

Supplementary information for Excitation-Aware On-Track System Identification at Full Scale: A Simulation Study

Abstract

This document holds material moved out of the main article: the residual model’s physics-augmented inputs and objective, the temporal residual and the exact reduction of its cost, the per-phase identification timing, the axis-by-axis comparison against the baseline pipeline as published, and the supporting result figures. Nothing in the main article’s argument depends on it. Sections, equations, figures and tables are numbered with an S prefix here; references of the form “Section 5.2 of the main article” point into the main article instead.

S1 Residual Model: Physics-Augmented Inputs and Objective

S1.1 Physics-augmented inputs: the slip angles

The baseline residual reads $\mathbf{z} = [v_x, v_y, \omega, \delta]$ and predicts $[e_{v_y}, e_\omega]$. The quantity it corrects, however, reaches the plant only through the slip angles of Eq. (3) of the main article: the tire law of Eq. (4) of the main article is a function of (α, F_z) and of nothing else in $\mathbf{z}$, so the target e is, to the extent the single-track structure holds at all, a function of $(\alpha_f, \alpha_r, F_{z,f}, F_{z,r})$ plus the genuinely unmodelled terms. The network is therefore asked to reconstruct α_f, α_r from $\mathbf{z}$ before it can represent anything.

That reconstruction is the worst part of the map to leave to a small network. Eq. (3) of the main article composes a division by v_x with an arctangent; a one-hidden-layer piecewise-linear network approximates a *ratio* only by tiling the (v_y, ω, v_x) domain, and this vehicle’s identification data spans 9.2–26.5 m/s, a $2.9\times$ range of the denominator. Eight LeakyReLU units cannot tile that, so the fitted residual

represents the same tire nonlinearity independently at each speed it happened to observe, with nothing tying those representations together. The consequence is specific to this pipeline: stage 2 evaluates the network along a synthetic rollout at a single speed and a steering ramp, i.e. exactly in the region where such a residual has no reason to be consistent, and stage 3 then reads the result as tire force.

Appending α_f, α_r , computed from Eq. (3) of the main article with the *measured* geometry of Table 1 of the main article, hands the network the two arguments of Eq. (4)

of the main article directly. Three properties motivated the choice. (i) The speed dependence becomes explicit instead of learned: a residual expressed in α transfers between the recorded lap’s speed and the rollout speed by construction, which matters because Section 5.2 of the main article deliberately drives those two apart. (ii) It costs 16 weights ($58 \rightarrow 74$ parameters) and no new sensor, against the several hundred a network would need to tile the division; augmentation is the cheaper way to buy the same function. (iii) It is compatible with the symmetry argument of Section S1.2: α_f, α_r are odd in (v_y, ω, δ) , so the mirrored batch mirrors the augmented inputs too and the symmetry penalty remains exact.

The raw four inputs are retained and not replaced, because the effects the residual exists to absorb (load transfer, actuator dynamics, aerodynamic load, combined slip) are not functions of α alone, and v_x in particular carries the load-dependent part. The geometry used for the augmentation is the same l_f, l_r the fit and the force recovery use, which is why $l_{wb} \equiv l_f + l_r$ is enforced (Section 5.5 of the main article): an augmentation computed from a different wheelbase than the one Eq. (7) of the main article assumes would inject a bias exactly where the network is meant to remove one.

This applies the physics-informed idea of [1, 2] at the input rather than at the loss. An input augmentation carries no penalty weight to tune, cannot trade against the data term, and costs nothing at inference; a loss term provides the same structure only softly and only where the data supports it. Deep Dynamics [2] makes the complementary choice of constraining coefficients *inside* the network; here the network is left unconstrained and its input space is made physical instead. The augmentation is available on its own (`physics_inputs`) and combined with the temporal residual (`s4` with `use_physics_inputs`), and is rank-agnostic, so the same code path serves a flat batch and a windowed sequence batch.

S1.2 Physics-informed objective

The optional physics-informed objective [1, 2] adds three penalties to the one-step MSE,

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda_s \mathcal{L}_{\text{steady}} + \lambda_m \mathcal{L}_{\text{sym}} + \lambda_g \mathcal{L}_{\text{smooth}}, \quad (\text{S1})$$

with $\lambda_s = 0.1$, $\lambda_m = 0.05$, $\lambda_g = 10^{-4}$. Each term addresses a specific problem when training a residual on one lap of ordinary driving.

All four terms are stated in the units of the residual network’s own output, which for the two dynamics residuals means scaling by T_s/m and T_s/I_z before squaring, as Section 5.1 of the main article derives. Without that scaling λ_s is not a relative weight: a force residual squared against a one-step increment in m/s puts 99.9996% of $\mathcal{L}$ on $\mathcal{L}_{\text{steady}}$ at initialisation on a real 30 s buffer of this vehicle, against 98.53% on $\mathcal{L}_{\text{data}}$ once it is applied.

Dynamics consistency

$\mathcal{L}_{\text{steady}}$ substitutes the *corrected* next state $\mathbf{x}_{k+1} = \hat{\mathbf{x}}_{k+1} + e_k$ back into Eqs. (1)–(2) of the main article, with the axle forces re-evaluated through (4) there at the corrected slip angles, and penalises the squared lateral and yaw balance residuals. Its purpose is to keep the two residual channels consistent with each other: nothing in the data

term prevents an e_{v_y} and an e_ω that individually reduce one-step error while jointly implying an axle force pair no tire could produce, and it is that force pair, not the state prediction, that stage 3 subsequently fits. The term is applied only on a quasi-steady mask ($|\dot{v}_y| < 0.8 \text{ m/s}^2$, $|\dot{\omega}| < 6 \text{ rad/s}^2$), because Eqs. (1)–(2) of the main article neglect load transfer and transient tire relaxation and therefore hold as written only where those terms are small; penalising the balance during a transient would ask the network to absorb the model’s known omissions rather than the plant’s unmodelled dynamics.

Left/right symmetry

The vehicle is laterally symmetric, so the true residual map is odd: $e(-v_y, -\omega, -\delta) = -e(v_y, \omega, \delta)$. $\mathcal{L}_{\text{sym}}$ penalises $\|f_{NN}(\mathbf{z}) + f_{NN}(-\mathbf{z})\|^2$ over the sign-mirrored batch. A circuit is rarely balanced in its corner directions, so 30 s of lap data covers one sign of δ far better than the other, and the synthetic sweep of stage 2 is run in one direction only. The mirror supplies the missing half of the input domain from geometry and not from data, at no data cost; the same mirroring is applied to the recorded trace before windowing (Section S2), so the two uses are consistent.

Input-gradient smoothness

$\mathcal{L}_{\text{smooth}}$ penalises $\|\partial e / \partial \mathbf{z}\|^2$, a soft Lipschitz bound on the residual. Stage 2 queries the network beyond the steering range it was trained on, and Section 5.3 of the main article shows what an unconstrained extrapolation there costs. The two mechanisms are complementary and both are kept: the extrapolation bound of Section 5.3 of the main article decides *where* the sweep stops asking, and the gradient penalty decides *how fast* the residual is allowed to grow inside that range.

The nominal-model half of Eq. (S1), the nominal rollout with the mirrored inputs and the constant terms of the symmetry residual, depends only on quantities fixed for the whole training call and is precomputed once per call (Section S2.2).

S2 Temporal Residual and Its Cost

S2.1 Temporal residual

The **s4** architecture is an S4D diagonal state-space residual [3] over a sliding window of $W = 20$ samples (0.7 s at $T_s = 0.035 \text{ s}$). The length is a configured value rather than a derived one: it spans roughly 30 times the time constant of the model’s fastest lateral mode at the identification speeds, so every window contains the transient the residual has to reproduce. Windows are cut so that none straddles the seam between the recorded trace and its mirrored copy. Each block carries $H = 4$ independent single-input single-output channels of complex state dimension $N = 4$, initialised in the S4D-Lin form ($\text{Re}(A_n) = -\frac{1}{2}$, $\text{Im}(A_n) = \pi n$) that approximates the HiPPO-LegS operator [4] in closed form, with a learnable per-channel discretisation step. As stated in Section 2 of the main article, the S4-family residual overlaps with concurrent work [5] and is not claimed as a contribution; Section S2.2 reports what we did change, which is its cost. Figure S1 shows stage 1 in this configuration.

The diagram specifies the stage 1 data flow. The six inputs are the four measured signals plus the two slip angles computed from the same measured state and geometry. A window of 20 samples passes through the projected S4D channels and produces the two residual outputs. The final panel shows two evaluations of the same learned system: convolution for windowed training and recurrence for the sequential rollout. The diagram identifies the inputs, temporal context, and evaluation paths; it does not by itself establish an accuracy improvement (Figure S1).

S2.2 Evaluating the temporal residual at CPU cost

A temporal residual is only usable on-vehicle if it fits inside the re-identification period on the hardware the car carries, which is typically a CPU. Profiled as first written, the S4D path cost 125.7 ms per epoch on a GPU and 898 ms per stage-2 rollout, putting a six-iteration identification at 181.4 s, longer than the 30 s data window it consumes. Three changes remove that cost without altering the arithmetic, and none is novel as a technique. The result is a pipeline that runs in real time on commodity hardware; equivalence is verified by regression testing, not assumed.

Closed-form kernel instead of a scan

A diagonal A makes the block’s impulse response available in closed form,

$$K_l = \text{Re}(C e^{A\Delta t l} \bar{B}), \quad l = 0, \dots, W - 1, \quad (\text{S2})$$

so a length- W window is one Vandermonde product followed by a causal Toeplitz matrix product, rather than W sequential steps [3]. At $W = 20$ the dense Toeplitz product is cheaper than the FFT convolution of the original S4 [6] and than a parallel associative scan [7]; both are the right choice at sequence lengths orders of magnitude longer than this one. Because the stack’s output layer reads only the window’s last timestep, that layer contracts the reversed kernel straight onto the window and skips $W - 1$ of its outputs. Cost: 125.7 ms to 26.7 ms per epoch.

