Inversion-Based Control for Autonomous Racing and Navigation

Harshavardhan Sameer Raje

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2026

Committee:
Santosh Devasia
Sawyer Fuller
Joseph Garbini

Program Authorized to Offer Degree:
Mechanical Engineering

University of Washington

Abstract

Inversion-Based Control for Autonomous Racing and Navigation

Harshavardhan Sameer Raje

Chair of the Supervisory Committee:
Santosh Devasia
Mechanical Engineering

Classical geometric path-following controllers for autonomous ground vehicles compute steering commands from cross-track and heading error. However, they do not directly check whether the requested motion can be achieved by the vehicle under tire force and friction limits. This thesis develops an *iterative force-consistency inversion* framework for the complementary problem: given an achievable trajectory demand and a vehicle model that is close enough to the real plant, compute a set of steering and longitudinal inputs that make the nonlinear tire-force model realize that demand. The needed inverse is solved online with a Newton or Levenberg–Marquardt update step using an enhanced bicycle model with nonlinear tire forces, load transfer, roll dynamics and steering lag.

The central result is that a pure feedforward inverse alone is not sufficient for path tracking because model error produces steady-state drift. To compensate for this, the inverse must be paired with an outer error-feedback loop. In simulations, improving the internal model from a kinematic approximation to the enhanced physics model lowers the RMS cross-track error of the feedback inverse on runs that stay within the track. This model improvement also increases the number of runs that leave the track bounds. A model-fidelity sweep across track shapes and speeds shows the limit of the current residual-correction layer: SINDy, Koopman/EDMD, and GP-SSM residual models can be fit using replay data, but they increase closed-loop failures when inserted directly into the Newton inverse. Model fidelity and closed-loop stability must be assessed together; learned residual corrections fit the replay data well but have not yet produced a closed-loop performance gain.

Planning under drift and friction constraints provides the trajectory-feasibility front end to this inverse-control problem. The planner tests whether a demanded trajectory or acceleration profile lies inside the available friction budget, and whether the path must be reshaped before inversion can produce meaningful inputs. The physical RC-car platform, equipped with dual IMUs, a 360° LiDAR, OptiTrack integration, telemetry, logging, and state estimation, provides the pipeline for future hardware validation. All quantitative controller benchmarks reported in this thesis are simulation-only. The hardware closed-loop controller comparison is reserved for future work.

Acknowledgments

I am deeply grateful to my professors and mentors who helped shape this work. Their guidance helped turn a collection of simulations, experiments, false starts, and half-working ideas into a thesis with a clearer direction.

I also owe a great deal to my friends and labmates, especially for lending an ear while I rambled through half-formed ideas at the whiteboard and for telling me when I needed to take a step back instead of spinning my wheels. The technical parts of this thesis benefited from those exchanges, and the process was easier because I had good people nearby when the work became difficult.

To my friends and family, thank you for carrying me through the parts of this degree that do not appear in the figures, tables, or equations. Your patience, humor, encouragement, and belief in my ability made the long stretches of work feel less isolating. This thesis belongs partly to the people who kept me grounded when the work became overwhelming and reminded me that finishing was possible.

This project also rests on the efforts of many people whose work often sits just outside the formal record. Thank you to the people who build and maintain the repositories, libraries, examples, and tools that made the experiments and writing possible; to the building staff whose daily work kept the spaces around this research running; to the baristas at the coffee shops on the Ave; to the staff at Pagliacci; and to the Route 32 drivers of King County Metro, who made more of this thesis possible than they probably know.

Contents

1 Introduction

Autonomous ground vehicles often need to operate near the limits of tire adhesion to execute rapid maneuvers such as tight turns, avoiding obstacles, or high-speed trajectory tracking. Classical trajectory tracking controllers are typically derived from the kinematic bicycle model and rely on assumptions of low speed, negligible inertial effects, and perfect tire grip [1, 2]. These assumptions allow for a simplified controller design, but they also limit performance when tire forces and friction constraints dominate the achievable motion.

Many widely used controllers treat steering and velocity as separate problems, with a lateral controller computing steering from cross-track and heading error and a longitudinal loop regulating speed independently. This decoupling works well at low to moderate speeds, but aggressive maneuvers often require lateral and longitudinal tire forces to be coordinated, and purely geometric commands can then request motion that the available friction cannot support, leading to tracking error, input saturation, or loss of stability.

Model inversion maps desired vehicle motion directly to control inputs, bypassing the decoupled geometric formulation. For the kinematic bicycle model this map can be obtained algebraically, but including tire dynamics and friction constraints makes the input-to-motion relationship nonlinear, state-dependent, and not globally invertible in closed form. Linearizing about an operating point recovers a tractable design but reintroduces the same structural limits as the kinematic model, and those limits matter most in the regimes where slip and nonlinear tire behavior are significant.

The inversion strategy developed here is based on force consistency rather than geometric inversion. Rather than solving directly for the steering angle that reproduces a desired trajectory, the solver first computes the lateral force needed to achieve the target curvature and speed, then solves for the steering input that matches the tire model's predicted force to that demand. A Newton or Levenberg–Marquardt update performs this at each control step and produces a feedforward command consistent with the vehicle's dynamic constraints.

The resulting controller consists of a trajectory planner that specifies curvature and velocity, a nonlinear inverse that converts that demand into tire-consistent control inputs, and a feedback controller that corrects for modeling error and disturbances. The planner screens the requested motion against the available friction budget; the inverse converts the resulting demand into actuator commands, while the feedback layer absorbs the residual mismatch between model and plant.

This thesis makes four main contributions. First, a force-based inversion framework is developed that connects trajectory curvature to achievable tire forces, and second, an iterative inverse algorithm is introduced that computes control inputs for a nonlinear vehicle model without requiring a closed-form inverse. Third, the framework is evaluated in MuJoCo simulation to measure how inverse-control performance depends on the fidelity of the internal model and on the presence of an outer feedback loop. Fourth, five different data-driven residual identification methods (SINDy, Koopman/EDMD, GP state-space model, hybrid sparse–neural network, and symbolic regression) are compared as model-refinement tools, and closed-loop results are used to expose the gap between one-step prediction accuracy and practical controller improvement.

All quantitative controller comparisons in this thesis are simulation-only. The physical RC-car platform provides sensing, logging, state estimation, LiDAR, and OptiTrack motion

capture as ground truth for future closed-loop experiments, but it is not yet the benchmark source because communication latency in the control network has not been characterized and compensated for.

Section 2 introduces the vehicle models including the enhanced bicycle model; Section 3 covers tire force models. Section 4 describes the standard geometric, hybrid, and optimization-based controllers, while Section 5 develops the force-based nonlinear inverse family and iterative solver. Section 6 addresses path planning and reference replanning in the context of the broader motion-planning literature [3, 4], and Section 7 presents data-driven residual identification. Experimental results and the simulation benchmark protocol appear in Sections 8 and 8.1.

Nomenclature

Symbol	Meaning	Unit
Vehicle Geometry		
L	Wheelbase	m
a	CG to front axle	m
b	CG to rear axle	m
h	CG height	m
W	Track width	m
Vehicle State		
X, Y	Inertial position	m
ϕ	Yaw heading angle	rad
v_x	Longitudinal velocity	m/s
v_y	Lateral velocity	m/s
$r = \dot{\phi}$	Yaw rate	rad/s
v	Speed	m/s
ϕ_{roll}	Roll angle	rad
$\dot{\phi}_{\text{roll}}$	Roll rate	rad/s
Inputs		
ψ (or δ)	Steering angle	rad
a_x	Longitudinal accel. cmd	m/s^2
F_{in}	Drive/brake force	N
Tire		
α_f, α_r	Slip angle (f/r)	rad
$C_{\alpha f}$, $C_{\alpha r}$	Cornering stiffness	N/rad
μ	Friction coefficient	–
$F_{z,f0}$, $F_{z,r0}$	Static normal load	N
$F_{z,f}$, $F_{z,r}$	Dynamic normal load	N
$F_{y,f}$, $F_{y,r}$	Lateral tire force	N
$F_{x,f}$, $F_{x,r}$	Longitudinal tire force	N
B, C, D, E	Pacejka MF factors	–
ΔF_z	Load transfer	N

Symbol	Meaning	Unit
Control / Path		
κ	Path curvature	1/m
e (CTE)	Cross-track error	m
θ_e	Heading error	rad
L_d	Lookahead distance	m
k, k_s	Stanley gain / softening	–
$w(t)$	Blend weight	–
$w_{\min}$, $w_{\max}$	Blend weight bounds	–
Enhanced Model		
I_z	Yaw inertia	kg m^2
I_x	Roll inertia	kg m^2
K_ϕ, D_ϕ	Roll stiffness/damping	N m/rad
C_d	Aero drag coefficient	–
A	Frontal area	m^2
ρ	Air density	kg/m^3
τ	Steering lag constant	s
Data-Driven		
$\Delta \dot{v}_y$	Lateral accel. residual	m/s^2
$\Delta \dot{r}$	Yaw accel. residual	rad/s^2
$\Delta F_{y,f}$, $\Delta F_{y,r}$	Tire force correction	N
$\Delta F_{z,f}$, $\Delta F_{z,r}$	Load correction	N
$\Delta\delta$	Steer correction	rad
J	Jacobian $\partial r / \partial u$	–
λ	LM damping factor	–

2 System Modeling

2.1 Vehicle Model

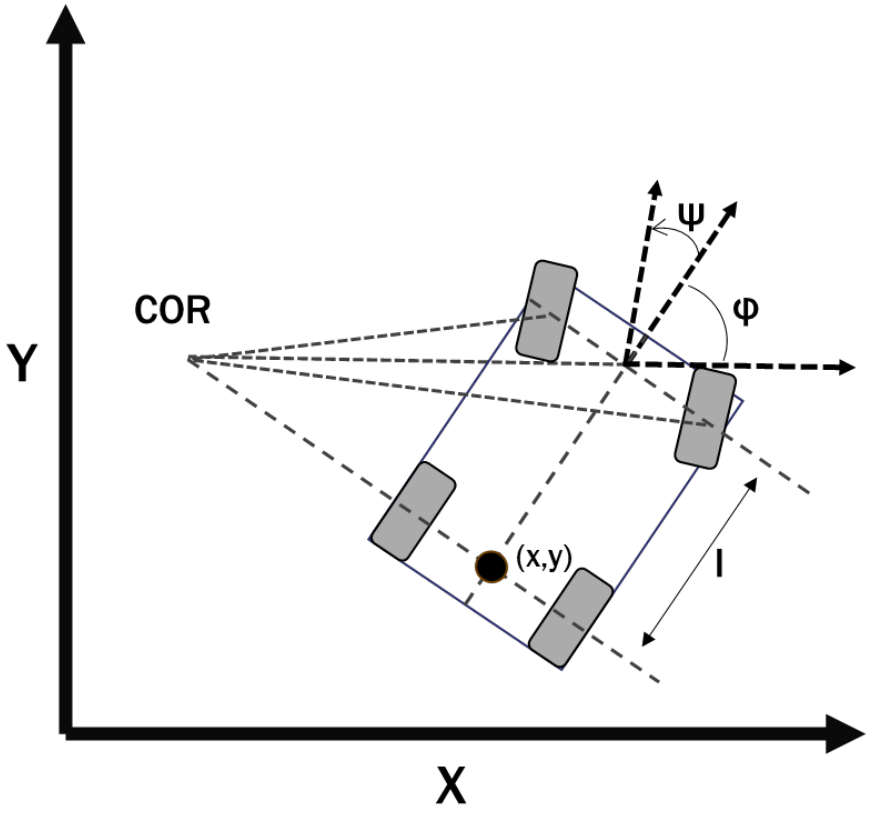


Figure 1: Ackermann Kinematics

Consider first the simplest kinematic representation: the Ackermann bicycle model. It assumes no tire slip, so it is most accurate at low lateral accelerations and low to moderate speeds. Following the kinematic bicycle model in [5, Chapter. 2] and the autonomous-driving control design literature [1, 2], the model can be described by

$$\begin{aligned} \dot{x} &= v\cos(\phi), \\ \dot{y} &= v\sin(\phi), \\ \dot{\phi} &= \frac{v}{L}\tan(\psi), \end{aligned} \tag{1}$$

where

- (x, y) is the rear axle center,
- ϕ is the heading angle,

- v is the forward velocity,
- ψ is the steering angle,
- L is the wheelbase.

Four assumptions bound the model's validity. The tire contact patches satisfy a pure rolling constraint, so they do not experience lateral slip. The velocity vector aligns with the wheel orientation, and the vehicle operates below the lateral tire-force saturation limit. Neglecting track width and differential effects, the front and rear wheel sets share a single ground contact point along their respective axles. Roll, pitch, suspension, load transfer, and compliance are all omitted. Finally, ψ is taken small enough that $\tan\psi$ remains finite. In simulation, this last assumption is also supported by hard steering limits.

The no-lateral-slip assumption can be written as:

$$-\sin(\phi)\dot{x} + \cos(\phi)\dot{y} = 0$$

This is a Pfaffian constraint from the rear wheel set rolling without slipping. It enforces zero velocity perpendicular to the heading and makes the kinematic vehicle model non-holonomic.

The kinematic bicycle model can be rewritten compactly by grouping the control inputs

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix}, \qquad \omega = \dot{\phi} = \frac{v}{L}\tan\psi,$$

Note: v and ω are virtual inputs corresponding to forward vehicle velocity and yaw rate. These are not actual inputs on the vehicle.
Using this representation, like terms can be collected so that each input multiplies a vector field.

$$\begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \omega \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + v \begin{bmatrix} 0 \\ \cos(\phi) \\ \sin\phi \end{bmatrix}$$

This yields the control-affine form

$$\begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{A(x) or drift}} + \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & \cos\phi \\ 0 & \sin\phi \end{bmatrix}}_{B(\phi)} \begin{bmatrix} \omega \\ v \end{bmatrix}.$$

Representing the dynamics in terms of yaw rate ω eliminates the explicit nonlinearity $\tan(\psi)$ from the input channel. In this form, the input v governs translational motion in the plane, while ω governs rotational motion of the vehicle.

2.2 Linearization and Controllability of the Kinematic Bicycle Model

The F1TENTH Gym Simulator is a commonly used environment for evaluating control strategies, and its physics are based on the standard kinematic bicycle model [1, 2, 5]. In

many control architectures, the longitudinal channel is regulated by a separate velocity loop, as in the Stanley controller, where velocity is set by a PI controller. For the linearization here, the steering angle and longitudinal acceleration are treated as inputs by augmenting (1) with

$$\dot{v} = a.$$

Define the state and input vectors

$$\mathbf{x} \coloneqq \begin{bmatrix} x & y & \phi & v \end{bmatrix}^\top, \qquad \mathbf{u} \coloneqq \begin{bmatrix} \psi & a \end{bmatrix}^\top, \tag{2}$$

so that $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$.

2.2.1 Continuous-time linearization

Linearizing about an operating point $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ yields

$$\delta\dot{\mathbf{x}} = A\,\delta\mathbf{x} + B\,\delta\mathbf{u}, \qquad A \coloneqq \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{(\bar{\mathbf{x}},\bar{\mathbf{u}})}, \quad B \coloneqq \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{(\bar{\mathbf{x}},\bar{\mathbf{u}})}. \tag{3}$$

For (1)–(2.2) with variables (ϕ, ψ) as defined above, the Jacobians are

$$A = \begin{bmatrix} 0 & 0 & -v\sin\phi & \cos\phi \\ 0 & 0 & v\cos\phi & \sin\phi \\ 0 & 0 & 0 & \frac{1}{L}\tan\psi \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{v}{L}\sec^2\psi & 0 \\ 0 & 1 \end{bmatrix}, \tag{4}$$

evaluated at $(\phi, v, \psi) = (\bar{\phi}, \bar{v}, \bar{\psi})$.

This model is underactuated in inertial coordinates. The inputs (ψ, a) or (ψ, v) do not directly actuate the inertial position states (x, y); instead, they influence position through vehicle heading and forward speed. Consequently, controllability of the linearized model (3) depends on the operating point, and the standard linear controllability rank test can be misleading if interpreted as a statement about the nonlinear system.

To make this explicit, consider the controllability matrix of the linearized continuous-time system

$$C \;=\; \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix}. \tag{5}$$

At low speed $v \approx 0$, the steering channel decouples from heading dynamics:

$$\frac{\partial\dot{\phi}}{\partial\psi} = \frac{\bar{v}}{L}\sec^2\bar{\psi} \approx 0,$$

and the linearized model loses the ability to affect the heading state through ψ. In that regime, the rank of C drops, showing that changing steering angle cannot change the vehicle pose when the vehicle is not moving. Physically, this is the stationary-car case: turning the steering wheel alone does not change the vehicle state.

For inversion-based or aggressive tracking control, this points toward a formulation that works with acceleration-level quantities rather than using small-signal controllability conclusions in inertial coordinates. The active controller implementation in this thesis therefore uses acceleration-domain inverse targets rather than a full dynamic-extension controller.

2.2.2 (ii) Dynamic single-track model (higher-speed)

State:

$$x_{\text{st}} \coloneqq \begin{bmatrix} X \\ Y \\ \delta \\ v \\ \phi \\ r \\ \beta \end{bmatrix},$$

where $r \coloneqq \dot{\phi}$ is yaw rate and β is the vehicle body slip angle at the center of mass.

The simulator uses the following position kinematics, consistent with the planar dynamic bicycle formulation used in autonomous-driving control design [1, 2]:

$$\boxed{\dot{X} = v\cos(\phi+\beta), \qquad \dot{Y} = v\sin(\phi+\beta).} \tag{6}$$

Steering and longitudinal channels:

$$\dot{\delta} = \omega_\delta, \qquad \dot{v} = a,$$

and yaw kinematics:

$$\dot{\phi} = r.$$

For compactness, define $g = 9.81$ and $L = l_f + l_r$. With these definitions, the single-track yaw-rate and slip-angle dynamics are:

$$\boxed{\begin{aligned} \dot{r} \;=\; & -\frac{\mu m}{vIL}\Big(l_f^2 C_{Sf}(gl_r - ah) + l_r^2 C_{Sr}(gl_f + ah)\Big)\, r \\ & + \frac{\mu m}{IL}\Big(l_r C_{Sr}(gl_f + ah) - l_f C_{Sf}(gl_r - ah)\Big)\,\beta \\ & + \frac{\mu m}{IL}\, l_f C_{Sf}(gl_r - ah)\,\delta, \\ \dot{\beta} \;=\; & \Big(\frac{\mu}{v^2 L}\Big(C_{Sr}(gl_f + ah)l_r - C_{Sf}(gl_r - ah)l_f\Big) - 1\Big)\, r \\ & - \frac{\mu}{vL}\Big(C_{Sr}(gl_f + ah) + C_{Sf}(gl_r - ah)\Big)\,\beta \\ & + \frac{\mu}{vL}\, C_{Sf}(gl_r - ah)\,\delta. \end{aligned}} \tag{7}$$

State-space form. Define $f_{\text{st}} : \mathbb{R}^7 \times \mathbb{R}^2 \to \mathbb{R}^7$ by

$$\dot{x}_{\text{st}} = f_{\text{st}}(x_{\text{st}}, u),$$

with

$$f_{\text{st}}(x_{\text{st}}, u) \;=\; \begin{bmatrix} v\cos(\phi+\beta) \\ v\sin(\phi+\beta) \\ \omega_\delta \\ a \\ r \\ \dot{r}(x_{\text{st}}, u) \\ \dot{\beta}(x_{\text{st}}, u) \end{bmatrix},$$

where $\dot{r}(\cdot)$ and $\dot{\beta}(\cdot)$ are given by (7).

2.2.3 Hybrid switching used in the simulator

The implementation switches between models based on speed:

$$\boxed{\text{if } |v| < v_{\text{th}},\ \dot{x} = f_{\text{ks}}(x,u); \qquad \text{else}\ \ \dot{x} = f_{\text{st}}(x,u),}$$

with $v_{\text{th}} = 0.5$ m/s in the provided code. The Jacobian matrices depend explicitly on the operating trajectory. Although these Jacobians can be computed analytically, the resulting linear system remains structurally underactuated in inertial coordinates.

In particular, longitudinal velocity acts only along the vehicle's body-fixed axis, and steering affects orientation indirectly through curvature. As a result, the inertial position states (x, y) are not directly actuated. Controllability is achieved only through integration over time.

This behavior appears in the controllability matrix of the linearized system, which is rank deficient when evaluated at a fixed operating point. The rank deficiency reflects a geometric limitation of the kinematic bicycle model expressed in inertial coordinates, not a shortcoming of the model.

Physically, the point is easiest to see for a vehicle moving on a flat plane with no external disturbances or landmarks. Absolute position in the plane does not affect the dynamics, and instantaneous control inputs cannot independently set motion in both inertial directions.

Consequently, linear controllability analysis of the kinematic bicycle model provides limited insight into tracking performance or inversion-based control. This is why the controller used later in the thesis works in the acceleration domain, targeting desired longitudinal/lateral acceleration and yaw-rate response.

2.3 Dynamic Single-Track Model

The planar single-track model is written in the body-fixed frame with the augmented state

$$x \coloneqq \begin{bmatrix} X & Y & \Phi & v_x & v_y & r & e & \Delta\psi \end{bmatrix}^\top, \qquad u \coloneqq \begin{bmatrix} a_x & \delta \end{bmatrix}^\top,$$

where X, Y are inertial position coordinates, Φ is yaw angle, v_x and v_y are the longitudinal and lateral body-frame velocities at the center of mass, and $r = \dot{\Phi}$ is yaw rate. The inputs are commanded longitudinal acceleration a_x and steering angle δ. The vehicle mass is m, yaw inertia is I_z, the distances from the center of mass to the front and rear axles are a and b, and the wheelbase is $L = a + b$. The center-of-mass height is denoted by h, the tire-road friction coefficient by μ, and the front and rear static normal loads by $F_{z,f0}$ and $F_{z,r0}$, consistent with the standard reduced-order bicycle-model setup used for force-balance vehicle dynamics [1, 2, 5]. Figure 2 provides the full four-wheel free-body layout used to interpret the force and moment balances in the model. The drawing serves as a geometric force map in the style used in vehicle-dynamics texts such as Pacejka's *Tire and Vehicle Dynamics* [6, Chapters 1–2] and Rajamani's *Vehicle Dynamics and Control* [5, Chapters 2–3]. The full-model notation in the figure is richer than the reduced planner model: it may distinguish per-wheel tire forces, multiple inertia components, angular deflection variables,

and load-transfer lever arms that do not appear explicitly in the planar single-track equations. The reduced model used for planning and inversion is obtained by summing wheel-pair forces and collapsing the rotational dynamics to the planar yaw channel, following the common kinematic/dynamic abstraction used in autonomous-driving control design [1, 2].

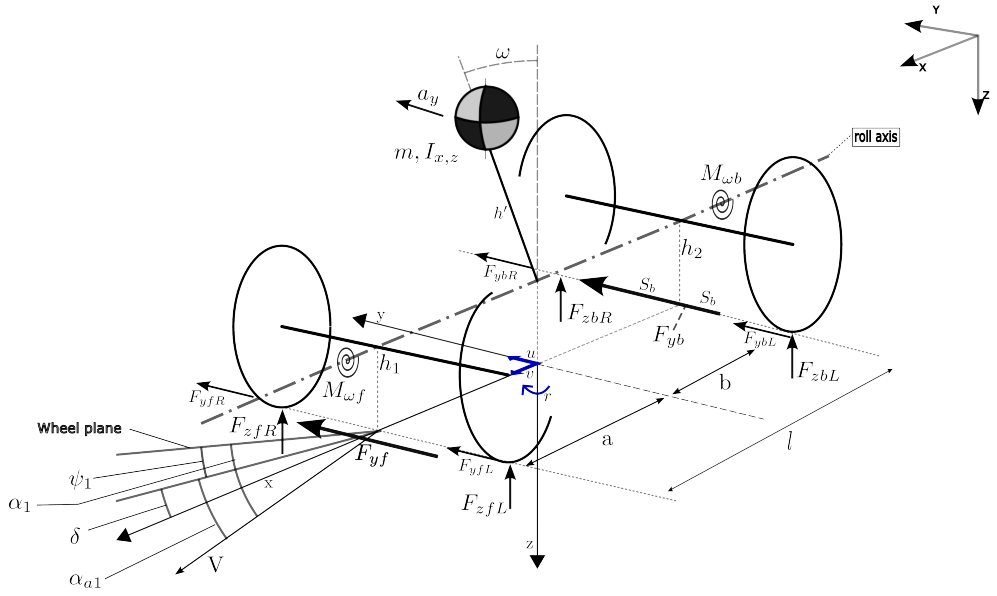


Figure 2: Four-wheel free-body diagram used to interpret the drift-capable vehicle model. The figure shows the body-fixed axes, steering angle, center-of-mass location, front/rear moment arms, load-transfer geometry, and the longitudinal, lateral, and normal force channels at the tire contact patches. The control-oriented model used later in the thesis is obtained by reducing this double-track force picture to equivalent front- and rear-axle resultant forces and a planar yaw moment balance.

The last two states are path-tracking error coordinates computed in the Frenet frame of the reference path. Here e is the signed lateral deviation (positive to the right of the path), and $\Delta\psi = \Phi - \psi_{\text{ref}}$ is the heading error relative to the local path tangent angle ψ_{ref}. The road curvature of the reference path is κ (positive for a right-turning path under the body-frame convention used here), and $\dot{s}$ is the path-progress speed, approximated as $\dot{s} \approx v_x$ for small $\Delta\psi$.

Four-wheel interpretation and reduction. Figure 2 should be read as the physical four-wheel or double-track model, in which each contact patch can carry its own longitudinal, lateral, and normal force components. In that setting one may use wheel-indexed quantities such as

$$F_{x,fl},\ F_{x,fr},\ F_{x,rl},\ F_{x,rr}, \qquad F_{y,fl},\ F_{y,fr},\ F_{y,rl},\ F_{y,rr}, \qquad F_{z,fl},\ F_{z,fr},\ F_{z,rl},\ F_{z,rr},$$

along with additional geometric or inertial variables that support roll, pitch, and load-transfer reasoning. This is the appropriate level for interpreting inside/outside tire unloading, brake-induced front loading, rear unloading during turn-in, and the force asymmetries that motivate drift-capable maneuvers.

The planner and inverse-controller model used in the present thesis is a reduced planar single-track model, obtained by summing left/right wheel pairs into equivalent front- and rear-axle resultants,

$$F_{x,f} = F_{x,fl} + F_{x,fr}, \qquad F_{x,r} = F_{x,rl} + F_{x,rr}, \tag{8}$$

$$F_{y,f} = F_{y,fl} + F_{y,fr}, \qquad F_{y,r} = F_{y,rl} + F_{y,rr}, \tag{9}$$

$$F_{z,f} = F_{z,fl} + F_{z,fr}, \qquad F_{z,r} = F_{z,rl} + F_{z,rr}, \tag{10}$$

and then retaining only the planar translational and yaw dynamics. In the reduced equations below, r denotes yaw rate, I_z is the yaw moment of inertia, and the front/rear axle resultants $(F_{x,f}, F_{y,f}, F_{z,f})$ and $(F_{x,r}, F_{y,r}, F_{z,r})$ carry the effect of the underlying four-wheel contact forces. The four-wheel model gives the physical interpretation layer; the single-track model is the control-oriented state-space form used for planning and inversion.

With the force layout of Figure 2, the state equations are

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\Phi} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{r} \\ \dot{e} \\ \Delta\dot{\psi} \end{bmatrix} = \begin{bmatrix} v_x \cos\Phi - v_y \sin\Phi \\ v_x \sin\Phi + v_y \cos\Phi \\ r \\ \dfrac{F_{x,f}\cos\delta - F_{y,f}\sin\delta + F_{x,r}}{m} + r v_y \\ \dfrac{F_{x,f}\sin\delta + F_{y,f}\cos\delta + F_{y,r}}{m} - r v_x \\ \dfrac{a(F_{x,f}\sin\delta + F_{y,f}\cos\delta) - b F_{y,r}}{I_z} \\ v_y + v_x \Delta\psi \\ r - \kappa \dot{s} \end{bmatrix}. \tag{11}$$

The path-error rows follow from the Frenet-frame kinematics: $\dot{e} = v_y + v_x \Delta\psi$ gives the lateral deviation rate under the small heading-error approximation, and $\Delta\dot{\psi} = r - \kappa\dot{s}$ is the yaw rate minus the rate at which the path tangent rotates.

The front and rear tire-frame velocities are

$$v^t_{x,f} = \sin\delta\,(v_y + ar) + \cos\delta\, v_x, \qquad v^t_{y,f} = \cos\delta\,(v_y + ar) - \sin\delta\, v_x, \tag{12}$$

$$v^t_{x,r} = v_x, \qquad v^t_{y,r} = v_y - br. \tag{13}$$

Sign and frame conventions. Throughout this work the vehicle body frame follows the convention shown in Figure 2: X points forward, Y points to the vehicle's right, and Z points downward. Yaw rate $r = \dot{\Phi}$ is positive for a right-turning, clockwise rotation about $+Z$, and steering angle δ is positive for a right steer. Slip angles α_f and α_r are defined as the tire velocity direction relative to the wheel-plane heading; under this definition, a positive slip angle produces a restoring lateral force in the negative Y direction in the linear

Fiala/Pacejka regime. Equivalently, when $v_y = r = 0$, a positive steering command gives a negative front slip angle and a positive rightward lateral force. The vehicle body-slip angle $\beta = \arctan(v_y/v_x)$ uses the body-frame velocity sign: positive β means the velocity vector points to the right of the vehicle nose.

Thus the front and rear slip angles are

$$\alpha_f = \arctan\left(\frac{\cos\delta\,(v_y + ar) - \sin\delta\,v_x}{\sin\delta\,(v_y + ar) + \cos\delta\,v_x}\right), \qquad \alpha_r = \arctan\left(\frac{v_y - br}{v_x}\right). \tag{14}$$

Equivalently,

$$\tan\alpha_f = \frac{\cos\delta\,(v_y + ar) - \sin\delta\,v_x}{\sin\delta\,(v_y + ar) + \cos\delta\,v_x}, \qquad \tan\alpha_r = \frac{v_y - br}{v_x}. \tag{15}$$

Figure 2 gives the global four-wheel force and moment layout, while Figure 3 isolates the axle-local slip geometry of the reduced single-track model. At each axle contact point, the velocity vector is set by the body-frame velocity and yaw rate. The slip angle is the angular difference between that velocity direction and the wheel-plane heading.

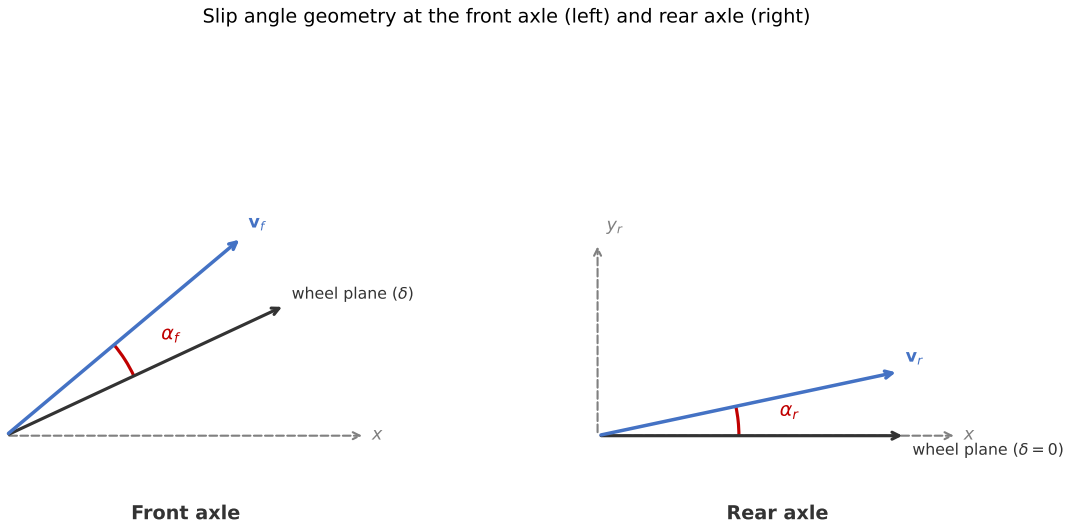


Figure 3: Slip angle geometry at the front axle (left) and rear axle (right). At the front, the slip angle α_f is measured between the steered wheel plane and the velocity vector $\mathbf{v}_f = (v^t_{x,f}, v^t_{y,f})$. At the rear, α_r is measured between the vehicle longitudinal axis ($\delta = 0$) and the rear contact-point velocity. The sign convention used in the benchmark scripts is checked by requiring positive steering at $v_y = r = 0$ to produce positive rightward lateral acceleration and positive yaw response.

Under longitudinal load transfer,

$$F_{z,f} = F_{z,f0} - \frac{mh}{L}a_x, \qquad F_{z,r} = F_{z,r0} + \frac{mh}{L}a_x. \tag{16}$$

This front–rear load shift corresponds directly to the normal-force channels shown in Figure 2: braking ($a_x < 0$) increases the front normal load and unloads the rear, while throttle does the

opposite. In the full four-wheel picture, the same mechanism is resolved into left/right wheel loads. The reduced single-track normal forces $F_{z,f}$ and $F_{z,r}$ should therefore be interpreted as the corresponding axle sums.

Define the Fiala stick-region operator

$$\mathcal{F}_{\text{stick}}(\tau, F_z) = -C_\alpha \tau + \frac{C_\alpha^2}{3\mu F_z} |\tau| \tau - \frac{C_\alpha^3}{27\mu^2 F_z^2} \tau^3, \tag{17}$$

and the slip-region operator

$$\mathcal{F}_{\text{slip}}(\alpha, F_z) = -\mu F_z \operatorname{sgn}(\alpha). \tag{18}$$

Using

$$\tau_f := \tan \alpha_f, \qquad \tau_r := \tan \alpha_r,$$

the front and rear lateral tire forces are selected according to the stick or slip regime at each axle.

The longitudinal tire forces are limited by the friction bounds

$$F_{x,f} = \operatorname{sgn}(F_{\text{in},f}) \min\left(|F_{\text{in},f}|, \sqrt{\max\left\{ (\mu F_{z,f})^2 - F_{y,f}^2,\, 0 \right\}} \right), \tag{19}$$

$$F_{x,r} = \operatorname{sgn}(F_{\text{in},r}) \min\left(|F_{\text{in},r}|, \sqrt{\max\left\{ (\mu F_{z,r})^2 - F_{y,r}^2,\, 0 \right\}} \right). \tag{20}$$

The state equations retain the same form in all regimes; only the definitions of $F_{y,f}$ and $F_{y,r}$ change. The four cases are given below.

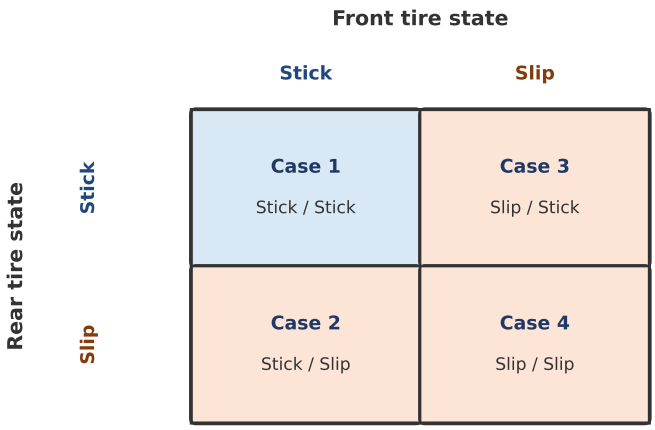


Figure 4: Combination of front and rear tire operating regimes used in the dynamic model.

Case 1: Front stick, rear stick

Here

$$|\alpha_f| < \arctan\left(\frac{3\mu F_{z,f}}{C_\alpha}\right), \qquad |\alpha_r| < \arctan\left(\frac{3\mu F_{z,r}}{C_\alpha}\right).$$

Thus

$$F_{y,f} = \mathcal{F}_{\text{stick}}(\tau_f, F_{z,f}), \qquad F_{y,r} = \mathcal{F}_{\text{stick}}(\tau_r, F_{z,r}), \tag{21}$$

or explicitly,

$$\begin{aligned} F_{y,f} = &-C_\alpha \frac{\cos\delta\,(v_y + ar) - \sin\delta\, v_x}{\sin\delta\,(v_y + ar) + \cos\delta\, v_x} \\ &+ \frac{C_\alpha^2}{3\mu\left(F_{z,f0} - \frac{mh}{L}a_x\right)} \left|\frac{\cos\delta\,(v_y + ar) - \sin\delta\, v_x}{\sin\delta\,(v_y + ar) + \cos\delta\, v_x}\right| \frac{\cos\delta\,(v_y + ar) - \sin\delta\, v_x}{\sin\delta\,(v_y + ar) + \cos\delta\, v_x} \\ &- \frac{C_\alpha^3}{27\mu^2\left(F_{z,f0} - \frac{mh}{L}a_x\right)^2} \left(\frac{\cos\delta\,(v_y + ar) - \sin\delta\, v_x}{\sin\delta\,(v_y + ar) + \cos\delta\, v_x}\right)^3, \end{aligned} \tag{22}$$

$$\begin{aligned} F_{y,r} = &-C_\alpha \frac{v_y - br}{v_x} + \frac{C_\alpha^2}{3\mu\left(F_{z,r0} + \frac{mh}{L}a_x\right)} \left|\frac{v_y - br}{v_x}\right| \frac{v_y - br}{v_x} \\ &- \frac{C_\alpha^3}{27\mu^2\left(F_{z,r0} + \frac{mh}{L}a_x\right)^2} \left(\frac{v_y - br}{v_x}\right)^3. \end{aligned} \tag{23}$$

Here both axles generate unsaturated lateral force, so β remains small and evolves smoothly. This regime corresponds to normal driving.

