to-go among them, with reachability screening (Zeng et al., 2023; 2025); *learned approximators* regress the stored cost-to-go on the state (Bao et al., 2024; Hashimoto et al., 2025). The position-indexed 1-NN pooled lookup of our study is the simplification of these selection rules, and Sec. 5.2 separates its failures from the families'. One work keeps the exact mixed-integer form and partitions the store per behavioural mode instead (Hashimoto & Hashimoto, 2025); mode partitioning restricts each evaluation to one iteration's data at a time, which suppresses the cross-iteration disparity we study—to our knowledge unremarked, and consistent with our mechanism. Classical soft-constraint MPC (Scokaert et al., 1999; Kerrigan & Maciejowski, 2000) supplies the exact-penalty theory the implementation's relaxed terminal constraint invokes (Appendix M).

M Implementation Details

Notation. Time indices $t, k \in \mathbb{N} = \{0, 1, 2, \dots\}$. Realized quantities carry a single subscript (x_t); predicted quantities carry a double subscript $x_{k|t}$, the state predicted k steps ahead at planning time t. Superscript j always indexes iterations. $U_t = (u_{0|t}, \dots, u_{N-1|t}) \in \mathcal{U}^N$ denotes a control sequence and $p(x)$ the position components of a state.

Stage cost. $\ell(x, u) = 3.0\, d_{\mathrm{lat}}(x)^2 + 0.1\, a^2 + 0.3\, \delta^2 - 0.8\, v$, where d_{lat} is the lateral distance to the track centerline. The negative speed term rewards progress; it is shared by *every* arm in every comparison, so no arm is handicapped by its objective.

Solver pipeline. Each step: build the warm start by shifting the previous solution (dropping the applied input, repeating the last); call SLSQP; on failure retry once from a pure-pursuit initialization with a larger iteration budget; if that also fails, apply the backup input u_b, a braking pure-pursuit law. Every arm within a given table uses the identical pipeline and tolerance, so no comparison is confounded by solver settings. Tolerances differ *between* studies according to what each probes, and we state them explicitly: $\texttt{ftol} = 10^{-3}$, $\texttt{maxiter} = 50$ for the closed-loop comparisons under a

Algorithm 1 LMPC with a per-trajectory terminal value

Require: Initial feasible trajectory; horizon N; distance weight λ; tolerance τ
1: $\mathcal{SS}^0 \leftarrow$ states of the initial trajectory; build its KD-tree
2: **for** iteration $j = 1, 2, \ldots$ **do**
3: $x_0 \leftarrow x_S$
4: **for** $t = 0, 1, \ldots$ until $x_t = x_F$ **do**
5: Solve (4) for U_t^*, with the slack-relaxed terminal constraint described below
6: **if** solver returns no solution **then** $u_t \leftarrow u_b(x_t)$ **else** $u_t \leftarrow u_{0|t}^*$
7: Apply u_t; observe x_{t+1}
8: **end for**
9: **if** iteration successful (Assumption 1) **then** $\mathcal{SS}^j \leftarrow \mathcal{SS}^{j-1} \cup \{x_0^j, \ldots, x_{T^j}^j\}$; add one KD-tree
10: **end for**

tracking stage cost (Appendix I), and $\texttt{ftol} = 10^{-4}$ with $\texttt{maxiter} = 40$–60 for the minimum-time LMPC loops, whose per-plant values are given with each study. We flag this because the terminal constraint's effect on the closed loop is genuinely tolerance-sensitive: at $\texttt{ftol} = 10^{-3}$ the terminal term is resolved too coarsely to bind reliably, an instance of the conditioning premise examined in Appendix E.

Relaxed terminal constraint. The solver operates on (4) with the terminal condition relaxed by a bounded slack, $d(x_{N|t}, \mathcal{SS}^{j-1}) \le \tau + \epsilon_t$ with $\epsilon_t \in [0, \bar{\epsilon}]$, penalized in the objective by $\rho_1 \epsilon_t + \rho_2 \epsilon_t^2$. When ρ_1 exceeds the terminal constraint's multiplier this ℓ_1 penalty is *exact* (Han & Mangasarian, 1979; Kerrigan & Maciejowski, 2000): every nearby local minimizer has $\epsilon_t^* = 0$ and solves the hard problem, so in obstacle-free operation—every experiment in the main text—the controller is pointwise the hard-terminal LMPC (4) and the theory of Sec. 4.2 applies unchanged. The slack exists for the obstacle scenarios of Appendix I, where a discrete-time control-barrier-function layer can conflict with terminal-set membership; that formulation and its guarantees (a constructive backup certificate, persistent solvability, closed-loop forward invariance) are developed in the supplementary material.