Loop-invariant physics-loss terms

Under the physics-informed objective, the nominal Pacejka rollout, the mirrored batch and the constant terms of the symmetry residual depend on the nominal model only, which is fixed for the whole training call, so they are precomputed once per call instead of once per epoch. The three forward passes the objective used to make (data, mirror, smoothness) collapse into one pass over a double-length batch, with the smoothness Jacobian taken off the same graph. Cost: 26.7 ms to 16.6 ms per epoch, with the loss value unchanged bit for bit.

The rollout belongs on a CPU

Stage 2 is 500 strictly sequential single-sample steps carrying the S4D hidden state, each with a host synchronisation. That is pure kernel-launch latency on a GPU: 898 ms on CUDA against 204 ms on a CPU copy of the same weights. The rollout is therefore run on a CPU copy whatever device training used, following the recurrent path of

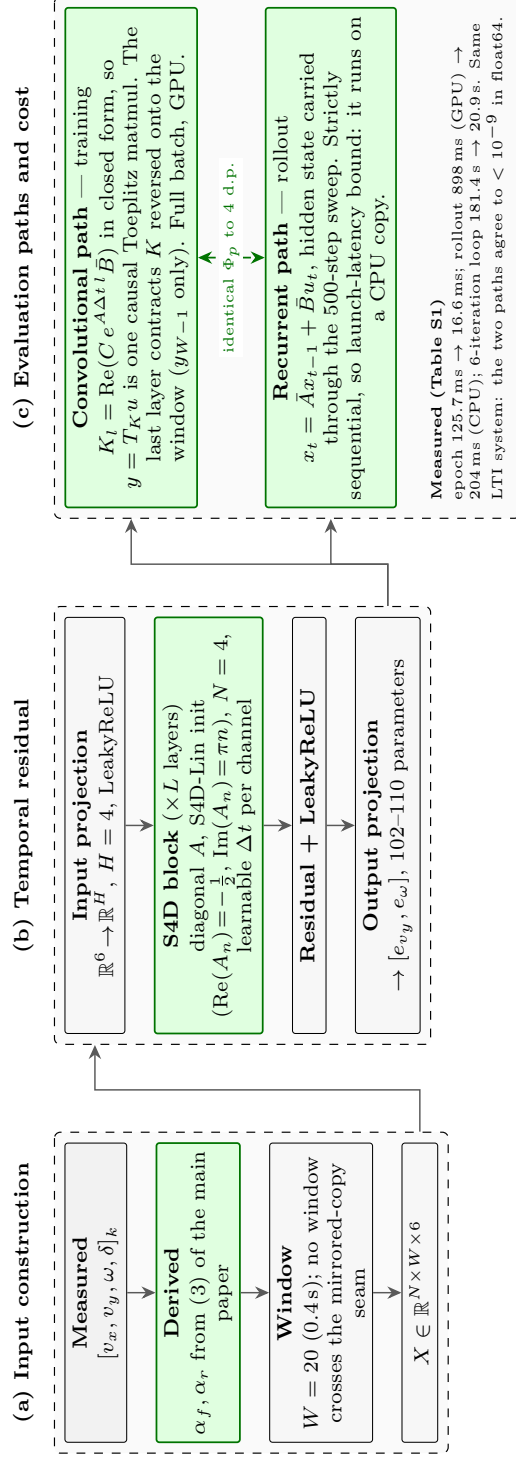


Fig. S1 Stage 1 data flow with physics-augmented inputs and the S4D temporal residual.

Figure S1(c), and only the full-batch training stays on the GPU, where it is $2.4\times$ faster than a CPU. This is the one place where the two evaluation paths of the same system have to agree exactly, and they are pinned by a regression test to $< 10^{-9}$ in double precision.

End to end this takes a six-iteration identification from 181.4s to 20.9s ($8.7\times$) on the same synthetic 30s lap, with the identified Pacejka coefficients agreeing to four decimal places (Table S1). We deliberately did *not* touch the knobs that trade accuracy for speed: epoch count, iteration count, early stopping, warm-starting the network from the previous iteration’s weights, or dropping the smoothness term. Those change the identified tire and are a modelling decision, not a free speedup.

S3 Identification Cost, per Phase

S3.1 Identification cost

Section S2.2’s claim is that the temporal residual is affordable on the hardware a vehicle carries; Table S1 is the measurement, on a synthetic 30s lap and the same six co-identification iterations throughout. The per-stage rows and the end-to-end totals were measured separately and are not a decomposition of each other: the stage rows reconstruct 156.2s of the 181.4s before and 21.1s of the 20.9s after, so the residual 25.2s of fixed cost visible before is not visible after. We report both as measured rather than reconciling them by attribution. Every step was verified against the implementation it replaced before being kept: the convolutional and recurrent paths of Figure S1(c) agree to $< 10^{-9}$ in double precision (pinned by a regression test), the rewritten physics-informed loss is bit-identical, and the full six-iteration loop lands on the same Pacejka coefficients to four decimal places, the remaining difference being float32 reordering over 1200 optimiser steps.

Table S1 Cost of one S4D identification on a synthetic 30s lap.

Stage	Device	Before	After
Training epoch: scan → kernel Eq. (S2)	GPU	125.7 ms	26.7 ms
Training epoch: physics-loss hoist and fusion	GPU	26.7 ms	16.6 ms
Stage-2 rollout, 500 sequential steps	GPU → CPU	898 ms	204 ms
Identification, 6 iterations	mixed	181.4 s	20.9 s

The optimisations are exact: no epoch count, iteration count, early-stopping rule or loss weight was changed, and the identified coefficients agree to four decimal places.

The first three rows are per-stage micro-benchmarks of the named stage; the last row is a separate end-to-end measurement of the whole six-iteration loop and includes everything the micro-benchmarks exclude: data preparation, the per-iteration network re-initialisation and the six Pacejka fits.

The two therefore do not sum: at 1200 epochs and six rollouts the stage rows account for 156.2s of the 181.4s before and 21.1s of the 20.9s after. The $8.7\times$ is the ratio of the two end-to-end measurements.

Two consequences follow. At 20.9s the identification fits inside the 30s re-identification period of Section 5.6 of the main article, so the temporal residual is a deployable option rather than an offline one; at 181.4s it was not. This table is an isolated measurement on a synthetic buffer, and the closed-loop runs publish accepted sets about 140s of simulation time apart, for the reasons Section 5.6 of the main article gives; what the 20.9s establishes is that the fit is no longer the term that dominates that cadence, not that the cadence is 30s. And the device split is not incidental: the stage-2 rollout is strictly sequential and single-sample, so it is $4.4\times$ *faster* on a CPU, while full-batch training over ≈ 3000 windows remains $2.4\times$ faster on the GPU. A pipeline pinned to one device pays one of those two penalties.

S4 The Baseline Pipeline Against This Work

S4.1 Baseline pipeline versus this work

Table S2 places the pipeline as published [8] beside the pipeline as it stands here, on the axes that changed. The residual model, the data budget, the four-stage loop and the steady-state force recovery are inherited unchanged; what differs is the configuration of the virtual-data step, the relaxation on the loop closing it, the regularisation topology, the measurement chain, and the contract at the output. Table 3 of the main article’s comparison is a subset of these axes: the rollout *speed* of Section 5.2 of the main article is taken from the measured lap in every run of Section 6 of the main article, so that comparison measures the remaining settings and not the rollout speed.

S5 Supporting Result Figures

These figures accompany Section 6 of the main article; every metric they show is also in its cross-run comparison table.

Figure S2 is the decaying-surface counterpart of the identified tire curve in Section 6.4 of the main article, and it carries two ground-truth curves rather than one because the plant is not the same tire at the two moments being compared. The corrected configuration’s last accepted set saturates at 631 N on the front axle against the 482 N the plant offers at its $\mu = 0.35$ floor, 31% high, which is the estimator’s static-surface bias plus one cycle of transport lag. The inherited set saturates far above the 721 N the plant offered at $\mu = 0.52$, the moment its run ended.

On the static $\mu = 0.70$ surface, the peak-friction errors are 0.145 and 0.142 for the corrected front and rear axles, compared with 1.300 for both axles in the inherited configuration. The figure therefore shows the same ordering on both axles, but it does not identify which component of the pipeline produced that difference (Figure S3).

On the decaying surface the same errors are 0.209 and 0.207 for the corrected front and rear axles and 1.408 for both inherited axles. The corrected figures are larger than their static-surface counterparts because the plant moves between identification cycles and the published value lags it; the inherited figures are larger because the plant moves away from a peak factor that stays on its box constraint. The bars do not separate those two causes, and Section 6.5 of the main article does (Figure S4).

Table S2 The baseline pipeline [8] against this work, on the axes that differ.

Axis	Baseline [8]	This work
Platform	1:10, 3.7 kg, $L = 0.32$ m	Full scale, 269.6 kg, $L = 1.53$ m
Rollout speed	$\text{clip}(\bar{v}_x, 2, 4)$ m/s	From the measured lap, floored at 15 m/s
Reachable μ	≈ 2.0 (adequate)	0.43 under the baseline clip; ≥ 1.2 after correction
Sweep amplitude	Fixed 0.4 rad	$1.5P_{99}(\delta_{\text{obs}})$, clipped to $[\delta_{\min}, \delta_{\max}]$
Regularisation	None; each fit is the next start point	Tikhonov to a <i>frozen</i> prior, $\lambda = 0.05$; start point still tracks
Integrator	Explicit Euler at T_s	RK4 shipped, Euler selectable
Steering signal	Command	Achieved road-wheel angle
Mass, geometry	Configuration file	Measured static wheel loads
Friction prior	—	Per-axle brush fit of (C_α, μ) from IMU and odometry, gated; applied two-sided and pinned for the per-axle fit; P_{99} of $ a /g$ as a floor-only fallback
Coefficient update	$\Phi_p \leftarrow \tilde{\Phi}_p$ each iteration	Relaxed, $\Phi_p \leftarrow (1 - \beta)\Phi_p + \beta\tilde{\Phi}_p$, $\beta = 0.20$
Solver	Trust-region, single start	Multi-start, plus simplex and global options
Output contract	Per-coefficient box constraints	Gates on derived quantities at the controller interface
Residual objective	One-step MSE	MSE plus quasi-steady balance, left/right symmetry and input-gradient penalties, Eq. (S1)
Residual inputs	State, command	<i>inherited</i> plus axle slip angles (Section S1.1)
Residual model	MLP, 58 parameters	<i>inherited</i> as default; S4D temporal residual shipped
Residual cost	Not reported	S4D identification 181.4 s $\rightarrow$ 20.9 s, exact (Table S1)
Data budget	30 s, one controller-free lap	<i>inherited</i>
Force recovery	Quasi-steady yaw balance	<i>inherited</i>
Ground truth	Steady-state manoeuvre in a large space	Simulator per-wheel force, slip and load

Rows marked *inherited* are unchanged and are listed so that the extent of the modification is unambiguous.