Case 2: Front stick, rear slip

Here

$$|\alpha_f| < \arctan\left(\frac{3\mu F_{z,f}}{C_\alpha}\right), \qquad |\alpha_r| \geq \arctan\left(\frac{3\mu F_{z,r}}{C_\alpha}\right).$$

Thus

$$F_{y,f} = \mathcal{F}_{\text{stick}}(\tau_f, F_{z,f}), \qquad F_{y,r} = \mathcal{F}_{\text{slip}}(\alpha_r, F_{z,r}), \tag{24}$$

that is,

$$\begin{aligned} F_{y,f} = &-C_\alpha \frac{\cos\delta\,(v_y + ar) - \sin\delta\, v_x}{\sin\delta\,(v_y + ar) + \cos\delta\, v_x} \\ &+ \frac{C_\alpha^2}{3\mu\left(F_{z,f0} - \frac{mh}{L}a_x\right)} \left|\frac{\cos\delta\,(v_y + ar) - \sin\delta\, v_x}{\sin\delta\,(v_y + ar) + \cos\delta\, v_x}\right| \frac{\cos\delta\,(v_y + ar) - \sin\delta\, v_x}{\sin\delta\,(v_y + ar) + \cos\delta\, v_x} \\ &- \frac{C_\alpha^3}{27\mu^2\left(F_{z,f0} - \frac{mh}{L}a_x\right)^2} \left(\frac{\cos\delta\,(v_y + ar) - \sin\delta\, v_x}{\sin\delta\,(v_y + ar) + \cos\delta\, v_x}\right)^3, \end{aligned} \tag{25}$$

$$F_{y,r} = -\mu\left(F_{z,r0} + \frac{mh}{L}a_x\right)\operatorname{sgn}(\alpha_r). \tag{26}$$

Here, rear lateral force saturates first, reducing rear stabilizing authority and tending to increase β more aggressively. This is the oversteer regime. Deliberately driving the transition from Case 1 to Case 2 can allow faster and tighter turns in some racing conditions. That possibility motivates trajectory planning and control designs that can use **Controlled Oversteer** without losing recoverability on the race track.

Case 3: Front slip, rear stick

Here

$$|\alpha_f| \geq \arctan\left(\frac{3\mu F_{z,f}}{C_\alpha}\right), \qquad |\alpha_r| < \arctan\left(\frac{3\mu F_{z,r}}{C_\alpha}\right).$$

Thus

$$F_{y,f} = \mathcal{F}_{\text{slip}}(\alpha_f, F_{z,f}), \qquad F_{y,r} = \mathcal{F}_{\text{stick}}(\tau_r, F_{z,r}), \tag{27}$$

that is,

$$F_{y,f} = -\mu\left(F_{z,f0} - \frac{mh}{L}a_x\right)\text{sgn}(\alpha_f), \tag{28}$$

$$\begin{aligned} F_{y,r} = &-C_\alpha\frac{v_y - br}{v_x} + \frac{C_\alpha^2}{3\mu\left(F_{z,r0} + \frac{mh}{L}a_x\right)}\left|\frac{v_y - br}{v_x}\right|\frac{v_y - br}{v_x} \\ &- \frac{C_\alpha^3}{27\mu^2\left(F_{z,r0} + \frac{mh}{L}a_x\right)^2}\left(\frac{v_y - br}{v_x}\right)^3. \end{aligned} \tag{29}$$

Here, front lateral force saturates first, limiting front turn-in authority and typically producing larger path-tracking error with more gradual growth in β. This is the understeer regime. In racing, a driver may mitigate it by briefly applying the brakes to shift load forward and increase steering authority.

Case 4: Front slip, rear slip

Here

$$|\alpha_f| \geq \arctan\left(\frac{3\mu F_{z,f}}{C_\alpha}\right), \qquad |\alpha_r| \geq \arctan\left(\frac{3\mu F_{z,r}}{C_\alpha}\right).$$

Thus

$$F_{y,f} = \mathcal{F}_{\text{slip}}(\alpha_f, F_{z,f}), \qquad F_{y,r} = \mathcal{F}_{\text{slip}}(\alpha_r, F_{z,r}), \tag{30}$$

or explicitly,

$$F_{y,f} = -\mu\left(F_{z,f0} - \frac{mh}{L}a_x\right)\text{sgn}(\alpha_f), \qquad F_{y,r} = -\mu\left(F_{z,r0} + \frac{mh}{L}a_x\right)\text{sgn}(\alpha_r). \tag{31}$$

In all four cases, the state equation remains

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\Phi} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{r} \\ \dot{e} \\ \Delta\dot{\psi} \end{bmatrix} = \begin{bmatrix} v_x \cos\Phi - v_y \sin\Phi \\ v_x \sin\Phi + v_y \cos\Phi \\ r \\ \dfrac{F_{x,f}\cos\delta - F_{y,f}\sin\delta + F_{x,r}}{m} + r v_y \\ \dfrac{F_{x,f}\sin\delta + F_{y,f}\cos\delta + F_{y,r}}{m} - r v_x \\ \dfrac{a(F_{x,f}\sin\delta + F_{y,f}\cos\delta) - bF_{y,r}}{I_z} \\ v_y + v_x \Delta\psi \\ r - \kappa \dot{s} \end{bmatrix}, \tag{32}$$

with $F_{y,f}$ and $F_{y,r}$ selected according to the active stick-slip regime.

In Case 4, both axles saturate, so the evolution of β is determined primarily by the friction envelope rather than tire cornering stiffness. This regime is strongly nonlinear, and additional steering input produces diminishing lateral force response. A sudden reduction in surface friction (for example, black ice) may cause rapid transitions from Cases 1–3 into Case 4 as slip angles exceed the stick threshold. Aggressive corrective inputs, such as simultaneous steering and braking, typically increase tire force demand and accelerate growth in β, often leading to spinout. Smooth inputs and wheel alignment with the velocity vector reduce tire force demand and promote recovery of lateral stability.

2.4 Enhanced Bicycle Model

The kinematic and dynamic models omit several physical effects that grow in importance at higher speeds and lateral accelerations. The enhanced bicycle model used in simulation adds these effects progressively on top of the planar bicycle model, with each effect enabled or disabled through the simulation configuration. The tire-force layer can use Pacejka Magic Formula (MF 5.2), Dugoff, or a linear cornering stiffness approximation, depending on the level of tire detail needed for a given run. Quasi-static lateral load transfer, $\Delta F_z = Mha_y/W$, redistributes normal load across the left and right sides under lateral acceleration, which changes the local friction limit at each tire. When roll dynamics are enabled, the model adds the states $(\phi, \dot{\phi})$ and couples roll angle into the load-transfer calculation, capturing the delay between lateral acceleration onset and full weight transfer. At higher speeds, the longitudinal force balance also includes quadratic aerodynamic drag, $F_{\text{aero}} = \frac{1}{2} C_d A \rho v_x^2$. Finally, the steering channel can include a first-order actuator lag, $\dot{\delta} = (\delta_{\text{cmd}} - \delta)/\tau$, so the simulated steering angle follows the commanded steering angle with a finite response time.

Figure 5 illustrates how these layers compose on top of the base bicycle model.

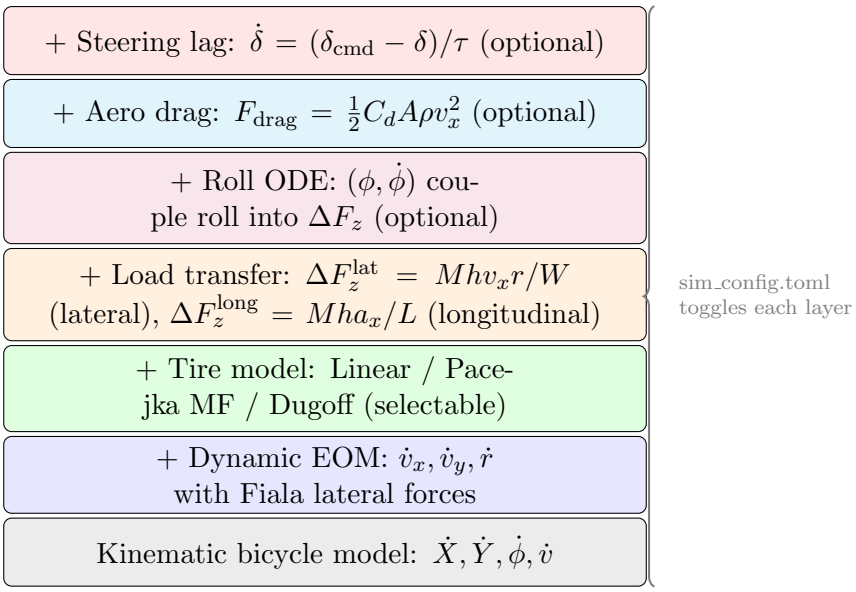


Figure 5: Enhanced bicycle model layer stack. Each row adds physics on top of the previous layer; all additions are individually configurable. The tire model selection (linear / Pacejka / Dugoff) and the optional roll, aero, and steering-lag layers can be enabled independently for ablation studies.

The three tire models in the enhanced bicycle model all saturate lateral force nonlinearly, each at a different level of fidelity. The **linear** model uses $F_y = C_\alpha \alpha$, which is valid only for small slip angles. The **Pacejka Magic Formula** [6] (MF 5.2) is

$$F_y = D \sin\Big(C \arctan\big(B\alpha - E(B\alpha - \arctan(B\alpha))\big)\Big), \tag{33}$$

where $D = \mu F_z$ is the peak force, C is the shape factor, B is the stiffness factor such that the initial slope equals BCD, and E controls the curvature at the transition to saturation. To capture the dependence of tire stiffness on vertical load, the stiffness factor is scaled as

$$B(F_z) = B_{\text{nom}} \left(\frac{F_{z,\text{nom}}}{F_z} \right)^p, \tag{34}$$

where the exponent p controls how strongly the tire stiffness changes with normal load. The **Dugoff** model [7] provides a more compact saturation law:

$$F_y = -C_\alpha \tan\alpha \cdot f(\lambda), \qquad f(\lambda) = \begin{cases} (2-\lambda)\lambda & \lambda < 1 \\ 1 & \lambda \geq 1 \end{cases}, \qquad \lambda = \frac{\mu F_z}{2C_\alpha |\tan\alpha|}, \tag{35}$$

where λ is a normalized saturation index. The quasi-static load transfer approximation uses $a_y \approx v_x r$ (centripetal acceleration), so the effective normal loads fed into Equations (33)–(35) are $F_{z,f} = F_{z,f0} - \Delta F_z$ and $F_{z,r} = F_{z,r0} + \Delta F_z$, with $\Delta F_z = Mhv_x r/W$.

Table 1 lists the parameter values used in the enhanced-model simulation studies.

Table 1: Default vehicle and enhanced-model parameters used in simulation.

Parameter	**Symbol**	**Value**	**Unit**
Vehicle Geometry			
Wheelbase	L	2.500	m
CG to front axle	a	1.250	m
CG to rear axle	b	1.250	m
Track width	W	1.400	m
CG height	h	0.500	m
Mass and Inertia			
Vehicle mass	M	1500	kg
Yaw inertia	I_z	2500	kg·m^2
Roll inertia	I_x	500	kg·m^2
Tire Properties			
Front cornering stiffness	$C_{\alpha f}$	80 000	N/rad
Rear cornering stiffness	$C_{\alpha r}$	90 000	N/rad
Peak friction coefficient	μ	0.90	–
Pacejka shape factor	C	1.90	–
Pacejka curvature factor	E	0.97	–
Load sensitivity exponent	p	0.30	–
Enhanced Model Additions			
Roll stiffness	K_ϕ	25 000	N·m/rad
Roll damping	D_ϕ	2 000	N·m·s/rad
Aero drag coeff.	C_d	0.35	–
Frontal area	A	2.20	m^2
Air density	ρ	1.225	kg/m^3
Steering lag	τ	0.05	s
Control Limits			
Max steering angle	$\delta_{\max}$	± 24	deg
Acceleration range	$[a_{\min}, a_{\max}]$	$[-3.0,\ 2.5]$	m/s^2

This enhanced model provides a physics baseline between the pure bicycle model and the full MuJoCo simulation. It also serves as the starting point for data-driven residual identification.

Implementation note: slip angle conventions. The enhanced-model implementation uses the small-steering-angle approximation for the front slip angle,

$$\alpha_f = \delta - \arctan\left(\frac{v_y + ar}{v_x}\right), \tag{36}$$

which is the linearisation of the exact formula in Section 2 for $\cos\delta \approx 1$, $\sin\delta \approx \delta$. With the $+Y$ right convention, a positive right steer from $v_y = r = 0$ gives the correct positive rightward response after the corresponding tire-force sign is applied. For the rear axle the convention is

$$\alpha_r = -\arctan\left(\frac{v_y - br}{v_x}\right), \tag{37}$$

where the leading minus sign keeps the rear slip variable paired with the restoring-force convention used by the tire model. Load transfer uses the quasi-static centripetal approximation $a_y \approx v_x r$, giving $\Delta F_z = Mhv_x r/W$. A higher-fidelity implementation would replace $v_x r$ with the body-frame lateral acceleration $\dot{v}_y + v_x r$. Both conventions produce the same result at steady cornering ($\dot{v}_y = 0$).

The reported scripts do not all store slip angle with the same intermediate sign. Some use $\alpha = \delta - \arctan((v_y + ar)/v_x)$ and a restoring-force convention, while the sweep implementation stores the opposite signed angle and pairs it with the corresponding tire-force law. The thesis uses the physically verified input-output convention throughout: a positive steering command at $v_y = r = 0$ must produce positive rightward lateral acceleration and positive yaw acceleration in the script that generated the reported table. The active sweep code now includes this sign-convention smoke test.

3 Tire Models

3.1 Fiala Tire Model

In the tire coordinate frame, the Fiala lateral-force approximation follows the classical tire-force formulation used by Fiala, Dugoff, and Pacejka [6–8].

$$Fy = \begin{cases} -C_\alpha \tan(\alpha) + \frac{C_\alpha^2}{3\mu F_z} |\tan(\alpha)| \tan(\alpha) - \frac{C_\alpha^3}{27\mu^2 F_z^2} \tan^3(\alpha); & |\alpha| < arctan(\frac{3\mu F_z}{C_\alpha}) \\ -\mu F_z sgn(\alpha); & \text{otherwise} \end{cases} \tag{38}$$

Where:
C_α is the tire cornering stiffness,
μ is the tire-road coefficient of friction,
α is the slip angle, and
F_z is the normal tire force.

The slip angle in the tire reference frame is defined as

$$\alpha = arctan(\frac{v_{y,tire}}{v_{x,tire}})$$

or

$$tan(\alpha) = \frac{v_{y,tire}}{v_{x,tire}}$$

This ratio is an intermediate variable that can be substituted directly into the Fiala tire model.

For the front tire frame, this gives the following lateral-force expression.

$$Fy = \begin{cases} -C_\alpha \frac{v_{y,tire}}{v_{x,tire}} + \frac{C_\alpha^2}{3\mu F_z} |\frac{v_{y,tire}}{v_{x,tire}}| \frac{v_{y,tire}}{v_{x,tire}} - \frac{C_\alpha^3}{27\mu^2 F_z^2} [\frac{v_{y,tire}}{v_{x,tire}}]^3; & |\frac{v_{y,tire}}{v_{x,tire}}| < \frac{3\mu F_z}{C_\alpha} \\ -\mu F_z sgn(arctan(\frac{v_{y,tire}}{v_{x,tire}})); & \text{otherwise} \end{cases} \tag{39}$$

The transformation from the tire frame to the vehicle frame uses the standard rotation matrix.

For the front tire set, this gives

$$\begin{bmatrix} F_{y,f} \\ F_{x,f} \end{bmatrix} = \begin{bmatrix} \cos\delta & \sin\delta \\ -\sin\delta & \cos\delta \end{bmatrix} \begin{bmatrix} F_{y,front\,tire} \\ F_{x,front\,tire} \end{bmatrix} \tag{40}$$

For the rear tire set, the steering rotation is the identity transformation.

$$\begin{bmatrix} F_{y,r} \\ F_{x,r} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} F_{y,rear\,tire} \\ F_{x,rear\,tire} \end{bmatrix}$$
$$\implies \begin{bmatrix} F_{y,r} \\ F_{x,r} \end{bmatrix} = \begin{bmatrix} F_{y,rear\,tire} \\ F_{x,rear\,tire} \end{bmatrix} \tag{41}$$

The same frame transformation applies to velocity:

$$\begin{bmatrix} v_{x,front\,tire} \\ v_{y,front\,tire} \end{bmatrix} = \begin{bmatrix} \cos\delta & \sin\delta \\ -\sin\delta & \cos\delta \end{bmatrix} \begin{bmatrix} v_x \\ v_y + ar \end{bmatrix} \tag{42}$$

Reordering the velocity components gives:

$$\begin{bmatrix} v_{y,front\,tire} \\ v_{x,front\,tire} \end{bmatrix} = \begin{bmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{bmatrix} \begin{bmatrix} v_y + ar \\ v_x \end{bmatrix} \tag{43}$$

Multiplying through yields

$$\begin{bmatrix} v_{y,front\,tire} \\ v_{x,front\,tire} \end{bmatrix} = \begin{bmatrix} \cos\delta(v_y + ar) - \sin\delta\, v_x \\ \sin\delta(v_y + ar) + \cos\delta\, v_x \end{bmatrix} \tag{44}$$

The front slip angle is therefore

$$tan(\alpha_f) = \frac{\cos\delta(v_y + ar) - \sin\delta\, v_x}{\sin\delta(v_y + ar) + \cos\delta\, v_x}$$

For the rear tire set, the velocity components are

$$\begin{bmatrix} v_{y,rear\,tire} \\ v_{x,rear\,tire} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_y - br \\ v_x \end{bmatrix} \implies \begin{bmatrix} v_{y,rear\,tire} \\ v_{x,rear\,tire} \end{bmatrix} = \begin{bmatrix} v_y - br \\ v_x \end{bmatrix} \tag{45}$$

The front lateral force in the tire frame becomes

$$F_{y,front\,tire} = \begin{cases} -C_\alpha \frac{\cos\delta(v_y+ar)-\sin\delta\, v_x}{\sin\delta(v_y+ar)+\cos\delta\, v_x} + \frac{C_\alpha^2}{3\mu F_z} |\frac{\cos\delta(v_y+ar)-\sin\delta\, v_x}{\sin\delta(v_y+ar)+\cos\delta\, v_x}| \frac{\cos\delta(v_y+ar)-\sin\delta\, v_x}{\sin\delta(v_y+ar)+\cos\delta\, v_x} & \\ -\frac{C_\alpha^3}{27\mu^2 F_z^2} [\frac{\cos\delta(v_y+ar)-\sin\delta\, v_x}{\sin\delta(v_y+ar)+\cos\delta\, v_x}]^3; & |\frac{v_{y,tire}}{v_{x,tire}}| < \frac{3\mu F_z}{C_\alpha} \\ -\mu F_z sgn(arctan(\frac{\cos\delta(v_y+ar)-\sin\delta\, v_x}{\sin\delta(v_y+ar)+\cos\delta\, v_x})); & \text{otherwise} \end{cases} \tag{46}$$

Similarly, the rear lateral force in the tire frame becomes:

$$F_{y,rear\,tire} = \begin{cases} -C_\alpha \frac{v_y - br}{v_x} + \frac{C_\alpha^2}{3\mu F_z} |\frac{v_y - br}{v_x}| \frac{v_y - br}{v_x} - \frac{C_\alpha^3}{27\mu^2 F_z^2} [\frac{v_y - br}{v_x}]^3; & |\frac{v_y - br}{v_x}| < \frac{3\mu F_z}{C_\alpha} \\ -\mu F_z sgn(arctan(\frac{v_y - br}{v_x})); & \text{otherwise} \end{cases} \tag{47}$$

3.2 Roll Dynamics and Load Transfer

For the bicycle model, longitudinal load transfer is represented using the standard quasi-static weight-transfer approximation from vehicle-dynamics texts [5, 6].

$$\begin{aligned} F_{z,f} &= F_{z,f0} - \Delta F_z, \qquad & F_{z,r} &= F_{z,r0} + \Delta F_z, \\ \Delta F_z &= \frac{h}{L} F_{x,\mathrm{cm}} = \frac{mh}{L} a_x. \end{aligned} \tag{48}$$

Forward acceleration unloads the front axle and loads the rear axle; deceleration reverses this effect.

This formulation is still vulnerable to wheel spin, where the requested force input F_{input} or throttle setting may not match the longitudinal force actually generated at the contact patch.

The input force can be split between the front and rear wheel sets using a weight w:

$$F_{\mathrm{in}} = F_{\mathrm{in},f} + F_{\mathrm{in},r}, \qquad F_{\mathrm{in},f} = w\, F_{\mathrm{in}}, \qquad F_{\mathrm{in},r} = (1-w)\, F_{\mathrm{in}},$$

For example, front wheel drive (FWD) uses $w = 1$, while rear wheel drive (RWD) uses $w = 0$. For the experimental vehicle, this work uses $w = 0.5$, an all-wheel-drive (AWD) approximation for a drivetrain that applies the same motor rotation to the wheels.

Each axle can generate only a friction-limited longitudinal force because the contact patch obeys Coulomb friction. In 2D, this limit is the *friction circle*

$$\sqrt{F_x^2 + F_y^2} \;\le\; \mu\, F_z,$$

which is the planar cross-section of the 3D friction cone at the wheel. The enhanced bicycle model also includes lateral load transfer. During cornering, centrifugal inertia shifts normal load from the inner tires to the outer tires. Under the quasi-static approximation $a_y \approx v_x r$, the lateral shift is

$$\Delta F_z^{\mathrm{lat}} = \frac{M h_{\mathrm{cg}}\, v_x r}{W}, \tag{49}$$

where W is the track width. Figure 6 illustrates this redistribution, and Figure 7 shows the resulting friction-circle constraint at each tire. When a tire must also produce lateral force F_y, the available longitudinal force budget is reduced according to

$$|F_{x,\{f,r\}}| \;\le\; F_{x,\max,\{f,r\}} \;\triangleq\; \sqrt{(\mu F_{z,\{f,r\}})^2 - F_{y,\{f,r\}}^2} \tag{50}$$

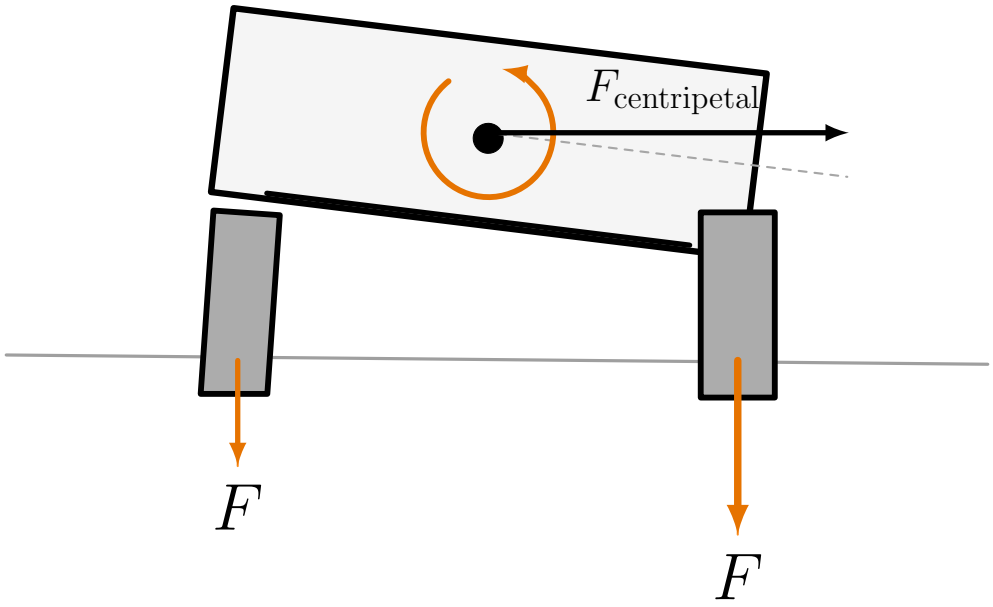


Figure 6: Lateral load transfer during cornering. The centripetal acceleration $a_y \approx v_x r$ shifts normal load $\Delta F_z^{\text{lat}} = M h_{\text{cg}} v_x r / W$ from the inner tire (left, lightly loaded, red) to the outer tire (right, heavily loaded, green). This redistribution reduces the inner tire's friction limit and increases the outer tire's.

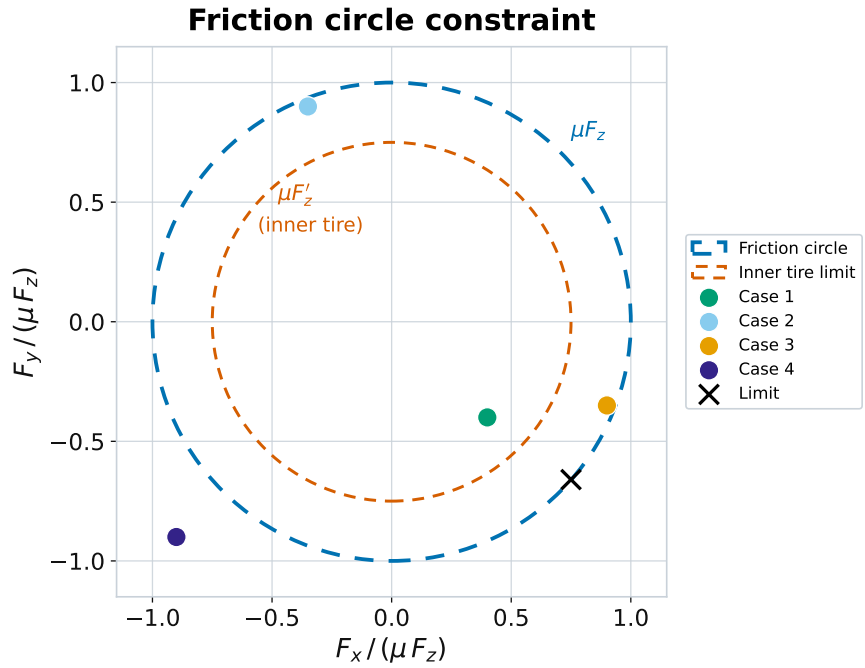


Figure 7: Friction circle constraint. Each tire's combined force vector (F_x, F_y) must remain within the friction circle of radius μF_z, while the inner tire's limit shrinks to $\mu F_z'$ after lateral load transfer. The plotted cases show representative operating points: Case 1 remains comfortably inside the limit, Case 2 and Case 4 show other feasible combined-slip configurations, and Case 3 lies near the full friction boundary.

3.3 Combined Friction Formulation

Substituting the load-transfer expressions $F_{z,f} = F_{z,f0} - \frac{mh}{L} a_x$ and $F_{z,r} = F_{z,r0} + \frac{mh}{L} a_x$ into the Fiala model, and using $\tau_f := \tan \alpha_f$ and $\tau_r := \tan \alpha_r$ from the slip-angle definitions above, the lateral force in the front tire frame can be written as a combined-slip approximation consistent with the Dugoff/Fiala tire literature [6–8]:

$$F_{y,f}^{\text{tire}} = \begin{cases} -C_\alpha \tau_f + \dfrac{C_\alpha^2}{3\mu F_{z,f}} |\tau_f| \tau_f - \dfrac{C_\alpha^3}{27\mu^2 F_{z,f}^2} \tau_f^3, & |\tau_f| < \dfrac{3\mu F_{z,f}}{C_\alpha}, \\ -\mu F_{z,f} \operatorname{sgn}(\alpha_f), & \text{otherwise}, \end{cases} \tag{51}$$

where $F_{z,f} = F_{z,f0} - \frac{mh}{L} a_x$. The corresponding lateral force in the rear tire frame is

$$F_{y,r}^{\text{tire}} = \begin{cases} -C_\alpha \tau_r + \dfrac{C_\alpha^2}{3\mu F_{z,r}} |\tau_r| \tau_r - \dfrac{C_\alpha^3}{27\mu^2 F_{z,r}^2} \tau_r^3, & |\tau_r| < \dfrac{3\mu F_{z,r}}{C_\alpha}, \\ -\mu F_{z,r} \operatorname{sgn}(\alpha_r), & \text{otherwise}, \end{cases} \tag{52}$$

where $F_{z,r} = F_{z,r0} + \frac{mh}{L} a_x$.

The slip-region branch uses $\operatorname{sgn}(\alpha)$ because $\operatorname{sgn}(\arctan(\tau)) = \operatorname{sgn}(\tau)$ for all finite τ. The vehicle-frame forces $F_{y,f}$ and $F_{y,r}$ follow from the tire-to-vehicle rotation matrices given in the Fiala section above.

3.4 Slip Angle Analysis

The slip angle β at the center of mass is the standard vehicle-body velocity angle used in bicycle-model derivations [1, 5]:

$$\beta = \tan^{-1} \frac{v_y}{v_x} \tag{53}$$

where v_x is the longitudinal velocity of the car in the body frame and v_y is the lateral velocity in the body frame.

The time derivative of the slip angle describes how the vehicle heading evolves relative to its velocity vector. This relationship becomes important when analysing understeer and oversteer behaviour in later sections.

$$\begin{aligned} \dot{\beta} &= \frac{d \tan^{-1} \frac{v_y}{v_x}}{dt} \\ &= \frac{1}{1 + (\frac{v_y}{v_x})^2} \frac{\dot{v_y}}{v_x} \\ &= \frac{1}{1 + (\frac{v_y}{v_x})^2} \frac{(v_x \dot{v_y} - v_y \dot{v_x})}{v_x^2} \\ &= \frac{(v_x \dot{v_y} - v_y \dot{v_x})}{v_x^2 + v_y^2} \end{aligned} \tag{54}$$

Substituting $\dot{v_y} = (a_y - \dot{\phi} v_x)$ and $\dot{v_x} = (a_x + \dot{\phi} v_y)$ gives

$$\begin{aligned} \dot{\beta} &= \frac{(v_x(a_y - \dot{\phi} v_x) - v_y(a_x + \dot{\phi} v_y))}{v_x^2 + v_y^2} \\ &= \frac{v_x a_y - v_x \dot{\phi} v_x - v_y a_x - v_y \dot{\phi} v_y}{v_x^2 + v_y^2} \\ &= \frac{v_x a_y - v_y a_x - \dot{\phi} v_y^2 - \dot{\phi} v_x^2}{v_x^2 + v_y^2} \\ &= \frac{v_x a_y - v_y a_x}{v_x^2 + v_y^2} - \dot{\phi} \end{aligned} \tag{55}$$

$$\boxed{\dot{\beta} = \frac{v_x a_y - v_y a_x}{v_x^2 + v_y^2} - \dot{\phi}}$$

This result gives the direction-of-travel relation $\theta = \phi + \beta$. The corresponding rate of change is

$$\begin{aligned} \dot{\theta} &= \dot{\phi} + \dot{\beta} \\ &= \dot{\phi} + \frac{v_x a_y - v_y a_x}{v_x^2 + v_y^2} - \dot{\phi} \qquad \text{from eq 55} \\ &= \frac{v_x a_y - v_y a_x}{v_x^2 + v_y^2} \end{aligned} \tag{56}$$

The accelerations can then be expressed through the forces acting at the center of mass.

$$F_{x_{cm}} = m a_x \qquad : \qquad F_{y_{cm}} = m a_y$$

$$\begin{aligned} F_{x_{cm}} &= F_{x,front\,tire} \cos\delta - F_{y,front\,tire} \sin\delta + F_{x,rear\,tire} \\ F_{y_{cm}} &= F_{x,front\,tire} \sin\delta + F_{y,front\,tire} \cos\delta + F_{y,rear\,tire} \end{aligned}$$

4 Controllers

This section covers several steering strategies for trajectory tracking. Each method reflects a different modeling perspective, ranging from purely geometric correction to model-informed prediction [9–12]. Comparing their assumptions and limitations provides context for the iterative inversion approach developed later.

Figure 8 organises the controllers into four families according to the information they use: purely geometric methods that operate on path errors without a vehicle model; kinematic inversion methods that exploit the bicycle model equations; model-predictive and sampling-based optimisation methods; and hybrid methods that blend feedforward and feedback paths.

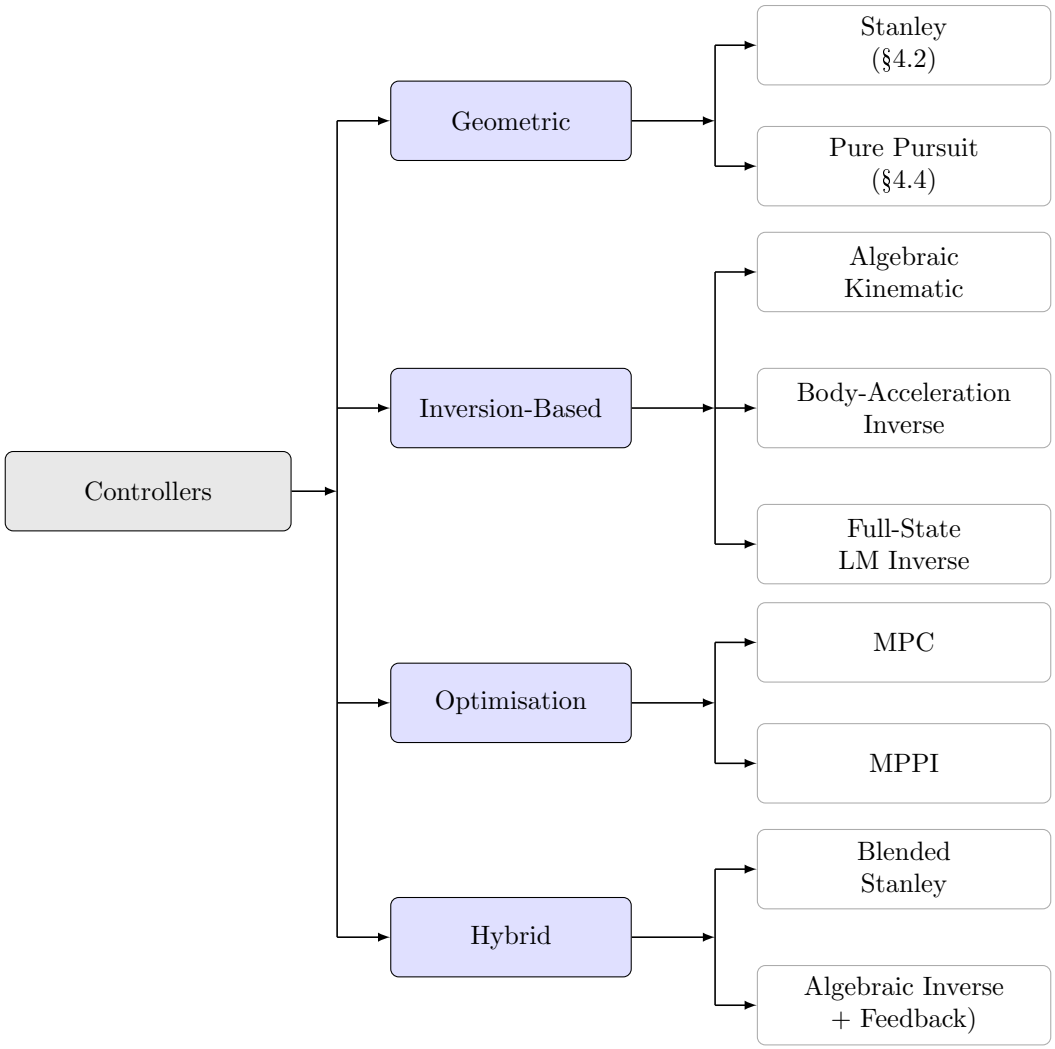


Figure 8: Taxonomy of the steering controllers evaluated in this work. Geometric controllers use path-error signals alone. Inversion-based controllers derive steering from a vehicle model. Optimisation-based controllers solve a receding-horizon problem. Hybrid controllers combine feedforward prediction with geometric or error-based feedback.

4.1 Algebraic Kinematic Inverse

Consider the planar kinematic bicycle model in 1.

$$\dot{x} = v\cos\phi, \qquad \dot{y} = v\sin\phi, \qquad \dot{\phi} = \frac{v}{L}\tan\psi, \tag{57}$$

Naive inertial-coordinate inversion (fails). A direct attempt to recover v from the inertial velocity components gives two conflicting expressions:

$$\begin{bmatrix} v \\ v \end{bmatrix} = \begin{bmatrix} \frac{1}{cos(\phi)}\dot{x} \\ \frac{1}{sin(\phi)}\dot{y} \end{bmatrix} \tag{58}$$

This formulation is ill-defined whenever $\cos\phi = 0$ or $\sin\phi = 0$. It also produces conflicting values for v unless the motion lies exactly along a coordinate axis. Thus, there are no valid answers for $\phi = \{0, \frac{n\pi}{2}\}$ where $n \in 1, 2, 3, ...$

Alternatively, the input velocity can be computed as:

$$\begin{bmatrix} v \end{bmatrix} = \begin{bmatrix} \pm\sqrt{\dot{x}^2 + \dot{y}^2} \end{bmatrix} \tag{59}$$

This removes the singularity but discards directionality and gives two roots. Since v is a signed scalar encoding forward or reverse motion, this mapping is not one-to-one and therefore unsuitable for inversion-based control.

Corrected body-frame inversion (well-posed). Both failure modes above are resolved by projecting the inertial velocity onto the vehicle's body frame before inverting:

$$v = \dot{x}\cos(\phi) + \dot{y}\sin\phi \tag{60}$$

In this form, there are no singularities at $\phi = \{0, \frac{n\pi}{2}\}$, and the velocity retains an associated direction. Using this velocity, the steering angle can be recovered algebraically as

$$\psi = \arctan\left(\frac{L\dot{\phi}}{\dot{x}\cos\phi + \dot{y}\sin\phi}\right). \tag{61}$$

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} \dot{x}\cos(\phi) + \dot{y}\sin\phi \\ \dot{\phi} \end{bmatrix} \tag{62}$$

This gives the corresponding velocity and steering inputs:

$$\begin{bmatrix} v \\ \psi \end{bmatrix} = \begin{bmatrix} \dot{x}\cos(\phi) + \dot{y}\sin\phi \\ \arctan\left(\frac{L\dot{\phi}}{\dot{x}\cos\phi + \dot{y}\sin\phi}\right) \end{bmatrix} \tag{63}$$

This representation defines a locally well-posed inverse of the kinematic bicycle model in the low-speed regime. For any sufficiently smooth trajectory $(x(t), y(t), \phi(t))$ satisfying $v(t) \neq 0$, the corresponding realizable velocity and steering inputs can be recovered uniquely and algebraically from the state derivatives.

4.2 Stanley Controller

The Stanley control law [9] regulates lateral tracking by combining heading alignment with lateral error correction. The steering command is defined as

$$\psi_{\text{stanley}} = \theta_e + \arctan\left(\frac{k \cdot \text{CTE}}{k_{\text{soft}} + v}\right), \tag{64}$$

where

- $\theta_e = \phi_{\text{ref}} - \theta$ denotes the heading error,
- CTE is the signed **C**ross-**T**rack **E**rror measured relative to the path,
- k is the gain applied to lateral correction,

- k_{soft} prevents excessive steering at low speeds. As $v \to 0$, $\tan^{-1}(\infty) \to \pm\pi/2$

The first term aligns the vehicle orientation with the local path tangent. The second term adds a velocity-scaled lateral correction that drives the vehicle toward the reference path. Figure 9 illustrates the key geometric quantities.