Per-study parameters. The distance weight is $\lambda = 2$ throughout (swept in Appendix C). Minimum-time loops: bicycle $N = 8$, $\tau = 0.30$, $\bar{\epsilon} = 2.0$; planar quadrotor $N = 12$, $\tau = 0.15$, $\bar{\epsilon} = 1.0$; MuJoCo quadrotor $N = 8$, $\tau = 0.60$, $\bar{\epsilon} = 1.5$; all with the ℓ_1 slack penalty alone, $\rho_1 = 20$, $\rho_2 = 0$, plus a tie-breaking input regularizer of $10^{-4} \|U_t\|^2$. In each case τ exceeds the fastest stored iteration's spatial sampling step, which Appendix N shows is necessary for the terminal constraint to chain along a fast stored tube. Tracking-cost and obstacle studies: $N = 6$, $\tau = 0.05$, $\bar{\epsilon} = 0.05$, $\rho_2 = 10^3$, $\rho_1 = 100$.

Spatial indexing. $\mathcal{SS}^j$ is indexed by a KD-tree, giving $\mathcal{O}(\log n)$ terminal-distance queries; under $Q^j_{\min}$ each appended trajectory adds one more tree.

The computational cost of $Q^j_{\min}$. $Q^j_{\min}$ needs one k-d query per stored trajectory instead of one globally, $\mathcal{O}(j \log(|\mathcal{SS}^j|/j))$ against $\mathcal{O}(\log |\mathcal{SS}^j|)$, and the measured query cost follows: $1.1\times$ at one stored iteration rising to $5.6\times$ at nine. The cost of a whole MPC solve, which is what the controller pays, does not follow, because the value also shapes the path the optimizer takes through it: on our two stores a solve ranges from $0.58\times$ to $2.2\times$ the pooled arm's, cheapest where the store is largest and SLSQP converges in fewer iterations on the better-behaved value. We report the range rather than the favourable end.

Compute. All experiments run on CPU on a standard laptop; every individual study reported here completes in under 20 minutes. Random seeds are fixed.

N The MuJoCo Quadrotor: Minimum-Time LMPC in Thirteen States

Plant and controller. MuJoCo 3.11, one free rigid body of mass $0.8\,\text{kg}$ carrying four thrust actuators at the corners of a $0.24\,\text{m}$ square, each limited to $[0, 6]\,\text{N}$. The state is 13-dimensional—position, linear velocity, unit quaternion, body angular velocity—integrated by RK4 at $0.02\,\text{s}$ with control every $0.1\,\text{s}$, so $\ell \equiv 0.3$ per control step as on the other two plants and the objective is the

terminal value alone. Hard constraints: $z \in [0.35, 3.5]$ m, tilt from vertical at most $45°$, and speed at most 4.0 m/s. The speed cap is not cosmetic: with unbounded speed the minimum-time solution is a dive, and the controller reaches the tilt limit and crashes within a second.

The controller predicts *through the simulator*. The horizon rollout is a call to MuJoCo, so the controller has no analytic model of the plant, no linearization and no hand-derived Jacobian; SLSQP differences the simulator. Nothing in the method needs a model—the store, the terminal value and the terminal constraint see only the index coordinates—and this is the cheapest available demonstration of that. Horizon $N = 8$, $\bar{\epsilon} = 1.5$, $\rho_1 = 20$, shifted warm start, one retry, and the braking backup u_b on failure.

The terminal tolerance must exceed the store's own sampling step. One implementation fact generalizes. A stored iteration is sampled at $\delta_i = v_i \Delta t$ in space, so a *fast* stored iteration is a *sparse* one: at 4 m/s and $\Delta t = 0.1$ s consecutive stored states sit 0.4 m apart. If $\tau < \delta_i$ the tube around that iteration has gaps, the terminal state cannot be chained along it, and the terminal *constraint* rather than the terminal value decides where the vehicle goes—we measured the ego held at $d = 0.288$ m against $\tau = 0.30$ with the slack identically zero and no solver failures, and the loop regressed as it got faster, which is exactly backwards. Setting $\tau = 0.60 > \max_i \delta_i$ removes the effect. This is the same quantity that appears in Prop. 6(i) as $\nu = \max_{i,i'} \delta_i / \delta_{i'}$, and it is the price of pricing one iteration's step against another's, seen from the constraint side rather than the cost side.