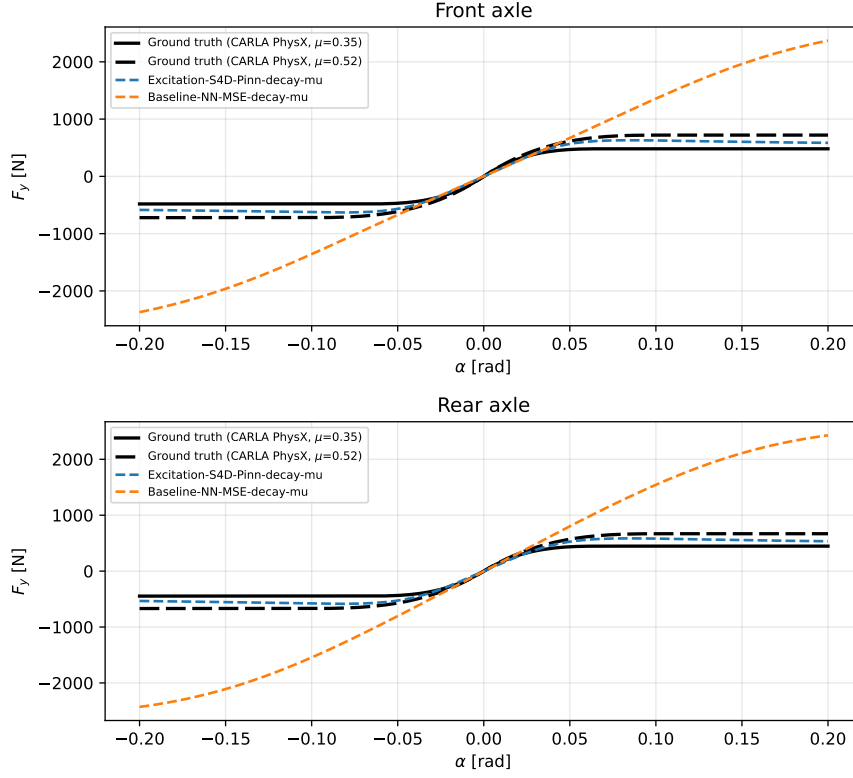


Fig. S2 Axle lateral force against slip angle on the decaying surface: the last accepted identification of each configuration against the plant's own PhysX curve at the floor the surface reaches ($\mu = 0.35$) and at the value it held when the inherited run ended ($\mu = 0.52$).

The force panels show the same ordering in two metrics. Front/rear RMSE is 27.15/33.227 N for the corrected configuration and 329.645/348.409 N for the inherited configuration. The corresponding R^2 values are 0.993/0.987 and $-0.160/-0.461$. The corrected estimates therefore have smaller errors and account for more of the measured force variation in these runs. The negative inherited R^2 values indicate performance below the constant-mean reference used for that statistic; they do not by themselves identify the source of the error (Figure S5).

For the constant surface, the force-error distributions have maximum absolute errors of 130/134 N for the corrected front/rear axles and 1249/1219 N for the inherited configuration. Their standard deviations are 27.1/32.8 N and 326.5/343.6 N, respectively. The inherited mean errors are also positive, 42.7 and 57.8 N, so its distribution has both much greater spread and a positive offset in the reported samples (Figure S6).

On the decaying surface, the corrected force RMSE is 32.883 N front and 36.469 N rear, with $R^2 = 0.984$ and 0.977. The inherited run is incomplete because the recorded comparison reports a collision; its observed RMSE is 171.429 N front and 178.341 N rear, with $R^2 = 0.708$ and 0.647. These partial-run values describe the available samples only and should not be read as a completed-lap comparison (Figures S7

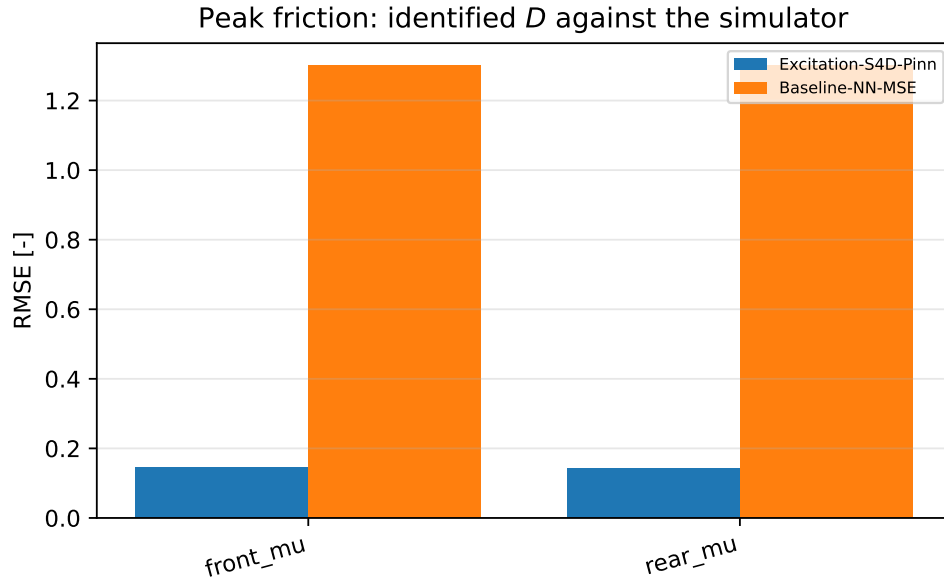
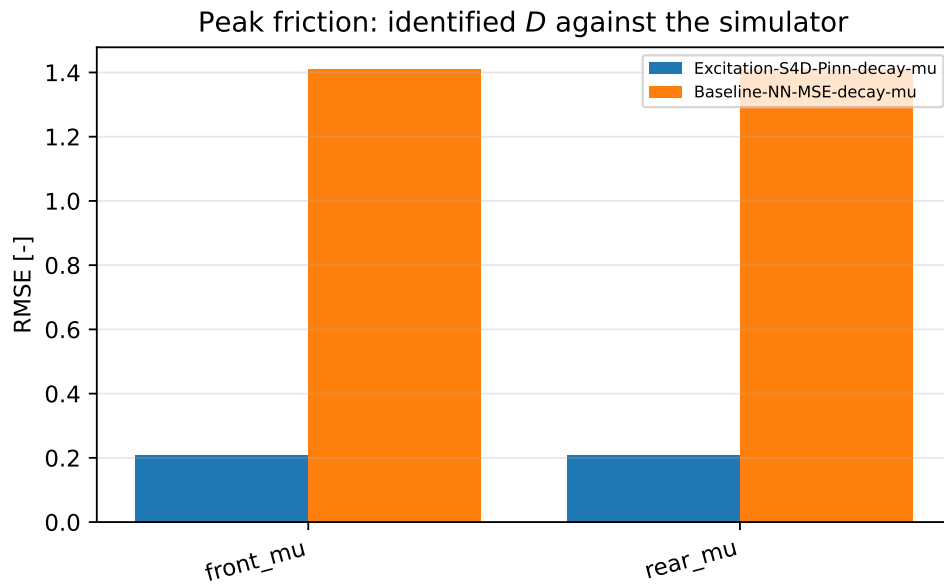


Fig. S3 Peak-friction error per axle against the effective coefficient the simulator reports per wheel, on the static $\mu = 0.70$ surface.



Baseline-NN-MSE-decay-mu failed in lap 2

Fig. S4 Peak-friction error per axle on the decaying surface, drawn as Figure S3. The inherited run is incomplete.