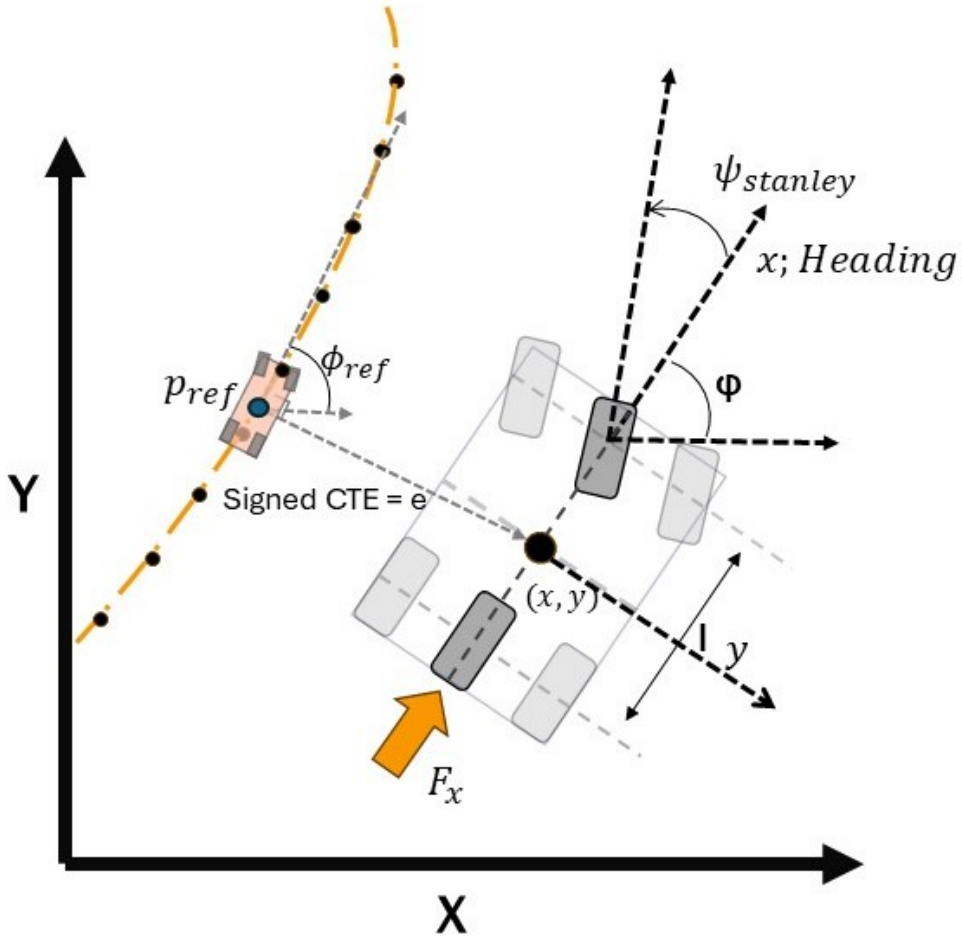


Figure 9: Stanley controller geometry. The front axle is projected onto the nearest reference-path point p_{ref} to compute the signed cross-track error e. The heading error $\theta_e = \phi_{\text{ref}} - \theta$ is the angular difference between the path tangent direction and the vehicle heading. The Stanley steering command ψ_{stanley} combines both signals.

Assumptions. This formulation implicitly assumes that steering directly determines curvature. Because it is based on the kinematic model, the same assumptions apply: pure rolling, no lateral slip, low speed, and negligible load-transfer effects. Tire forces and friction limits are not modelled.

Advantages. The controller requires only geometric information about the path and the vehicle pose. This simplicity gives it low computational cost and predictable lateral conver-

gence in nominal operating conditions. Relying only on relative position and orientation, it is straightforward to implement and tune without tire parameters or vehicle mass.

Limitations. Because the controller does not model tire forces or dynamic constraints, it may command curvature that the vehicle cannot execute. In high-speed or high-curvature manoeuvres, the pure-rolling and no-slip assumptions fail, and load transfer is no longer negligible. In these regimes, force limits rather than geometry govern the vehicle response, so the controller may produce oscillations, saturation, or degraded tracking performance. No feasibility check against the friction ellipse is performed, since the commanded curvature is geometric and may be physically unrealisable.

4.3 Blended Stanley Controller

The Blended Stanley controller augments the geometric Stanley law with a feedforward curvature term derived from the reference path. Rather than relying only on error correction, it blends a curvature-feedforward command ψ_{ff} with the Stanley feedback command ψ_{st} using a CTE-driven weight [9, 10]:

$$\psi(t) = w(t)\,\psi_{\text{ff}} + \big(1 - w(t)\big)\,\psi_{\text{st}}, \tag{65}$$

where the blend weight is

$$w(t) = \text{clamp}\big(e^{-a\,|\text{CTE}(t)|},\ w_{\min},\ w_{\max}\big)\,, \tag{66}$$

with default parameters $w_{\min} = 0.30$, $w_{\max} = 0.90$, and $a > 0$ controlling how quickly the weight decays with cross-track error. The signal-flow architecture is shown in Figure 10.

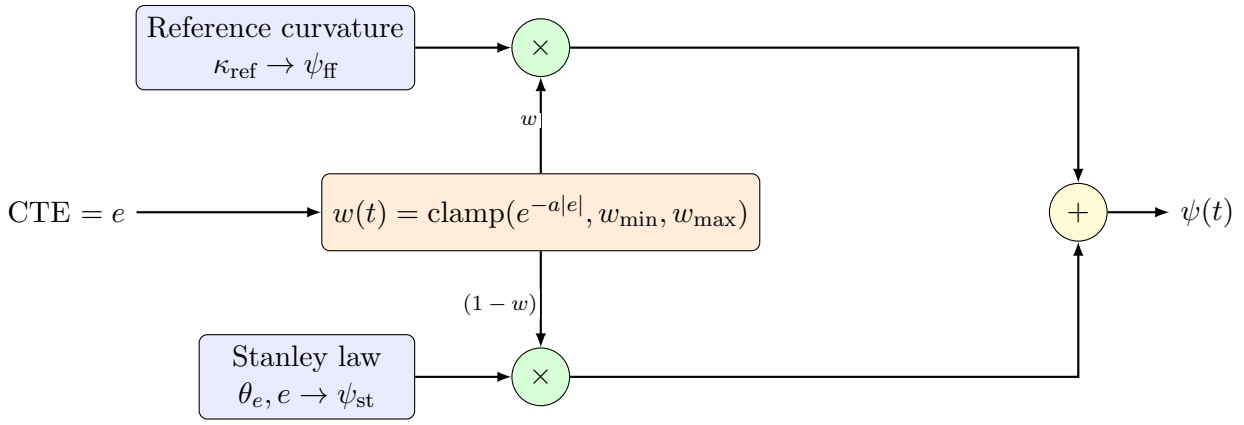


Figure 10: Blended Stanley block diagram. An exponential weight $w(t)$, driven by the cross-track error, interpolates between the feedforward curvature command ψ_{ff} (from the reference path) and the Stanley feedback command ψ_{st} (from heading and CTE). Near the path, $w \to 0.90$ gives mostly feedforward; as error grows, $w \to 0.30$ shifts authority to Stanley correction.

Weight interpretation. When $|\text{CTE}| \approx 0$, the vehicle is on path and $w \to 0.90$, so the command is 90% feedforward curvature and 10% Stanley correction. As CTE grows, w decays toward 0.30, shifting authority toward the Stanley term to pull the vehicle back to the path.

Assumptions. The feedforward curvature κ_{ref} is assumed to be realizable. The Stanley component is valid under the kinematic, low-slip assumptions described in Section 4, and CTE is treated as a reliable lateral error signal.

Advantages. The single additional tuning parameter a controls the transition rate. Near the path, the feedforward term reduces steady-state lag and improves tracking on curved segments without requiring tire-force knowledge. If the deviation grows, the Stanley correction pulls the vehicle back toward the path.

Limitations specific to friction-limited conditions. As the operating regime approaches the friction limit, the blended Stanley structure inherits several limitations from its kinematic feedback component. The Stanley correction ignores tire-force feasibility, so a large steering command can request lateral force outside the friction bound. The CTE-based weight also cannot distinguish between path error caused by saturation and path error caused by model bias, even though those failure modes require different responses. Because neither ψ_{ff} nor ψ_{st} is projected onto the friction ellipse, the blended command can remain infeasible. The blend is also insensitive to speed, slip angle, and lateral acceleration; at high speed, the floor $w_{\min} = 0.30$ may preserve too much feedforward authority. Finally, when CTE is small but heading error is large, the controller shifts toward feedforward even though curvature feedforward alone cannot correct the heading transient.

Sections 5–8 show how the inversion-based controller addresses these five deficits by construction, with Section 8 providing quantitative validation against the geometric baselines.

Relation to feedforward methods. The structure $\psi = w\,\psi_{\text{ff}} + (1-w)\,\psi_{\text{st}}$ anticipates the architecture used by the *Curvature Feedback Inverse*: a feedforward command augmented by a geometric error-correction term. Replacing ψ_{ff} with a Newton-inverse feedforward command while retaining the feedback component yields a hybrid that is both dynamically aware and geometrically self-correcting. This motivates replacing purely kinematic corrections with model-aware residual corrections from the data-driven methods of Section 7.

4.4 Pure Pursuit

Pure pursuit [10] computes a steering command by fitting a circular arc from the vehicle's current pose to a lookahead point on the reference path at distance L_d. In the vehicle frame, the lookahead point has coordinates $(x_{\text{la}}, y_{\text{la}})$. The required arc curvature and corresponding steering angle are

$$\kappa = \frac{2y_{\text{la}}}{L_d^2}, \qquad \psi = \arctan(L\kappa). \tag{67}$$

Figure 11 illustrates the geometry.

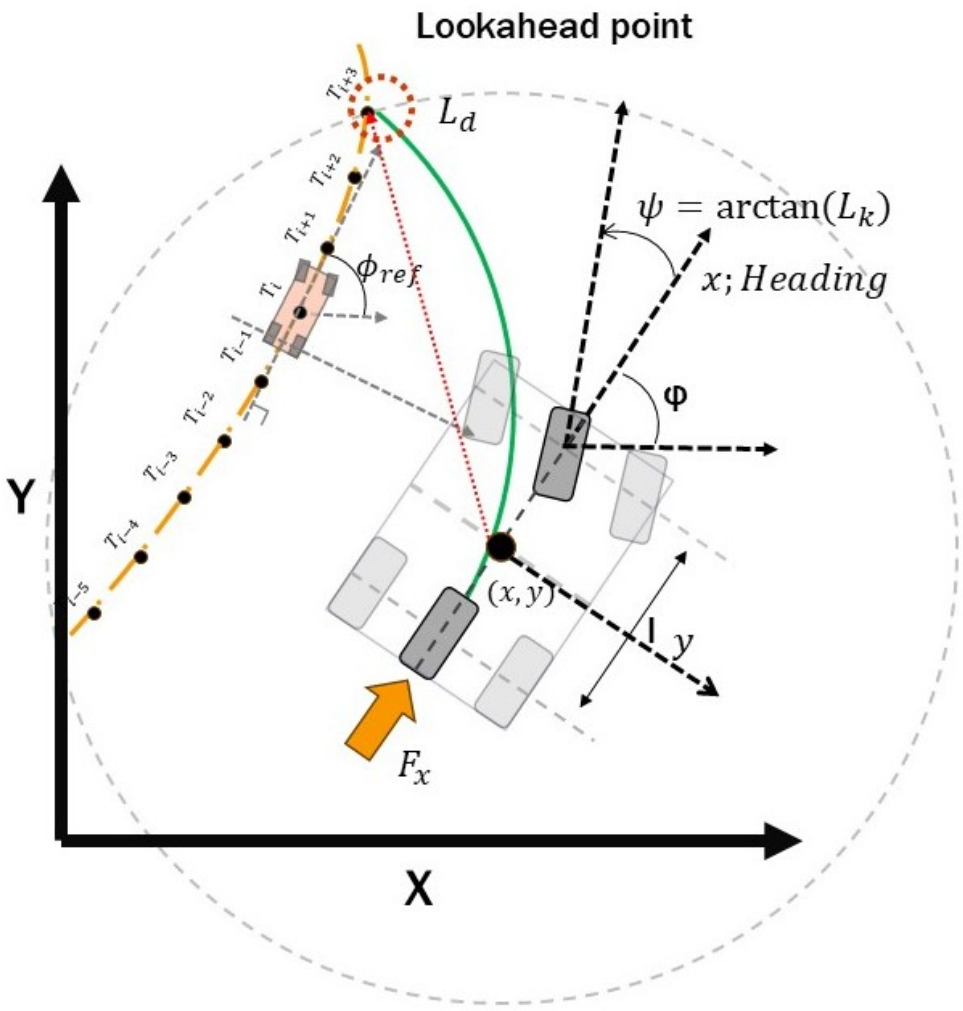


Figure 11: Pure pursuit geometry. A circular arc (green) is fit from the rear axle to the lookahead point at distance L_d on the reference path. The dashed circle indicates the lookahead radius, and the lateral offset of the lookahead point in the vehicle frame, y_{la}, determines the arc curvature $\kappa = 2y_{\text{la}}/L_d^2$ and the corresponding steering command ψ.

Assumptions. The method assumes that a valid lookahead point at distance L_d exists on the reference path, that the path curvature at the lookahead distance is realizable (i.e., κ does not exceed the steering limit), and that slip angles are small enough for the kinematic bicycle model to provide an adequate steering-to-curvature mapping.

Advantages. Pure pursuit provides inherently smooth anticipatory steering. The controller responds to upcoming path geometry rather than only current error, which reduces oscillation near the reference. The formulation is computationally cheap and straightforward to implement, and the lookahead distance L_d gives a single intuitive tuning parameter that trades path-following tightness for smooth transients.

Limitations. Performance is sensitive to the choice of L_d: too short a lookahead can lead to oscillation, while too long a lookahead can cause corner-cutting on tight curves. The method does not account for actuator limits or vehicle dynamics, and it may produce infeasible curvature commands at high speeds or on sharp curves. Steady-state lateral error is not explicitly corrected; unlike Stanley, there is no direct cross-track error feedback term.

4.5 Geometric Replanning with Circular Arcs

An alternative to direct error correction is to generate a locally feasible path at each timestep. One approach connects the current vehicle pose to a lookahead point using a circular arc, which is the same geometric idea used in classic path-tracking formulations such as pure pursuit [4, 10]. Expressing the lookahead point in the vehicle frame gives the arc curvature

$$\kappa = \frac{2\, y_{\text{local}}}{x_{\text{local}}^2 + y_{\text{local}}^2}, \tag{68}$$

with the corresponding steering command

$$\psi = \arctan(L\, \kappa). \tag{69}$$

This formulation replaces explicit feedback with local geometric planning. The controller determines a feasible path segment and then commands the curvature required to follow it.

Assumptions. This approach relies on the same geometric assumptions as the kinematic bicycle model. Steering uniquely determines curvature by assumption, and the vehicle is taken capable of realizing the arc. Feasibility follows from the vehicle pose and lookahead point alone, with no account for tire forces, load transfer, or dynamic limits.

Advantages. Because steering is derived from a locally planned arc rather than instantaneous error, the resulting commands are typically smooth and anticipatory. The method naturally incorporates future path geometry through the lookahead point, which can improve stability compared to purely reactive feedback. Curvature varies continuously with the local geometry, helping avoid abrupt steering changes.

Limitations. Since feasibility is judged only from geometry, the method does not account for tire-force limits or saturation. Performance depends strongly on the choice of lookahead distance: short lookahead can cause oscillations, while long lookahead can delay correction. When the curvature required by the arc exceeds what the vehicle can physically realize, the controller has no mechanism to enforce feasibility, and tracking may degrade.

Motivation for Iterative Inversion. All of the preceding methods operate under the implicit assumption that a commanded curvature is realizable. In practice, achievable curvature is limited by tire forces, friction, and vehicle dynamics. When commanded curvature exceeds these limits, geometric controllers may generate oscillations, saturation, or loss of tracking.

This limitation motivates a different formulation. Instead of commanding curvature directly, the controller should determine the inputs required to produce the lateral force associated with the desired trajectory. Because tire forces depend nonlinearly on the control inputs, this mapping is not generally available in closed form. The iterative inversion method

introduced in the next section addresses this problem by solving for control inputs that satisfy the dynamic force balance at each timestep.

4.6 Hybrid Control (Feedforward + Stanley Feedback)

To improve nominal tracking, geometric feedback in the Stanley controller can be combined with a feedforward steering prediction obtained from a model-based inverse, following the standard hybrid feedforward/feedback pattern used in path tracking [9, 10]. Let ψ_{ff} denote the steering angle predicted from the desired curvature. The combined control law is

$$\psi = w(t)\,\psi_{\text{ff}} + \big(1 - w(t)\big)\,\psi_{\text{stanley}}, \tag{70}$$

where the weighting function

$$w(t) = \max\big(w_{\min}, \min\big(w_{\max}, e^{-a|\text{CTE}|}\big)\big) \tag{71}$$

adjusts the balance between feedforward and feedback.

When tracking is accurate, the controller relies primarily on feedforward curvature prediction. As deviations grow, the feedback term takes over. This structure separates trajectory tracking into two components: a predictive term that enforces nominal curvature and a corrective term that compensates for disturbances and modeling errors.

Assumptions. This formulation inherits the same assumptions as the Stanley controller. Steering determines curvature, and the commanded inputs are achievable by the vehicle. The feedforward term therefore predicts the curvature required by the reference trajectory, while the feedback term is intended only to correct deviations, not to enforce physical feasibility of the motion.

Advantages: By incorporating feedforward prediction, the controller can reduce steady-state tracking error and avoid excessive reliance on corrective steering. The resulting commands are typically smoother because the feedforward term tracks reference curvature directly, reducing the corrective steering needed at each step. At the same time, the feedback component preserves robustness to disturbances and modeling inaccuracies, allowing the controller to recover when deviations arise.

Limitations: Despite these improvements, the formulation still assumes that the vehicle can execute the commanded curvature. If the feedforward prediction requests a manoeuvre requiring more force than the tires can provide, the controller has no mechanism to enforce command feasibility. The feedback term can only react after the deviation has occurred. Consequently, performance may degrade in aggressive maneuvers where force limits, rather than geometric considerations, govern the achievable motion.

This limitation motivates controllers that reason directly about the forces required to produce the desired motion, rather than relying solely on curvature prediction.

Thus, while the hybrid formulation improves nominal behavior, it does not fundamentally address feasibility of the commanded motion.

4.7 Model Predictive Control

Model Predictive Control (MPC) [11] formulates trajectory tracking as a finite-horizon optimal control problem. At each time step, MPC chooses a sequence of future control inputs

that minimizes a cost function penalizing tracking error, control effort, and input variation, subject to the vehicle dynamics and actuator constraints. Only the first control input of the optimized sequence is applied, and the optimization is repeated at the next time step using updated state information.

Let the discrete-time prediction model be written as

$$x_{k+1} = f(x_k, u_k),$$

where x_k is the state and $u_k = [a_k, \psi_k]^\top$ denotes longitudinal acceleration and steering input. Given a prediction horizon H, MPC solves

$$\min_{\{u_k\}_{k=0}^{H-1}} \sum_{k=0}^{H-1} \left(\|e_{\text{cte},k}\|_{Q_{\text{cte}}}^2 + \|e_{\text{head},k}\|_{Q_{\text{head}}}^2 + \|e_{v,k}\|_{Q_v}^2 + \|u_k\|_R^2 + \|\Delta u_k\|_{R_\Delta}^2 \right) \tag{72}$$

subject to the system dynamics and actuator limits

$$u_{\min} \le u_k \le u_{\max}, \qquad \Delta u_{\min} \le \Delta u_k \le \Delta u_{\max}. \tag{73}$$

Here,

- e_{cte} denotes cross-track error,
- e_{head} denotes heading error,
- e_v denotes speed tracking error,
- $\Delta u_k = u_k - u_{k-1}$ penalizes rapid control variation.

The resulting optimization produces control inputs that explicitly balance tracking accuracy, smoothness, and feasibility. Unlike geometric controllers, MPC accounts for actuator limits directly during planning and can incorporate additional constraints such as steering rate limits, acceleration bounds, or tire-force approximations. MPC is included here as a comparison baseline that characterises constraint-aware receding-horizon tracking. A four-speed numerical benchmark (1.0, 4.0, 7.0, 10.0 m/s; ellipse and circle trajectories; friction-plant simulation) is reported in Section 8 alongside Stanley and MPPI; see Table 13.

Modeling assumptions. The controller assumes that the prediction model adequately represents the vehicle dynamics over the horizon. In practice, a kinematic or reduced dynamic model is used, and receding-horizon feedback corrects disturbances or modeling errors. The approach further assumes that the optimization can be solved within the available control update time.

Advantages. MPC explicitly enforces actuator constraints and produces dynamically feasible control inputs. By looking ahead over a finite horizon, it incorporates future trajectory information, improving tracking smoothness and stability at higher speeds. The framework also provides a systematic way to balance competing objectives such as tracking error, control effort, and smoothness.

Limitations. The method requires solving an optimization problem at each time step, which increases computational cost. Performance depends on the accuracy of the prediction model and the tuning of the cost weights. If the model is inaccurate or the solver cannot converge within the control period, tracking performance may degrade.

Relation to inversion-based control. Where inversion computes the control required to achieve a desired instantaneous motion, MPC searches for a sequence of feasible controls that best approximate the desired trajectory over a horizon. This makes MPC particularly suitable when actuator limits or dynamic constraints prevent exact inversion of the motion equations.

4.8 Model Predictive Path Integral Control

Model Predictive Path Integral (MPPI) control [12] is a sampling-based optimal control method that approximates the solution of a finite-horizon stochastic optimal control problem through Monte Carlo simulation. MPPI evaluates many randomized control sequences and computes a cost-weighted average over predicted performance, without requiring a deterministic optimizer. Similar to MPC, MPPI is included as a comparison baseline representing the sampling-based end of the controller spectrum. A four-speed numerical benchmark is reported in Section 8 alongside Stanley and MPC (Table 13), providing quantitative context for where the Newton-type inversion approach sits relative to both deterministic optimisation (MPC) and stochastic sampling (MPPI).

Let the system dynamics be written in discrete time as

$$x_{k+1} = f(x_k, u_k),$$

where x_k is the vehicle state and $u_k = [a_k, \psi_k]^\top$ denotes acceleration and steering input. At each time step, a nominal control sequence

$$U = \{u_0, u_1, \ldots, u_{H-1}\}$$

is perturbed by additive noise samples

$$u_k^{(i)} = u_k + \epsilon_k^{(i)}, \qquad i = 1, \ldots, K,$$

where $\epsilon_k^{(i)}$ are drawn from zero-mean Gaussian distributions with prescribed variances.

For each sampled trajectory, a cumulative cost is evaluated:

$$S^{(i)} = \sum_{k=0}^{H-1} \Big(\|e_{\text{cte},k}^{(i)}\|_{Q_{\text{cte}}}^2 + \|e_{\text{head},k}^{(i)}\|_{Q_{\text{head}}}^2 + \|e_{v,k}^{(i)}\|_{Q_v}^2 + \|u_k^{(i)}\|_R^2 \Big). \tag{74}$$

The control update is then computed using a soft-min weighting:

$$\Delta u_k = \frac{\displaystyle\sum_{i=1}^{K} \exp\left(-\frac{1}{\lambda} S^{(i)}\right) \epsilon_k^{(i)}}{\displaystyle\sum_{i=1}^{K} \exp\left(-\frac{1}{\lambda} S^{(i)}\right)}, \tag{75}$$

where λ is a temperature parameter controlling selectivity. The nominal control sequence is updated by

$$u_k \leftarrow u_k + \Delta u_k, \tag{76}$$

and only the first control input is applied before the process is repeated at the next time step.

Modeling assumptions. MPPI assumes that the forward simulation model captures the dominant vehicle dynamics over the prediction horizon. Unlike gradient-based MPC, MPPI does not require differentiability of the dynamics and can accommodate nonlinearities or discontinuities. The approach also assumes that enough samples can be generated within the control period.

Advantages. MPPI naturally handles nonlinear dynamics and can incorporate actuator constraints and cost shaping without analytic gradients. It is well suited to highly nonlinear regimes where inversion or quadratic programming approaches may struggle. The method is also highly parallelizable because trajectory rollouts are independent.

Limitations. Performance depends on the number of samples and the noise tuning. A large sample count increases computational cost, while insufficient sampling may lead to suboptimal control updates. Unlike deterministic MPC, constraint satisfaction is typically enforced through cost penalties rather than strict guarantees.

Relation to inversion-based control. Inversion attempts to compute the control input that produces a desired instantaneous motion. MPC searches deterministically for a feasible control sequence that minimizes tracking error over a horizon. MPPI instead estimates an optimal control update statistically through sampling. This makes MPPI attractive in regimes where dynamic nonlinearities or tire-force saturation make closed-form inversion inaccurate, while still retaining a receding-horizon structure.

4.9 Comparison of Control Strategies

The controllers evaluated in this work cover geometric, inversion-based, and optimization-based approaches [9–12]. Geometric controllers, such as Pure Pursuit and Stanley, compute steering commands directly from the vehicle's instantaneous geometric relationship to the reference path. These methods are computationally efficient and simple to implement, but they do not explicitly account for actuator limits or vehicle dynamics.

Inversion-based methods compute the control inputs required to reproduce a desired vehicle motion derived from the reference trajectory. When the underlying model accurately captures the vehicle dynamics, inversion can provide precise tracking performance. Exact inversion may fail, however, or produce infeasible commands when the desired motion exceeds the physical limits imposed by tire friction or steering constraints.

Optimization-based approaches determine control inputs by solving a finite-horizon optimal control problem. Deterministic methods such as Model Predictive Control (MPC) compute the control sequence that minimizes a cost function subject to system dynamics and

actuator limits. Sampling-based methods such as Model Predictive Path Integral (MPPI) estimate an optimal control update by evaluating many randomized control sequences and weighting them according to predicted cost. These approaches are more computationally demanding, but they naturally accommodate nonlinear dynamics, actuator constraints, and friction limits.

In practice, geometric controllers trade model complexity for simplicity and robustness. Inversion-based controllers track accurately when the model is valid, and optimization-based methods handle nonlinear dynamics and tire-force limits most flexibly.

Title	**Mathematical formulation**	**Notes (A / + / −)**
Inverse (curvature)	$\psi_{\text{ff}} = \arctan(L\kappa_{\text{ref}})$	**A:** Kin model; low slip; feasible κ. **+:** Smooth feedforward; low compute; simple tuning. **−:** No disturbance reject; infeasible κ; dynamics ignored.
Blended Stanley (Sec. 4.3)	$\psi = w\,\psi_{\text{ff}} + (1-w)\psi_{\text{st}}$, $\psi_{\text{st}} = \theta_e + \arctan\left(\frac{k\,\text{CTE}}{k_s+v}\right)$	**A:** Kin steering$\rightarrow$ κ; CTE sign correct; mild slip. **+:** Better nominal tracking; robust correction; smooth blend. **−:** Model bias leaks; no force limits; tuning sensitive.
Inverse + FB (Inverse FB)	$\psi = \psi_{\text{ff}} + \psi_{\text{fb}}$ (feedback from geometric error)	**A:** Additive steering valid; errors measurable; small angles. **+:** Cancels drift; improves transients; modular design. **−:** Can oversteer; saturations hidden; dynamics ignored.
Lookahead (arc / pure pursuit style)	$\kappa = \frac{2y_{\text{la}}}{x_{\text{la}}^2+y_{\text{la}}^2}$, $\psi = \arctan(L\kappa)$	**A:** Local arc feasible; lookahead point valid; mild slip. **+:** Anticipatory steering; smooth curvature; fewer oscillations. **−:** L_d sensitive; corner cutting; infeasible curvature.
Stanley PI	$\psi = \psi_{\text{st}} + k_I \int \text{CTE}\,dt$	**A:** Bias is slow; CTE integrable; bounded windup. **+:** Rejects steady bias; reduces offset; simple add-on. **−:** Windup risk; oscillations; needs anti-windup.
Newton (body-accel inverse)	Solve $[\dot{v}_x; \dot{v}_y] = f(F_{\text{in}}, \delta)$ via Newton	**A:** Tire model valid; derivatives smooth; good warm-start. **+:** Enforces dynamics; handles coupling; reduces slip error. **−:** Iteration cost; can diverge; needs model params.

Table 2: Summary of control methods (part 1 of 2). Notes: A = assumptions, + = advantages, – = limitations.

Title	Mathematical formulation	Notes (A / + / −)
Newton LA	$[ax, ay]_{\text{des}} = [ax_{\text{PI}}, ay_{\text{ref}} + ay_{\text{la}}]$; Newton solve	**A:** Lookahead errors reliable; model valid; stable step. **+:** Better robustness; anticipates curvature; fewer overshoots. **−:** More tuning knobs; compute cost; still saturation-limited.
Newton Pred LA	Predict under u_{ff}, compute (CTE, θ_e) on predicted state, add ψ_{corr}	**A:** Predictor close; dt small; errors smooth. **+:** Compensates delay; cleaner steering; improved stability. **−:** Predictor mismatch; extra compute; corner cases.
Pure Pursuit	$\kappa = \frac{2y_{\text{la}}}{L_d^2}$, $\psi = \arctan(L\kappa)$	**A:** Lookahead point valid; arc feasible; mild slip. **+:** Anticipatory; single tuning param L_d; smooth transients. **−:** L_d sensitive; corner-cutting; no CTE feedback.
Body-accel inverse	Solve $\dot{v}_y = a_y^{\text{des}}$ for (F_{in}, δ); opt. lookahead	**A:** Dyn. model valid; a_y^{des} computable; feasible inputs. **+:** Explicit dynamics; lookahead variant reduces delay. **−:** Model error propagates; compute cost; saturation blind.
Full-state LM inverse	LM solve for (F_{in}, δ): match $(v_x, v_y, r)_{\text{next}}$; + yaw-rate feedforward	**A:** State targets computable; LM converges; yaw ff stable. **+:** Jointly inverts all states; curvature-aware ff; robust. **−:** Most expensive; requires warm-start; three target states.

Table 3: Summary of control methods (continued, part 2 of 2). Notes: A = assumptions, + = advantages, – = limitations.

5 Non-linear Inversion: Newton Inverse

5.1 Algebraic vs. Iterative Inversion

Model inversion, the problem of finding the control input that produces a desired vehicle motion, takes two fundamentally different forms depending on the complexity of the vehicle model.

Algebraic (closed-form) inversion. When the vehicle model is linear or low-degree and invertible, a closed-form expression can give the steering angle directly, without iteration. For the *kinematic bicycle model*, tire lateral force tracks slip angle instantaneously and sideslip is negligible by assumption. Under those assumptions, a single algebraic step gives the required steering angle for reference curvature κ_{ref} and speed v_{ref}:

$$\delta_{\text{alg}} = \arctan\big(L\,\kappa_{\text{ref}}\big), \tag{77}$$

where L is the wheelbase. This mapping is one-to-one, requires no iteration, and evaluates immediately. It underlies the geometric controllers (Stanley, Pure Pursuit) and simple kinematic feedforward terms, where desired curvature maps directly to steering angle through geometry without consulting a tire force model. The algebraic inverse therefore acts on the *geometric* structure of the vehicle, mapping path curvature to wheel angle. It remains valid only while the kinematic assumptions of negligible sideslip and linear tire behavior hold.

Iterative (Newton-type) inversion. Once tire nonlinearity, saturation, or state-dependent effects are included, a closed-form map from desired motion to control input is generally no longer available. The *enhanced* bicycle model, with Fiala or Pacejka tire force, defines a map $u \mapsto \dot{\mathbf{x}}$ that is nonlinear in both the steering input δ and the current state. Finding the input $u^\star$ that produces a desired acceleration $\mathbf{a}_{\text{ref}}$ requires solving the nonlinear equation

$$f(u^\star;\, \mathbf{x}) = \mathbf{a}_{\text{ref}} \tag{78}$$

iteratively. The iterative inverse therefore acts on the vehicle's *force-generation* structure. It searches for the steering angle whose tire forces are *consistent* with the desired acceleration, while checking feasibility against the friction ellipse at every Newton step.

Table 4 summarises the key behavioral differences.

Table 4: Algebraic vs. iterative inversion: key behavioral differences.

Property	**Algebraic inverse**	**Iterative inverse (Newton-LM)**
Closed form	Yes - single expression	No - solved numerically
Model	Kinematic / linear tire	Nonlinear (Fiala, Pacejka, enhanced)
Acts on	Geometric curvature	Tire force balance
Friction check	Not explicit	At each Newton step (friction ellipse)
Computation	$O(1)$	$O(n_{\text{iter}})$, typically 3–5 steps
Validity	Low slip, low curvature	Any regime within model validity
Failure mode	Silently infeasible command	Convergence failure / saturation flag

The most important practical difference is in how each method fails. When the reference curvature exceeds the friction-limited capability of the tire, the algebraic inverse still returns a steering command; the formula produces a value whether or not the tire can generate the required force. The iterative inverse either converges to a feasible solution inside the friction ellipse or raises a convergence flag when none exists, giving explicit per-timestep feasibility information that geometric controllers lack. Near the friction limit, this makes iterative inversion more robust, since it verifies feasibility rather than assuming it.

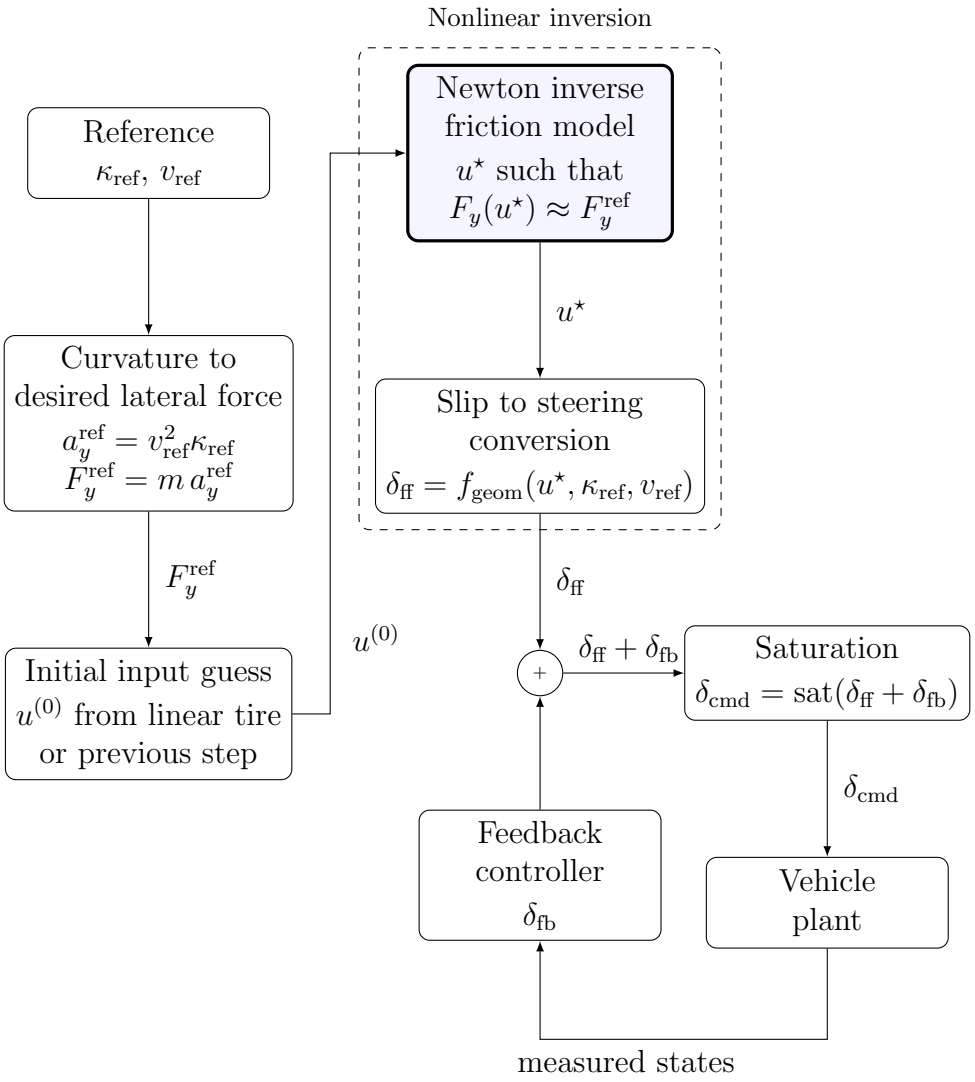


Figure 12: Block diagram of the nonlinear inverse using the Newton method to compute a feedforward steering command from the force implied by the reference trajectory.