Point to point: a boundary, where the pooled lookup is safe. The task flies from hover at the origin to a randomized goal 13–15 m away and must *stop* there (0.25 m, 0.5 m/s), which is Assumption 1's terminal equilibrium. A position-only index cannot express it: hovering at the goal and crossing it at the speed cap are the same point in the metric and get the same cost-to-go, so the value prices an overshoot as an arrival and the loop dithers around the goal instead of stopping in it. The index map therefore carries velocity, scaled so the speed cap is on the order of the position tolerances, identically for both arms.

The result is a clean null. Over 6 loops of 8 iterations both arms complete $48/48$, both reach a best of 12.35 and a final of 12.65 from a common iteration 0 of 61.00, the paired comparison at the last common iteration is 2–2 ($p = 1$, mean gap $+0.00$), and the placebo holds on all six loops.

We report this as a boundary on the claim rather than a failure of it, because the mechanism is legible and it is the mechanism the theory names. Prop. 6(ii) needs stored states from *different* iterations that are close in the index metric and far apart in cost-to-go. Here the quantity that separates two iterations' costs-to-go—how fast they were flying—is now part of the metric that decides which is nearest, so a slow stored flight and a fast one are no longer neighbours and the pooled lookup has nothing to hop to. The prediction this makes is not vacuous: it says the pooled lookup is safe exactly when the index map already separates the iterations, and unsafe when it does not. The literature's own experience corroborates the boundary from the other side. Published rest-to-rest quadrotor LMPC selects neighbours in the *full* state—position, velocity and attitude—for the same reason our index carries velocity, and reports clean iterative improvement on hardware with no hop pathology (Li et al., 2022): on that task class, with that metric, the defect cannot be triggered, which is our account of the null, observed independently. The racing formulations, indexed by track position (Rosolia & Borrelli, 2021; Zeng et al., 2023), are where the exposure lives—and our index here, position plus *weighted* velocity, is coarser than the full state those rest-to-rest studies use, which can only make hopping easier; the null appearing anyway says the boundary is robust in the conservative direction.

A closed circuit: where the premise fails, and its repair. The null above predicts where the separation should return: a closed circuit, where a lap has no terminal rest condition, the index can be positional, and stored laps pass near one another at *different phases* of the task—the configuration Prop. 6(ii) needs. Our first attempt (a 32.2 m closed 3-D curve, Frenet indexing, tube 3.0×2.0 m) did not discriminate: SLSQP returned nothing on 3–18% of each iteration's steps, that noise is larger than any effect a terminal value has, and both arms completed all 18 laps while wandering without converging. What broke is the exact-optimization premise of Thm. 4, not the comparison, and Appendix K closes that reading quantitatively: degrading the planar loop's solver budget until its backup rate enters the same band reproduces the null on completions exactly, while the cost separation persists throughout. The repaired protocol was fixed in advance. Probe rounds against a pre-set gate (at most 2 backups per lap before any seeds are spent) selected `ftol` $= 10^{-2}$, `maxiter` $120/200$, terminal tolerance 0.9 and, decisively, a horizon of 6; move blocking, a lower speed cap and

a wider tube all made failures worse, and we record that. The confirmatory run comprised 12 seeds of 8 iterations—twice the point-to-point study's six—executed in two batches of six. The first batch's interim record comparison was 5–0 for the per-trajectory minimum ($p = 0.0625$, not significant on its own), and the run was extended to twelve rather than stopped there; we disclose the interim so the sequential structure is visible, and note that treating the two looks sequentially (a Bonferroni split of α across them) leaves the final 8–0 result significant.

The circuit, repaired: result. The premise now holds where the comparison needs it: the per-trajectory minimum's backup rate beyond the placebo lap is 0.56%, while the pooled arm's is 3.69% *at identical solver settings*—the finding of Appendix K, that the non-monotone value is also harder to optimize, reproduced at 13 states. Both arms complete 96 of 96 laps, which at that backup rate is exactly what the sweep predicts survival to show, and the cost at the last common iteration does not resolve either (7–5, $p = 0.77$): the loop wanders under both arms on this hard task. What separates is the record, the quantity Thm. 5 bounds: the best lap under the per-trajectory minimum beats the pooled lookup's on all 8 discordant seeds (8–0, $p = 0.0078$, means 29.88 against 31.00 s), and the $j{=}1$ lap is bit-identical between arms on all 12 seeds, as Prop. 3 requires. The defect's cost at 13 states is real, and it lives where the theory says it lives—in how good a lap the loop ever finds, not in whether a crippled solver finishes laps.