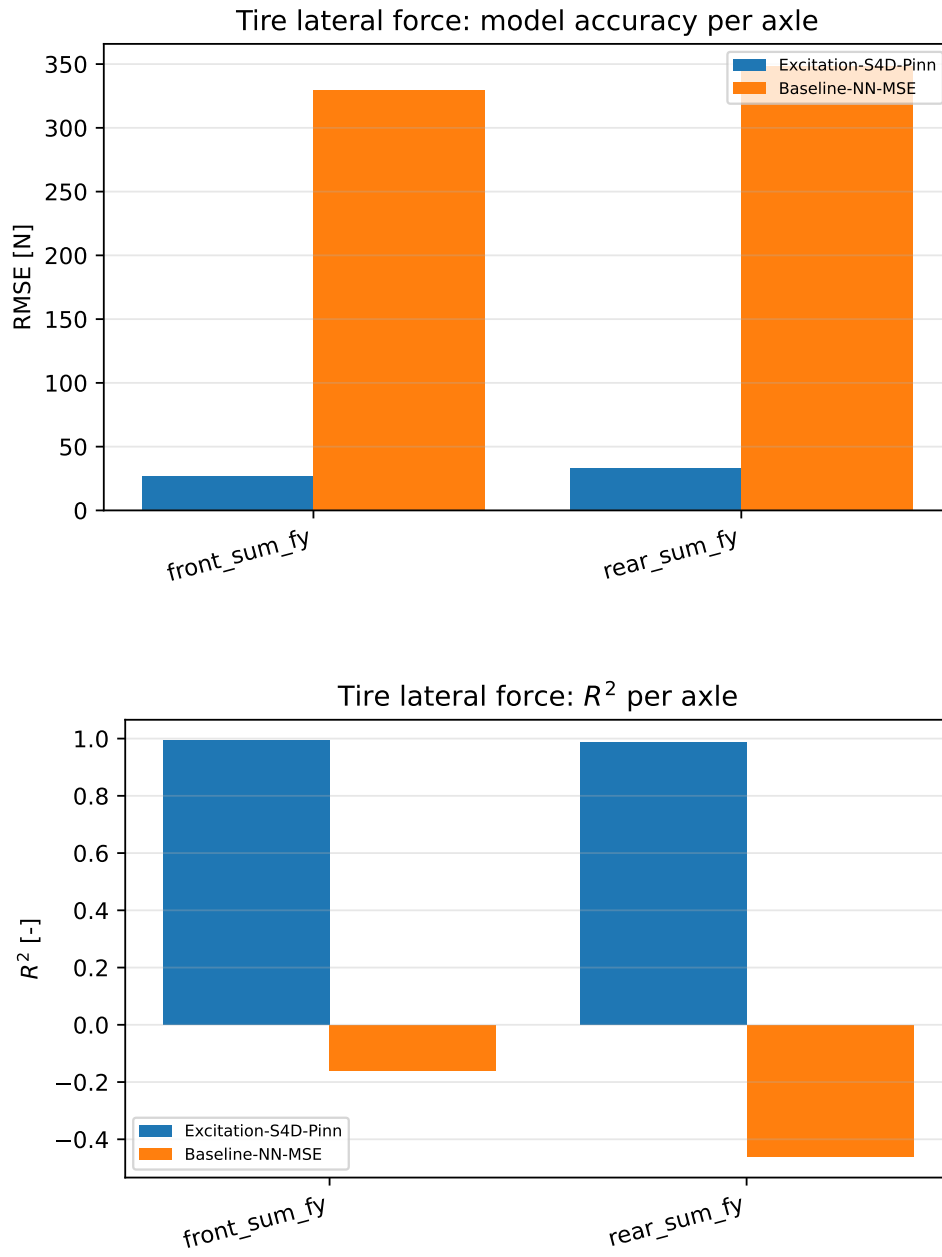


Fig. S5 Axle lateral-force comparison with per-wheel plant telemetry: RMSE (top) and R^2 (bottom).

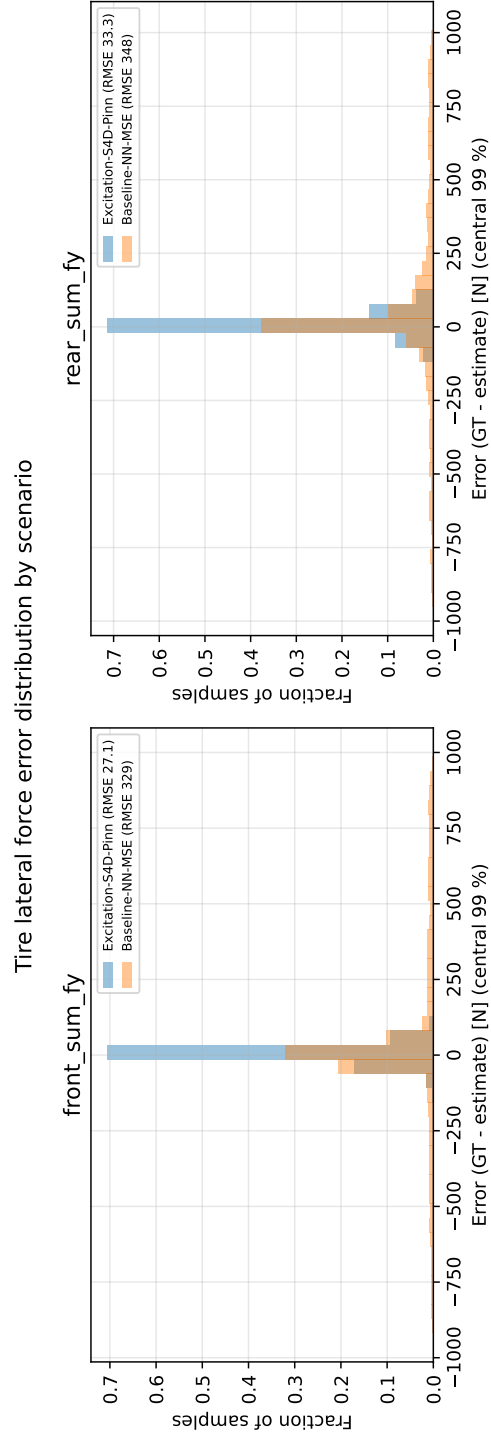


Fig. S6 Distribution of axle lateral-force error (truth minus estimate) on the static surface.

and S8). The inherited R^2 is positive here and negative on the static surface, which the sample counts explain rather than the model: that run is scored over 2597 samples against 10744 and ends before the surface reaches its floor.

The force time series compare each estimate with the ground truth recorded in the same run. On the constant surface, the corrected trace has front/rear RMSE of 27.15/33.227 N, whereas the inherited trace has 329.645/348.409 N. The traces are from separate laps, so visual alignment at the same time does not constitute a pointwise comparison between configurations; the valid comparison is each estimate against its own telemetry and the reported summary metrics (Figure S9).

On the decaying surface the ground-truth envelope contracts over the run as the plant loses grip, and the corrected estimate follows it at 32.883/36.469 N front/rear RMSE. The inherited trace ends at the collision, so its later samples are absent rather than small (Figure S10).

At the model step of 0.035 s, neither configuration improves on persistence for both reported states. For the corrected configuration, the lateral-velocity RMSE is 0.014 m/s versus 0.011 m/s for persistence, and the yaw-rate RMSE is 0.049 rad/s versus 0.005 rad/s. The corresponding inherited values are 0.156 and 0.104 m/s or rad/s, respectively. Thus the force-recovery result and the one-step state-prediction result measure different properties in these runs; the former does not imply an improvement over persistence for the latter (Figure S11).

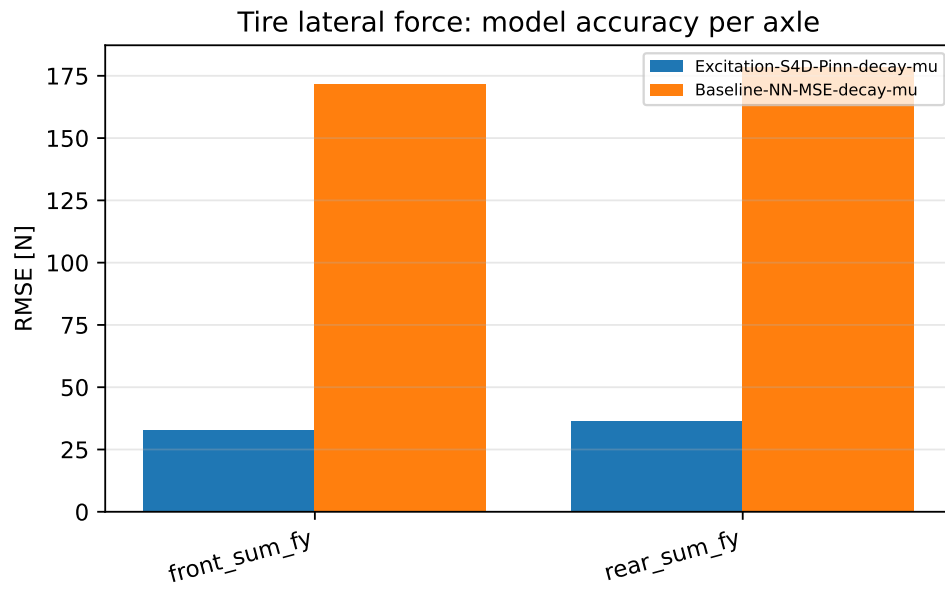
The decaying surface gives the same ordering and the same conclusion about persistence. The corrected lateral-velocity and yaw-rate RMSE are 0.011 m/s and 0.037 rad/s against persistence values of 0.007 m/s and 0.004 rad/s; the incomplete inherited run reports 0.055 m/s and 0.105 rad/s against 0.083 m/s and 0.010 rad/s. Persistence is again unbeaten on both signals for both configurations, so the one-step metric carries no more weight here than on the static surface (Figure S12).

The lateral-velocity error histogram is narrower for the corrected configuration on the constant surface. Its standard deviation is 0.0139 m/s, compared with 0.1551 m/s for the inherited configuration, and the RMSE values are 0.014 and 0.156 m/s. The corrected model still has a persistence RMSE of 0.011 m/s, so the narrower model-error distribution does not imply an improvement over that baseline (Figure S13).

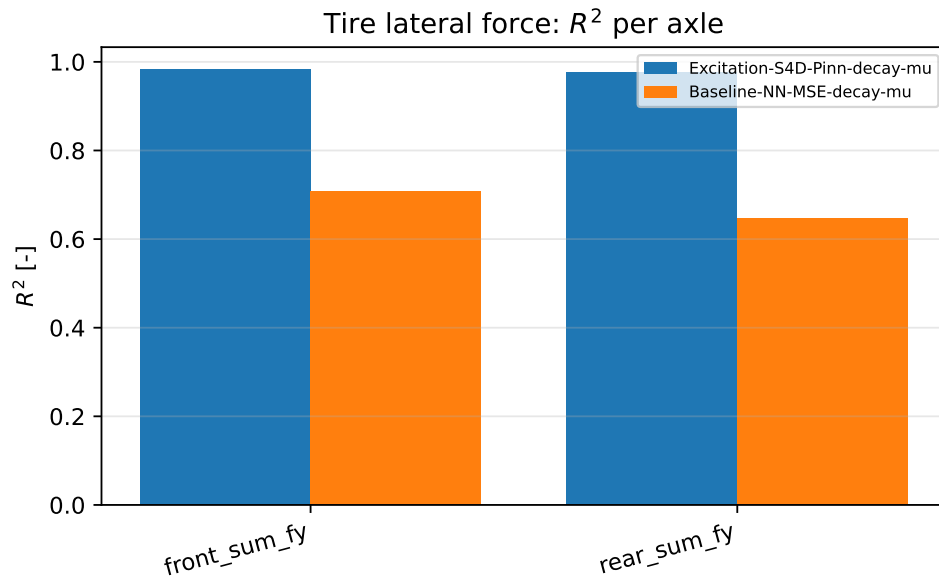
On the decaying surface, the corrected lateral-velocity RMSE is 0.011 m/s, compared with 0.055 m/s for the incomplete inherited run. The corresponding persistence values are 0.007 and 0.083 m/s. The inherited distribution contains a worst observed error of 1.28 m/s, so the available samples include a tail much larger than their central spread; the incomplete run limits the scope of that comparison (Figure S14).