The nonlinear inverse pipeline maps a desired path to a feedforward steering command by inverting the nonlinear tire model. As shown in Fig. 12, the reference path enters as curvature and speed profiles, $\kappa_{\text{ref}}(s)$ and $v_{\text{ref}}(s)$. At each step, these define a desired lateral acceleration a_y^{ref}, which converts to an equivalent desired lateral tire force F_y^{ref}. That force, together with the current tire and state parameters, feeds into the nonlinear inverse friction block. The

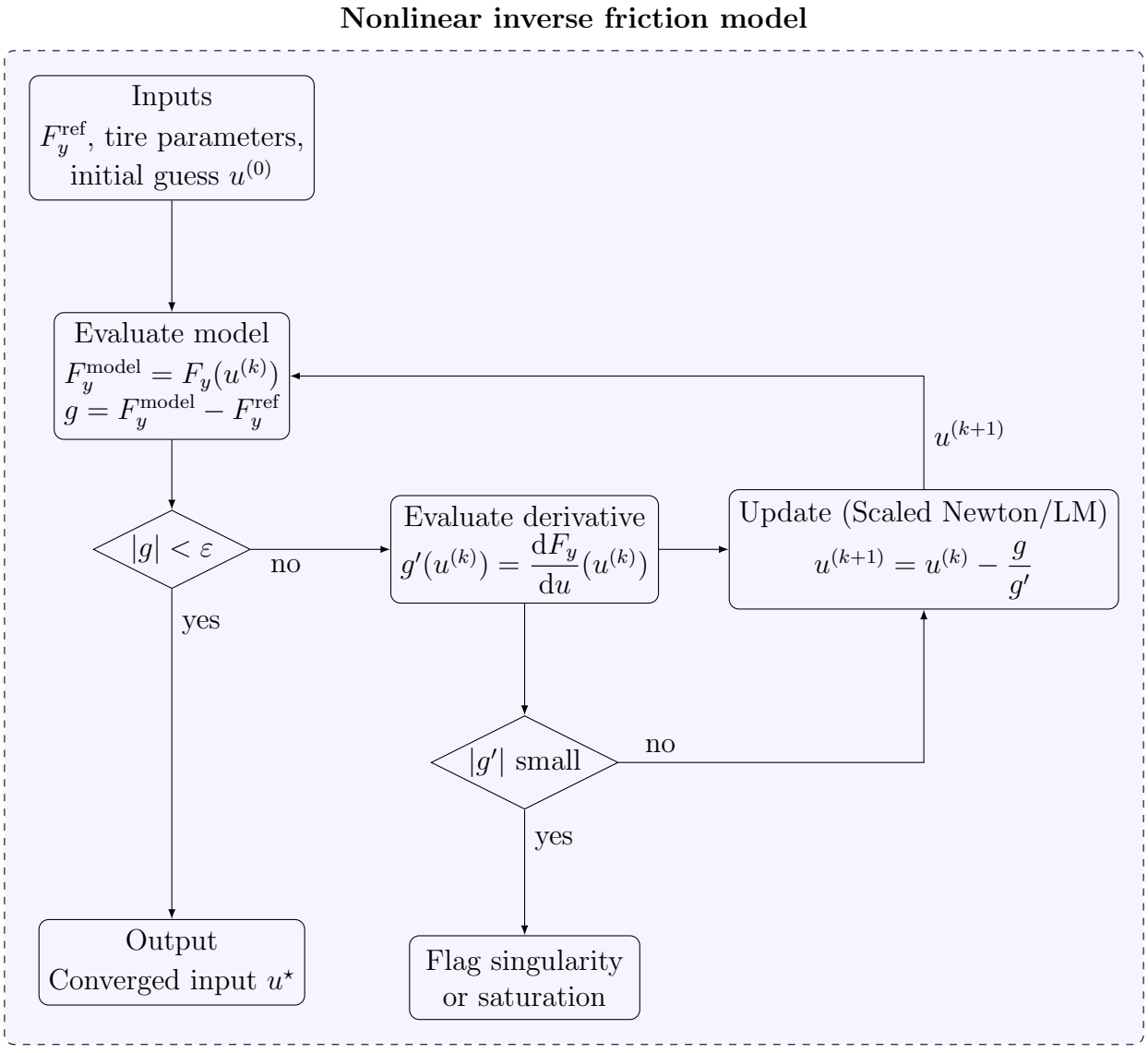


Figure 13: Flowchart for the Newton-iteration procedure in the nonlinear inversion block of Fig. 12, showing model evaluation, convergence checks, and update steps used to solve for the converged input u^*.

initial internal tire-input guess $u^{(0)}$ comes either from a linearized tire approximation or from the converged solution at the previous time step.

The flowchart in Fig. 13 summarizes the internal operation of the nonlinear inverse friction model. Given the current guess $u^{(k)}$, the tire model is evaluated to produce $F_y^{\text{model}} = F_y\big(u^{(k)}\big)$, and the scalar residual $g\big(u^{(k)}\big)$ measures the mismatch between the model and the desired lateral force. If $|g(u^{(k)})|$ is below a prescribed tolerance, the algorithm declares convergence and returns the current iterate as the solution u^*. Otherwise, it performs an update step to reduce the residual. The system also guards against singularities and records diagnostic logs when $|g'(u^{(k)})|$ is near zero; in that case the residual manifold is locally flat, so changing the input produces little change in the residual.

Once a consistent tire input u^* is found, it is mapped through the geometric tire–steering relationship to a feedforward steering angle δ_{ff},

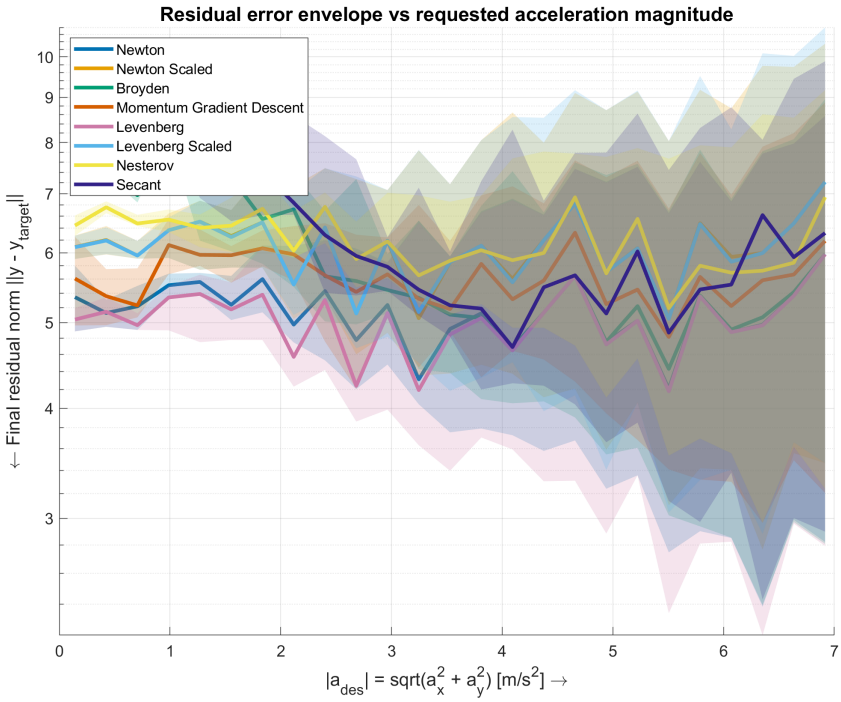
$$\delta_{\text{ff}} = f_{\text{geom}}(u^*, \kappa_{\text{ref}}, v_{\text{ref}}) \,, \tag{79}$$

which represents the steering command required to generate the desired lateral force at the current operating point. This feedforward term is then combined with the feedback correction δ_{fb} and passed through the steering saturation block to produce the final commanded steering input δ_{cmd} to the vehicle plant, as indicated in Fig. 12.

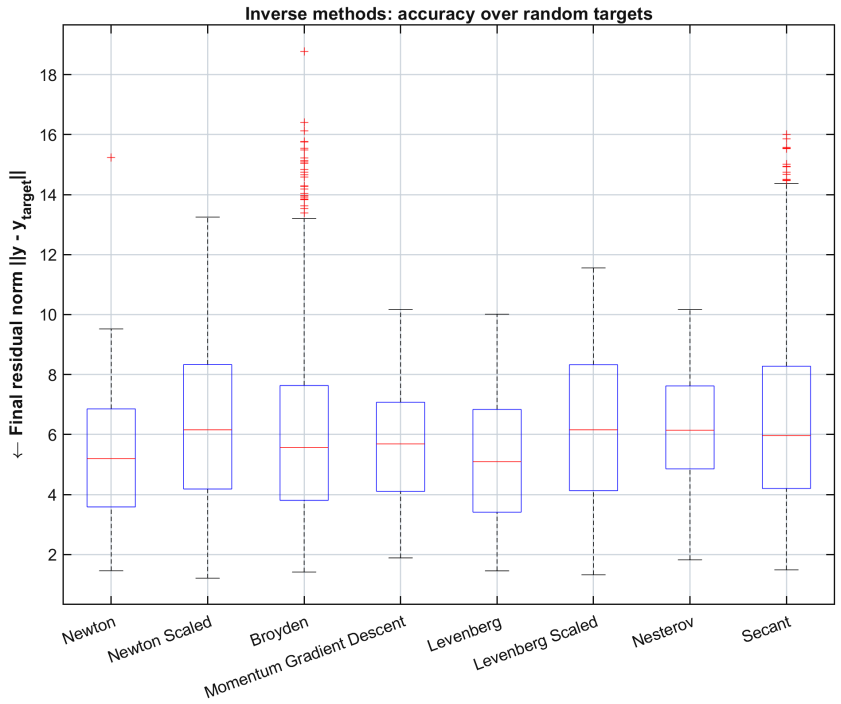
The implementation uses a Newton update to invert the nonlinear friction model, although other update rules are possible. Alternatives include gradient descent, momentum-based methods (Momentum GD and Nesterov Accelerated Gradient), quasi-Newton updates such as Broyden's method, and one-dimensional secant updates. Short of a precomputed lookup table with interpolation, Newton's method is the fastest, most reliable, and simplest option in this setting. It converges rapidly while keeping the implementation light enough for low-power onboard processors such as the Raspberry Pi 3 or Pi 5, even when the full state estimator and trajectory logic run concurrently.

To assess the suitability of different update rules for feedforward control, the inversion routine was evaluated over randomly sampled desired accelerations from a fixed initial state. The resulting metrics, summarized in the following figures, show each method's convergence behavior, final accuracy, iteration count, and feasibility rate. These results guide the choice of update step when inverse dynamics is embedded in a fast closed-loop controller.

For a fixed initial state with randomly sampled desired accelerations, Figures 14 and 15 summarize the residual envelope, final accuracy, iteration-count distribution, and feasibility rate for the update rules.

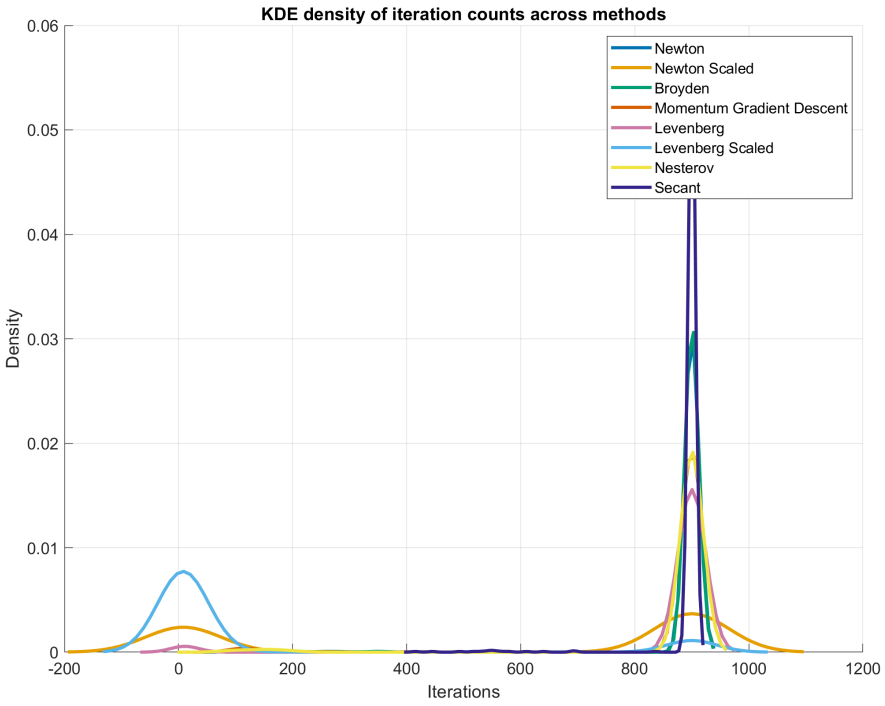


(a) Residual error envelope vs. requested acceleration magnitude.

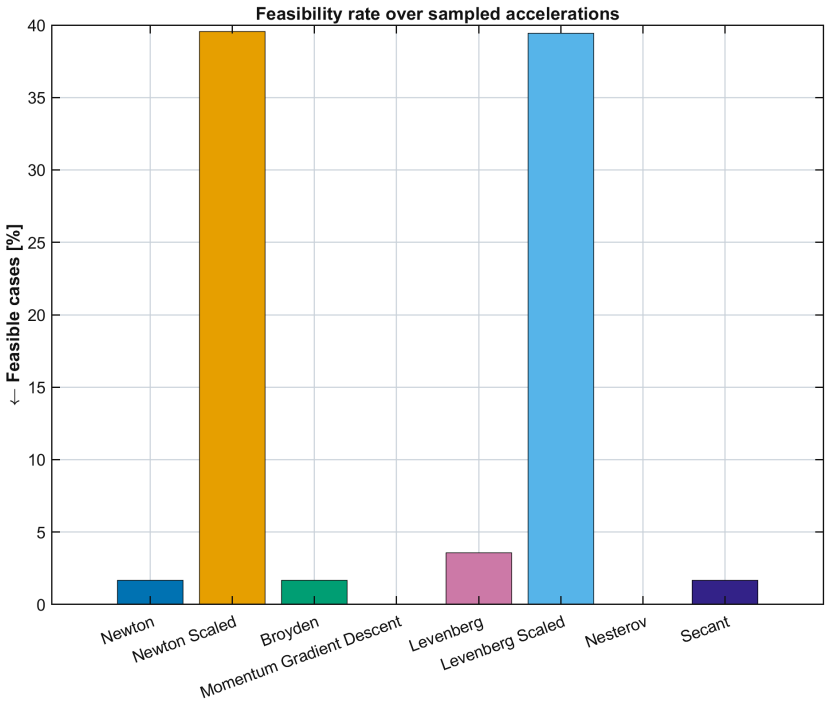


(b) Boxplot of final residual norms across methods.

Figure 14: Comparative evaluation of inverse-dynamics update rules (1/2): residual envelope and accuracy boxplot.



(a) KDE of iteration counts across all update schemes.



(b) Feasibility rate over randomly sampled accelerations.

Figure 15: Comparative evaluation of inverse-dynamics update rules (2/2): iteration count distribution and feasibility rate.

Scaled Newton and especially Scaled Levenberg–Marquardt [13, 14] show better convergence robustness, lower iteration counts, and higher feasibility than unscaled or first-order

methods.

These results show that the inverse-dynamics update step strongly affects convergence and feasibility when solving for the wheel force (throttle) and steering input that reproduce a desired body-frame acceleration. Newton and Gauss–Newton-like schemes can reach low residuals, but their reliability deteriorates when the Jacobian is ill-conditioned or the requested acceleration lies near the edge of the friction circle. The high iteration counts and low feasibility rates reflect this failure mode.

Both scaled variants, Newton and Levenberg–Marquardt, show higher feasibility and lower variance in residual error. The diagonal scaling improves numerical conditioning, especially for the Scaled LM method, by preventing overshoot and suppressing directions with amplified curvature. Its damping mechanism allows the update to move smoothly between gradient-like and Newton-like behavior, producing the most consistent performance.

The Newton method preserves quadratic convergence when the linearization is accurate and the Jacobian is well-conditioned. LM damping reduces the convergence order to at most super-linear, but it improves robustness when the Jacobian is ill-conditioned. Consequently, Newton converges in fewer iterations under favorable, low-transition acceleration demands, while LM is preferred in poorly conditioned cases.

Table 5: Summary of statistical comparison of inversion update steps.

Method	**Feas. Rate [%]**	**Iter. Mean**	**Iter. Median**	**Residual Median**	**Residual Mean**	**Runtime [s]**
Newton	1.67	889.07	900	5.25	5.19	0.0360
Newton Scaled	39.56	547.43	900	6.24	6.16	0.0238
Broyden	1.67	889.28	900	5.99	5.57	0.0217
Momentum Gradient Descent	0.00	875.16	900	5.68	5.69	0.0342
Levenberg–Marquardt	3.56	868.40	900	5.15	5.09	0.0361
Levenberg–Marquardt Scaled	39.44	121.72	9	6.25	6.16	0.0071
Nesterov	0.00	875.62	900	6.10	6.14	0.0343
Secant	1.67	894.62	900	6.50	5.96	0.0206

Table 6: Summary of update rules for inverse-dynamics solvers.

Method	**Update Equation**	**Notes**
Newton	$u^{(k+1)} = u^{(k)} - \left(J^\mathsf{T} J\right)^{-1} J^\mathsf{T} r$	1D: $u^{(k+1)} = u^{(k)} - \frac{g}{g'}$.
Newton Scaled	$u^{(k+1)} = u^{(k)} - S\,\left(J^\mathsf{T} J\right)^{-1} J^\mathsf{T} r$	Same as Newton, but with a diagonal scaling matrix S: $S = \mathrm{diag}(1/\|J_{\cdot i}\|)$.
Broyden's first	$u^{(k+1)} = u^{(k)} - B_k^{-1} r^{(k)}$ $B_{k+1} = B_k + \frac{(\Delta r_k - B_k\,\Delta u_k)\,\Delta u_k^\mathsf{T}}{\Delta u_k^\mathsf{T} \Delta u_k}$	Where $\Delta r_k = r_{k+1} - r_k,\ \Delta u_k = u_{k+1} - u_k$.
Momentum Gradient Descent	$v^{(k+1)} = \beta v^{(k)} - \eta\, J^{(k)\mathsf{T}} r^k$ $u^{(k+1)} = u^{(k)} + v^{(k+1)}$	With momentum coefficient $0 < \beta < 1$.
Nesterov Accelerated Gradient	$y^{(k)} = u^{(k)} + \beta(u^{(k)} - u^{(k-1)})$ $u^{(k+1)} = y^{(k)} - \eta\,\nabla_u\left(\frac{1}{2}\|r(y^{(k)})\|^2\right)$	
Levenberg–Marquardt (Damped least squares)	$u^{(k+1)} = u^{(k)} - \left(J^\mathsf{T} J + \lambda I\right)^{-1} J^\mathsf{T} r$	Where $\lambda > 0$ tunes between Gauss–Newton and gradient descent.
Levenberg–Marquardt (Scaled)	$u^{(k+1)} = u^{(k)} - \left(J^\mathsf{T} W J + \lambda I\right)^{-1} J^\mathsf{T} W r$	With diagonal weighting matrix $W = \mathrm{diag}(w_i)$.
Secant Method	(1D equivalent approximation) $u^{(k+1)} = u^{(k)} - g(u^{(k)}) \frac{u^{(k)} - u^{(k-1)}}{g(u^{(k)}) - g(u^{(k-1)})}$	Where $g(u) = r(u)$.

In the update expressions above, the control vector is $u = [F_{in},\, \delta]^{\mathsf{T}}$, and the residual is $r(u) = y(u) - y_{ref}$, where $y(u) = [\, dv_x(u),\, dv_y(u),\, \dot{r}(u)\,]^{\mathsf{T}}$ denotes the model-predicted body accelerations and y_{target} is the desired acceleration vector. Newton-type and gradient-based methods use the Jacobian $J = \partial r/\partial u \in \mathbb{R}^{3\times 2}$, computed numerically by finite differences. The parameters $\eta > 0$ and $0 < \beta < 1$ denote the gradient-descent step size and momentum coefficient, respectively. Quasi-Newton schemes use an approximate Jacobian or inverse Jacobian B_k, updated from the increments $\Delta u_k = u^{(k+1)} - u^{(k)}$ and $\Delta r_k = r^{(k+1)} - r^{(k)}$. Levenberg-Marquardt variants introduce the damping factor $\lambda > 0$, while scaled formulations use diagonal weighting matrices S or W. Momentum-based methods add an auxiliary velocity variable $v^{(k)}$, and Nesterov acceleration evaluates the gradient at the extrapolated point $y^{(k)} = u^{(k)} + \beta\big(u^{(k)} - u^{(k-1)}\big)$.

5.2 Nonlinear Inverse Family and Solver Variants

The abstract Newton inverse framework appears at two distinct layers. Controller families define the inverse target and feedback architecture, while solver variants define the numerical update used inside that inverse. The main controller families are the *Body-Acceleration Inverse*, the *Lookahead Inverse*, the *Curvature Feedback Inverse*, and the *Full-State LM Inverse*. Table 7 compares solver update rules, not ten separate controller architectures.

Residual function. The original MATLAB/Fiala inverse variants use the same derivative-level residual. For state (v_x, v_y, r) and control candidate $u = [F_{\text{in}},\, \delta]^{\top}$, define

$$\hat{y}(u) \in \mathbb{R}^3$$

as the Fiala/Pacejka model prediction for the body-frame acceleration vector $[\dot{v}_{x,\text{pred}},\, \dot{v}_{y,\text{pred}},\, \dot{r}_{\text{pred}}]$. The inverse solve compares this prediction with the target

$$y^{\star} = \begin{bmatrix} a_x^{\text{des}} + r\, v_y \\ a_y^{\text{des}} - r\, v_x \\ \dot{r}^{\text{des}} \end{bmatrix}, \tag{80}$$

where the first two components convert desired body-frame accelerations into time derivatives of v_x and v_y, while $\dot{r}^{\text{des}}$ specifies the desired yaw acceleration. The residual is then $\rho(u) = \hat{y}(u) - y^{\star} \in \mathbb{R}^3$.

The MuJoCo sweep and oracle controllers use a closely related, but not identical, residual. They compare Coriolis-corrected body accelerations $[a_{\text{long}}, a_{\text{lat}}, \dot{r}]$ rather than the raw state derivatives $[\dot{v}_x, \dot{v}_y, \dot{r}]$. This distinction is easy to miss, but it matters when results are compared across tables because the two residual definitions are not interchangeable. The results and provenance sections therefore identify the artifact behind each table so that the formulation remains traceable.

Jacobian structure. The 3×2 Jacobian $J = \partial\hat{y}/\partial u \in \mathbb{R}^{3\times 2}$ is approximated by finite differences at each Newton iteration:

$$J \approx \left[\frac{\hat{y}(u + \varepsilon_F\, e_1) - \hat{y}(u)}{\varepsilon_F} \quad \frac{\hat{y}(u + \varepsilon_\delta\, e_2) - \hat{y}(u)}{\varepsilon_\delta} \right], \tag{81}$$

with $\varepsilon_F = \max(1, |F_{\text{in}}|) \times 10^{-6}$ N and $\varepsilon_\delta = 10^{-6}$ rad. The weighted Gauss–Newton step

$$\Delta u = -(WJ^\top JW^\top + \lambda I)^{-1} J^\top W^\top \rho$$

is applied with diagonal weight $W = \text{diag}(1, 1, 0.2)$ that de-emphasises the yaw channel and Levenberg–Marquardt damping λ for regularisation.

Body-Acceleration and Lookahead Inverses. The *Body-Acceleration Inverse* family inverts the equations of motion at the body-frame acceleration level. Given desired lateral acceleration a_y^{des} and current state (v_x, v_y, r), the inverse solve finds (δ, F_{in}) such that $\hat{y}(u) \approx y^\star$ with $\dot{r}^{\text{des}} = 0$. The *Lookahead Inverse* variant adds a geometric correction to the desired lateral acceleration:

$$a_y^{\text{des}} \leftarrow a_y^{\text{des}} + K_{\text{la}} \frac{e_{\text{la}}}{T_{\text{look}}^2}, \tag{82}$$

where e_{la} is the projected cross-track error at lookahead time T_{look} and K_{la} is a gain (default 1.0). The predictive variant first propagates the vehicle state one timestep under the feedforward command, then computes the geometric correction from that predicted state. This reduces sensitivity to communication and actuation delay.

Architectural requirement: inversion must be paired with feedback. A critical architectural distinction separates the *Body-Acceleration Inverse* from the *Curvature Feedback Inverse*. The former is *pure feedforward*; the latter wraps feedforward inversion with error feedback.

The *Body-Acceleration Inverse* computes (δ, F_{in}) from a desired lateral acceleration a_y^{des} derived only from the reference curvature at the nearest waypoint. If the vehicle drifts off the path, a_y^{des} does not change. The controller continues to target the same curvature, so the vehicle can track a permanently offset path. Any steady-state model error, including incorrect $C_{\alpha f}$, tire mismatch, or load-transfer error, becomes an uncorrected force bias that integrates into growing position error.

The *Curvature Feedback Inverse* closes this loop by augmenting the inversion target with heading-error and cross-track-error feedback terms, analogous to Stanley's correction but applied at the force level rather than the steering-angle level. When the vehicle drifts e_{lat} m off-path, the feedback term adjusts a_y^{des} proportionally, driving the inversion toward a steering command that returns the vehicle to the reference.

This distinction strongly affects benchmark performance. In the Enhanced bicycle-model simulations, pure feedforward inversion is sensitive to steady-state model error because it has no path-error term to pull the vehicle back after drift begins. When a residual correction changes the effective tire response, the *Body-Acceleration Inverse* can command proportionally less steering, understeer, and diverge. The *Curvature Feedback Inverse* sees the same model change, but its error-feedback term can correct the resulting drift. The numerical results in Section 8.5 therefore compare inversion architectures as much as model fidelity levels.

The practical implication is unambiguous: the force-consistency benefit of iterative inversion is only realised when inversion is used as the feedforward component of a feedforward+feedback architecture. In the tested benchmarks, a

pure feedforward inversion controller is less stable than a simple geometric controller with feedback, because geometric controllers such as Stanley inherently include cross-track and heading correction. Inversion should therefore be paired with an outer error-feedback loop. The *Curvature Feedback Inverse* follows this pattern and is the preferred architecture for future hardware deployment.

Chassis-coupling instability, solver geometry, and inversion extension. The `fixed_v3` ablation quantifies this boundary: tire-only corrections leave the pure body-acceleration inverse stable at 4.58 m CTE, but adding chassis and roll correctors causes divergence because the identified state-coupling coefficients ($r \to \Delta\dot{r} \approx -29$ rad/s^2, $v_y \to \Delta\dot{v}_y \approx 16$ m/s^2) inject forces proportional to yaw rate and lateral velocity at every RK4 substep. Those forces grow monotonically through a corner and go unmodelled by the inversion target.

The Newton–LM solver is *not* trapped in a local minimum, flat basin, or saddle point. It converges in 3–5 iterations to the exact root of the *uncorrected* residual. The solution is numerically accurate, but it solves the residual of the uncorrected model, not of the corrected plant. Because $\Delta F_y^{\text{chassis}}(\mathbf{x})$ depends on $\mathbf{x}$ and not on the control input u, it does not reshape the residual landscape seen by the solver. The residual surface is unchanged, and the solver converges to its root without difficulty. What fails is the *equation itself*. The corrected plant adds a state-dependent force offset at runtime that the solver never sees. The closed-loop geometry is therefore neutral-stability with a growing disturbance input, not a saddle or flat minimum. Without feedback to arrest drift, errors accumulate monotonically rather than oscillating or plateauing.

To extend the inversion to handle state coupling, fold the chassis prediction directly into the Newton residual before each solve step. Let $\Delta a_y^{\text{ch}}(\mathbf{x}) = (1/m)\,\Xi_{\text{chassis}}\,\phi(\mathbf{x})$ be the lateral acceleration perturbation predicted by the chassis corrector at the current state. Replace the standard residual with the augmented form

$$r(u) \;=\; F_y(u;\,\mathbf{x}) \;+\; m\,\Delta a_y^{\text{ch}}(\mathbf{x}) \;-\; F_y^{\text{des}}. \tag{83}$$

Since Δa_y^{ch} depends on $\mathbf{x}$ but not on u, the Jacobian $\partial r/\partial u$ is unchanged and no additional solver iterations are needed. The augmentation is a free pre-shift of the inversion target, so the solver finds the steering angle consistent with the *corrected* plant. This would allow the pure body-acceleration inverse to remain stable under all-subsystem corrections without relying solely on the outer feedback loop used by the *Curvature Feedback Inverse*.

Full-State LM Inverse. The *Full-State LM Inverse* controller inverts the full (v_x, v_y, r) state jointly. Desired body-frame velocities at the next timestep are obtained from the reference curvature and speed. The controller then uses scaled Levenberg–Marquardt to find the (F_{in}, δ) pair that drives the model prediction toward those targets, with yaw-rate feedforward defined by

$$\dot{r}^{\text{des}} = \frac{(\kappa_{\text{next}} - \kappa_{\text{now}})\,v_{\text{ref}}}{\Delta t} \tag{84}$$

This term comes from the reference curvature change rate. Input preconditioning uses scales $S = \text{diag}(mg, 0.3)$, with force scaled to vehicle weight and angle scaled to $\approx 17°$, which improves numerical conditioning across the full speed range.

Ten inversion variants benchmarked. A broader family of ten numerical update rules was benchmarked against the same inverse residual:

#	Key name	Description
1	`Newton`	Weighted+scaled Newton (Gauss–Newton with W, S)
2	`Broyden`	Rank-1 quasi-Newton (Good Broyden)
3	`Momentum`	Heavy-ball Newton (momentum-damped gradient)
4	`Levenberg`	Scaled+weighted Levenberg–Marquardt
5	`Nesterov`	Nesterov-accelerated gradient descent
6	`Secant`	Frozen-Jacobian secant method
7	`LevenbergPlain`	Plain LM (no W or S preconditioning)
8	`NewtonPlain`	Plain damped least-squares Newton
9	`Adam`	Adaptive moment estimation (Adam)
10	`RMSProp`	RMS-gradient adaptive step size

Table 7: Ten inversion solver variants. All share the residual in (80) and Jacobian in (81); variants differ in step computation and preconditioning. Methods 1–6 directly mirror the six MATLAB solver implementations.

Across the MuJoCo ellipse benchmark at $v_{\text{ref}} = 7$ m/s, the scaled Levenberg–Marquardt and scaled Newton variants achieved the lowest residual norm per iteration. Both benefited from the S preconditioner, which balances the force ($\sim mg = 14700$ N) and angle (~ 0.3 rad) scales. Plain methods (7–8) and momentum methods (3, 5, 9, 10) converged more slowly under the default 5-iteration budget, though RMSProp produced slightly smoother control histories.

Solver conditioning, low-speed behaviour, and feasibility. Three operating conditions require explicit treatment:

Low-speed singularity. Below approximately $v_x < 0.5$ m/s the slip-angle denominators approach zero (the front denominator is $\sin\delta\,(v_y + ar) + \cos\delta\, v_x$; the rear is simply v_x). The resulting Jacobian columns are ill-conditioned and any Newton step becomes unreliable. The implementation gates the inversion below this speed threshold, falling back to a geometric (Stanley) command rather than invoking the nonlinear solve.

Friction saturation. In the full-slip regime ($|\alpha| \geq \alpha_{\text{slip}}$), the Fiala force saturates at μF_z and $\partial F_y / \partial \delta \approx 0$. The Jacobian column corresponding to δ effectively vanishes, making the normal equations ill-conditioned. The Levenberg–Marquardt damping term λI regularises this case, since as $\|J^\top J\|$ shrinks, λ dominates and the step reduces to a small gradient descent move. Increasing λ when saturation is detected (e.g., when $\|J^\top J\| < \epsilon_{\text{sat}}$) provides an additional safeguard.

Feasibility. A control candidate $u^\star$ is considered feasible if, after convergence: (a) the residual satisfies $\|\rho(u^\star)\| < \rho_{\max}$, (b) $|F_{\text{in}}| \leq \mu mg$ (within the friction-limited traction circle),

and (c) $|\delta| \leq \delta_{\max}$. If any bound is violated after the iteration budget is exhausted, the command is clipped to the nearest feasible point and the solver reinitialises from zero on the next timestep. This ensures that the inversion never issues a command that is geometrically or physically infeasible, directly addressing the limitation of the geometric controllers described in Section 4.

6 Path Planning and Reference Replanning

A feedforward controller reads a desired curvature $\kappa_{\text{ref}}(t)$ from the global reference path and converts it instantaneously into a steering command. When the vehicle is exactly on the path this works well. When the vehicle deviates, the nominal curvature no longer points toward the path, and the feedforward command propagates the error rather than correcting it. A *replanning* layer sits between the global reference and the feedforward computation. Reading the current pose error, it produces a locally corrected curvature target, which the feedforward controller then converts into a steering command. This section describes the five concrete replanning strategies implemented in the codebase, ordered from the simplest to the most structured [4, 9, 10].

6.1 Reference Curvature Extraction

Every controller begins by projecting the current vehicle position onto the reference path via a monotone nearest-point search. The search window $[\, i-5,\, i+500\,]$ around the previous index i prevents the match from retreating while still allowing a small backward correction. The reference curvature at the matched index i is

$$\kappa_{\text{ref}}(t) = \frac{\dot{\phi}_r(t)}{v_r(t)}, \tag{85}$$

and the nominal feedforward steering command is

$$\delta_{\text{ff}} = \arctan\big(L\,\kappa_{\text{ref}}\big). \tag{86}$$

Starting from this base curvature estimate, all feedforward and blended controllers compute their commands. Without any correction, this command tracks the reference curvature exactly *at the nearest point* but generates no restoring force when the vehicle is displaced from the path.

6.2 Lookahead Geometric Correction

The first replanning strategy substitutes the current nearest-point error for an error measured at a point L_d metres ahead along the reference arclength. The lookahead distance is proportional to speed, matching the path-tracking framing used in classic autonomous-vehicle steering methods [4]:

$$L_d = \max\big(L_{d,\min},\; v_x \cdot T_{\text{look}}\big), \tag{87}$$

with $T_{\text{look}} = 1.0$ s and $L_{d,\min} = 0.5$ m as defaults. The lookahead index j is obtained by walking forward from i along the reference until the accumulated arclength $\sum_{k=i}^{j-1} \|p_{k+1} - p_k\| \geq L_d$. At index j, the heading error and front-axle CTE are evaluated:

$$\theta_e^{(j)} = \phi_r(j) - \psi, \tag{88}$$

$$e^{(j)} = \sin\phi_r(j)\,\big(x_{\text{fa}} - x_r(j)\big) - \cos\phi_r(j)\,\big(y_{\text{fa}} - y_r(j)\big), \tag{89}$$

where $(x_{\text{fa}}, y_{\text{fa}}) = (x + L\cos\psi,\ y + L\sin\psi)$ is the front-axle position. The correction applied on top of the feedforward inverse output δ_{inv} is

$$\psi_{\text{corr}} = \theta_e^{(j)} + \arctan\left(\frac{K_{\text{la}}\, e^{(j)}}{L_d}\right), \qquad \delta = \delta_{\text{inv}} + \psi_{\text{corr}}. \tag{90}$$

This correction is used by the *Lookahead Corrected Body-Acceleration Inverse* family with gain $K_{\text{la}} = 1.0$. The effect is a local path replan. By connecting the vehicle's predicted front-axle position to the lookahead point on the reference, the correction generates a restoring curvature command even when the Newton inverse output is exactly on the nominal curvature.

In the *Predictive Lookahead Inverse*, the lookahead index is computed in *time steps* rather than arclength: $j = \min(k + \lfloor T_{\text{look}}/\Delta t \rceil,\ N-2)$. The vehicle state is propagated one step under the current feedforward command before evaluating $\theta_e^{(j)}$ and $e^{(j)}$, which compensates for actuator lag between command and actuation.

6.3 Curvature-Space PD Correction (Feedback Inverse)

The *Curvature Feedback Inverse* performs replanning in the curvature domain rather than the steering domain. The feedforward curvature $\kappa_{\text{ff}} = \tan\delta_{\text{ff}}/L$ is augmented by a PD correction computed from the current heading and lateral errors, in the same geometric-feedback spirit as classical Stanley-style path correction [4, 9]:

$$\kappa_{\text{fb}} = K_{p,\phi}\,\theta_e + K_{d,\phi}\,\dot{\theta}_e + K_{p,e}\,e + K_{d,e}\,\dot{e}, \tag{91}$$

with default gains $K_{p,\phi} = 0.8$, $K_{d,\phi} = 0.10$, $K_{p,e} = 0.6$, $K_{d,e} = 0.15$. To prevent the curvature correction from dominating at low speed (where geometric controllers are well-conditioned) and avoid large transients, the raw correction is gated by a speed-dependent sigmoid and low-pass filtered:

$$g(v_x) = \frac{1}{1 + e^{-(v_x - v_{\text{mid}})/k_\sigma}}, \quad v_{\text{mid}} = 0.7\ \text{m/s},\ k_\sigma = 0.25\ \text{m/s}, \tag{92}$$

$$\kappa_{\text{fb}}^* = \alpha\,\kappa_{\text{fb,prev}}^* + (1-\alpha)\,g(v_x)\,\kappa_{\text{fb}}, \quad \alpha = 0.85. \tag{93}$$

The total curvature command and steering output are

$$\kappa_{\text{tot}} = \kappa_{\text{ff}} + \kappa_{\text{fb}}^*, \qquad \delta_{\text{cmd}} = \arctan\big(L\,\kappa_{\text{tot}}\big). \tag{94}$$

Since the correction enters in the curvature domain, it is geometrically consistent; the resulting steering command always corresponds to a physically realisable circular arc of radius $1/\kappa_{\text{tot}}$. The low-pass filter with $\alpha = 0.85$ prevents chattering from derivative terms, and the sigmoid gate keeps the correction negligible below ~ 0.7 m/s where the denominator of the kinematic model becomes ill-conditioned.

6.4 CTE-Driven Blending (Blended Stanley)

The Blended Stanley controller (Section 4.3) can be understood as a replanning strategy that selects between two reference representations depending on how far the vehicle is from

the path. When the CTE is small, the controller trusts the global reference curvature (feedforward); when the CTE is large, it switches toward a locally corrective path implied by the Stanley law [9, 10]:

$$\delta = w\,\delta_{\text{ff}} + (1-w)\,\delta_{\text{stanley}}, \qquad w = \text{clamp}\big(e^{-a|e|},\ w_{\min},\ w_{\max}\big). \tag{95}$$

In this view, the Stanley term $\delta_{\text{stanley}} = \theta_e + \arctan(k\,e/(k_s + v))$ implicitly defines a locally corrected heading target - one that points the vehicle back toward the reference with a curvature proportional to the lateral error. The exponential weight w continuously interpolates between the global reference plan and this local geometric replan. The default parameters $w_{\min} = 0.30$ and $w_{\max} = 0.90$ ensure that even when the vehicle is far off-path, the feedforward contributes at least 30% of the command, preventing the controller from abandoning the global trajectory entirely.

Compared to the lookahead correction in Section 5.2, the Blended Stanley replan is reactive (it acts on the *current* CTE rather than a future one) and uses a kinematic correction that does not account for tire force limits. The lookahead correction, by contrast, uses the force-aware Newton inverse as its base and adds a geometric correction on top, so the physics constraints are satisfied first.

6.5 Yaw-Rate Anticipation Feedforward (Full-State LM Inverse)

When the reference curvature changes rapidly, as it does on entry to a tight hairpin corner, a purely reactive replanner incurs a lag equal to the controller's settling time. The *Full-State LM Inverse* controller reduces this lag by passing the desired yaw acceleration directly into the inversion as a feedforward term, extending the standard model-based feedforward and replanning idea used in trajectory-tracking literature [9, 11]:

$$\dot{r}_{\text{des}} = \frac{v_{r,k+1}\,\kappa_{k+1} - v_{r,k}\,\kappa_k}{\Delta t}, \tag{96}$$

where $\kappa_k = \dot{\phi}_r(k)/v_r(k)$ is read from the reference tuple one step ahead. Passed as the third component of the target vector $y^\star$ in (80), this term lets the Levenberg–Marquardt solver account for the anticipated curvature change when finding (F_{in}, δ). The result is a steering command that begins rotating the vehicle before the corner geometry is encountered, reducing the transient CTE on corner entry. This anticipatory feedforward is only meaningful when the reference curvature derivative is non-negligible; on straight segments or constant-radius arcs ($\dot{r}_{\text{des}} \approx 0$) the term has no effect.