For yaw rate on the constant surface, both error distributions remain close to zero in their central mass, while the inherited configuration is wider. The interquartile widths are 0.021 and 0.049 rad/s and the standard deviations are 0.049 and 0.103 rad/s for the corrected and inherited configurations, respectively. The figure therefore supports a difference in dispersion, not a large systematic shift in the centre (Figure S15).

On the decaying surface, the yaw-rate error distribution for the incomplete inherited run contains an observed tail reaching 4.17 rad/s, whereas the corrected distribution has no comparable observed excursion. This is a statement about the available error samples, not a diagnosis of the event that produced the tail (Figure S16).



Baseline-NN-MSE-decay-mu failed in lap 2



Baseline-NN-MSE-decay-mu failed in lap 2

Fig. S7 Axle lateral-force comparison on the decaying surface, drawn as Figure S5: RMSE (top) and R^2 (bottom). The inherited run is incomplete and its bars cover a shorter window.

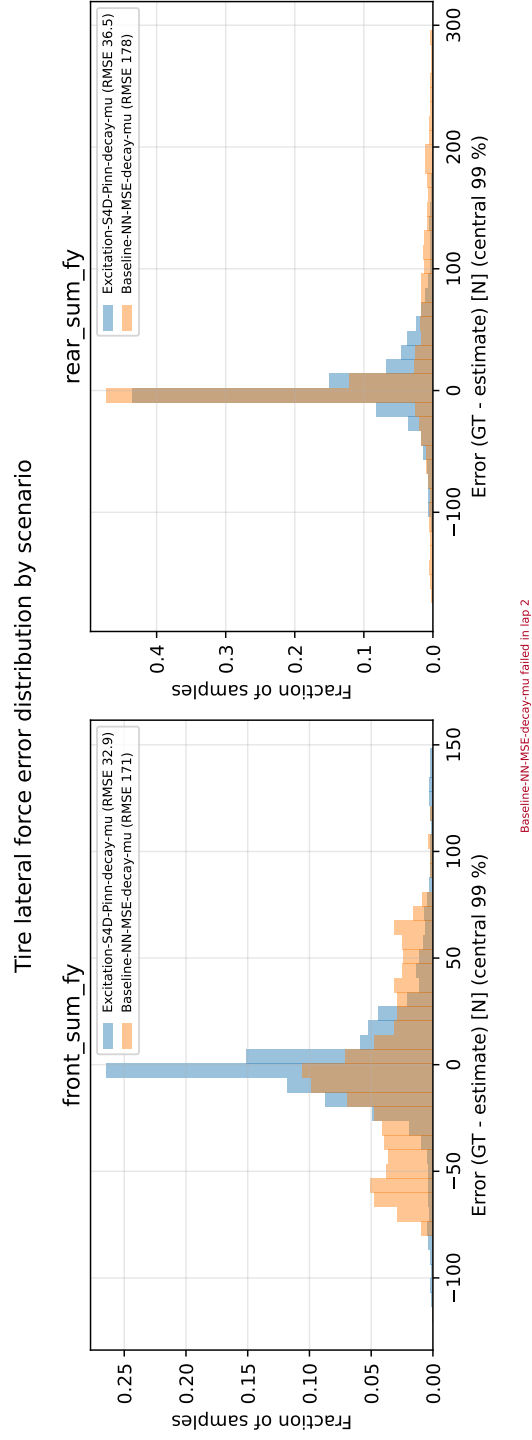


Fig. S8 Distribution of axle lateral-force error (truth minus estimate) on the decaying surface.

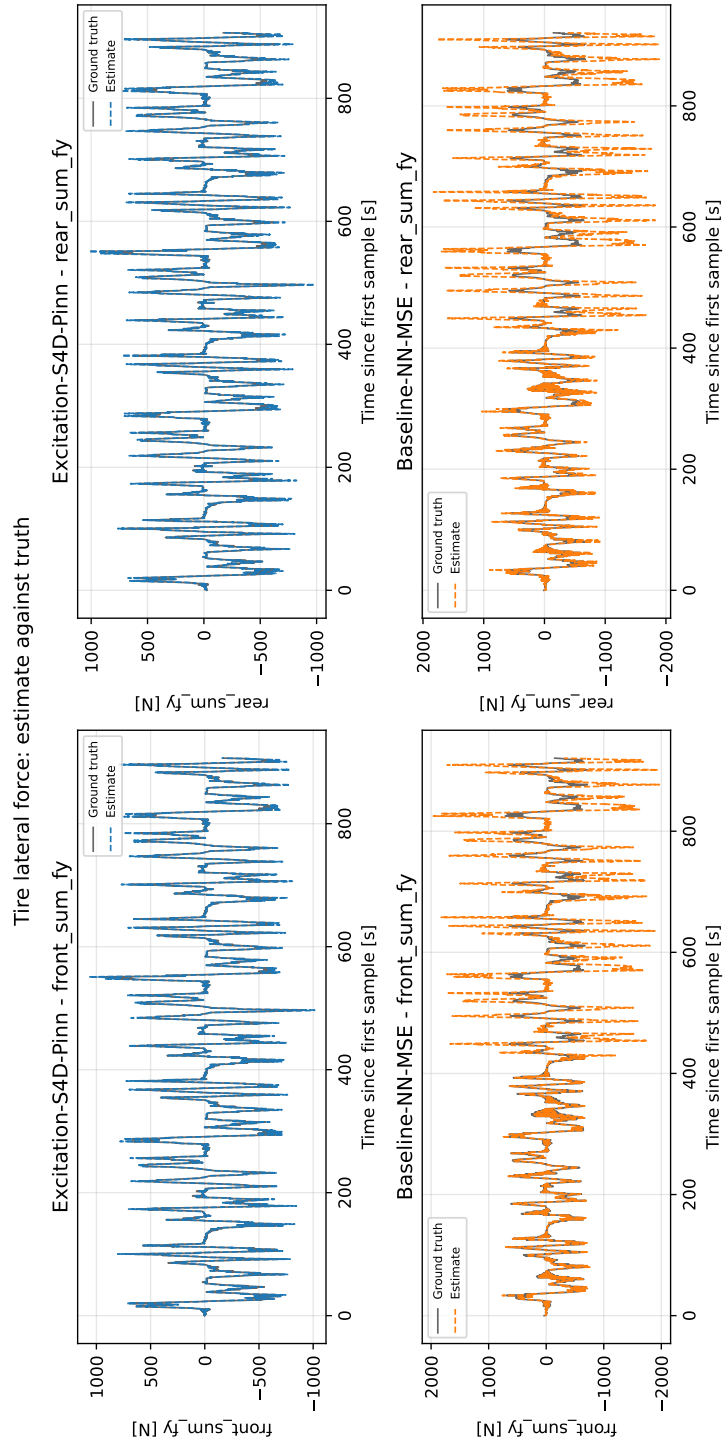
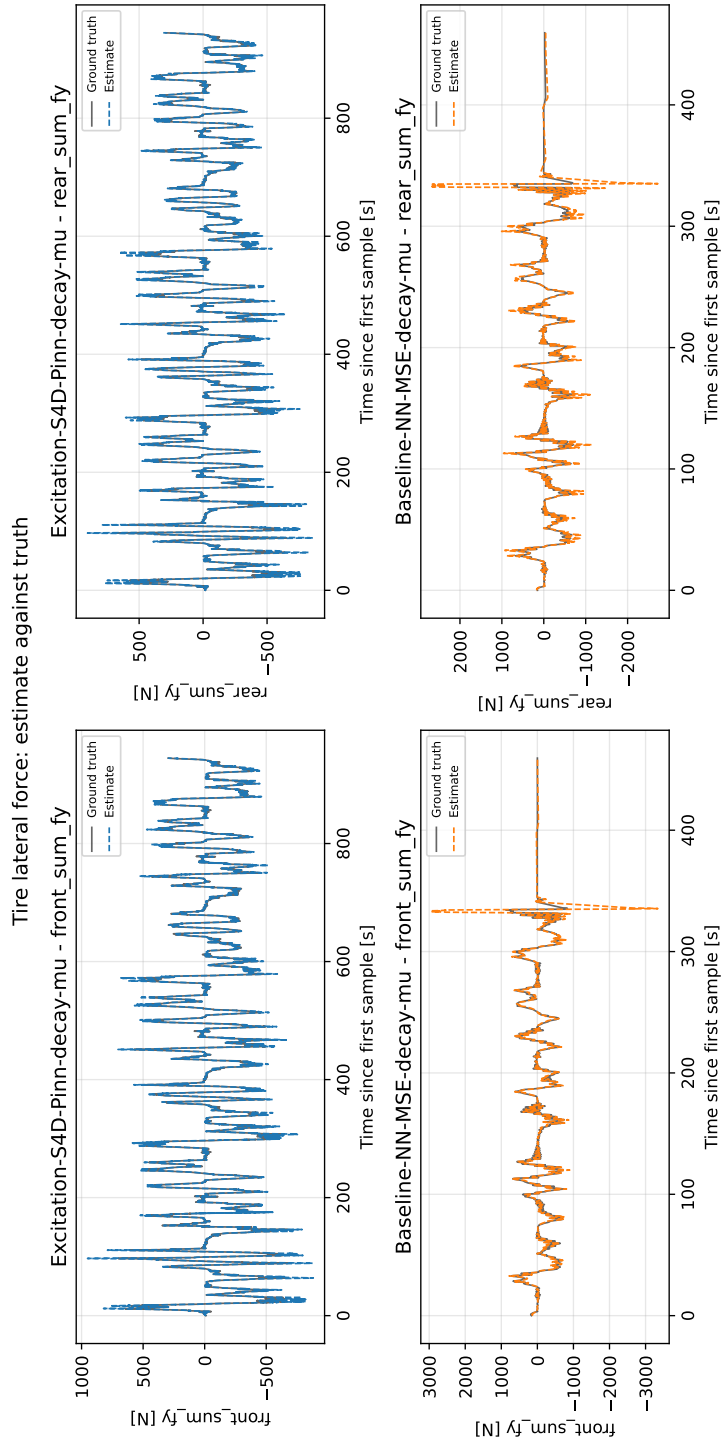


Fig. S9 Axle lateral force and ground truth over separate constant-surface runs.