Corner-entry CTE improvement from yaw-rate anticipation feedforward is quantified in Section 8.

6.6 Minimum-Time Drift-Aware Corner Planning

The replanning methods above modify the curvature target used by a tracking controller. A separate question is whether the reference itself should change when the demanded motion is close to, or outside, the tire friction boundary. In this thesis, drift-aware planning serves as a feasibility layer, establishing whether the requested trajectory lies within the achievable

friction budget before the inverse controller computes inputs. A minimum-time corner planner implements this idea by comparing a grip-constrained optimal-control problem with a drift-allowed variant on the same corner geometry. The planner uses the reduced dynamic bicycle model from Section 2.4, with state

$$x = [X,\ Y,\ \psi,\ v_x,\ v_y,\ r]^\top$$

and either the two-control input $u_2 = [\delta,\ a_x]^\top$ or the explicit four-control input

$$u_4 = [\delta,\ \text{throttle},\ \text{brake},\ \text{handbrake}]^\top.$$

Both formulations share the same vehicle dynamics, path geometry, track bounds, speed floor, actuator limits, and tire-utilization checks; they differ in the permitted sideslip envelope and the warm-start family used to initialize the nonconvex solve.

For a fixed radius R, turn angle $\Delta\psi_c$, entry speed v_0, and friction coefficient μ, the planner minimizes corner traversal time subject to discrete dynamics,

$$x_{k+1} = f(x_k, u_k),$$

and feasibility constraints on path deviation, tire utilization, speed, and terminal progress. The drift-allowed solve is therefore not counted as a success merely because it produces a shorter nominal trajectory; it must also satisfy the same admissibility checks as the grip solution. In the analysis tables, a scenario is classified as a drift win only when both plans are feasible and

$$T_{\text{grip}} - T_{\text{drift}} > 0.05 \text{ s}.$$

Smaller differences are treated as ties.

The four-input transient maneuver planner inserts an outer parameter search before local refinement. Rather than optimizing a full control sequence from scratch, it samples over interpretable maneuver phases: braking onset and magnitude, preload and flick steering, throttle reapplication, handbrake timing and duration, target sideslip, and catch gain. The best rollout candidate then warm-starts local direct shooting, keeping the discrete search over maneuver families separate from the continuous refinement step.

Planner results are interpreted with a mechanism label. A low-sideslip drift win, with $|\beta_{\text{peak}}| < 5^\circ$, is labelled a *line-shaping win*: the drift-allowed optimizer found a faster admissible line, but the result is not evidence of high-angle drifting. Wins with $5^\circ \le |\beta_{\text{peak}}| < 10^\circ$ are labelled moderate-sideslip wins. A controlled-oversteer claim requires $|\beta_{\text{peak}}| \ge 10^\circ$ together with recovery evidence after the corner exit. Flick or handbrake claims are reserved for trajectories whose explicit four-control sequence contains the brake/preload/flick/catch structure and whose transfer is validated without collapsing the handbrake channel into a net longitudinal acceleration.

Finally, the planning claim is tied to recoverability. A drift-allowed plan is counted as practical evidence only if it is faster than the grip plan, passes the nominal feasibility checks, and recovers from bounded perturbations in entry speed, lateral offset, heading offset, and friction. This definition prevents a faster but fragile trajectory from being counted as a useful drift benefit.

7 Data-Driven Residual Identification

7.1 Motivation

Even with the enhanced bicycle model, MuJoCo rollouts show a persistent gap between predicted and observed state derivatives. Since this gap appears across trajectories and operating points, the analysis treats it as a structured residual rather than random numerical error. Residual identification learns a correction $\Delta f(x, u)$ so that the corrected model, $\dot{x} = f_{\text{enhanced}}(x, u) + \Delta f(x, u)$, better matches the MuJoCo-observed dynamics. This follows the residual-learning framing used in modern data-driven dynamics work [15–19].

The methods surveyed here serve different modeling roles. SINDy and symbolic regression are most useful when interpretability matters, since they recover sparse, physically grounded terms. Gaussian process state-space models add explicit uncertainty estimates and provide a route toward online adaptation, which is useful when measurements arrive incrementally. For real-time prediction, Koopman and eDMD embeddings trade some representational flexibility. Hybrid sparse-neural models sit between these extremes, combining a readable physics backbone with the additional capacity of a learned residual [15–19].

7.2 Four-Subsystem Architecture

Residual identification is decomposed into four physically motivated subsystems. Each subsystem is fitted independently and applied in cascade at each RK4 substep. The methods span sparse discovery [15], Koopman lifting [16], variational latent dynamics [17], and symbolic regression [18, 19]:

Order	Subsystem	Features	Target	Applied as
1	Steering	δ_{cmd}, v_x	$\Delta\delta$	Corrects effective steer before slip angles
2	Roll	a_y, v_x, r	ΔF_{zf}, ΔF_{zr}	Corrects normal load before tire eval
3	Tire	α_f, α_r, F^*_{zf}, F^*_{zr}	ΔF_{yf}, ΔF_{yr}	Injected inside EOM as extra lateral force
4	Chassis	v_x, v_y, r, δ	$\Delta\dot{v}_y$, $\Delta\dot{r}$	Post-hoc state-rate residual

Table 8: Cascade architecture for residual identification. Each subsystem is fitted independently; subsystems are applied in order at each integration step.

The tire correction is injected at force level before $\dot{v}_y$ and $\dot{r}$ are computed, so all four RK4 substeps integrate the corrected forces. Injecting at force level prevents the coupling loss that post-hoc state-rate addition would otherwise introduce. Figure 18 illustrates the signal flow.

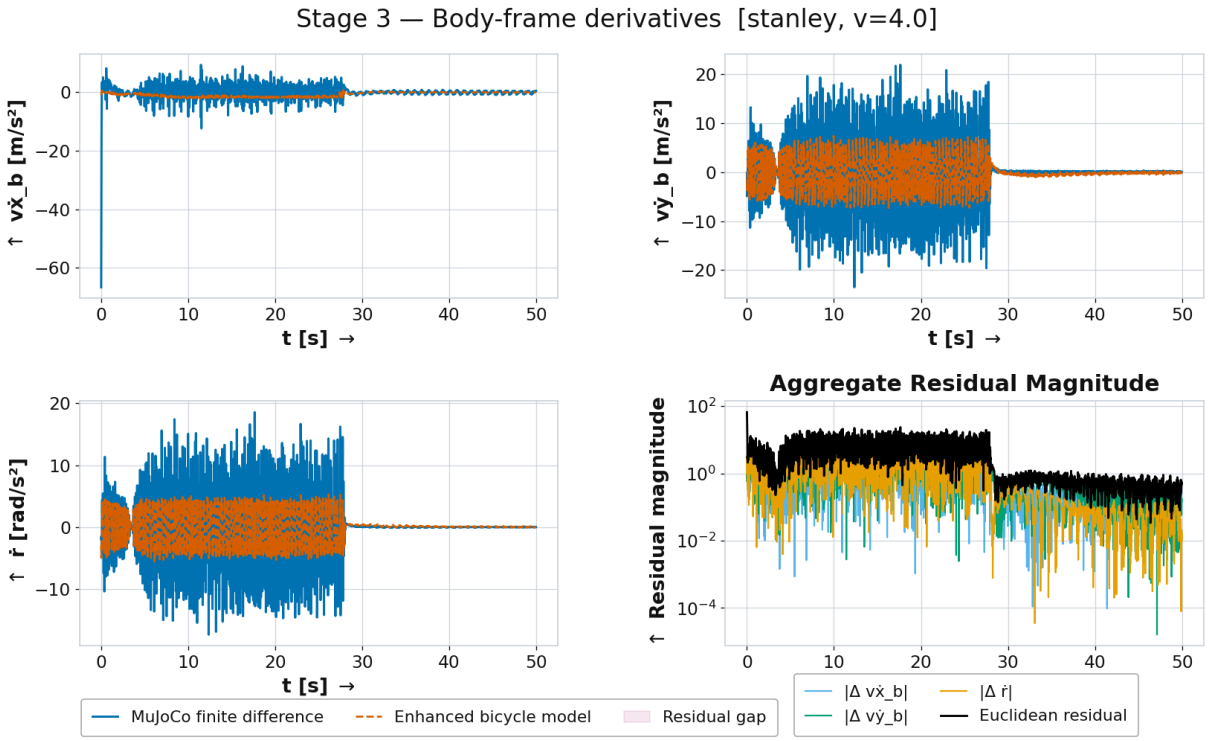


Figure 16: Body-frame state-derivative comparison: enhanced bicycle model prediction (orange) vs. MuJoCo finite-difference ground truth (blue). The shaded gap between the two traces is the systematic residual $\Delta\dot{v}_y$ and $\Delta\dot{r}$ that the data-driven correctors must learn. The residual is persistently one-sided rather than random, indicating a structural gap in the physics model. Legends appear to the right of each subplot panel.

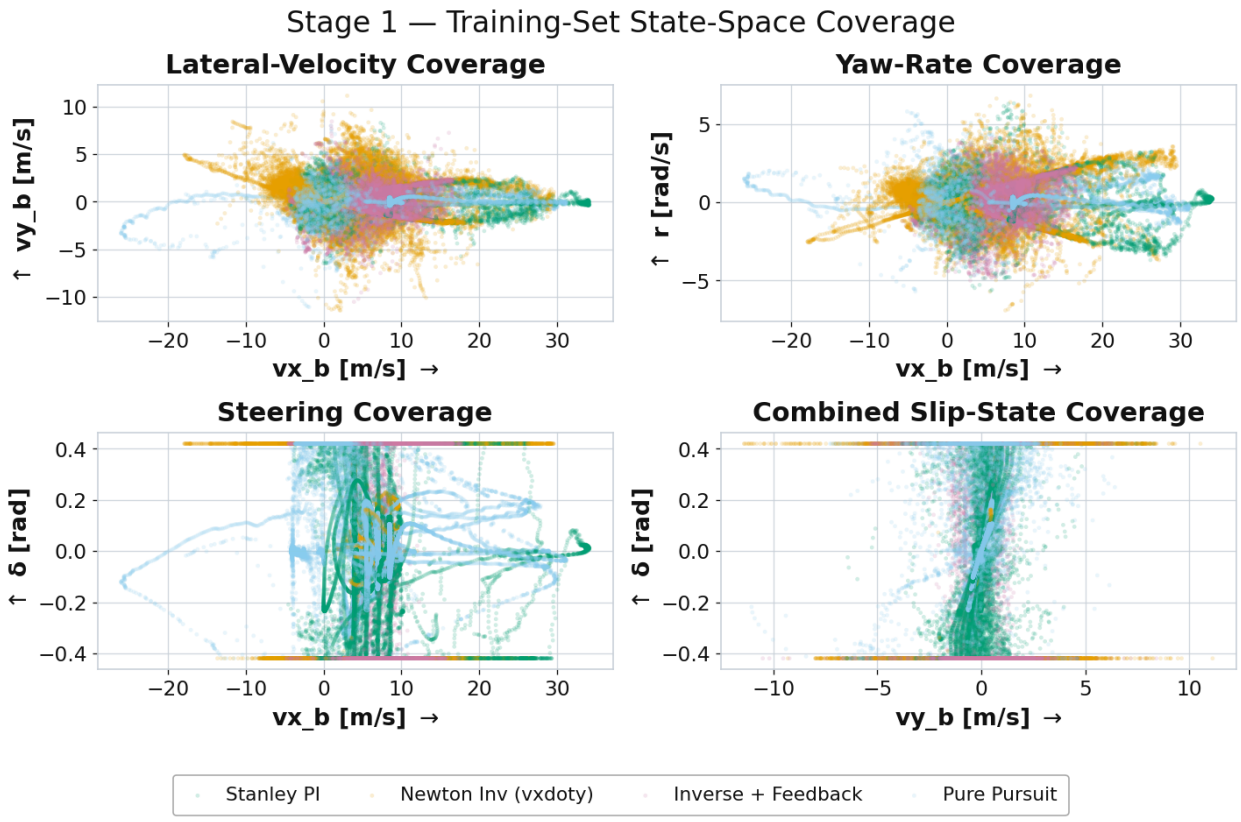


Figure 17: Training-set state-space coverage of the enriched dataset (≈320 000 MuJoCo samples, 4 trajectories × 4 speeds × 4 controllers). Each panel shows one 2-D slice of the (v_x, v_y, r, δ) state space. Dense coverage at low v_x and sparse coverage at high $|v_y|$ and $|r|$ reflects the operating regime of the reference trajectories; the correctors should be applied with caution at operating points outside the dense region.

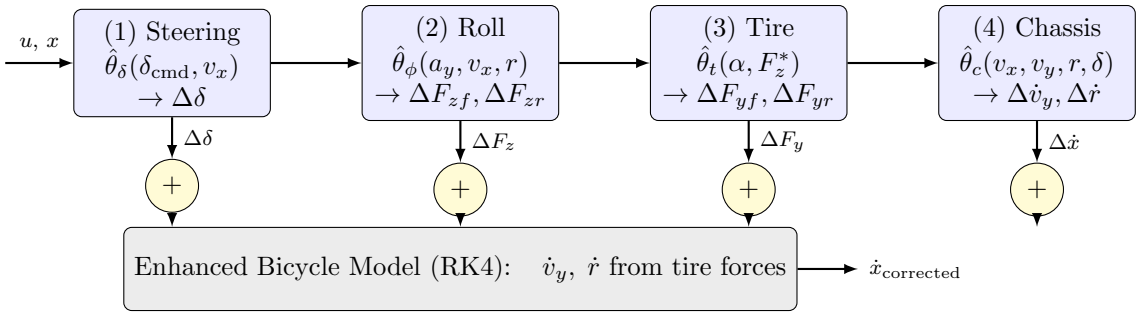


Figure 18: Four-subsystem cascade architecture. Subsystems (1)–(4) are fitted independently and applied in order at each RK4 substep. Steering correction adjusts the effective steer angle before slip angles are evaluated; roll correction adjusts normal loads before tire force evaluation; tire correction injects extra lateral force at force level; chassis correction adds a post-hoc state-rate residual.

7.3 Methods Compared

The residual study evaluates five method families. **SINDy** [15] (Sparse Identification of Nonlinear Dynamics) uses mixed-integer optimization (MIOSR) to select a sparse set of basis functions from a physics-inspired library. **Koopman/EDMD** [16] lifts the state into an expanded observable space using polynomial and RBF features, then fits a linear predictor by least squares. **GP-SSM** [17] applies sparse variational Gaussian processes, providing uncertainty estimates that can gate the applied correction. **Hybrid SINDy+NN** [19] uses SINDy for the interpretable part of the residual and a multilayer perceptron for the remaining error. **Symbolic Regression** [18] uses a physics-constrained genetic search to recover compact analytical expressions.

SINDy is the baseline because it produces readable, differentiable corrections. Sparse polynomial or bilinear terms can be read directly from the coefficient vector. That readability is important when the learned correction is embedded inside an inverse controller. The chassis, roll, and steering subsystems fit this structure well because their governing relationships are low-order in the chosen variables. The tire subsystem presents a harder identification target. Pacejka-style force laws contain $\sin(\arctan(\cdot))$ compositions that ordinary polynomial libraries cannot reproduce exactly, so the study includes methods with stronger approximation capacity.

Gaussian-process residual models return a mean correction together with an uncertainty estimate. In an online setting, the uncertainty estimate could gate the feedforward command in low-confidence regions. The posterior covariance structure also opens a path to friction adaptation, though neither has been validated in closed-loop hardware or MuJoCo experiments here. This thesis does not claim that such online adaptation has been validated on hardware or in closed-loop MuJoCo experiments; here the GP-SSM is evaluated as an uncertainty-aware residual predictor.

Koopman/eDMD represents the dynamics with a linear operator in a chosen lifted feature space. That structure suits real-time prediction: one matrix–vector multiply propagates the lifted state at each step, with no nonlinear solve. In practice, a single fitted Koopman

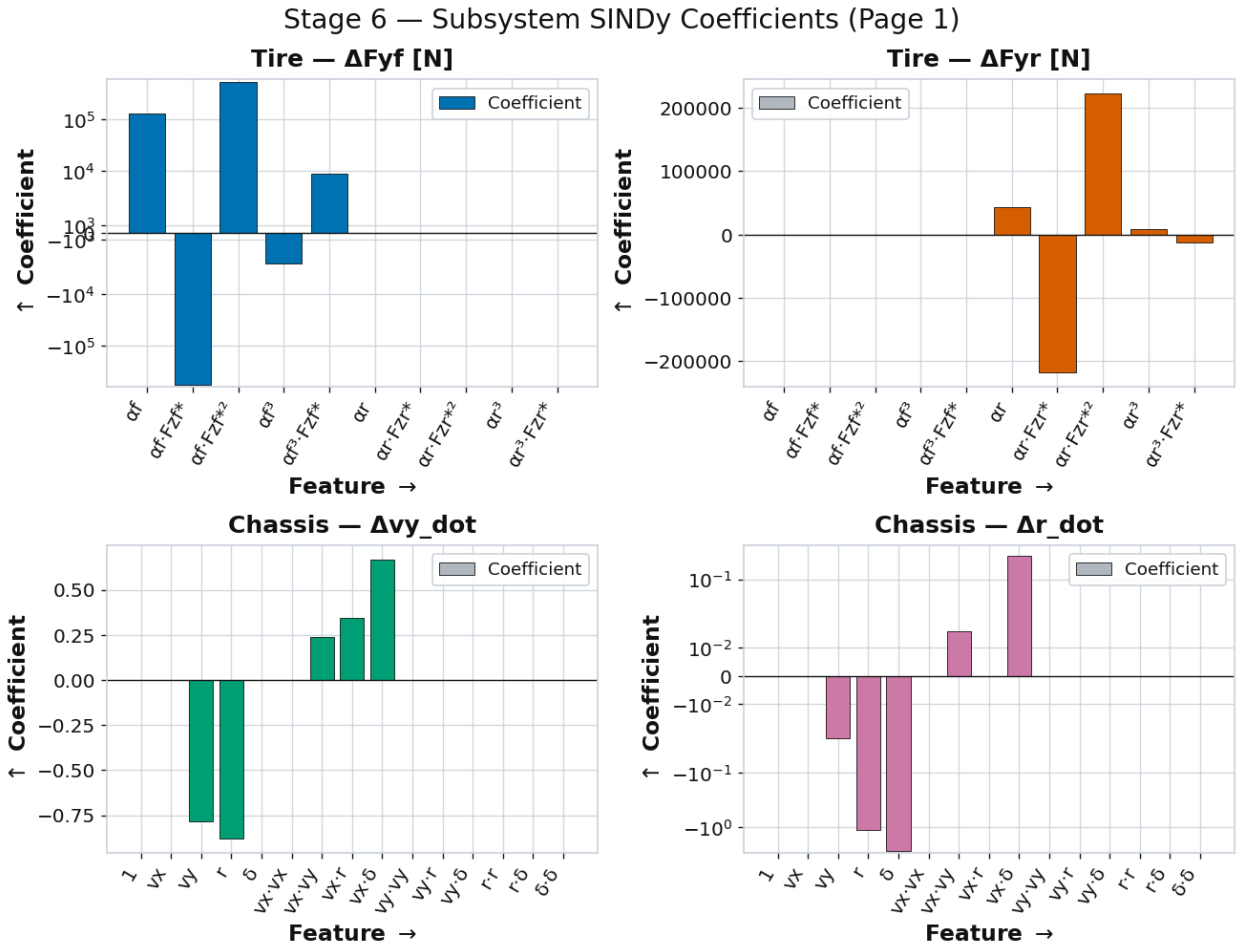


Figure 19: SINDy-identified coefficients for each of the residual-identification subsystems. Non-zero bars indicate which library terms were selected by the sparse solver; zero bars indicate terms that were pruned. The physics-constrained tire library (odd-in-α, no cross-axle terms) is visible in the tire and chassis-related panels.

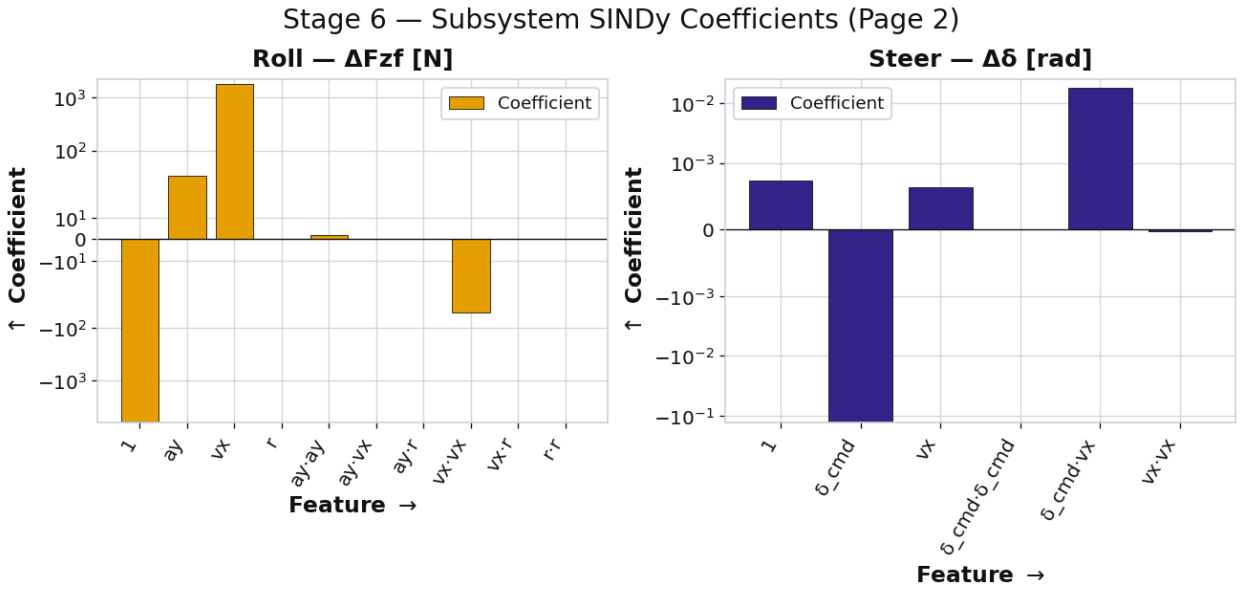


Figure 19: SINDy-identified coefficients for each of the residual-identification subsystems (continued).

operator must cover low-slip, high-slip, and saturated operating regimes simultaneously. Koopman therefore serves as a complement to the inversion framework rather than evidence that a single lifted linear model can replace the nonlinear controller.

Hybrid SINDy+NN and symbolic regression anchor opposite ends of the accuracy-interpretability tradeoff. Keeping the sparse SINDy component, the hybrid model adds a neural residual to absorb structure the library misses. Symbolic regression runs offline as a structure-discovery diagnostic; it is not embedded in the runtime controller. If symbolic regression finds a compact equation that differs from the assumed Pacejka structure, the implication is that the physics baseline itself needs revision before any residual corrector is added.

7.4 Physics-Constrained Tire Library

For the tire subsystem, the basis function library is constrained to match known tire physics: basis terms are odd in slip angle α because cornering force is antisymmetric, cross-axle coupling terms are excluded, and constant offsets are not permitted because zero slip should produce zero lateral force. These constraints cut overfitting, make the identified correction interpretable, and align with the tire-model assumptions in the bicycle and Magic-Formula literature [5–8].

7.5 Wheel-Resolved Normal-Load Identification

The two-wheel bicycle model used by the inverse controller treats each axle as a single lumped contact with a fixed normal load. Real vehicles redistribute load dynamically under braking, acceleration, and cornering. That redistribution introduces wheel-specific Pacejka asymmetry that the bicycle model cannot capture. This subsection extends the residual-identification stack to four-wheel normal-load prediction, using MuJoCo contact forces as ground truth.

Calibrated four-wheel analytical baseline. A spring–damper suspension correction is added per wheel:

$$F_{z,w} = F_{z,w}^{\text{static}} + k_{\text{fs}}\, x_w + c_{\text{fs}}\, \dot{x}_w, \tag{97}$$

where x_w is the suspension deflection at wheel w, and differential evolution fits k_{fs} and c_{fs} globally to the MuJoCo contact force dataset. Benchmark v8 yields $k_{\text{fs}} = -141.9$ kN/m and $c_{\text{fs}} = +12{,}983$ N·s/m. Figure 22 shows the step response of the calibrated suspension and the resulting normal-load prediction on a representative front-left wheel rollout, where the uncalibrated proxy achieves RMSE 2476 N, dropping to 622 N after the $k_{\text{fs}}/c_{\text{fs}}$ correction, a 75% reduction.

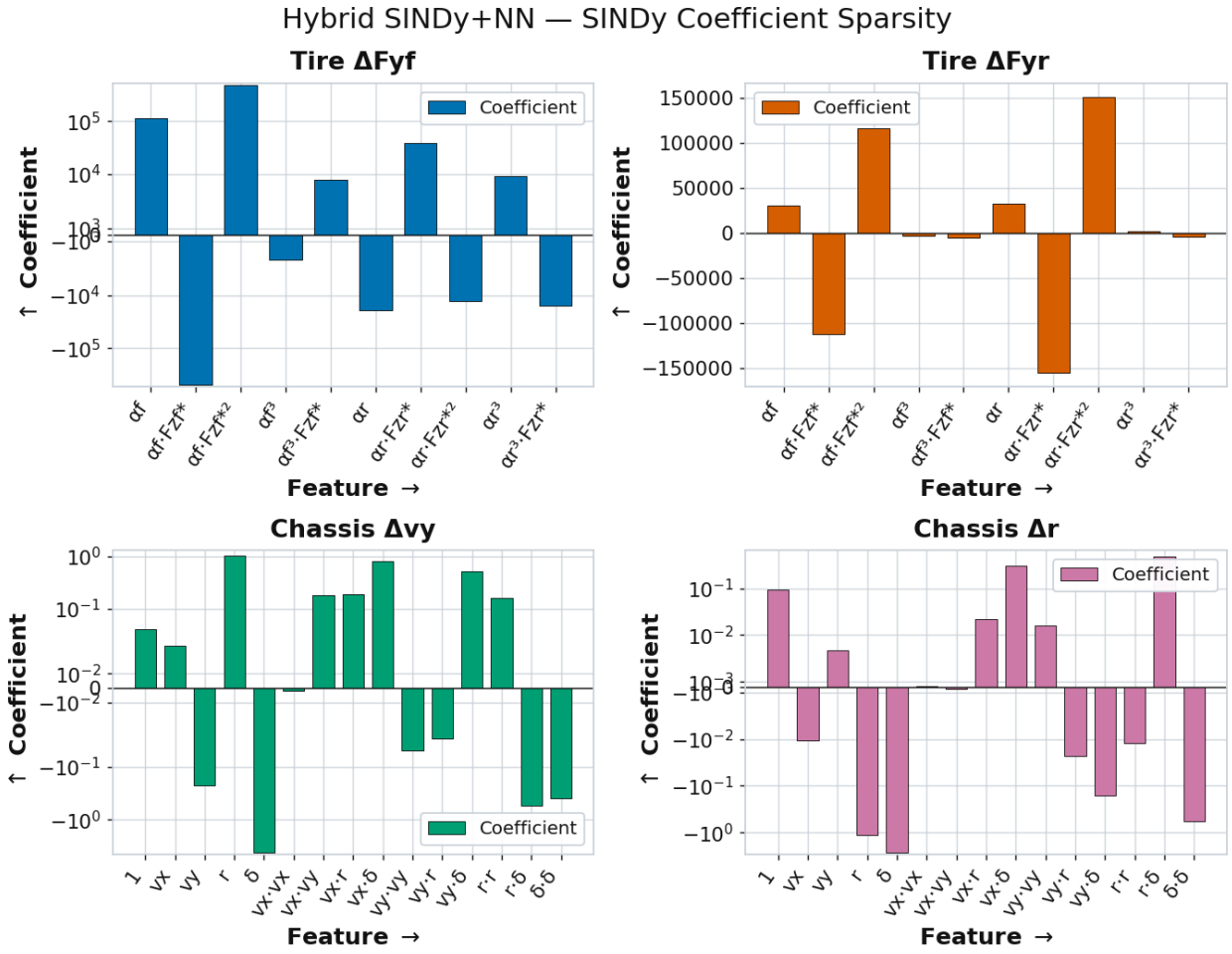


Figure 20: SINDy/Hybrid coefficient sparsity pattern across all four subsystems. Bar height is coefficient magnitude; features pruned to zero by the sparsity threshold are absent. Only selected terms are non-zero, confirming interpretable structure in the residual correction.

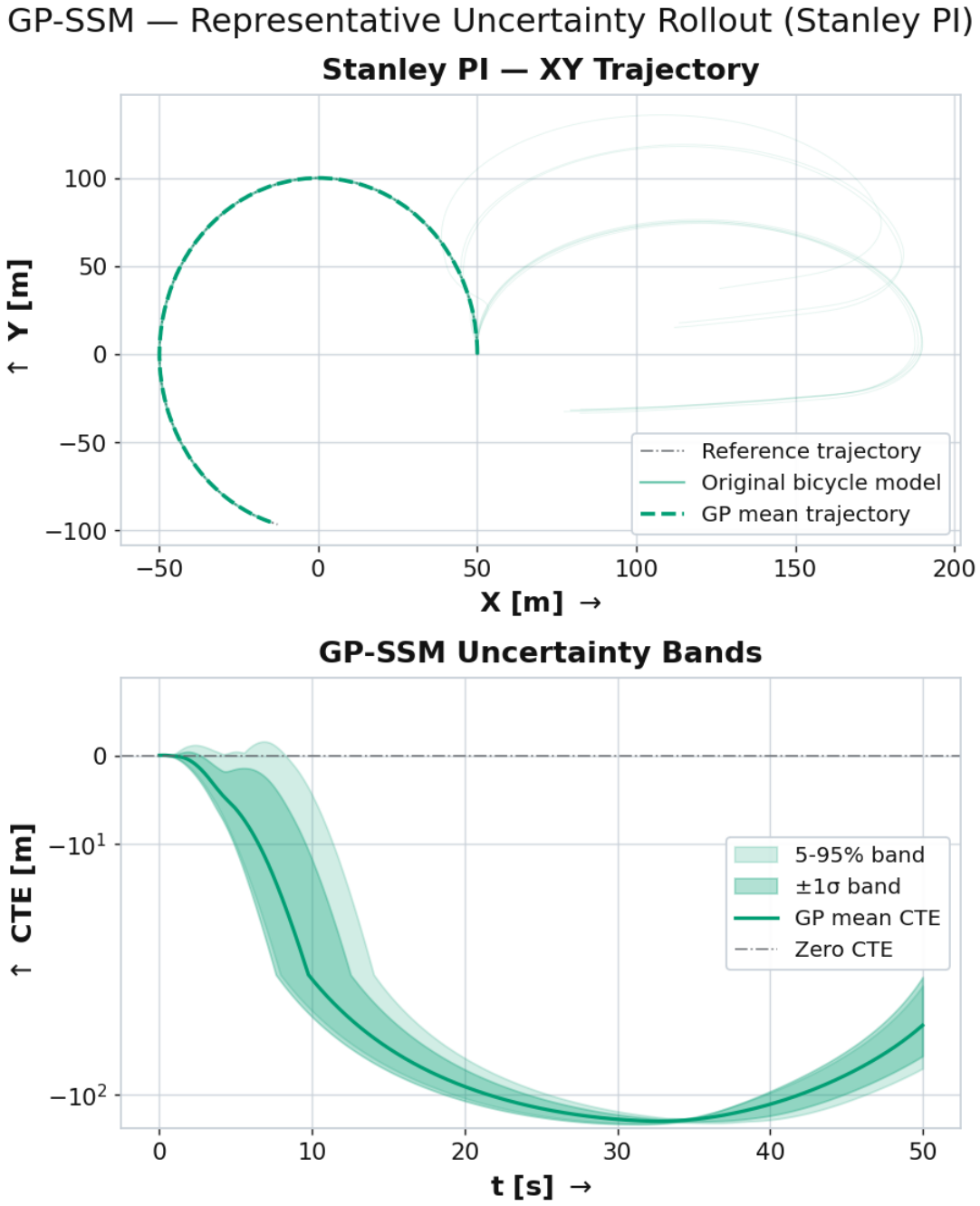


Figure 21: GP-SSM uncertainty rollout (Stanley PI). *Top*: XY trajectory; reference (dash-dot), bicycle model (solid), GP mean (dashed). *Bottom*: CTE with $\pm 1\sigma$ and 5–95% bands; expanding bands mark low-confidence regions.

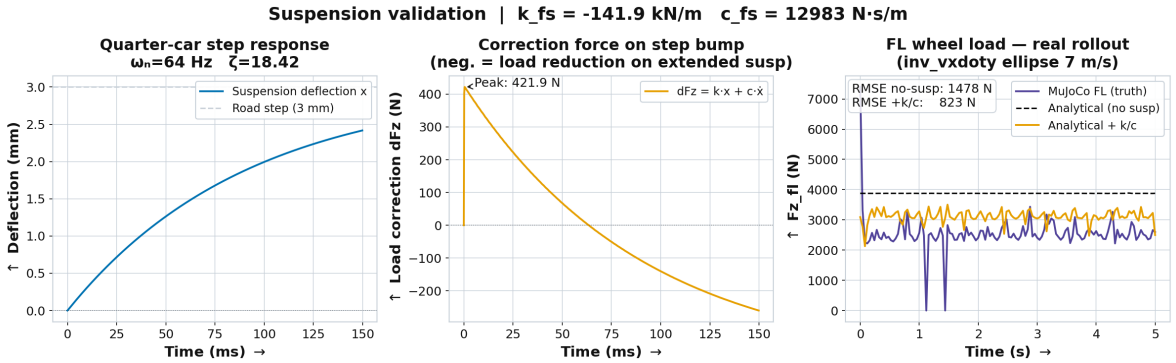


Figure 22: Suspension calibration validation. (Left) Quarter-car step response of the calibrated spring–damper; natural frequency $\omega_n = 64$ Hz, damping ratio $\zeta = 18.4$. (Centre) Correction force ΔF_z during a representative rollout, peak ≈400 N. (Right) Front-left wheel normal load: MuJoCo ground truth vs. two-wheel proxy vs. calibrated four-wheel model. RMSE drops from 2476 N (proxy) to 622 N (calibrated). Script provenance is listed in Appendix D.

Data-driven residual models. All four data-driven models are trained and tested against the calibrated four-wheel physics baseline, using a 29-feature library that includes a roll-rate proxy, longitudinal load transfer, and per-axle slip terms. SINDy uses DAgger iteration and MIOSR best-subset selection to keep the correction sparse. Koopman/Hankel-DMD uses a 5-delay Hankel embedding with spectral-norm clipping. Of the models evaluated, it is the clearest candidate for future real-time linear-predictor deployment. GP-SSM uses an ARD kernel and SVD-ICM coregionalization with a capped training set and multiple restarts to expose uncertainty. Hybrid SINDy+NN keeps the sparse SINDy component and clips the neural correction to suppress high-slip extrapolation. Symbolic regression is run as an offline structure-discovery diagnostic for this benchmark and is excluded from the aggregate RMSE table.

Benchmark results. Table 9 reports aggregate per-wheel RMSE across all four controllers on a chronological hold-out from the same ellipse trajectory at 7 m/s. Figure 23 shows the grouped bar chart by model and controller.

Table 9: Aggregate wheel-load RMSE (N) for the four-wheel normal-load benchmark, evaluated on a chronological hold-out from the same ellipse 7 m/s rollouts. "Proxy" is the two-wheel axle-level baseline; improvement is computed relative to that proxy. Script and CSV provenance are listed in Appendix D.

Model	**Aggregate RMSE (N)**	**vs. Proxy**
2-wheel proxy (baseline)	1512	–
4-wheel physics	1147	+24.2%
SINDy	1201	+20.6%
Koopman/Hankel-DMD	1044	+30.9%
GP-SSM (ARD)	1067	+29.4%
Hybrid SINDy+NN	1033	+31.7%

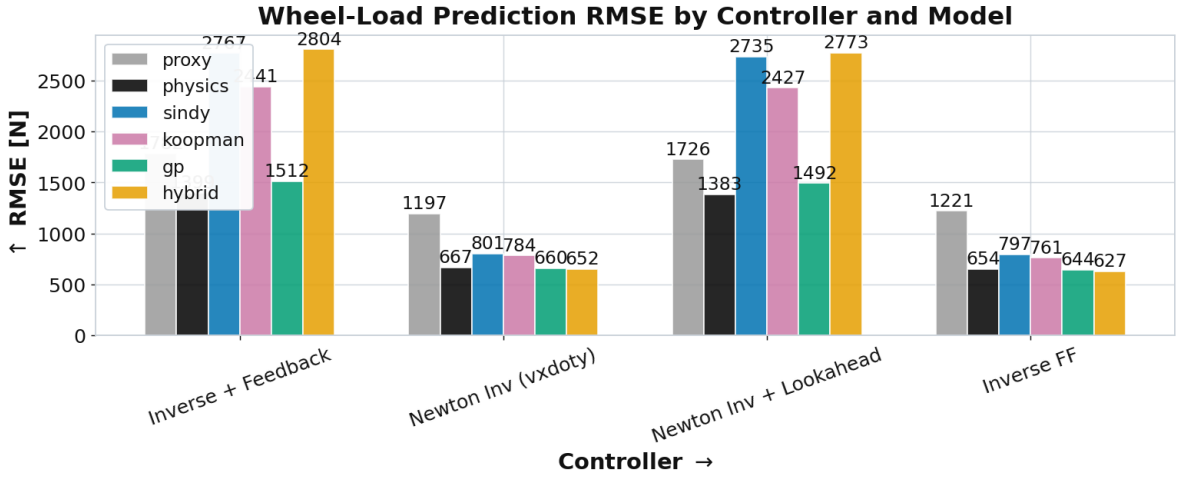


Figure 23: Grouped aggregate wheel-load RMSE by model and controller for the ellipse 7 m/s hold-out. Hybrid SINDy+NN has the lowest aggregate RMSE (1033 N, 31.7% improvement vs. the two-wheel proxy), and Koopman/Hankel-DMD is second (1044 N, 30.9%), making it the best linear-predictor candidate for future real-time evaluation.