Baseline-NN-MSE-decay-mu failed in lap 2

Fig. S10 Axle lateral force and ground truth over separate decaying-surface runs.

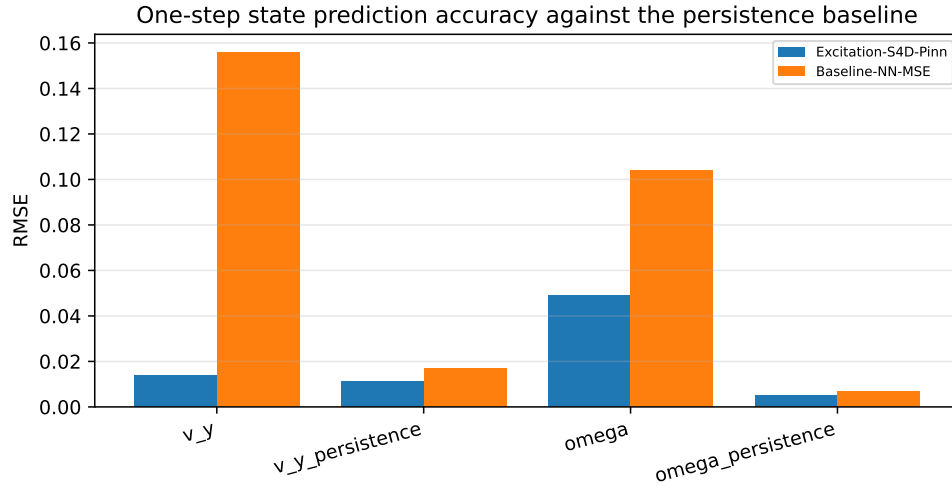
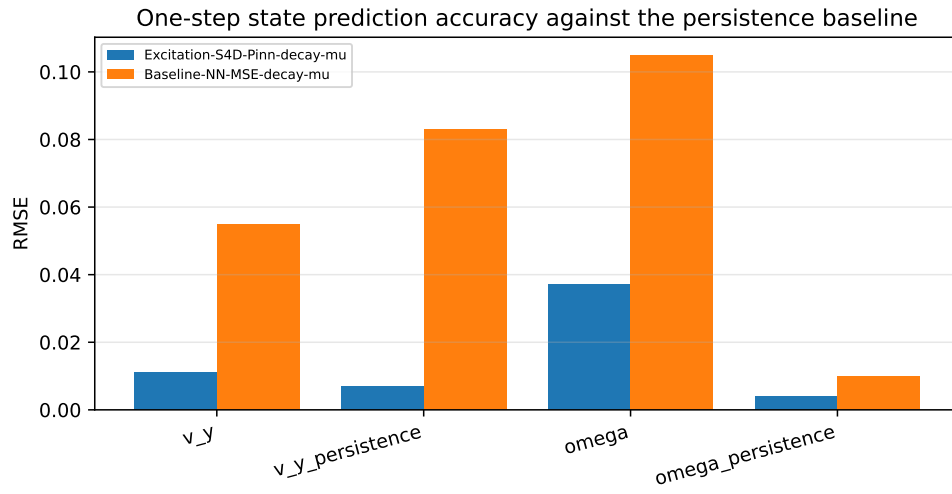


Fig. S11 One-step prediction RMSE for v_y and ω compared with persistence.



Baseline-NN-MSE-decay-mu failed in lap 2

Fig. S12 One-step prediction RMSE on the decaying surface, drawn as Figure S11.

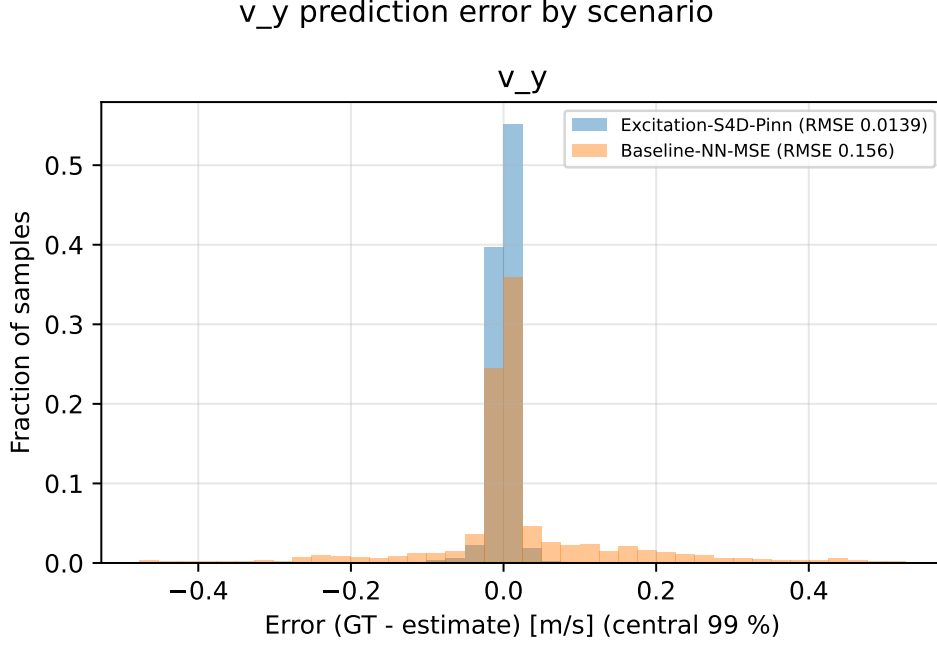


Fig. S13 Lateral-velocity prediction error (truth minus estimate) on the static surface.

The state traces largely overlap their corresponding ground truth over the reported constant-surface runs. The summary CSV records 0.014 m/s and 0.049 rad/s RMSE for the corrected lateral velocity and yaw rate, compared with 0.156 m/s and 0.104 rad/s for the inherited configuration. The error distributions therefore expose differences that are difficult to resolve in the compressed traces. The separate runs do not define paired observations (Figure S17).

On the decaying surface the corrected traces overlap their ground truth for the whole run at 0.011 m/s and 0.037 rad/s RMSE, while the inherited traces stop at the collision and their last samples carry the departure rather than a prediction error. The separate runs again do not define paired observations (Figure S18).

For the constant-surface tracking error, the corrected configuration has a smaller standard deviation, 0.256 m versus 0.318 m, while the means are similar, 0.021 m and 0.024 m. The reported statistics therefore locate the difference primarily in spread, not in the mean signed offset; they do not establish that the two runs had identical initial conditions (Figure S19).

On the decaying surface, the corrected tracking-error standard deviation is 0.264 m, within 3% of its constant-surface value. The incomplete inherited run has a reported RMS tracking error of 21.505 m and a maximum absolute error of 38.387 m; its histogram contains a second mode near 38.4 m with 29% of the available samples. A

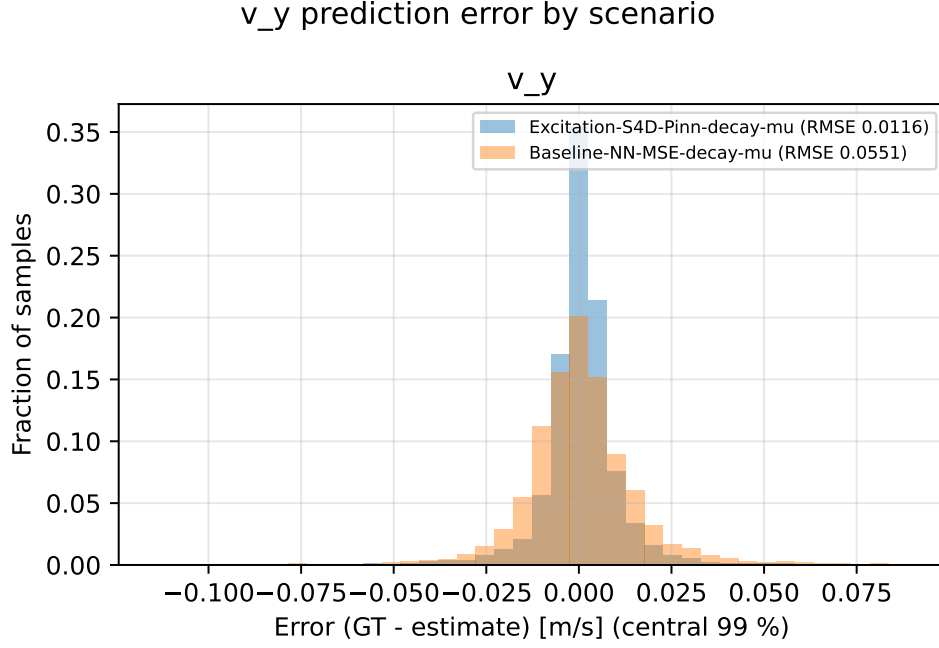


Fig. S14 Lateral-velocity prediction error (truth minus estimate) on the decaying surface.

single mean and standard deviation would not represent that shape, and the collision means the result is not a completed-lap performance measure (Figure S20).

The ITAE difference is concentrated in the MPC phase. On the constant surface, the corrected and inherited configurations have overall values of 16041.5 and 25501.9. Their pure-pursuit contributions are 109.5 and 106.9, whereas their MPC contributions are 15925.3 and 25384.2. The phase breakdown shows that the overall separation is not produced by the pure-pursuit portion; it is dominated by the portion in which the identified model is active (Figure S21).

The decaying surface puts the same breakdown two orders of magnitude apart. The corrected configuration reports 18044.1 overall, 136.2 under pure pursuit and 17887.7 under MPC; the incomplete inherited run reports 1316108.3, 110.8 and 1315977.4. The pure-pursuit contributions stay within 23% of each other while the MPC contributions differ by a factor of 74, so the separation again falls entirely in the phase where the identified model is in the loop. The inherited MPC value is the track departure integrated over time and is not a tracking quality (Figure S22).