All four implemented data-driven models reduce wheel-load RMSE below the two-wheel proxy on this hold-out. Hybrid SINDy+NN achieves the lowest aggregate RMSE (1033 N). This result is enabled by constraining the neural correction to ±2000 N, which prevents over-correction in the high-slip extrapolation regime. Among the wheel-load predictors, Koopman/Hankel-DMD is the strongest candidate for future real-time evaluation. It achieves a 30.9% improvement. The underlying linear Koopman operator supports direct linearised control synthesis, which the nonlinear SINDy and GP models do not. The SINDy model (1201 N) performs below both Koopman and GP despite having the most interpretable structure. The learned coefficients show that DAgger iteration broadens the training distribution but also brings in distribution shift that the sparse polynomial library cannot fully absorb.

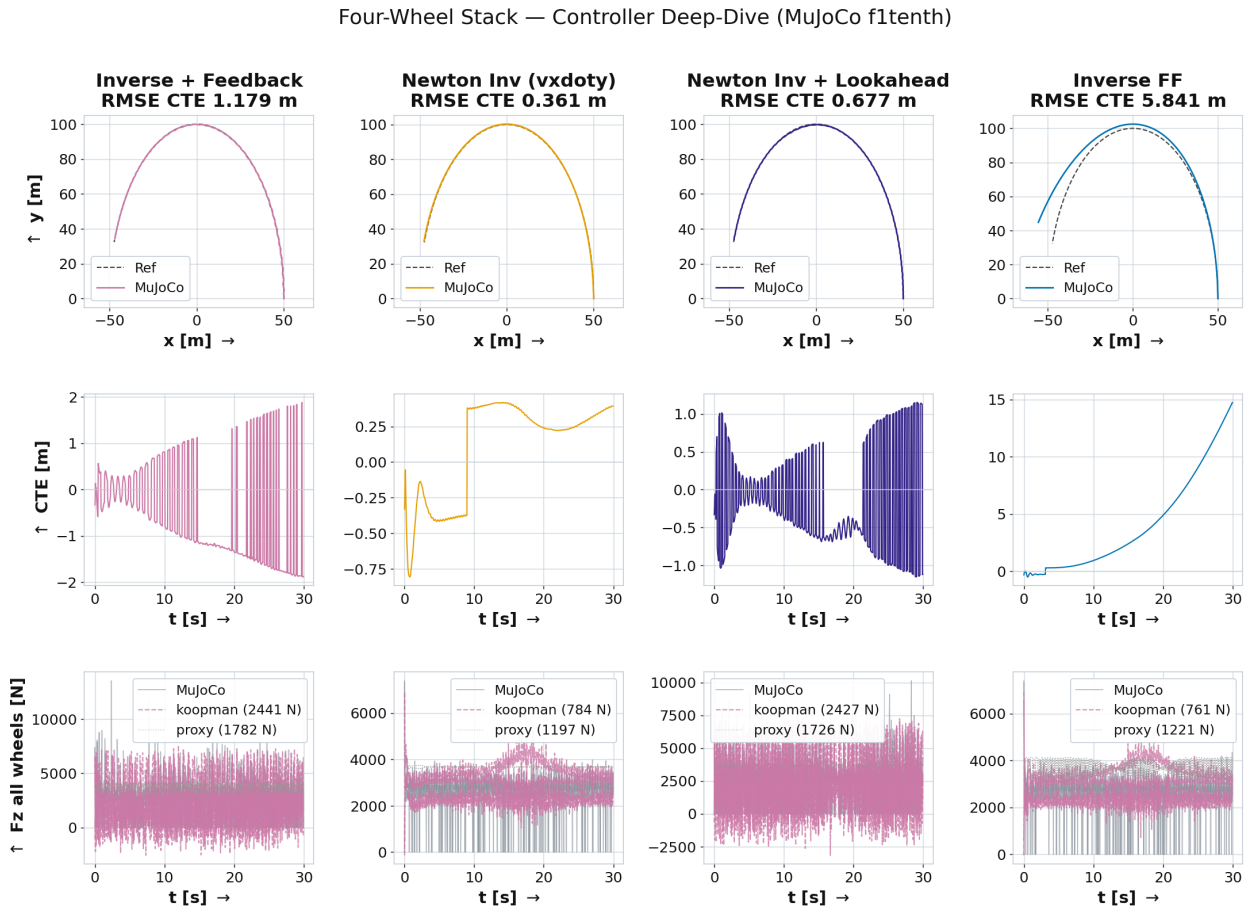


Figure 24: Per-controller four-wheel normal-load deep dive: XY trajectory (top row), cross-track error (middle row), and per-wheel F_z time series comparing MuJoCo ground truth, the two-wheel proxy, and the best learned model (bottom row). Four columns correspond to the four benchmark controllers.

7.6 Results Summary

Table 10 reports the one-step prediction RMSE of each method at dataset points drawn from the enriched MuJoCo dataset (4 trajectories × 4 speeds × 4 controllers, ≈320k samples).

Table 10: Residual-identification error on an 8000-sample MuJoCo evaluation set. The table reports end-to-end one-step residual prediction error for all methods. Per-subsystem values are reported only for SINDy because it is the only decomposed pipeline whose intermediate tire, chassis, and roll targets are exported separately; the other methods are evaluated as end-to-end correctors. The raw generation artifacts are listed in Appendix D.

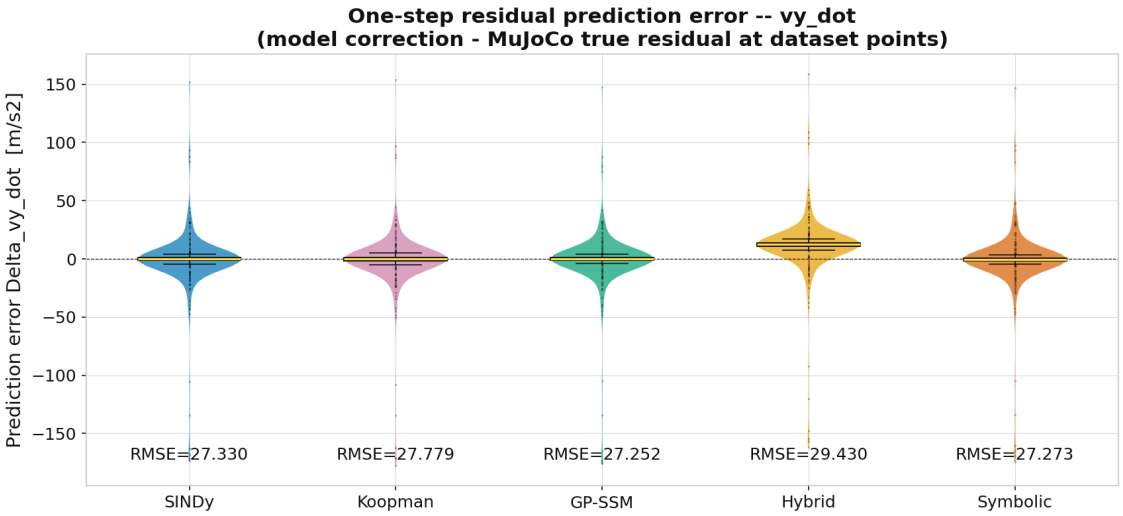
Method	RMSE $\Delta\dot{v}_y$ (m/s^2)	RMSE $\Delta\dot{r}$ (rad/s^2)	Notes
SINDy	27.33	12.54	sparse interpretable baseline
Koopman/eDMD	27.78	12.57	lifted linear predictor
GP-SSM	27.25	12.30	uncertainty-aware predictor
Hybrid SINDy+NN	29.43	12.57	sparse model plus neural residual
Symbolic Reg.	27.27	12.39	offline expression search

The residual RMSE of approximately 27–29 m/s^2 in $\Delta\dot{v}_y$ is larger than the typical lateral acceleration magnitude ($a_y \in [-5, +5]$ m/s^2 at $v_x = 7$ m/s in the dataset). At this stage, the identified corrections do not reliably recover the true MuJoCo residual. The finding is negative and diagnostic. While the learned residual models can be fit and compared, the current data and model structure are insufficient to improve closed-loop controller performance reliably. Two interpretations are consistent with this result. First, the four-subsystem cascade decomposition may not cleanly isolate residual sources, so errors in upstream subsystem fits contaminate downstream targets. Second, the enriched dataset likely under-represents the high-slip operating points, where the model gap is largest. Future work should explore tighter data coverage near the friction limit or end-to-end identification without cascade decomposition.

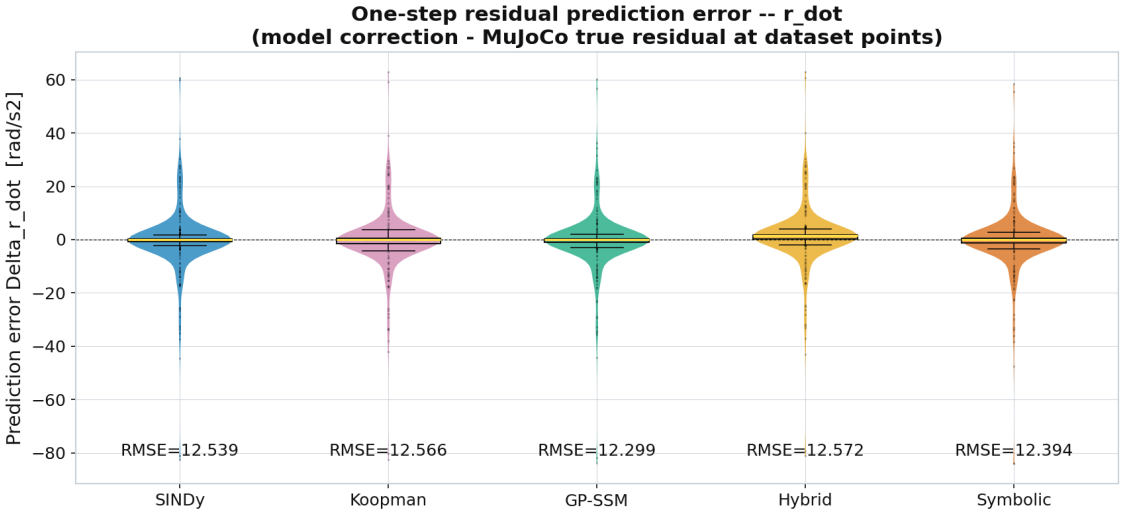
Section 8 evaluates the residual identification pipeline by its effect on closed-loop tracking. The evaluation criterion is the trustworthiness of the resulting controller when plugged into the inversion framework, not the one-step prediction RMSE of the regressor.

8 Experimental Evaluation

Section 8 benchmarks the iterative inverse controller in MuJoCo simulation. The section also addresses model-fidelity dependence, the speed-tracking caveat that confounds raw CTE comparisons, and the sensitivity of the inverse controller to friction-model mismatch. For all quantitative controller and model comparisons in this section, results come from MuJoCo simulation rather than hardware. The physical RC-car platform, telemetry stack, OptiTrack integration, and LiDAR/IMU state-estimation workflow are documented in Appendix A as the intended hardware-validation pipeline. No hardware CTE benchmark appears in the present work.

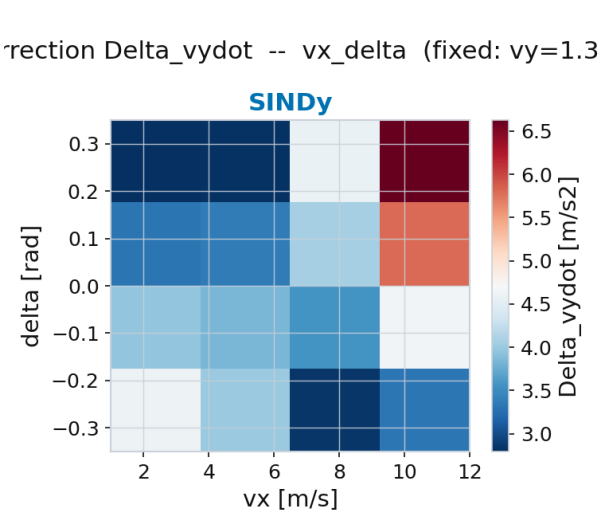


(a) $\Delta\dot{v}_y$ prediction error distribution per method.

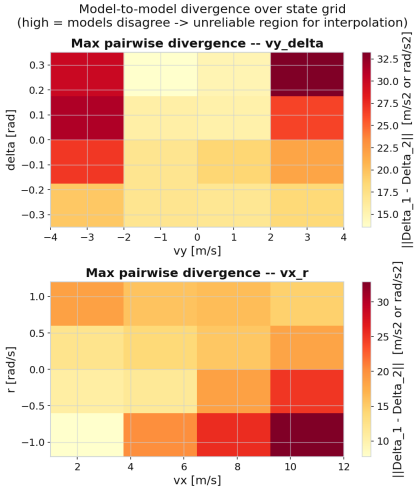


(b) $\Delta\dot{r}$ prediction error distribution per method.

Figure 25: Violin and box plots of one-step residual prediction error at 8000 randomly sampled MuJoCo dataset points. All methods show similar RMSE ($\approx$27–29 m/s^2 for $\Delta\dot{v}_y$, $\approx$12.3–12.6 rad/s^2 for $\Delta\dot{r}$), indicating that the gap is systematic rather than method-dependent.



(a) $\Delta\dot{v}_y$ correction surface: v_x–δ slice.



(b) Max pairwise model disagreement over state grid.

Figure 26: State-grid analysis. (a) Heatmap of total $\Delta\dot{v}_y$ correction predicted by the SINDy corrector over the (v_x, δ) state grid; largest corrections at high speed and high steering angle. (b) Maximum pairwise disagreement between any two methods at each grid point, with high-disagreement regions indicating where model uncertainty is largest.

8.1 Simulation Benchmark Protocol

The following conditions produced the quantitative controller and residual-identification results in Table 10 and Figures 29–31; all runs can be reproduced from the MuJoCo model and the dataset collection loop [20].

Reference trajectories. The model-fidelity benchmark uses four analytically parameterized reference trajectories. The benchmark parameterizes each trajectory as a closed-form $(X_{\text{ref}}(s), Y_{\text{ref}}(s))$ curve with a corresponding curvature profile $\kappa_{\text{ref}}(s)$. The ellipse (semi-axes 20 m and 10 m) subjects the controller to sustained cornering with slowly varying curvature. A 15 m-radius circle isolates the constant-curvature regime for a steady-state stability check. The figure-8, formed by two 10 m-radius circles joined at a crossing, reverses lateral acceleration sign once per lap. The slalom uses a 5 m sinusoidal lateral offset with 40 m pitch, producing the most rapid steering transients in the benchmark.

Reference speeds. The benchmark evaluates each trajectory at four reference speeds: 4.0, 5.5, 7.0, and 8.5 m/s. At 4.0 m/s the slip-angle linearisation breaks down, placing the controller in the high-slip low-speed regime; at 8.5 m/s the lateral acceleration on the tightest circle segment reaches ≈ 4.8 m/s^2 at the nominal $\mu = 0.90$, approaching the friction limit.

MuJoCo simulation settings. All simulation benchmarks run in the race-car MuJoCo scene, which includes a 6-DOF free joint, per-wheel suspension, and six-dimensional frictional contact. The MuJoCo timestep is 4 ms, and the model-fidelity sweep evaluates the controller

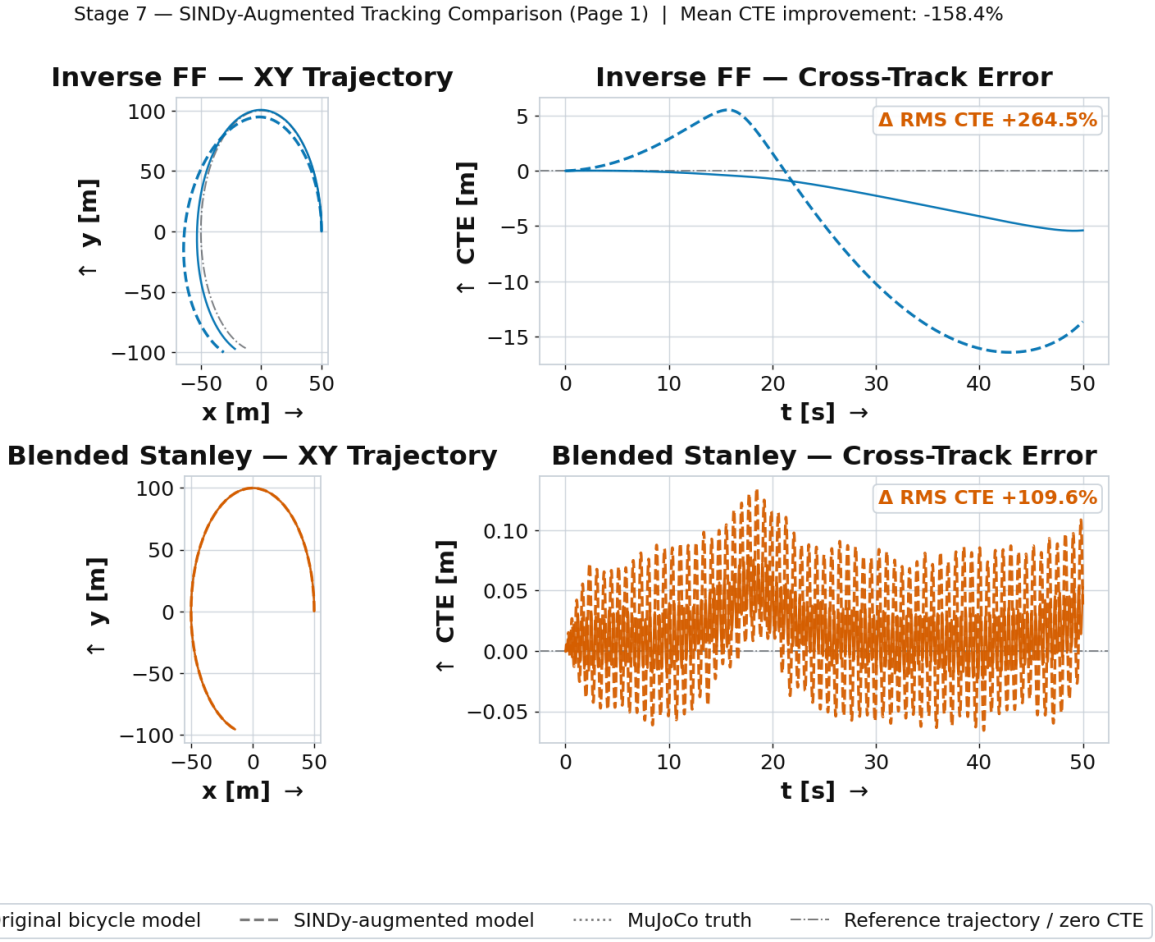


Figure 27: Trajectory and cross-track-error comparison for the ellipse reference at 7 m/s. The controller-wise panels are split across two figure pages so each row remains legible. Each row contrasts the original enhanced bicycle model, the SINDy-corrected model, and the MuJoCo ground truth against the same reference path.

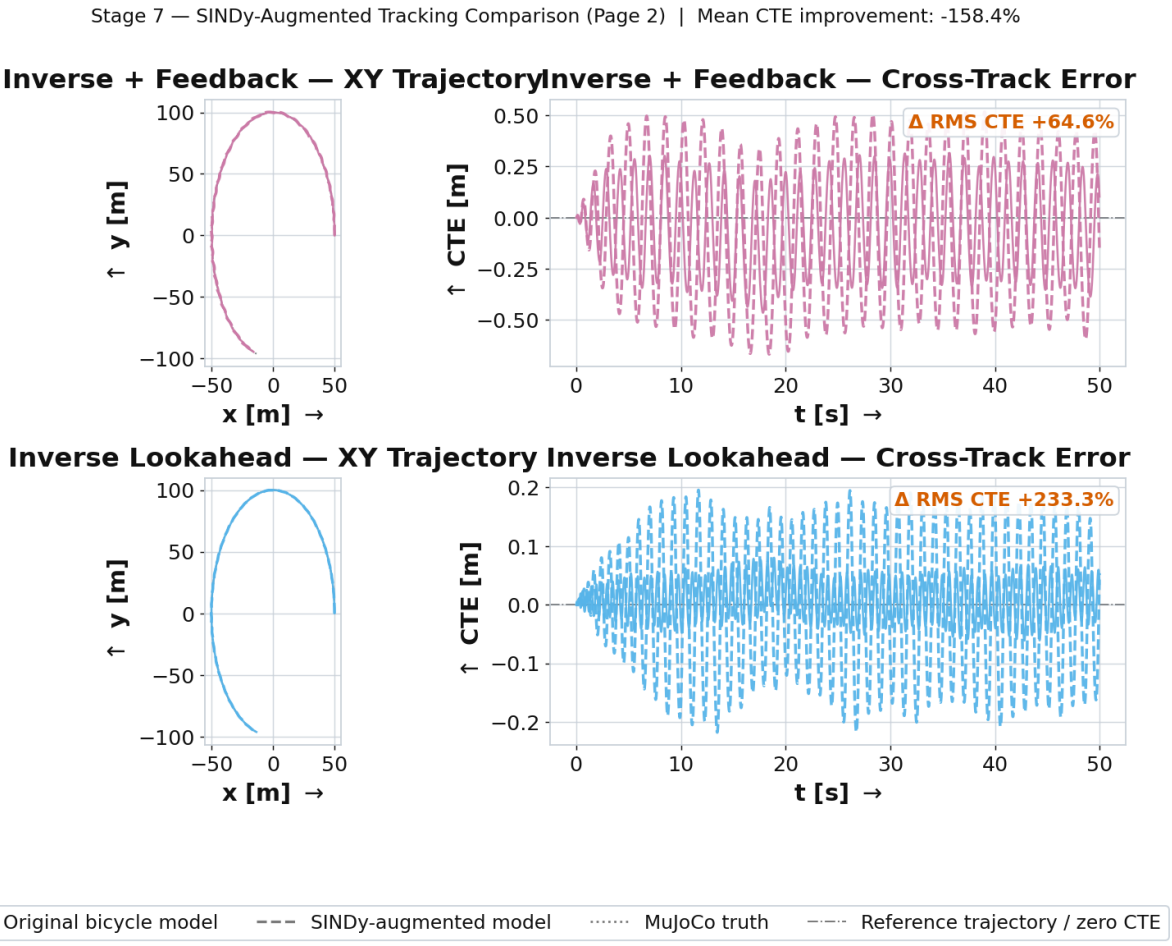


Figure 27: Trajectory and cross-track-error comparison for the ellipse reference at 7 m/s (continued).

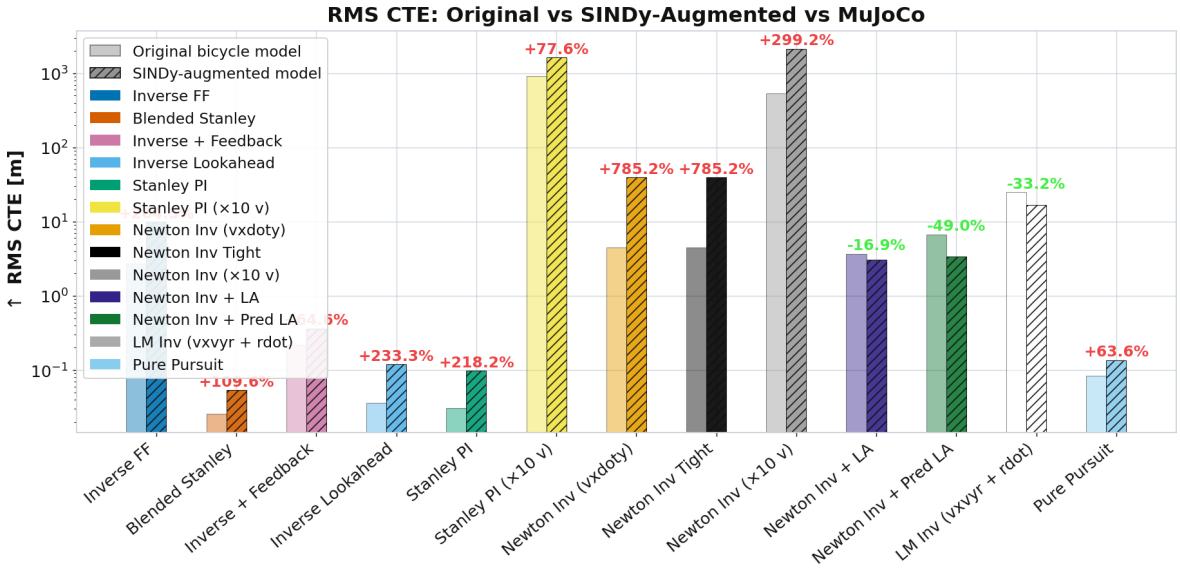


Figure 28: RMS cross-track error by controller: original enhanced bicycle model (solid) vs SINDy-corrected (hatched). Controllers where the SINDy corrector reduces CTE are those whose operating regime is covered by the training dataset.

at the same interval. Gravity is the standard 9.81 m/s^2. The simulation applies no wind disturbance, and the vehicle starts at the reference-trajectory origin with zero velocity. The analysis excludes the first 0.5 s of each run so that suspension transients do not dominate the tracking statistics.

Controller saturation limits. All controllers share the same actuator and solver limits. The controller saturates steering at the servo hardware limit $|\delta| \leq 24°$ (Table 1) and clips longitudinal acceleration to $a_x \in [-3.0,\ 2.5]$ m/s^2. For $v_x < 0.5$ m/s, the Newton/LM solver falls back to Stanley so the inverse does not operate at the zero-velocity singularity.

Success metrics. RMS cross-track error (RMS CTE) is the root-mean-square signed lateral error over the evaluated portion of a run, reported in metres. RMS CTE measures path-tracking accuracy. A controller that reduces speed to track a tighter path will show a lower CTE without necessarily being more capable; the speed-tracking record must be read alongside the CTE.

Heading RMSE is the root-mean-square heading error $|\Delta\psi|$ over the same evaluated interval, reported in degrees when used in tables. For inverse controllers, the feasibility rate is the fraction of control steps at which the inverse solve returns a command that passes all three checks: the residual threshold, the iteration budget, and the actuator limits. Because geometric controllers do not run an inverse solve, the benchmark records their feasibility rate as N/A, not zero.

Divergence criterion. The benchmark flags a run as *diverged* when the lateral error exceeds $|e| > 2.0$ m at any point, or when the MuJoCo contact solver reports a body–

boundary penetration event. Diverged runs are tabulated separately from the non-diverged RMS CTE summaries; a finite RMS value from a failed trajectory would otherwise overstate the controller's tracking accuracy. When a table reports a non-diverged mean, the divergence count must be read alongside it.

8.2 Data Collection for Model Training and Controller Benchmarking

Simulation results vs. hardware results. All quantitative results in Sections 7 and 8 come from **MuJoCo simulation**: the residual RMSE table (Table 10), controller RMS CTE bar chart (Figure 29), trajectory overlays (Figure 30), and CTE time-series plots (Figure 31). No hardware CTE measurements are reported numerically. Appendix A describes the physical validation pipeline, including sensor architecture, state estimation, LiDAR-based odometry, OptiTrack calibration, telemetry, and the full 3D MuJoCo model provenance. A full hardware CTE comparison across the controller set requires a long straight-and-curved track with continuous OptiTrack coverage; this infrastructure was not available for the present study.

The study collects MuJoCo [20] simulation data across a structured grid of operating conditions to train the data-driven residual correctors and establish a physics ground truth for controller comparison. The residual-identification dataset covers four trajectory types: an ellipse (steady-state cornering), a circle (constant curvature), a figure-8 (lateral-acceleration sign reversal), and a slalom (rapid steering transients). For each trajectory type, the data-collection loop logs data at four reference speeds (4.0, 5.5, 7.0, and 8.5 m/s) using Stanley, the *Body-Acceleration Inverse*, the *Curvature Feedback Inverse*, and Pure Pursuit. This yields 64 MuJoCo runs and approximately 320,000 samples. The expanded controller-sweep runner supports a larger robustness grid: eight synthetic trajectories, eight integer speeds, and five deterministic initial-condition perturbations per trajectory-speed pair. Those fields are logged separately from the residual-identification dataset to keep the provenance clear.

Each sample contains the MuJoCo body-frame state (v_x, v_y, r, δ), its finite-difference derivative, and the corresponding enhanced bicycle model prediction. The difference forms the residual target for each subsystem corrector. Each fitted corrector is evaluated on the same dataset; Section 7 reports the resulting RMSE metrics.

The same MuJoCo dataset is also used for the simulation controller comparison. The numerical CTE results below come from MuJoCo simulation, not the hardware LiDAR pipeline. That pipeline is documented as the intended physical-validation path for future work. In future hardware runs, wall-boundary curves extracted by DBSCAN/OPTICS/BIRCH from consecutive scans can provide a local reference path; OptiTrack or AprilTag pose would then supply the ground-truth alignment needed to measure cross-track error.

8.3 Representative Closed-Loop Simulation Comparisons

Figures 29–31 present grouped RMS CTE and per-controller CTE time histories for representative conditions; the aggregate model-fidelity and robustness summaries follow in Section 8.5.

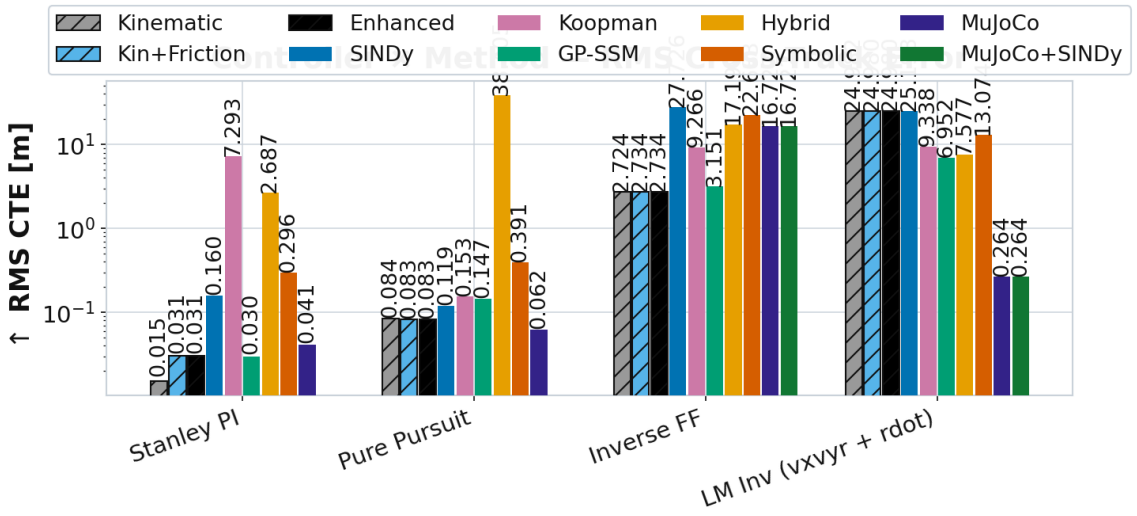
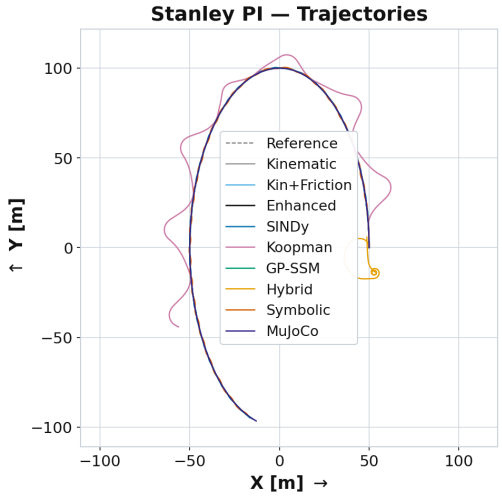
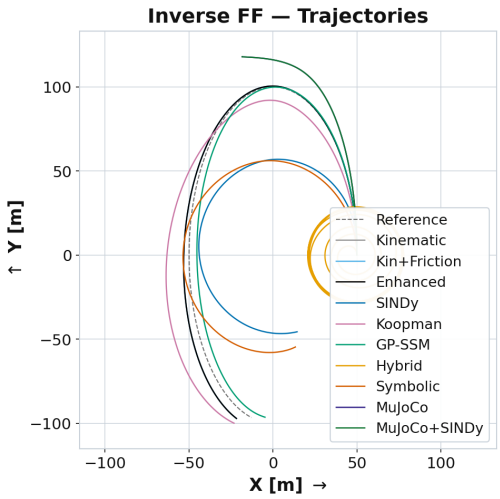


Figure 29: Grouped RMS cross-track error for all controllers across physics levels and residual-correction methods. Physics baselines are shown with a hatch pattern. The main comparison is model-fidelity dependence: the feedback inverse can achieve lower non-diverged CTE with the enhanced model, but divergence count must be read alongside the bars. Learned residual corrections are judged by closed-loop stability rather than one-step fit alone.

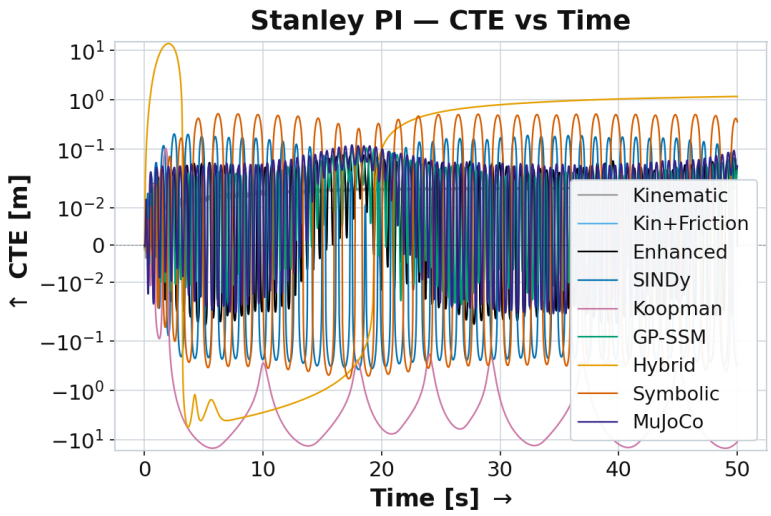


(a) Stanley controller XY trajectories.

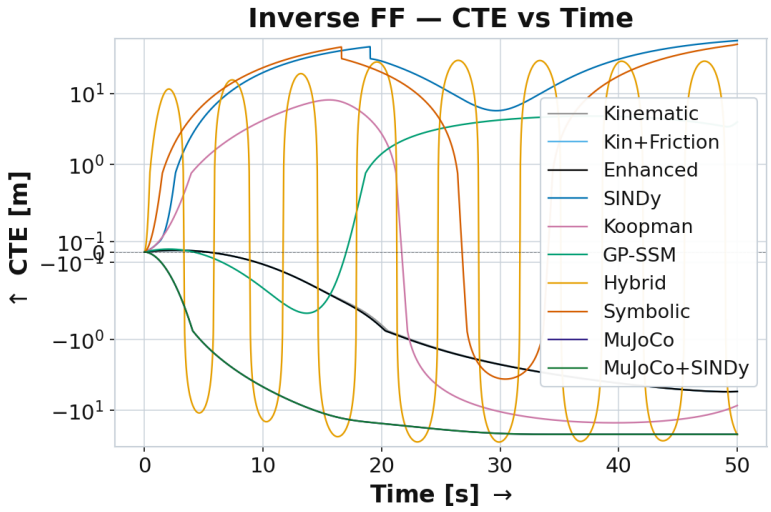


(b) *Body-Acceleration Inverse* controller XY trajectories.

Figure 30: XY trajectory overlays for Stanley (left) and the *Body-Acceleration Inverse* (right) under all physics/data-driven levels. Each curve is one method; MuJoCo ground truth is shown as a dotted line.



(a) Stanley controller CTE vs time.



(b) *Body-Acceleration Inverse* CTE vs time.

Figure 31: Cross-track error over time for Stanley (top) and the *Body-Acceleration Inverse* (bottom). The plot illustrates the architectural difference between geometric feedback and feedforward inversion: without an outer error-feedback term, an inverse controller can drift when model mismatch changes the effective force response. The aggregate model-fidelity and divergence results are summarized separately in Table 12.

Table 11 adds a rebuilt single-condition quality check for the same ellipse trajectory at 7 m/s. This table is not the canonical aggregate result. The inverse-family controllers can fail badly under a single condition even when their aggregate non-diverged means look competitive; this case exposes that failure mode.

Table 11: Single-run controller quality metrics at the Enhanced model level (MuJoCo simulation, ellipse trajectory at 7 m/s, $\mu = 1.0$). RMS speed error is computed from logged body-frame longitudinal speed relative to the prescribed 7 m/s reference. Feasibility rate is the inverse-controller feasibility flag logged by the sweep; geometric controllers have no inversion step (N/A). This diagnostic rebuild includes the *Full-State LM Inverse* row and uses the same divergence criterion as the model-fidelity sweep. Script and CSV provenance are listed in Appendix D.

Controller	RMS CTE (m)	RMS Δv (m/s)	Diverged	Feasib. rate	LM iter
Stanley	0.802	6.625	No	N/A	N/A
Pure Pursuit	0.819	6.625	No	N/A	N/A
Body-Acceleration Inverse	8.833	6.495	Yes	0.008	4.0
Full-State LM Inverse	8.831	6.495	Yes	0.008	4.0
Curvature Feedback Inverse	8.833	6.495	Yes	0.008	4.0

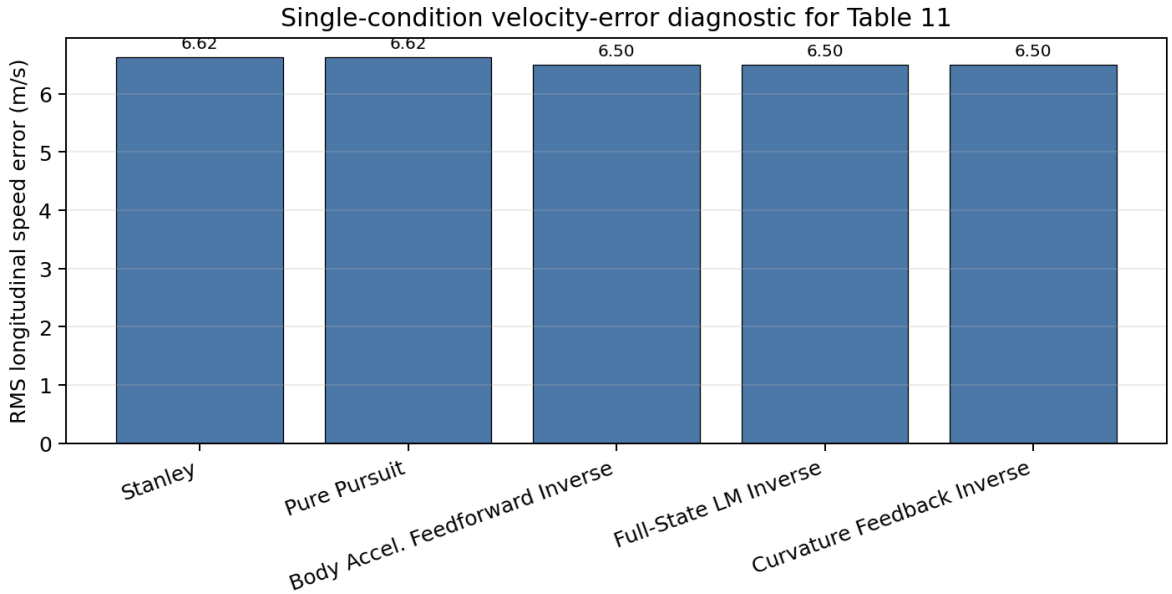


Figure 32: RMS longitudinal speed error for the single-condition diagnostic in Table 11. The large errors show that this diagnostic condition is not a clean speed-tracking success case; it is retained to expose failure modes rather than to support an aggregate controller ranking.

8.4 Drift-Aware Trajectory Feasibility Results

This chapter treats drift-aware planning as a trajectory-feasibility layer; it does not assert that drifting is broadly superior to conventional path following. The planner asks whether the requested path, speed, and acceleration demand lie inside the available friction budget. When the demand is infeasible, the reference trajectory must be reshaped before the Newton inverse can compute valid inputs.

The simulation results justify a limited, condition-specific role for this planner. In selected near-limit corner cases, allowing sideslip or flick-style motion yields faster achievable references that remain recoverable within the tested bounds; the conditions and recoverability evidence are documented in Appendix B. Those references are useful to the thesis only when they produce an achievable demand for the inverse controller. Supporting artifacts (diagnostic plots, recoverability notes, and open blockers) appear in Appendix B.

8.5 Model-Fidelity Dependence of Inverse Control

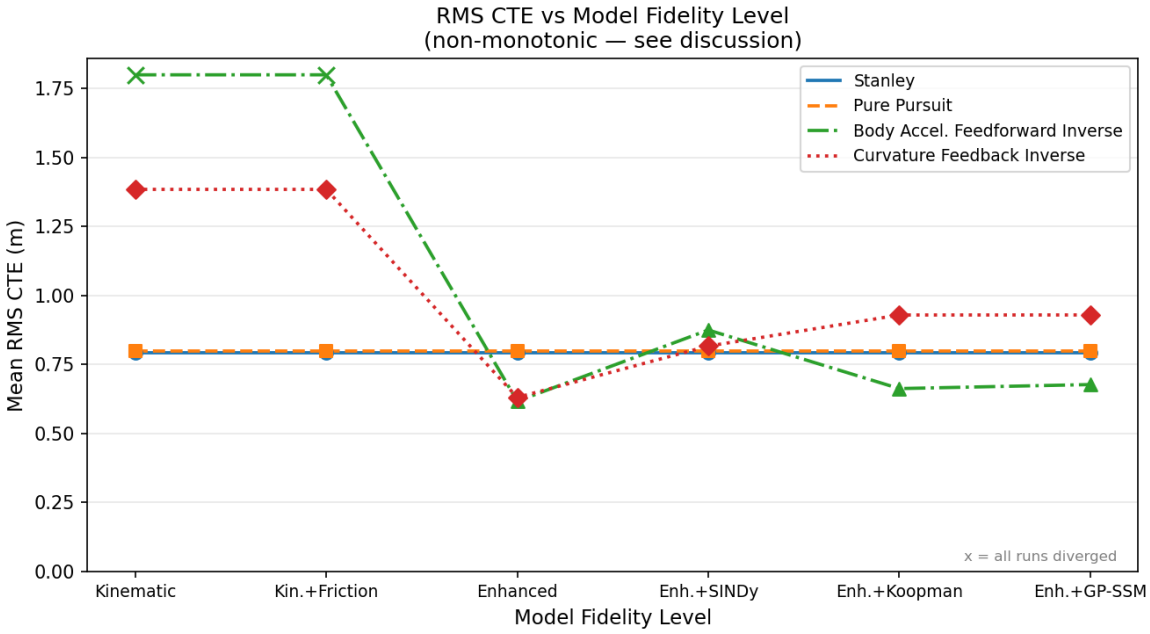


Figure 33: RMS cross-track error as a function of model fidelity level for each controller, averaged over 16 conditions (4 trajectories × 4 speeds, $\mu = 1.0$). An × marker indicates all runs at that level diverged (mean CTE undefined). Script and CSV provenance are listed in Appendix D.

The model-fidelity sweep (Figure 33, Table 12) aggregates 16 MuJoCo runs per cell (4 trajectories × 4 speeds, $\mu = 1.0$). Geometric controllers are unaffected by model level: Stanley holds 0.793 m mean RMS CTE with 2 divergences out of 16 across all six fidelity levels. Diverging on all kinematic runs, the *Body-Acceleration Inverse* recovers to 0.617 m non-diverged mean RMS CTE at the Enhanced level, with 9/16 divergences. At the Enhanced level, the *Feedback Inverse* improves to 0.630 m non-diverged mean RMS CTE (from 1.385 m at kinematic), though its divergence count worsens from 6/16 to 9/16. These results confirm that inversion quality is model-fidelity-limited. They do not establish that the inverse controller is uniformly more stable than geometric feedback.

None of the three learned residual corrections (SINDy, Koopman/Hankel-DMD, or GP-SSM) improves closed-loop stability relative to the physics-only Enhanced model; all three increase both divergence count and mean CTE (Table 12). Among learned corrections,

the best for the *Body-Acceleration Inverse* is Koopman (0.662 m, 14/16 div.), yet it still performs worse than Enhanced alone (0.617 m, 9/16 div.). This open-loop/closed-loop gap indicates that residual corrections fit on replay data extrapolate unfavourably into the tight-curvature and high-speed regimes; these are the same conditions that trigger divergence in the Newton–LM loop. Section 7 discusses this issue further.

Table 12: Mean RMS CTE (m) and divergence count across model fidelity levels (MuJoCo simulation, 16 runs per cell: 4 trajectories × 4 speeds, $\mu = 1.0$). Cell format: *mean RMS (div./16)*. N/A indicates all runs diverged, so no finite mean is reported. Kinematic and Kinematic+Friction are identical because the friction-correction path does not affect the bicycle model used for controller prediction. Script and CSV provenance are listed in Appendix D.

Controller	Kinematic (=Kin.+Fric.)	Enhanced	Enh.+SINDy	Enh.+Koop.	Enh.+GP
Stanley	0.793 m (2/16)	0.793 m (2/16)	0.793 m (2/16)	0.793 m (2/16)	0.793 m (2/16)
Pure Pursuit	0.799 m (2/16)	0.799 m (2/16)	0.799 m (2/16)	0.799 m (2/16)	0.799 m (2/16)
Body-Acceleration Inverse	N/A (16/16)	0.617 m (9/16)	0.875 m (14/16)	0.662 m (14/16)	0.677 m (14/16)
Curvature Feedback Inverse	1.385 m (6/16)	0.630 m (9/16)	0.816 m (14/16)	0.929 m (12/16)	0.929 m (13/16)

8.6 Speed-Tracking Fidelity and the Validity of CTE Comparisons

When controllers regulate speed differently, cross-track error alone gives an incomplete picture of performance. A controller that implicitly reduces cornering speed operates in a kinematically easier regime, and any CTE advantage it achieves is partly attributable to the lower friction demand rather than superior path-following authority. A separate matched speed-tracking benchmark quantifies this effect; its results should not be pooled with the 16-run model-fidelity sweep unless the benchmark difference is stated.

In the friction-plant benchmark (Table 13), Stanley's RMS speed error grows monotonically with reference speed: 0.09 m/s at 1.0 m/s (9%), 0.47 m/s at 4.0 m/s (12%), 1.23 m/s at 7.0 m/s (17%), and 2.23 m/s at 10.0 m/s (22% below reference). MPC and MPPI hold effectively zero speed error at all four conditions. The *Feedback Inverse* also holds speed error near 0.15 m/s across the higher-speed cases, while the *Body-Acceleration Inverse* exposes a separate low-speed singularity by diverging at 1 m/s on both reference trajectories. Stanley's speed deficit follows from the tire-force allocation: Stanley commands a geometrically correct steering angle without consulting the friction ellipse; at high cornering speed the lateral tire force demand approaches $\mu m v^2/R$, partially saturating the tire; the residual tire capacity for longitudinal thrust is reduced; the PI speed controller cannot overcome this drag; and the vehicle settles at a lower equilibrium speed at which the kinematic bicycle-model assumptions become approximately valid again. The result is that Stanley's low CTE numbers reflect an operating point that Stanley itself selected, one where its own modelling assumptions hold, not the commanded reference condition. The two operating points are not equivalent for comparing tracking authority.

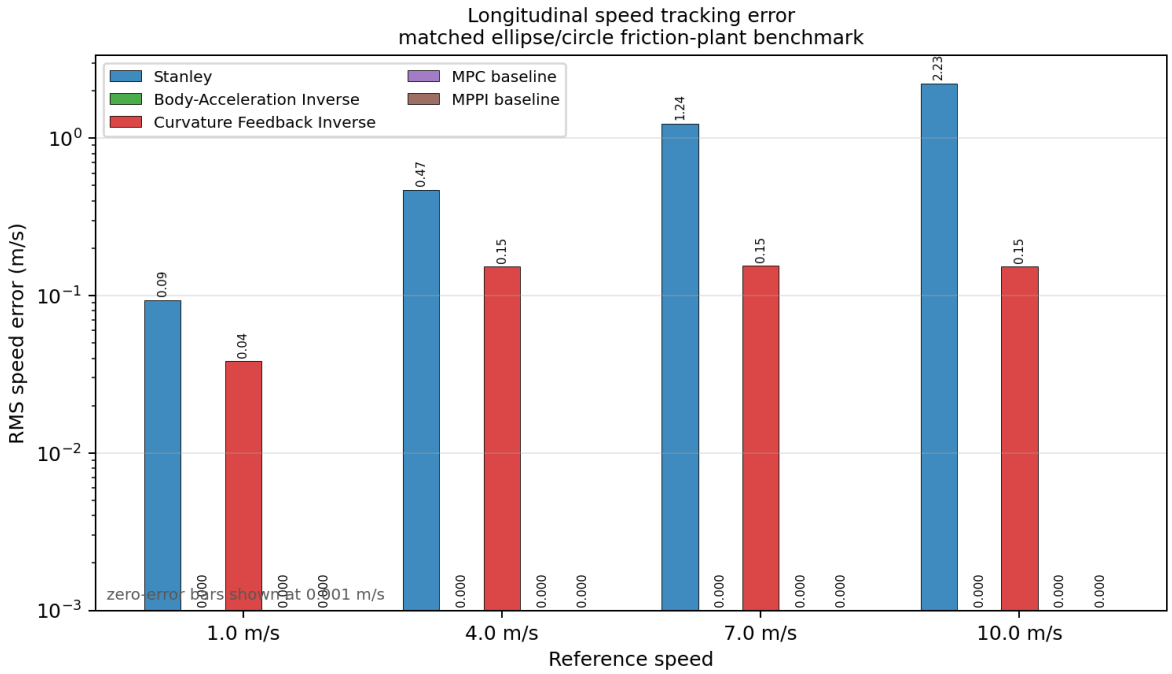


Figure 34: RMS longitudinal speed error for Stanley, the inverse-family controllers, MPC, and MPPI at 1.0, 4.0, 7.0, and 10.0 m/s reference speeds, averaged over ellipse and circle trajectories. Numeric labels above the bars report RMS speed error in m/s. The y-axis is logarithmic; zero-error MPC/MPPI bars are plotted at a small floor for visibility. This is a separate matched speed-tracking benchmark, not part of the 16-run model-fidelity sweep. Script and CSV provenance are listed in Appendix D.

Unlike Stanley, the inverse-family controllers target the commanded speed and lateral acceleration simultaneously through the joint $(v_x, \dot{v}_y)$ or (v_x, v_y, r) inversion. This places the full commanded friction demand on the tire simultaneously, producing a harder control problem at the same nominal reference. The correct reading of the CTE table is that Stanley *reframes* the tracking problem by self-regulating to a lower-speed regime. Its apparent tracking advantage is partially attributable to operating at a self-selected condition rather than the commanded one. The intended role of the inverse controller is to track an achievable commanded-speed trajectory in the tire-saturation regime, precisely the regime where geometric controllers cannot maintain reference speed and path simultaneously without an explicit force-feasibility check.

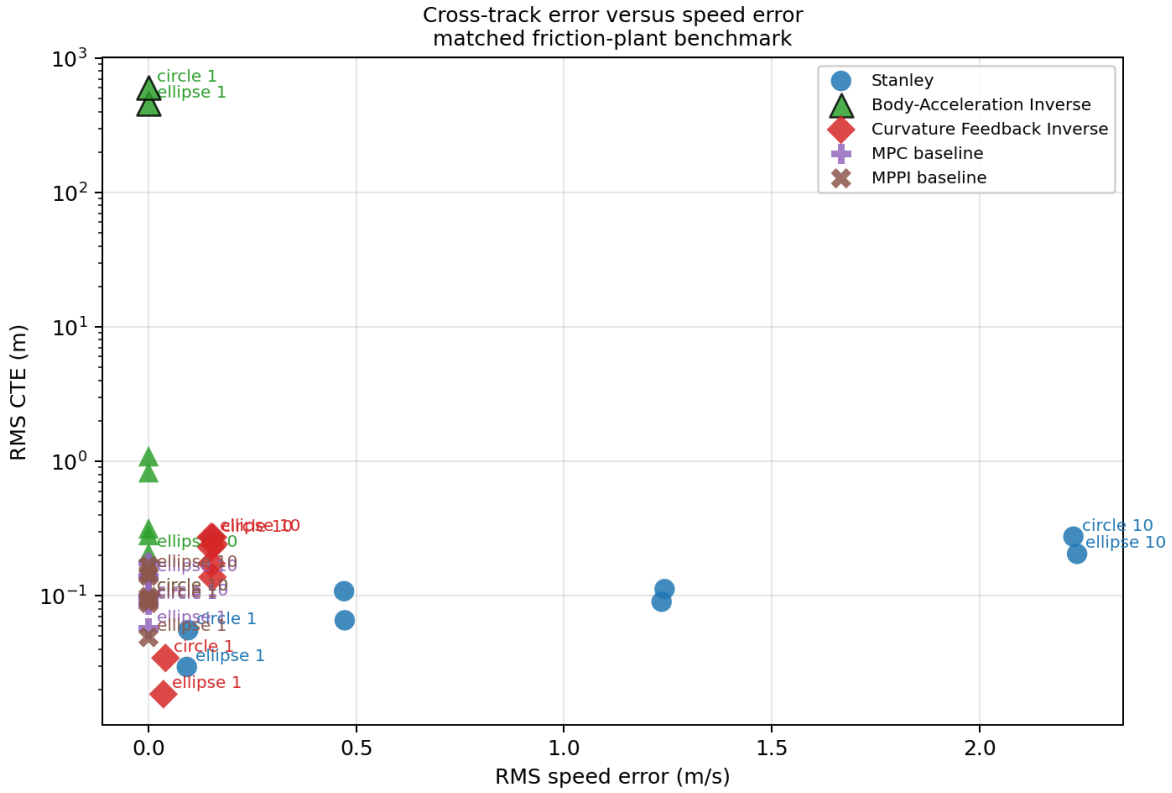


Figure 35: RMS CTE versus RMS speed error per benchmark run (one point per controller × trajectory × speed condition; four speeds 1.0–10.0 m/s). Divergent low-speed *Body-Acceleration Inverse* runs are retained so the plot shows both tracking quality and failure cases.

Table 13: Four-speed matched friction-plant benchmark: RMS cross-track error (m) for all controllers on ellipse and circle trajectories. Regime labels refer to Pacejka tire operating point at $R = 100$ m ($a_{\text{lat}} = v^2/R$): stick ≤ 0.5 m/s^2, transitional ≈ 0.5 m/s^2, slip ≥ 1.0 m/s^2. MPC uses $H = 10$ with SLSQP and warm-start; MPPI uses $H = 25$, $K = 500$, $\lambda = 5.0$. Diverged rows are marked "div." Script and CSV provenance are listed in Appendix D.

		RMS CTE (m)			
Controller	**Traj**	**1.0 m/s (stick)**	**4.0 m/s (stick)**	**7.0 m/s (trans.)**	**10.0 m/s (slip)**
Stanley	ellipse	0.030	0.108	0.090	0.205
	circle	0.055	0.066	0.112	0.275
Body-Acceleration Inverse	ellipse	div.	0.829	0.316	0.207
	circle	div.	1.091	0.283	0.099
Curvature Feedback Inverse	ellipse	0.019	0.173	0.242	0.274
	circle	0.034	0.137	0.229	0.272
MPC	ellipse	0.057	0.175	0.140	0.137
	circle	0.086	0.098	0.098	0.091
MPPI	ellipse	0.049	0.167	0.140	0.147
	circle	0.088	0.093	0.093	0.098
Stanley RMS Δv (m/s)		0.09	0.47	1.23	2.23
Curvature Feedback Inverse RMS Δv (m/s)		0.04	0.15	0.15	0.15
MPC / MPPI RMS Δv (m/s)		0.00	0.00	0.00	0.00

8.7 Friction Coefficient Sensitivity

Experimental scope note. The friction-sensitivity sweep varies the assumed friction coefficient μ passed to the Enhanced bicycle-model prediction (the controller's internal model) while holding the MuJoCo plant at its nominal friction ($\mu_{\text{plant}} = 1.0$). The sweep is a *mu-sensitivity* test, which quantifies how controller performance changes when the operator's friction estimate is conservative relative to the plant value. The present study defers the full surface-friction degradation test (modifying MuJoCo geom friction directly) to future work (Section 8.9).

Table 14 reports divergence counts and mean RMS CTE as the assumed μ drops from 1.0 to 0.5, Enhanced model level, 16 runs per cell.

Table 14: Divergence count and mean RMS CTE (non-diverged runs) as a function of assumed friction coefficient μ in the Enhanced controller model (MuJoCo plant held at $\mu_{\text{plant}} = 1.0$). 16 runs per cell (4 trajectories $\times$ 4 speeds). Script and CSV provenance are listed in Appendix D.

Controller	$\mu = 1.0$	$\mu = 0.8$	$\mu = 0.6$	$\mu = 0.5$
Stanley	0.793 m (2/16)	0.793 m (2/16)	0.793 m (2/16)	0.793 m (2/16)
Pure Pursuit	0.799 m (2/16)	0.799 m (2/16)	0.799 m (2/16)	0.799 m (2/16)
Body-Acceleration Inverse	0.617 m (9/16)	all div. (16/16)	0.874 m (13/16)	1.295 m (15/16)
Curvature Feedback Inverse	0.630 m (9/16)	0.581 m (15/16)	0.736 m (14/16)	1.295 m (15/16)

Geometric controllers are completely unaffected by the assumed μ because they do not query the vehicle physics model. Their 2/16 baseline divergences reflect the hardest trajectory–speed combinations (figure-8 and slalom at 8.5 m/s), not friction sensitivity. Inverse controllers are highly sensitive. Both the *Body-Acceleration Inverse* and the *Curvature Feedback Inverse* diverge significantly more at $\mu < 1.0$ because the Newton–LM feasibility region shrinks as the friction estimate decreases, making it harder to find a valid steering command within the physical saturation limit. The assumed friction coefficient must be close to the plant value; an overly conservative estimate shrinks the Newton–LM feasibility region and can induce divergence even when the actual surface provides adequate grip.

8.8 Robustness to Model Parameter Error

A systematic one-variable-at-a-time parameter mismatch sweep is left as future work. That sweep would vary cornering stiffness C_α, mass, inertia, and wheelbase in the controller's model while holding the MuJoCo plant fixed. The friction coefficient sensitivity results (Section 8.7) indicate that tire-related parameters (C_α and μ) are the most critical axes of model uncertainty for the Newton–LM loop, because they set the feasibility boundary against which each candidate steering command is clipped.

Main thesis claim. The central claim of this work is that iterative inverse control provides a fast, force-consistent input-generation method for aggressive path tracking, provided the trajectory is physically achievable under the tire-force constraints and the model used inside

the inverse is close enough to the plant. The supporting evidence is organized around five points:

1. **Force feasibility gap.** Classical geometric controllers ignore tire force feasibility: they command curvature without checking whether the required lateral force falls within the friction ellipse. Geometric controllers also avoid part of this failure mode by self-regulating to lower cornering speeds where kinematic assumptions hold (Section 8.6): Stanley's RMS speed error grows from 0.09 m/s at 1 m/s to 2.23 m/s at 10 m/s (Table 13), operating at effective cornering speeds progressively below reference. This speed self-regulation reduces lateral friction demand and partially explains the competitive CTE of the geometric baseline in the stick-stick regime. The inversion controller targets the commanded speed and path simultaneously, placing greater demand on the friction budget and making the comparison operationally harder for the inversion approach.

2. **Model dependency.** Inverse methods require model fidelity. An iterative Newton–LM inversion is only as good as the mapping from steering input to lateral force that the vehicle model provides.

3. **Model-fidelity dependence.** Across the 16-run model-fidelity benchmark (Table 12), non-diverged mean RMS CTE for the *Feedback Inverse* falls from 1.385 m (kinematic, 6/16 div.) to 0.630 m (Enhanced, 9/16 div.), confirming model-fidelity limitation while also showing a higher divergence count. Stanley holds 0.793 m throughout all six levels, unaffected by model choice. Adding learned residual corrections (SINDy, Koopman, GP) increases divergence to 12–14/16 and raises mean CTE, revealing an open-loop/closed-loop accuracy gap: corrections that reduce prediction error in replay nonetheless destabilize the Newton–LM inversion loop at higher speeds and tighter curvatures.

4. **Friction coefficient sensitivity.** The assumed friction coefficient μ is a critical parameter for inverse control. Geometric controllers are completely insensitive to assumed μ (they do not query the physics model), while both inverse controllers show markedly increased divergence when the controller's assumed μ drops below 1.0 (Table 14). This supports accurate friction estimation as a practical requirement for the current Newton–LM inversion loop. A true surface-friction degradation experiment is future work.

5. **Mismatch threshold.** A full one-variable-at-a-time parameter mismatch sweep (cornering stiffness C_α, mass, inertia, wheelbase) is identified as future work. The friction coefficient sensitivity result (point 4 above) suggests that tire parameters are the most critical axis of model uncertainty for the inversion loop.

Taken together, these results support a pipeline view of the thesis. In this pipeline, planning produces an achievable force/acceleration demand and the Newton inverse converts that demand into steering inputs. Feedback then corrects residual tracking error, while model refinement narrows the gap between the inverse model and the plant. The Fiala and

Pacejka tire models (Section 3) show how geometric controllers command curvature without accounting for saturation. At the force level, the Newton-inverse framework (Section 5) explicitly checks feasibility against the friction ellipse. The data-driven residual identification results (Section 7) quantify the remaining model gap, though one-step residual accuracy does not automatically translate into closed-loop improvement.

8.9 Limitations and Scope Boundary

The quantitative controller benchmark in this thesis is simulation-only. No head-to-head comparison has yet been completed on the physical RC car, and the data-driven residual models (SINDy, Koopman, GP-SSM, and hybrid sparse-neural corrections) were trained and evaluated on MuJoCo-generated trajectories rather than hardware data. The effective vehicle model used by the controller was fitted from OptiTrack-grounded experiments, but those identification runs occurred at lower speeds than the 7–8.5 m/s simulation benchmark. Hardware closed-loop evaluation across the full controller set has not been completed and is reserved for future work.

The drift-aware planning results also have a conditional scope. They do not show that drift is universally faster than grip; the evidence contains selected wins, grip wins, ties, infeasible cases, and invalid or timed-out rows. Several drift-allowed OCP wins have small peak sideslip and are best interpreted as faster line-shaping results, not as evidence that high-angle drifting is the active mechanism. Recoverability is plan-specific. Broad perturbation sweeps show bounded recovery for ready case families, but exact replay has not yet been completed for every exported OCP drift win. The four-control handbrake transfer remains unvalidated because the current MuJoCo screen collapses $U_4 = [\delta, \text{throttle}, \text{brake}, \text{handbrake}]$ to a two-control approximation. The tight-radius hairpin test case is one of the corner families used to stress-test planner-to-MuJoCo transfer. Its tuned transfer profile can be made to settle after corner exit, but the resulting times are slow and do not constitute evidence of drift superiority.

9 Conclusion and Future Work

This thesis examined iterative inverse control as a fast input-generation layer for aggressive path tracking. The central result is that inversion is useful only as part of a pipeline. To succeed, the planner must request an achievable force and acceleration demand, the inverse must use a model close enough to the plant, and feedback must correct the remaining tracking error. In the current 16-run model-fidelity benchmark (4 trajectories $\times$ 4 speeds, $\mu = 1.0$), the *Curvature Feedback Inverse* improves from 1.385 m non-diverged mean RMS CTE with the kinematic model (6/16 diverged) to 0.630 m with the Enhanced physics model (9/16 diverged), while Stanley holds a constant 0.793 m across the same model levels. This supports a narrow model-fidelity claim that improving the model inside the inverse improves non-diverged tracking error, but does not by itself solve reliability or divergence.

The data-driven residual results expose a second boundary. Residual models can reduce one-step prediction error in replay, but the current SINDy, Koopman, and GP-SSM corrections increase divergence when inserted directly into the Newton–LM closed-loop controller.

The learned model therefore cannot be judged only by fit quality; it must also be judged by whether it improves the closed-loop inverse. The friction-coefficient sensitivity sweep reinforces the same point: the inverse controller depends on the friction estimate used inside the model, while geometric controllers do not query that estimate directly.

The CTE comparison must also be read alongside speed tracking. In the MuJoCo benchmark, the geometric baseline self-regulates to a lower effective speed in high-curvature corners, reducing its friction demand and making the path-following problem easier than the commanded reference implies. The inverse controller targets the commanded speed and path simultaneously. That is the harder operating condition, but it is also the condition needed for racing-style trajectory tracking. Under this interpretation, the thesis does not claim that inversion is universally better than a geometric controller. It claims that, when the trajectory is feasible and the model is sufficiently accurate, iterative inversion provides the fast force-consistent input map that geometric steering laws lack.

The drift-aware planning evidence extends this claim from tracking a fixed reference to choosing a feasible aggressive reference. Its role is to determine whether the requested motion is inside the achievable friction budget and to reshape the trajectory when it is not. The present results support a conditional statement: drift-allowed and flick-style planning can improve corner time in selected near-limit cases while remaining recoverable in bounded tests. The benefit depends on radius, speed, friction, and corner angle. Many current OCP wins are low-sideslip line-shaping wins rather than high-angle drift maneuvers, and faithful four-control handbrake transfer remains future work until the MuJoCo plant accepts direct throttle, brake, and handbrake commands.

The results also show that controller quality and model quality cannot be separated cleanly in this problem. The Newton/Levenberg–Marquardt inversion method is only as effective as the map from steering and longitudinal input to lateral force, yaw response, and feasibility margin. For that reason, a substantial portion of the thesis focused on improving the modeling layer through enhanced bicycle dynamics and data-driven residual correction. The residual-identification results show that a gray-box workflow is practical: begin with a physically interpretable model, measure where it fails against MuJoCo, and learn only the remaining structured discrepancy. In this formulation, the performance gain traces to better agreement between commanded accelerations and the forces the model predicts the vehicle can realize.

At the same time, the present draft has an important scope boundary. The quantitative controller benchmark reported here is simulation-only. The RC-car platform, telemetry stack, OptiTrack integration, and LiDAR/IMU sensing pipeline establish the hardware infrastructure needed for future closed-loop validation. They do not yet form a complete hardware comparison across all controllers. The hardware contribution of this thesis is therefore infrastructural and methodological: it provides the sensing, logging, synchronization, and system-identification pipeline required to bring the same inversion idea onto the physical vehicle in a controlled way.

Several extensions follow naturally from these results. The most immediate next step is a structured hardware validation campaign in which steering and throttle maps are identified from OptiTrack-grounded experiments, an effective vehicle model is fitted, and the iterative controller is tested against Stanley under matched conditions. The MuJoCo study should also be expanded with explicit robustness sweeps over friction, parameter mismatch, latency, and

sensor noise so that the dependence of inversion performance on model fidelity is quantified directly rather than inferred qualitatively. The current inverse controller should then be extended from locally parameterized curvature references to richer waypoint and racing-line formulations, including explicit braking authority and combined-slip longitudinal and lateral coordination. The drift-aware planner needs clean full-grid OCP sweeps, exact-plan recoverability replay for every counted drift win, planned-trajectory inverse-solver tracking benchmarks, and true four-control MuJoCo handbrake validation. Finally, the data-driven residual layer should be judged not only by one-step prediction error, but by closed-loop improvement per unit model complexity. The practical question is not simply which regressor fits best; it is which one produces the most trustworthy controller.

Overall, this thesis argues for a specific control philosophy for autonomous racing and aggressive path tracking: treat feedforward control as a model-inversion problem at the force level, treat feedback as a *necessary* correction layer rather than an optional refinement, and treat model refinement as the main lever that determines whether iterative inversion succeeds. In the tested benchmark, the pure *Body-Acceleration Inverse* is less stable than Stanley because it lacks cross-track and heading correction; model error therefore integrates into drift rather than being rejected by feedback. The force-consistency advantage of inversion is realized only when inversion serves as the feedforward component of a feedforward plus feedback pair. Feedback provides stability, and inversion provides physical feasibility. Under that view, the central engineering question is how accurate the model and trajectory-feasibility layer must be before iterative inversion becomes more useful than a geometric baseline.

10 Bibliography

References

[1] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1094–1099. [Online]. Available: https://doi.org/10.1109/IVS.2015.7225830

[2] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 812–818. [Online]. Available: https://doi.org/10.1109/IVS.2017.7995816

[3] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016. [Online]. Available: https://doi.org/10.1109/TITS.2015.2498841

[4] J. M. Snider, "Automatic steering methods for autonomous automobile path tracking," *Technical Report CMU-RITR-09-08*, 2009. [Online]. Available: https://publications.ri.cmu.edu/automatic-steering-methods-for-autonomous-automobile-path-tracking/

[5] R. Rajamani, *Vehicle Dynamics and Control*, 2nd ed., ser. Mechanical engineering series. Netherlands: Springer Nature, 2011. [Online]. Available: https://doi.org/10.1007/978-1-4614-1433-9

[6] H. B. Pacejka, *Tire and Vehicle Dynamics*, 3rd ed. Butterworth-Heinemann, 2012. [Online]. Available: https://shop.elsevier.com/books/tire-and-vehicle-dynamics/pacejka/978-0-08-097016-5

[7] H. Dugoff, P. S. Fancher, and L. Segel, "An analysis of tire traction properties and their influence on vehicle dynamic performance," SAE International, SAE Technical Paper 700377, 1970. [Online]. Available: https://doi.org/10.4271/700377

[8] E. Fiala, "Seitenkräfte am rollenden luftreifen," *VDI Zeitschrift*, vol. 96, no. 29, pp. 973–979, 1954. [Online]. Available: https://books.google.com/books/about/Lateral_Forces_at_the_Rolling_Pneumatic.html?id=B-wJngEACAAJ

[9] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rubin, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006. [Online]. Available: https://doi.org/10.1002/rob.20147

[10] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie Mellon University, Robotics Institute, Tech. Rep. CMU-RI-TR-92-01, 1992. [Online]. Available: https://www.ri.cmu.edu/publications/implementation-of-the-pure-pursuit-path-tracking-algorithm/

[11] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing, 2017. [Online]. Available: https://sites.engineering.ucsb.edu/~jbraw/mpc/

[12] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017, pp. 1714–1721. [Online]. Available: https://doi.org/10.1109/ICRA.2017.7989202

[13] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944. [Online]. Available: https://doi.org/10.1090/qam/10666

[14] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963. [Online]. Available: https://doi.org/10.1137/0111030

[15] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of*

the National Academy of Sciences, vol. 113, no. 15, pp. 3932–3937, 2016. [Online]. Available: https://doi.org/10.1073/pnas.1517384113

[16] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data-driven approximation of the Koopman operator using extended dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015. [Online]. Available: https://doi.org/10.1007/s00332-015-9258-5

[17] R. Frigola, Y. Chen, and C. E. Rasmussen, "Variational Gaussian process state-space models," *Advances in Neural Information Processing Systems*, vol. 27, 2014. [Online]. Available: https://papers.neurips.cc/paper/5375-variational-gaussian-process-state-space-models

[18] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, "Discovering symbolic models from deep learning with inductive biases," in *Advances in Neural Information Processing Systems*, vol. 33, 2020. [Online]. Available: https://arxiv.org/abs/2006.11287

[19] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, 2nd ed. Cambridge University Press, 2022. [Online]. Available: https://doi.org/10.1017/9781009098167

[20] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033. [Online]. Available: https://doi.org/10.1109/IROS.2012.6386109

[21] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," *IEEE International Conference on Rehabilitation Robotics (ICORR)*, pp. 1–7, 2011. [Online]. Available: https://doi.org/10.1109/ICORR.2011.5975346

[22] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008. [Online]. Available: https://doi.org/10.1109/TAC.2008.923738

[23] Espressif Systems, "Esp32-s3 about," https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/about.html, 2026, official ESP-IDF documentation for the ESP32-S3 series.

[24] STMicroelectronics, "Lsm6dsox datasheet," https://www.st.com/resource/en/datasheet/lsm6dsox.pdf, 2024, official datasheet for the 6-axis IMU used in the platform.

[25] LDROBOT, "Lidar ld08 datasheet," https://www.ldrobot.com/images/2023/03/02/LDROBOT_LD08_Datasheet_CN_V1.3_aoMhAtqQ.pdf, 2023, official LD08 datasheet from the manufacturer.

[26] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407. [Online]. Available: https://april.eecs.umich.edu/papers/details.php?name=olson2011tags

[27] Espressif Systems, "Led control (ledc) - esp32-s3," https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/api-reference/peripherals/ledc.html, 2026, official ESP-IDF documentation for PWM generation on ESP32-S3.

[28] ——, "Adc - arduino esp32," https://docs.espressif.com/projects/arduino-esp32/en/latest/api/adc.html, 2026, official Arduino-ESP32 documentation for ADC attenuation and resolution settings.

[29] OptiTrack, "Natnet sdk," https://docs.optitrack.com/developer-tools/natnet-sdk, 2026, official documentation for streaming motion-capture data via NatNet.

[30] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 1981, pp. 674–679. [Online]. Available: https://www.ri.cmu.edu/publications/an-iterative-image-registration-technique-with-an-application-to-stereo-vision-ijcai/

[31] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 1994, pp. 593–600. [Online]. Available: https://www.ri.cmu.edu/publications/good-features-to-track/

[32] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991. [Online]. Available: https://doi.org/10.1109/34.88573

[33] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 1996, pp. 226–231. [Online]. Available: https://f.aaai.org/Library/KDD/1996/kdd96-037.php

[34] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1999, pp. 49–60. [Online]. Available: https://doi.org/10.1145/304182.304187

[35] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1996, pp. 103–114. [Online]. Available: https://doi.org/10.1145/235968.233324

[36] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Information Theory*, vol. 28, no. 2, pp. 129–137, 1982. [Online]. Available: https://doi.org/10.1109/TIT.1982.1056489

A Hardware Validation Pipeline

The hardware material in this appendix documents the physical validation path for the iterative inverse controller. It is retained for reproducibility and future experiments, while the quantitative controller comparisons in the main text remain simulation-only.

A.1 Experimental Platform Overview

The hardware platform used for sensing and state estimation is a 1/10-scale RC car instrumented with dual LSM6DSOX+LIS3MDL IMU/magnetometer breakouts (front and rear), an LD08 360° single-plane LiDAR at 10 Hz, an ESP32 microcontroller running Madgwick and Mahony AHRS filters for onboard yaw fusion [21, 22], and a Windows laptop as the logging host [23, 24]. An OptiTrack MoCap system provides ground-truth vehicle pose through NatNet telemetry (130 sessions logged, 33 with meaningful vehicle motion). The controller architecture in firmware version v19 supports three modes: pure teleoperation, OptiTrack-guided waypoint following, and inversion-based autonomous steering. The calibration workflow fits an effective bicycle model from OptiTrack-grounded experiments using `fit_effective_bicycle_model.py`; this model is the source of `effective_vehicle_model.json` used by all simulation experiments.

The experimental vehicle is a custom Ackermann-steered racecar equipped with multiple sensors for real-time state estimation, mapping, and controller evaluation. It integrates:

1. dual MARGs for body-frame inertial sensing,
2. a LiDAR for range-based motion cues and environmental structure,
3. one or more cameras for AprilTag-based global localization,
4. an OptiTrack motion-capture interface for ground-truth validation, and
5. an ESP32 microcontroller for low-latency telemetry transmission. All data streams are fused on a Raspberry Pi 5 running a Python-based state estimator and control pipeline.

A.1.1 Onboard Sensors and Electronics

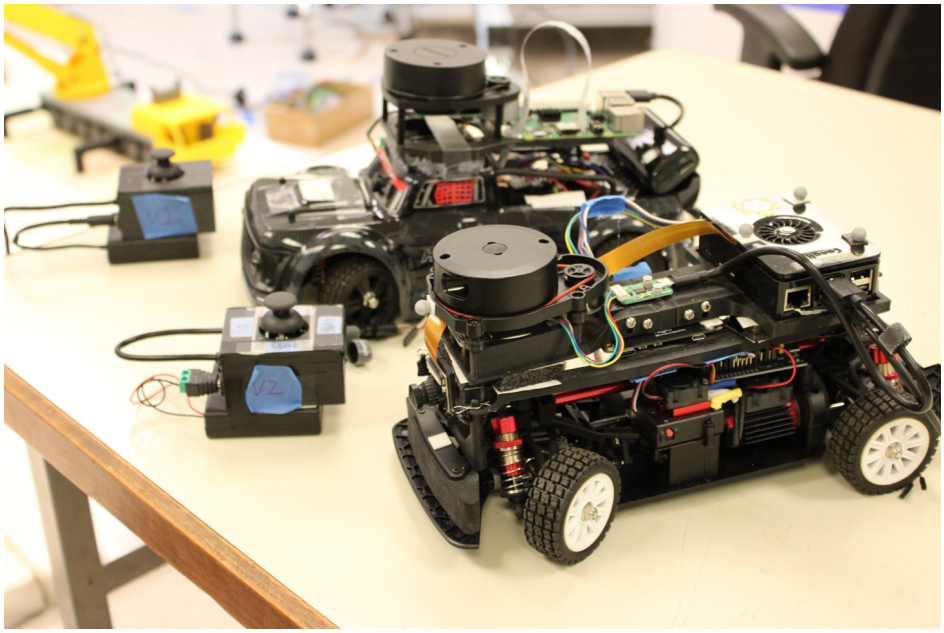


Figure 36: 1/10-scale RC car modified for Wi-Fi operation

- **Dual IMUs (front and rear):** LSM6DSOX modules mounted at two longitudinally separated locations to measure angular rate, linear acceleration, and magnetic heading. The spatial separation enables yaw-rate consistency checks and vibration rejection.
- **RPLiDAR A1M8 or LD08:** 360° scanning LiDAR providing between 2,000 – 8,000 points per revolution, used for odometry, motion cues, and local map reconstruction [25].
- **PiCamera v2:** Used for AprilTag detection via the `pupil_apriltags` library, providing 6DoF tag poses for global localization in the field [26].
- **ESP32 Telemetry Board:** Streams IMU data, remote-control inputs (steering, throttle), and timing packets over UDP with a fixed 21-float packed binary protocol. Supports fallback serial forwarding and passes throttle and steering as PWM inputs to the motors.
- **Raspberry Pi 5:** Serves as the central compute unit, running multi-threaded Python processes for sensor ingestion, filtering, logging, visualization, and control.
- **OptiTrack Motion Capture:** A NatNet 3.1 client running on the Pi consumes rigid-body poses for ground-truth comparison and calibration. The same telemetry is received by the controlling laptop for logging and diagnostics.