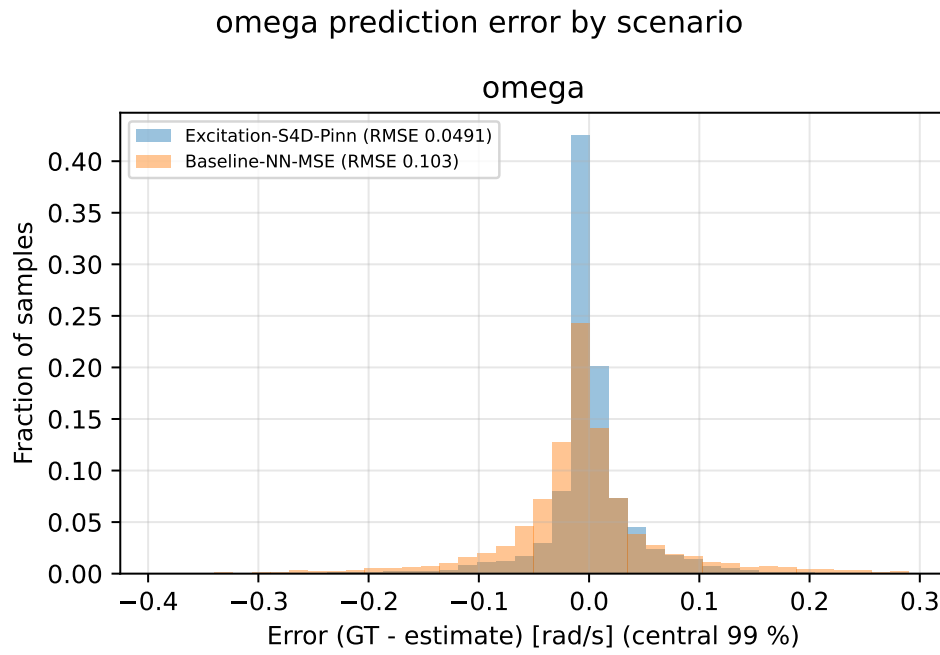
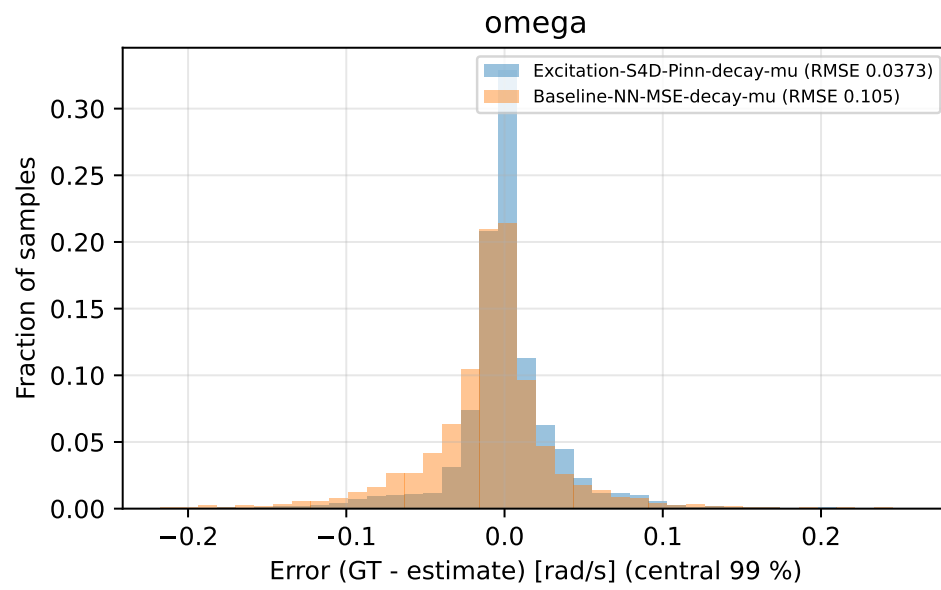


Fig. S15 Yaw-rate prediction error (truth minus estimate) on the static surface.

omega prediction error by scenario



Baseline-NN-MSE-decay-mu failed in lap 2

Fig. S16 Yaw-rate prediction error (truth minus estimate) on the decaying surface.

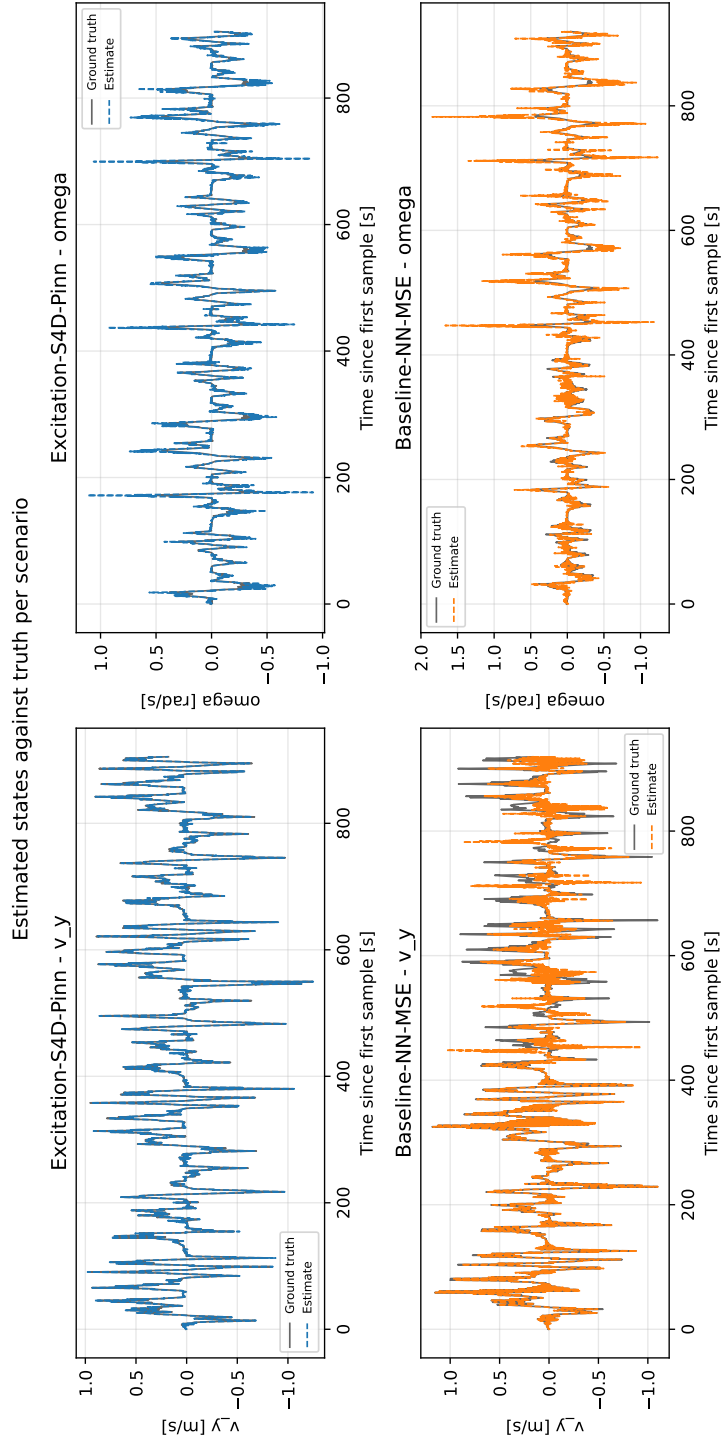
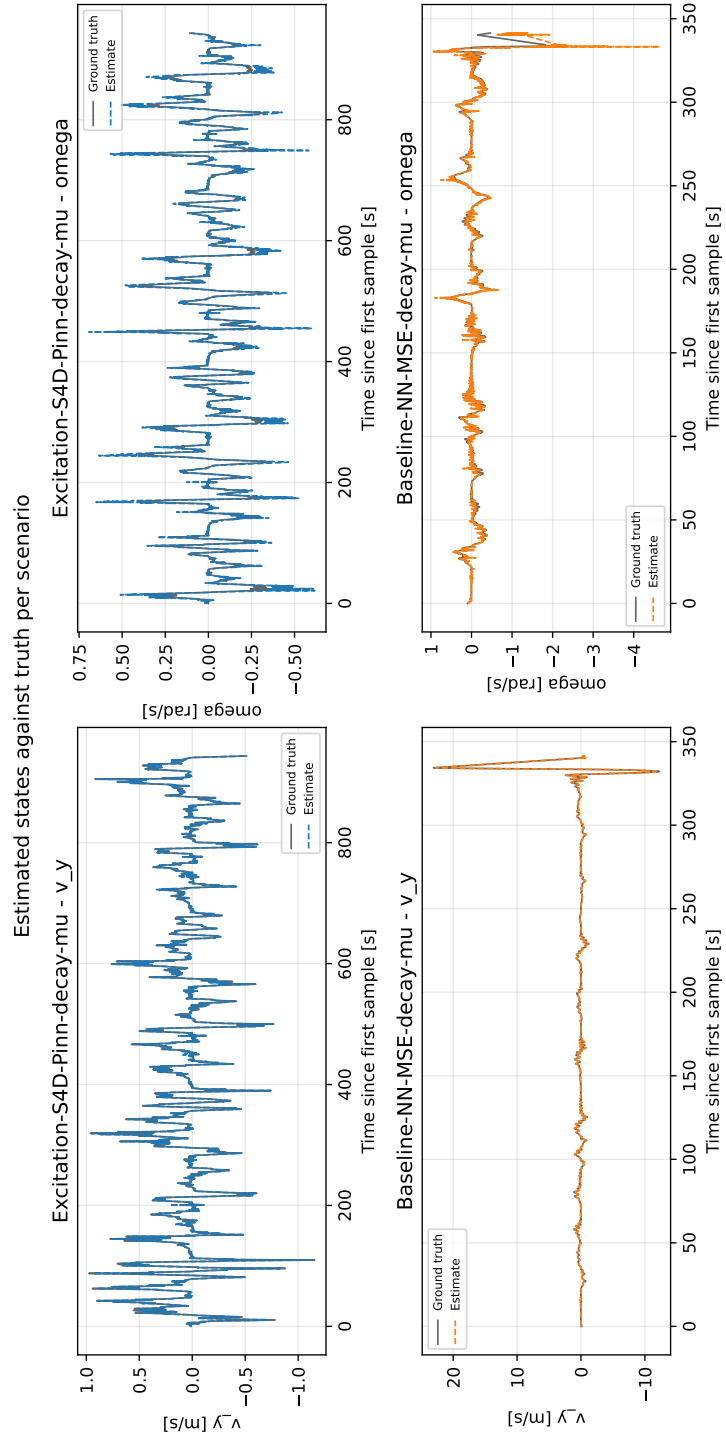


Fig. S17 Estimated states and ground truth over separate constant-surface runs.



Baseline-NN-MSE-decay-mu failed in lap 2

Fig. S18 Estimated states and ground truth over separate decaying-surface runs.

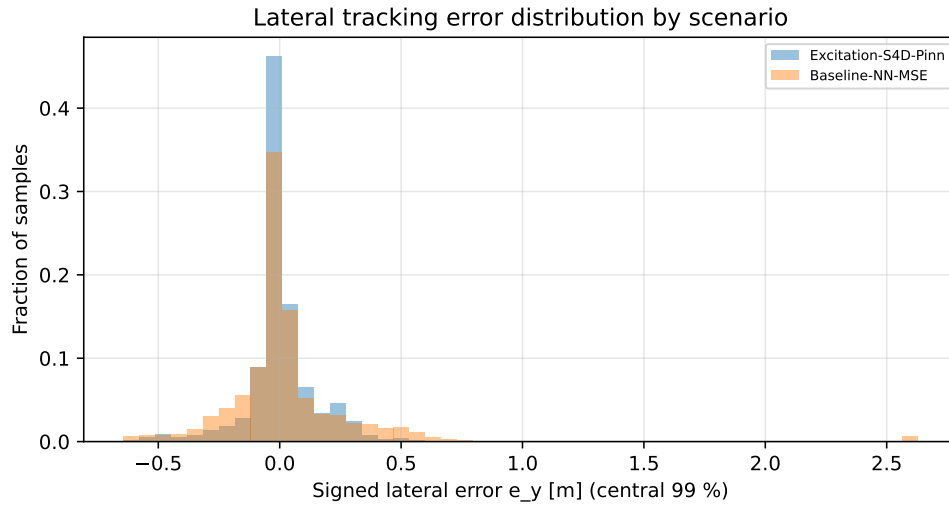


Fig. S19 Signed lateral tracking error over the static-surface runs.

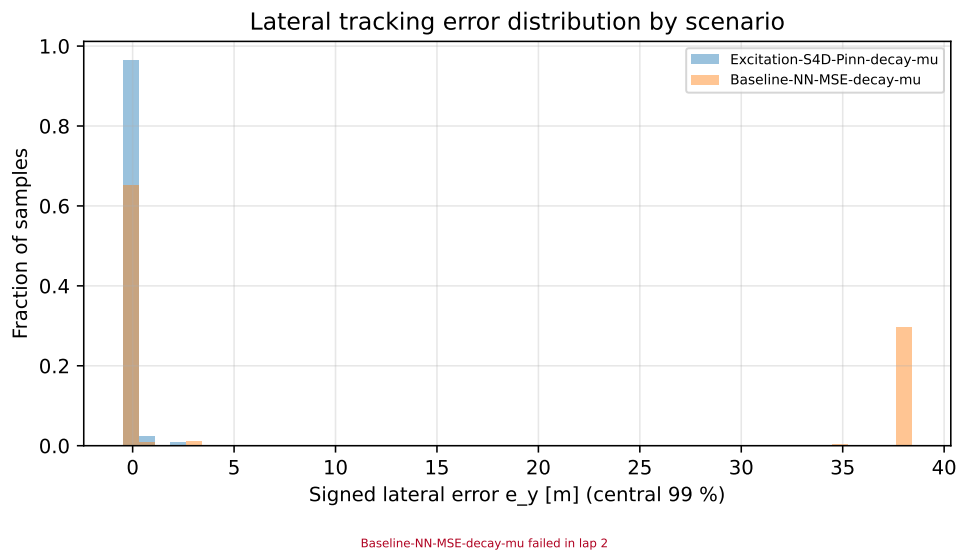


Fig. S20 Signed lateral tracking error over the decaying-surface runs.

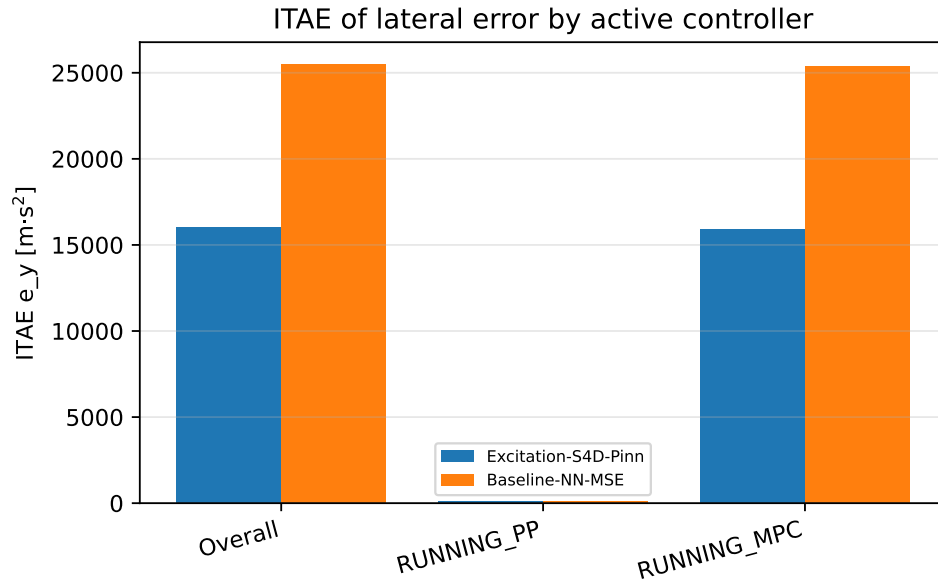
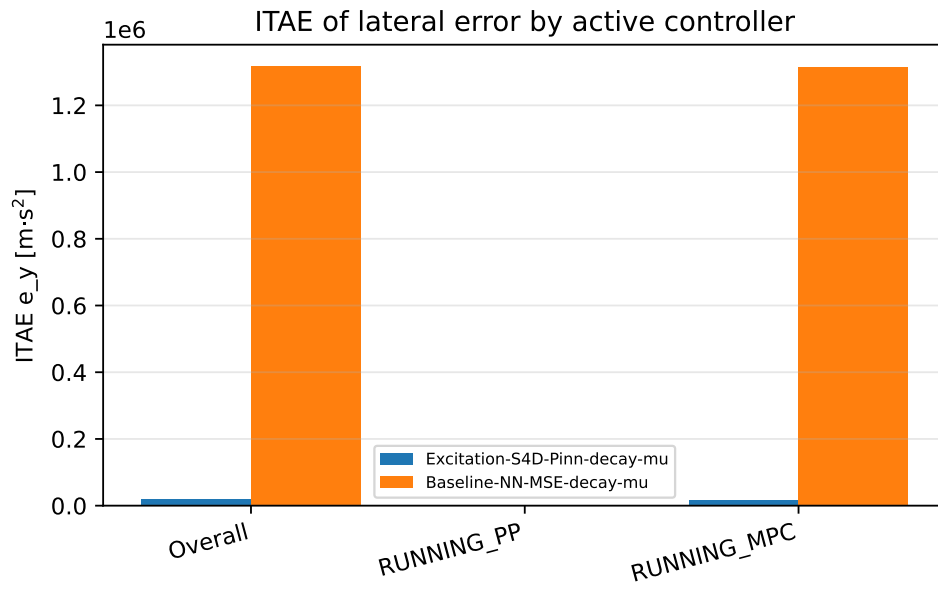


Fig. S21 Integrated time-weighted absolute lateral error, overall and by active controller.



Baseline-NN-MSE-decay-mu failed in lap 2

Fig. S22 Integrated time-weighted absolute lateral error on the decaying surface, drawn as Figure S21.

References

- [1] Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **378**, 686–707 (2019).
- [2] Chrosniak, J., Ning, J. & Behl, M. Deep dynamics: Vehicle dynamics modeling with a physics-constrained neural network for autonomous racing. *IEEE Robotics and Automation Letters* **9**, 5292–5297 (2024). Preprint: arXiv:2312.04374.
- [3] Gu, A., Goel, K., Gupta, A. & Ré, C. *On the parameterization and initialization of diagonal state space models*. *Advances in Neural Information Processing Systems (NeurIPS)* (New Orleans, LA, USA, 2022). Preprint: arXiv:2206.11893.
- [4] Gu, A., Dao, T., Ermon, S., Rudra, A. & Ré, C. *HiPPO: Recurrent memory with optimal polynomial projections*. *Advances in Neural Information Processing Systems (NeurIPS)* (2020).
- [5] Wu, Z., Hu, C., Wang, Y., Xie, L. & Su, H. Vision-augmented on-track system identification for autonomous racing via attention-based priors and iterative neural correction (2026). URL <https://arxiv.org/abs/2603.09399>. 2603.09399.
- [6] Gu, A., Goel, K. & Ré, C. *Efficiently modeling long sequences with structured state spaces*. *Proc. International Conference on Learning Representations (ICLR)* (Virtual, 2022). Preprint: arXiv:2111.00396.
- [7] Smith, J. T. H., Warrington, A. & Linderman, S. W. *Simplified state space layers for sequence modeling*. *International Conference on Learning Representations (ICLR)* (2023).
- [8] Dikici, O. *et al.* Learning-based on-track system identification for scaled autonomous racing in under a minute. *IEEE Robotics and Automation Letters* **10** (2025). Preprint: arXiv:2411.17508. Code: <https://github.com/ForzaETH/On-Track-SysID>.