A.2 Data Acquisition Pipeline

The hardware dataflow uses separate acquisition, parsing, and synchronization stages. Sensor streams are processed asynchronously, time-stamped, and synchronized between the Raspberry Pi and the laptop before being compared with OptiTrack ground truth.

A.2.1 ESP32 Telemetry Stream

The ESP32 firmware runs on a dual-core Xtensa LX7 processor (240 MHz) [23]. The two ESP32 nodes, a hand-held *transmitter* and an onboard *receiver*, communicate over **Wi-Fi UDP sockets** (802.11b/g/n, 2.4 GHz); the proprietary ESP-NOW layer is *not* used. The receiver runs as a Wi-Fi Access Point, and the transmitter associates to it as a station, as described in Section A.2.2. The receiver firmware drives the ESC and servo on GPIO 32 and GPIO 33 using the LEDC peripheral at 50 Hz with 16-bit resolution [27], mapping the 1000–2000 μs pulse-width range via the formula duty $= \lfloor \tau_{\mu s}/20{,}000 \times 65535 \rfloor$. For the ESC, a pulse of 1500 μs corresponds to neutral/brake; values above 1500 μs command forward throttle, and values below the neutral band engage reverse after a mandatory brake pulse.

The transmitter reads two 12-bit ADC channels (0–4095 counts, 3.3 V reference) from the RC joystick and encodes them as a comma-separated ASCII pair transmitted at approximately 10 Hz during manual operation. In the closed-loop telemetry configuration the packet is extended to a fixed-length binary structure:

$$\text{Packet} = \{\underbrace{a_x,\, a_y,\, a_z,\, g_x,\, g_y,\, g_z}_{\text{IMU front}},\ \underbrace{a'_x,\, a'_y,\, a'_z,\, g'_x,\, g'_y,\, g'_z}_{\text{IMU rear}},\ \underbrace{\theta_{\text{steer}},\, \tau_{\text{throttle}}}_{\text{RC}},\ \underbrace{\Delta t,\, \ldots}_{\text{timing+misc}}\}$$

consisting of 21 single-precision floats (84 bytes) plus a CRC16 checksum. The Pi decodes packets using a dedicated UDP listener thread, providing:

- Raw accelerometer ($\pm 16\, g$) and gyroscope ($\pm 2000\,^\circ$/s) readings from both LSM6DSOX IMUs sampled at 104 Hz internally,
- RC command inputs (steering and throttle as normalised $[-1, +1]$ floats),
- A precise integration timestep Δt for dead-reckoning,
- Optional magnetometer readings from the LIS3MDL (if fitted).

The ESP32 sends at 100–150 Hz during closed-loop operation, and the Pi control loop runs at the same rate, consuming the latest packet at each step. Serial debugging is available at 115200 baud for both transmitter and receiver nodes.

A.2.2 End-to-End Command and Telemetry Loop

The full communication architecture separates a *command path* (laptop $\rightarrow$ car actuators) from a *telemetry path* (car sensors $\rightarrow$ laptop). Both run concurrently. Figure 37 summarizes the data flow.

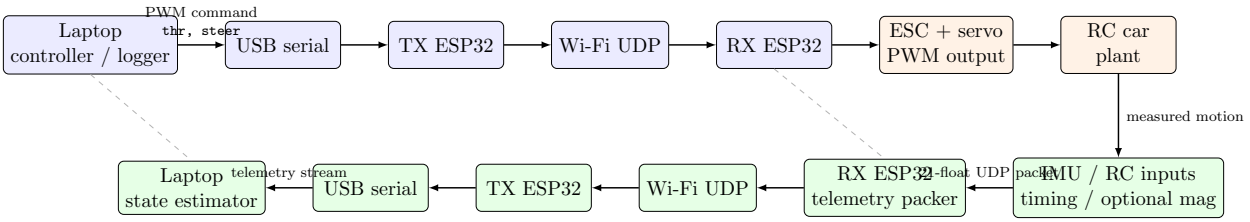


Figure 37: End-to-end command and telemetry loop for the RC car platform.

Command path. The host laptop generates throttle and steering commands either from a human operator or autonomously through the `SerialCarDriver` class. Commands are expressed as pulse-width microsecond values on the standard RC range of 1000–2000 μs, with 1500 μs serving as neutral for both channels:

$$\text{command} = \underbrace{\tau_{\mu s}}_{\text{throttle}}\ ,\ \underbrace{\delta_{\mu s}}_{\text{steering}}\ , \quad \tau, \delta \in [1000, 2000]\ \mu\text{s}.$$

These are encoded as a comma-separated ASCII string and written over a USB serial link at 115 200 baud to the **transmitter** ESP32:

```
"<throttle_us>,<steering_us>\n"
```

The transmitter and receiver communicate over a **Wi-Fi UDP socket**, not the Bluetooth-style ESP-NOW protocol. The **receiver** ESP32 runs as a **Wi-Fi Access Point** (AP) with SSID `"ESP32_AP"`, assigned the fixed IP `192.168.4.1`, and listens for UDP datagrams on port 12345. The **transmitter** ESP32 connects as a Wi-Fi Station (STA) to this AP and targets `192.168.4.1:12345` for every outgoing packet. An event-driven auto-reconnect loop (`pumpWiFiReconnect()`) retries the association every 2 s after any disconnection and re-initialises the UDP socket on the next successful IP lease, so a brief RF dropout does not require a power cycle.

The transmitter has two operating modes selected by a debounced push-button (GPIO 33):

- **Joystick mode** (default-off, button-selected): reads two 12-bit ADC channels (GPIO 34 and GPIO 35, range 0–4095 counts, 11 dB attenuation for 0–3.3 V full scale) from the physical RC joystick potentiometers [23, 28]. A software deadband of ± 20 μs around the calibrated neutral point (ADC count 1850) maps each axis to PWM microseconds using the piecewise linear function mapVoltage($\cdot$), then clamps to $[1000, 2000]$ μs.

- **Serial mode** (default-on): reads newline-terminated commands from the USB-UART at 115 200 baud in the format `"thr_us,str_us\n"`. Values are parsed as floats, rounded to integers, and clamped to $[1000, 2000]$ μs. The last valid command is re-sent every loop iteration ($\sim$100 Hz) so the car does not coast to a stop on a slow host.

In both modes the outgoing UDP payload is a comma-separated ASCII string of the two microsecond values:

$$\text{UDP payload} = \texttt{"}\ \tau_{\mu s}\texttt{,}\ \delta_{\mu s}\texttt{"} \qquad \tau_{\mu s},\ \delta_{\mu s} \in [1000, 2000].$$

Packets are sent at approximately **100 Hz** (10 ms loop delay).

On receiving a UDP datagram the **receiver** firmware parses the payload into throttle and steering integer values. The values are then converted to 16-bit LEDC duty-cycle counts via a helper that maps microseconds to the 65535-count register for a 20 ms PWM period [27]:

$$\text{duty} = \left\lfloor \frac{\tau_{\mu s}}{20{,}000\ \mu\text{s}} \times 65535 \right\rfloor .$$

LEDC channels 0 and 1 drive GPIO 32 (ESC) and GPIO 33 (servo) respectively at 50 Hz [27]. A two-stage reverse arming sequence prevents the ESC from ignoring a reverse command: when $\tau_{\mu s}$ falls below the neutral band the firmware first emits a brake pulse for 300 ms, then issues the reverse PWM. Forward throttle and neutral are written directly without delay.

Telemetry path. While the car is running, the onboard **Raspberry Pi 5** executes three concurrent Python processes. Each feeds a streaming SSH subprocess to the host laptop:

1. **IMU / actuator telemetry** (`test_serial.py` on the Pi, forwarded via SSH to the host). The receiver ESP32 echoes every received UDP packet to its USB-UART at 115 200 baud, appending the computed PWM values. The line format matches the 24-column `raw_stream_log` layout (Table 1): interleaved front and rear IMU readings (accelerometer, gyroscope, magnetometer), the onboard Madgwick + Mahony fused yaw [21, 22], received throttle and steering, and the integration timestep Δt. The host's `parse_esp32_line()` function de-interleaves these fields into `accel1`/`accel2`, `gyro1`/`gyro2`, `mag1`/`mag2`, `avg_yaw`, `throttle`, `steering`, and `dt` columns, then enqueues each row for flushing to `raw_stream_log_<TS>.csv`.

2. **LiDAR scan stream** (`ld08_parser.py /dev/ttyLIDAR` on the Pi, forwarded via SSH). The LD08 LiDAR connects to the Pi via `/dev/ttyLIDAR` at its native baud rate. The parser assembles raw UART frames into (*angle_deg*, *distance_mm*) tuples and prints one tuple per beam. On the host, `parse_lidar_line()` extracts tuples using a fixed regex, converts to Cartesian coordinates $(x, y) = (d\cos\theta, d\sin\theta)$ m, and appends to `lidar_stream_log_<TS>.csv`. This stream runs at roughly 10 Hz (one full 360° revolution per frame) and provides the primary positioning input to all eight odometry algorithms [25].

3. **Camera / AprilTag stream** (`RTAT_corner.py` on the Pi, forwarded via SSH). The PiCamera captures frames at the Pi's native rate. Each frame is processed by the `pupil_apriltags` detector, which returns the tag ID, 3×3 rotation matrix, translation vector (x, y, z), and four corner pixel coordinates. Detections are prefixed with `[TAG]` and printed to stdout; the host's `parse_apriltag_line()` recovers the full 6-DoF pose from the rotation matrix and logs to `apriltag_log_<TS>.csv`. AprilTag poses serve as drift-free position resets for the inertial state estimator.

All three SSH streams run as daemon threads and populate in-memory `deque` buffers that the PyQtGraph UI reads at ∼30 Hz. A dedicated logger thread drains three `queue.Queue` objects to disk asynchronously so that disk I/O never blocks the parsing threads. An `atexit` handler guarantees all queued rows are flushed on exit.

Timing and synchronisation. Each log row is timestamped at the host using `time.time_ns()` at parse time. LiDAR rows also retain the original Unix wall-clock from `time.time()` at the moment the SSH line arrives. Frame boundaries in the LiDAR log are recovered offline from timestamp gaps > 50 ms. All downstream odometry and benchmark scripts match sessions across streams by the shared `YYYY-MM-DD_HH-MM-SS` suffix appended to every log filename.

A.2.3 RPLiDAR / LD08 LiDAR Stream

A Python LiDAR driver (a port of the C++ LDPKG/LiPkg parser) performs:

1. byte-level packet assembly,
2. angle and distance extraction,
3. flipping, angle unwrapping, and time ordering,
4. per-beam filtering (invalid ranges, angle jumps),
5. transformation into Cartesian coordinates.

Each completed scan also contains a timestamp allowing frame-to-frame motion estimation or ICP-based scan matching.

AprilTag Pose Estimation Camera images are processed with the `pupil_apriltags` library, which implements the AprilTag fiducial pipeline described by Olson [26]. For each detected tag:

$$(\mathbf{r}, \mathbf{t}) = \text{solvePnP}(\text{tag corners}, K, D)$$

yielding a 6DoF camera-to-tag pose. When tags are placed in known global frames, this provides:

$$T_{\text{world}\leftarrow\text{car}} = T_{\text{world}\leftarrow\text{tag}}\, T_{\text{tag}\leftarrow\text{cam}}\, T_{\text{cam}\leftarrow\text{car}}.$$

AprilTag detections serve as drift-free position resets for the inertial estimator.

A.2.4 OptiTrack Mocap

A NatNet Python client streams ground-truth pose data [29]:

$$(\mathbf{p}_{\text{mocap}}, \psi_{\text{mocap}}, \theta_{\text{mocap}}, \phi_{\text{mocap}})$$

at $\sim 120\,\text{Hz}$. This is used for:

- validating state-estimator performance,
- calibrating LiDAR and IMU extrinsics,
- benchmarking inverse-model controllers.

A.3 Filtering and State Estimation

A.3.1 IMU Filtering

Front and rear IMUs are processed independently:

- Madgwick or Mahony filters estimate quaternion orientation [21, 22].
- Bias-compensated gyroscope integration refines yaw-rate consistency.
- Accelerometer low-pass filtering rejects vibration from the motors.

Yaw estimates from both IMUs are fused via:

$$\psi = w_f \psi_f + w_r \psi_r,$$

where w_f, w_r are confidence weights determined by vibration metrics and variance of gyroscope readings.

A.3.2 Global State Estimator

The estimator maintains the state:

$$\mathbf{x} = \begin{bmatrix} x & y & \psi & v_x & v_y & b_g \end{bmatrix}^T.$$

The prediction step uses:

$$\dot{\psi} = \omega_z - b_g, \quad \mathbf{v}_{t+1} = \mathbf{v}_t + R(\psi)\, \mathbf{a}_{\text{imu}}\, \Delta t,$$

with gravity removal and optional suspension filtering.

Measurement updates incorporate:

- AprilTag pose (full 6DoF) [26],
- LiDAR-derived scan-matching displacement,
- Mocap pose when available.

A complementary or EKF-like structure blends inertial and exteroceptive cues.

A.4 LiDAR-Based Odometry

LiDAR odometry operates in two modes and follows the standard feature-tracking and scan-alignment ideas used in classical image and point-cloud odometry [30–32]:

- **Real-time mode:** Frame-to-frame motion from scan centroids, ICP seeds, or point-density shifts.
- **Offline mode:** Dense reconstruction via KLT optical flow on LiDAR range images.

The range image is created by mapping each measured beam:

$$(x_i, y_i) = (d_i \cos\theta_i,\ d_i \sin\theta_i)$$

into a normalized grid. Lucas–Kanade optical flow produces motion vectors approximating:

$$\Delta\mathbf{p} \approx \text{median over features.}$$

A.5 LiDAR Scan Clustering

Several clustering and feature-based grouping strategies were evaluated to extract stable geometric structures from consecutive 2D LiDAR scans. The goal is to obtain coherent point groups corresponding to track boundaries, walls, or large obstacles. These groups can then be approximated by smooth curves and used for local map building and odometry.

A.5.1 Shi–Tomasi Feature-based Grouping

In the first variant, image-space features derived from LiDAR scans drive the clustering. Each 2D LiDAR frame is rasterized into a fixed-resolution depth image, and salient point features are detected using the Shi–Tomasi "Good Features to Track" criterion [31]. Let $I(\mathbf{x})$ denote the intensity (here, normalized depth) at pixel location $\mathbf{x} = [x, y]^\mathsf{T}$. The Shi–Tomasi score at $\mathbf{x}$ is defined as

$$R(\mathbf{x}) = \min(\lambda_1, \lambda_2), \tag{98}$$

where λ_1 and λ_2 are the eigenvalues of the second-moment matrix

$$M(\mathbf{x}) = \sum_{\mathbf{u} \in \mathcal{N}(\mathbf{x})} w(\mathbf{u}) \begin{bmatrix} I_x(\mathbf{u})^2 & I_x(\mathbf{u}) I_y(\mathbf{u}) \\ I_x(\mathbf{u}) I_y(\mathbf{u}) & I_y(\mathbf{u})^2 \end{bmatrix}, \tag{99}$$

with I_x, I_y the spatial image gradients and $w(\cdot)$ a local weighting kernel. Points with $R(\mathbf{x})$ above a threshold are retained as salient features.

These features are then tracked across consecutive frames using a pyramidal Lucas–Kanade optical flow method [30], producing short feature tracks over time. For LiDAR scans, the tracked points are re-projected into the original (x, y) scan coordinates. Spatially nearby feature tracks are grouped into clusters, for example through connected components in feature space. Each cluster is then fitted with a robust polynomial curve using the PCA-aligned least-squares model described in Section A.5. This "Shi–Tomasi version" favors regions where the local depth map has strong corner or edge structure, which often coincides with curb lines or wall boundaries in the environment.

A.5.2 Density- and Hierarchy-based Clustering of Raw Points

In the second family of methods, clustering is performed directly on the raw (x, y) LiDAR points without rasterization. The evaluation includes several standard clustering algorithms:

DBSCAN Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [33] groups points into clusters when they lie within an ε-neighborhood of sufficiently many neighbors. Formally, a point is a core point if its neighborhood $\mathcal{N}_\varepsilon(\mathbf{p})$ contains at least `min_samples` points. Clusters are grown by connecting core points and their density-reachable neighbors, while points not assigned to any cluster are treated as noise. In the implementation, DBSCAN is applied to each scan to obtain labels $\ell_i \in \{-1, 0, 1, \dots\}$, and only clusters with at least 80 points are retained for curve fitting. The tunable parameter search explored the range $\varepsilon \in [0.04, 0.18]$ m and `min_samples` $\in [5, 20]$; the nominal operating point was $\varepsilon = 0.12$ m and `min_samples` $= 10$.

OPTICS Ordering Points To Identify the Clustering Structure (OPTICS) [34] extends DBSCAN by computing an augmented ordering of the data points together with reachability distances. This captures cluster structure over a range of density parameters. In this work, the `sklearn` OPTICS implementation with the ξ-based cluster extraction is used, with `min_samples` $\in [3, 40]$ and $\xi \in [0.01, 0.20]$ as the tuning bounds (nominal: `min_samples` $= 10$, $\xi = 0.05$). The resulting labels are post-processed with the same 80-point minimum-size threshold as for DBSCAN.

BIRCH Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [35] incrementally builds a tree of clustering features (CF-tree) and clusters points using cluster feature summaries, which is efficient for large point sets. We employ BIRCH in its single-pass mode with a threshold radius in $[0.05, 0.50]$ m controlling leaf-node splitting. This tends to produce a moderate number of spatially compact clusters around structural elements such as walls or barriers.

K-means For comparison against density- and hierarchy-based methods, a centroid-based k-means clustering method [36] is also evaluated with $k \in [2, 12]$ clusters. Given a preset number of clusters k, the algorithm iteratively assigns points to the closest centroid and updates each centroid as the mean of its assigned points. Although simple and fast, k-means assumes roughly spherical, equal-variance clusters and does not explicitly model noise. As a result, elongated track boundaries can be fragmented or merged inappropriately.

The full scan pipeline subsamples each LiDAR frame to at most 4000 points before clustering, using random subsampling when the raw count exceeds this limit. It operates on consecutive scan pairs separated by a fixed time gap of 50 ms and computes a composite score weighting coverage, compactness, curve fit quality ($\text{RMSE}_{\text{ref}} = 0.03$ m), inter-frame alignment match, and cluster count. Hyperparameter tuning used a gradient-free surrogate optimizer (Bayesian-style with random-perturbation jitter) with up to 120 LiDAR frames per evaluation run drawn from hardware scan logs.

A.5.3 Scoring and Curve Extraction

For each clustering method, clusters are first filtered by size. Each surviving cluster is then passed through the PCA-based polynomial fitting routine implemented in `curve_from_cluster`. The quality of a cluster curve is measured by a span–RMSE score and, for pairs of consecutive scans, by how well the best curve matches a corresponding cluster in the next frame under an SE(2) alignment [32]. These metrics are combined with coverage, compactness, and silhouette-based cohesion into a scalar score used to compare clustering strategies across long sequences of LiDAR frames.

A.6 Intended Closed-Loop Hardware Control Pipeline

The combined pipeline for racecar control proceeds as:

1. Acquire multi-sensor data (IMUs, LiDAR, RC commands).

2. Fuse IMU orientation and estimate global yaw using Madgwick/Mahony AHRS updates [21, 22].

3. Compute global velocity and position.

4. Incorporate AprilTags or Mocap if available [26].

5. Feed state into the nonlinear inverse controller or MPC.

6. Send commanded steering/throttle to the ESC/servo.

7. Log all data for replay, SLAM evaluation, and controller tuning.

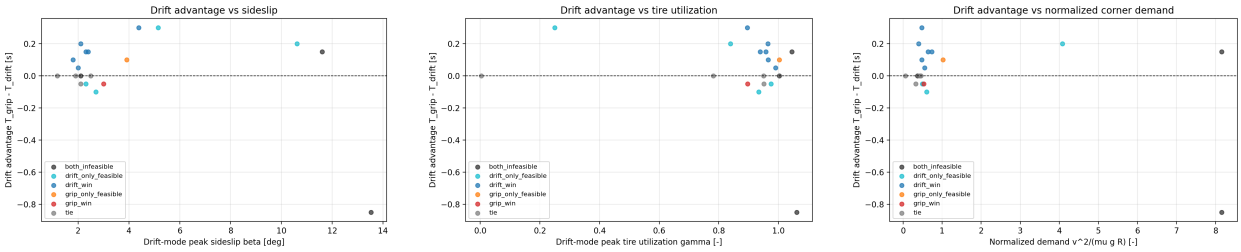
B Drift Planner Evidence Appendix

The drift-aware planner experiments are organized around the constraint question that precedes inversion: when is the requested trajectory physically achievable, and when should the reference be reshaped before the inverse controller is asked to compute inputs? Allowing sideslip or flick-style motion is useful only if it creates an achievable acceleration demand that remains recoverable. The current evidence table is built from the minimum-time OCP summaries, the OCP sweep artifacts, the recoverability perturbation sweep, MuJoCo corner replay, and the four input transient maneuver rollout screens. These artifacts are normalized into `normalized_evidence_table.csv`; Table 15 lists the planner-specific derived tables.

Table 15: Planner-evidence artifacts used to answer the drift/grip research questions. The current row counts reflect the available local artifacts; the table deliberately distinguishes completed evidence from staged or blocked evidence.

Artifact	**Rows**	**Role in the argument**
`planner_regime_map.csv`	64	Grip/drift feasibility, corner time, sideslip, tire utilization, and verdict by scenario.
`planner_mechanism_labels.csv`	64	Separates low-beta line-shaping wins from moderate/high-sideslip candidates.
`planner_recoverability_by_ocp_win.csv`	12	Links drift wins to available perturbation evidence and marks exact-plan recovery gaps.
`mujoco_planner_replay_results.csv`	336	Closed-loop MuJoCo replay evidence for the radius-8 m, 90-degree corner and tight-radius hairpin case families.
`mujoco_4ctrl_replay_results.csv`	6	Approximate collapsed $U_4 \rightarrow U_2$ transient maneuver replay plus the true four-control replay blocker.
`warm_start_ranking.csv`	29	Random/CEM rollout and OCP-mode warm-start comparison rows, with equal-budget rows marked separately.
`inverse_tracking_fidelity.csv`	337	Existing controller replay rows plus an explicit not-run marker for planned-trajectory tracking.

The current OCP-derived regime map is useful as an evidence organizer, but it is not yet a final broad-regime proof. Existing rows include exploratory sweep outputs and targeted OCP summaries; rows known to originate from incomplete or stale sweep artifacts are marked as not final evidence. In the available rows, the verdict distribution contains drift wins, grip wins, ties, one-sided feasible cases, infeasible cases, and invalid/timeout rows. This already rules out a universal "drift is always faster" claim, but a clean full Cartesian sweep is still required before making a broad surface-regime claim about gravel, tarmac, or ice.



(a) Sideslip (b) Tire use (c) Demand

Figure 38: Planner diagnostic plots generated from the current regime-map artifact. Positive values on the vertical axis indicate that the drift-allowed solution is faster than the grip solution. The plots are used to distinguish a faster drift-allowed line from a genuine high-sideslip mechanism.

The mechanism labels show that the current OCP drift wins are primarily low-sideslip line-shaping wins rather than high-angle drift evidence. The defensible interpretation is therefore that the drift-allowed optimizer can find faster admissible cornering lines in selected near-limit cases, not that large-angle drifting is broadly proven to be the mechanism. A controlled-oversteer claim is deferred unless the plan reaches the high-sideslip threshold and recovers under the exact perturbation sweep defined in Section 6.6.

Recoverability is presently stronger for broad case families than for individual OCP wins. The white-box perturbation sweep completed all tested ready cases and recovered sideslip in those cases, but the normalized link table marks most current OCP drift wins as requiring exact-plan perturbation replay before they can be counted as robust drift wins. This distinction is intentional. Recoverability must be tied to the same exported plan that produced the time advantage.

MuJoCo replay provides conditional transfer evidence. The radius-8 m, 90-degree corner case supports a positive drift/flick signal under shared-complete perturbation comparisons. The tight-radius hairpin test case, by contrast, fails in direct MuJoCo transfer because of large exit cross-track error. A tuned transfer profile makes the corner executable, but only conservatively and at slow pace, so it is not counted as a drift performance win. The collapsed $U_4 \rightarrow U_2$ transient maneuver replay is a negative screen; faithful handbrake validation remains blocked until MuJoCo exposes direct throttle, brake, and handbrake channels.

Taken together, the planning evidence currently supports a conditional feasibility claim: drift-allowed and flick-style planning can identify faster achievable references in selected near-limit 90-degree cases and can remain recoverable in bounded tests. The result should be interpreted as support for the inversion pipeline, not as a standalone proof that drifting is broadly superior. A full regime-level claim still requires clean long sweeps, exact-plan recoverability for every counted OCP win, planned-trajectory inverse tracking, and true four-control MuJoCo replay.

C RPLiDAR SDK and LiDAR Odometry Appendix

C.1 RPLiDAR SDK Setup

C.1.1 Prerequisites

Before proceeding, ensure you have the following installed:

- A C++ compiler (GCC, Clang, or MSVC)
- CMake $\geq$ 3.10
- RPLIDAR SDK (download from github.com/Slamtec/rplidar_sdk)
- Git

C.1.2 Project Setup

1. Clone the SDK repository:

```
git clone https://github.com/Slamtec/rplidar_sdk.git
cd rplidar_sdk
```

2. Create a new project directory:

```
mkdir -p examples/new_project
cd examples/new_project
```

3. Add a `CMakeLists.txt` linking the SDK static library.

C.1.3 Minimal Application

```
#include "rplidar.h"
#include <iostream>
using namespace rp::standalone::rplidar;

int main() {
    RPlidarDriver *driver =
        RPlidarDriver::CreateDriver(DRIVER_TYPE_SERIALPORT);
    if (!driver) {
        std::cerr << "Failed to create RPLIDAR driver." << std::endl
           ;
        return -1;
    }
    // application code
    RPlidarDriver::DisposeDriver(driver);
    return 0;
}
```

C.1.4 CMake Build

```
cmake_minimum_required(VERSION 3.10)
project(MyRPLidar)

set(SDK_DIR ../../sdk)
include_directories(${SDK_DIR}/include)
add_library(rplidar_sdk STATIC ${SDK_DIR}/src/rplidar_driver.cpp)
add_executable(${PROJECT_NAME} main.cpp)
target_link_libraries(${PROJECT_NAME} rplidar_sdk)
```

Build and run:

```
mkdir build && cd build && cmake .. && make
./MyRPLidar
```

C.2 LiDAR Odometry Pipeline

This section documents the software pipeline that implements and benchmarks eight 2D LiDAR odometry algorithms against OptiTrack motion-capture ground truth. All scripts reside in `car_data_logs/`.

C.2.1 Pipeline Overview

Script	Role
`cluster_chain_methods.py`	Core primitives: clustering, polynomial fitting, ego-motion
`cluster_chain_temporal.py`	Temporal extensions: Kalman tracker, particle filter
`cluster_chain_fit_lidar_motion.py`	Scan-pair pipeline: produces per-frame SE(2) deltas
`primitive_map_odometry.py`	Algorithm 8: wall-primitive map with corner anchoring
`Curl_script_lidar.py`	Algorithm 7: Circulation-Flux (Green's theorem) odometry
`lidar_mocap_benchmark.py`	Benchmark runner: all algorithms vs. MoCap, 33 sessions
`collate_benchmark_runs.py`	Aggregates per-session outputs into summary CSV/plots

Figure 39: LiDAR odometry pipeline scripts. `lidar_mocap_benchmark.py` imports the primitives from `cluster_chain_methods.py` and dispatches each algorithm over the session log pairs.

C.2.2 `cluster_chain_methods.py` — Core Primitives

This module provides the building blocks shared by all polynomial-fitting algorithms (Algorithms 1–6):

`_pca_local_frame(P)` Returns a 2D rotation R, centroid $\mathbf{t}$, and local coordinates for a cluster point cloud P. The first principal axis is forced into the $+x$ world half-plane to prevent inter-frame sign flips that accumulate into systematic heading bias.

`_irls_huber(X, y, delta, iters)` Iteratively re-weighted least squares with a Huber loss scaled to 3 cm (matching the LD08 inter-point spacing at 2 m range). Used internally by `curve_from_cluster` to fit polynomials robustly to noisy wall returns.

`curve_from_cluster(pts, degree)` Fits a degree-d polynomial in the PCA local frame. Returns a `PolyModel` dataclass with coefficients, inlier ratio, support length, and mean squared residual. Inlier ratio < 0.5 or support < 0.3 m causes the cluster to be rejected.

`ego_delta_from_reports(prev, curr, ...)` Matches polynomial models between consecutive frames (nearest-centroid + orientation guard) and solves the SE(2) rigid-body constraint via weighted least-squares. Returns $(dx_{\text{world}}, dy_{\text{world}}, d\theta)$.

`accumulate_poses(deltas)` Dead-reckons the full trajectory from a list of SE(2) deltas.

C.2.3 `cluster_chain_temporal.py` — Temporal Extensions

Implements three components used by Algorithm 6 (Temporal Cluster Odometry):

`ClusterKalmanFilter` Constant-velocity Kalman filter tracking a cluster centroid in the world frame. Residual velocity from the KF provides a gentle drift-correction signal: under zero-ego-motion the environment should be stationary, so any estimated cluster velocity indicates estimator drift.

`ClusterTracker` Multi-object tracker using the Hungarian algorithm (`scipy.optimize.linear_sum_assignment`) for frame-to-frame cluster assignment. Maintains persistent IDs so that the Kalman filter can accumulate multi-frame evidence. The Hungarian assignment requires stable cluster identities, which is why radius-connected-components clustering (single-linkage, $r = 0.12$ m) is preferred over DBSCAN for this algorithm: DBSCAN's noise-point handling can split or merge clusters at frame boundaries, causing track fragmentation.

`SlidingWindowOdometry` Main pipeline wrapper. Accepts raw per-frame SE(2) deltas from `cluster_chain_methods`, applies the KF velocity correction, and returns a 5-frame median-smoothed pose. The window length was tuned empirically: windows shorter than 3 frames provided no smoothing; windows longer than 7 frames introduced latency visible as a lag in the MoCap ATE time-series.

C.2.4 `lidar_mocap_benchmark.py` — Benchmark Runner

The benchmark script is the authoritative entry point for reproducing the metrics reported in this thesis and in the MECC 2026 papers.

Inputs. For each session the script expects a matched triplet in `car_data_logs/`:

- `lidar_stream_log_<TS>.csv` — per-frame LiDAR scan arrays
- `raw_stream_log_<TS>.csv` — IMU/RC telemetry (throttle filter)

- `logs/Telemetry_<TS>.csv` — OptiTrack NatNet rigid-body pose

Sessions with < 0.5 m of measured MoCap displacement are excluded so stationary captures do not pollute the statistics.

Algorithms dispatched.

1. SVD-ICP (plain)
2. Shi-Tomasi ICP
3. Poly-RANSAC
4. PolyFit-Shi (Poly-Shi)
5. Fused (ICP + polynomial)
6. Temporal Cluster (SlidingWindowOdometry)
7. Circulation-Flux (Stokes / Green's theorem)
8. Curvature ICP

Metrics computed per session per algorithm.

ATE (m) Absolute Trajectory Error: RMSE of the estimated path vs. the Umeyama-aligned MoCap path.

RTE (m) Relative Trajectory Error: mean step-wise displacement error (frame-to-frame).

HDG (°) Heading MAE: mean absolute heading error after zeroing both sequences at frame 0.

Outputs.

- `comparison_results/mocap_benchmark_<TS>/per_session_<TS>.png` — 3-panel figure (XY trajectory, ATE time-series, heading time-series)
- `comparison_results/mocap_benchmark_<TS>/metrics.csv` — full numerical table (one row per algorithm per session)

Canonical benchmark numbers. The final full 9-algorithm results (33 sessions, source: `mocap_benchmark_20260421_200203`) are reproduced in Table 16 for reference.

Table 16: Full 9-algorithm LiDAR odometry benchmark (33 MoCap sessions). Lower ATE/RTE is better; lower HDG is better. *Best per column in bold. (Source: hardware runs, OptiTrack ground truth.)*

Algorithm	ATE (m)	RTE (m)	HDG (°)	Wins
Circ-Flux (Stokes)	**0.508**	0.051	40.6	4
Shi-Tomasi ICP	0.516	0.057	33.6	4
SVD-ICP (plain)	0.525	**0.048**	35.4	6
Poly-Shi	0.539	0.060	32.6	2
Poly-RANSAC	0.544	0.051	41.3	7
Fused	0.636	0.065	40.4	2
Temporal Cluster	0.725	0.052	**25.1**	5
Curvature ICP	0.792	0.092	56.2	0

C.2.5 Clock Synchronisation and Coordinate Convention

NatNet timestamps and the host-PC LiDAR timestamps are not guaranteed to be synchronised. Trajectory comparison therefore uses *frame-index parameterisation*: ATE and HDG are computed after SE(2) Umeyama alignment (handles any fixed rotation + translation offset between the two coordinate frames). In the benchmark script, both streams are first converted to elapsed time from their own first sample; MoCap pose is then interpolated onto the LiDAR frame times wherever the two windows overlap. This removes constant start-time offsets without assuming a shared hardware clock. Heading traces are unwrapped and then zeroed at frame 0 before comparison, so the reported HDG metric captures relative turning drift rather than absolute yaw offset.

MoCap coordinate frame: Z-up right-hand; floor-plane position (x, y); heading $= \arctan 2\big(2(q_w q_z + q_x q_y),\ 1 - 2(q_y^2 + q_z^2)\big)$ (standard aerospace/ROS Z-up yaw). LiDAR odometry pose is accumulated as an SE(2) transform in the floor plane, with heading extracted from the rotation block as $\arctan 2(R_{21}, R_{11})$. Because the LD08 scan angle increases clockwise when viewed from above, sign handling is normalised inside each algorithm wrapper before the benchmark metrics are computed; the appendix should therefore be read as comparing already-normalised headings rather than raw sensor angles.

C.3 MuJoCo Simulation Pipeline

The quantitative controller and residual-identification benchmarks reported in Sections 7 and 8 are generated entirely in MuJoCo simulation. The relevant scripts live in `mojuco_sim/`.

C.3.1 `race_car_env.py` — Environment Wrapper

This file is a lightweight `gym.Env` sandbox used for interactive simulation, not the main closed-loop benchmark harness used for the thesis figures. It loads a MuJoCo XML model,

exposes a 4-channel control vector via `step(action)`, switches wheel friction when the vehicle enters the named "ice" or "oil" zones, and returns a synthetic LiDAR observation along with AprilTag detections and a triangulated pose estimate in the `info` dictionary.

In other words, `race_car_env.py` is best viewed as a perception-rich playground for testing contact changes and onboard sensing hooks. The quantitative controller benchmarks reported in Chapters 7–8 instead use the dedicated sweep / comparison scripts described below, which log body-frame state, solver diagnostics, and MuJoCo ground-truth trajectories directly.

C.3.2 `run_racecar_sim.py` — Hardware Log Replay

Replays a recorded hardware run (`raw_stream_log_<TS>.csv`) inside MuJoCo by applying the same steering/throttle PWM commands to the simulated model and logging the body-frame sensor outputs alongside the real IMU stream. Used for calibrating the simulation parameters and for the sim-vs-real IMU comparison in Section 8.

C.3.3 Data Collection for Controller Benchmark

The 64-run enriched dataset (4 trajectories × 4 speeds × 4 controllers, ≈320,000 samples) is collected by running the environment wrapper in a loop and recording body-frame state, finite-difference derivative, and the enhanced bicycle model prediction at each step. The per-step residual (state derivative minus model prediction) forms the target for the SINDy, Koopman/EDMD, GP-SSM, Hybrid, and Symbolic regression correctors. Fitted correctors are then frozen and used to close the loop for the grouped RMS CTE benchmark (Figure 29).

Trajectory types and speed schedule.

Trajectory	Reference speeds (m/s)	Purpose
Ellipse	4.0, 5.5, 7.0, 8.5	Steady-state cornering
Circle	4.0, 5.5, 7.0, 8.5	Constant curvature
Figure-8	4.0, 5.5, 7.0, 8.5	Lateral-acceleration sign reversal
Slalom	4.0, 5.5, 7.0, 8.5	Rapid steering transients

Controller success criteria. A run is flagged as diverged if the lateral error $|e| > 2.0$ m or the simulation contacts a boundary. Diverged runs are excluded from the CTE statistics and counted separately as failure events.

D Script and Figure Appendix

Table 17 lists the major scripts used to generate the quantitative results, figures, and tables in this thesis. Paths are written relative to the workspace root `CSE 583/` unless a row explicitly begins with `Thesis/`.

Table 17: Script-to-figure mapping for reproducibility. Paths are relative to the `CSE 583/` workspace root unless otherwise noted.

Script	Purpose	Figure/Table generated	Key inputs	Key outputs
`Thesis/mojuco_sim/run_sweep.py`	Multi-controller sweep runner	`tab:model_fidelity`, `tab:controller_quality_metrics`	`--controllers`, `--model-levels`, `--mu`	`Thesis/mojuco_sim/results/*/metrics_summary.csv`
`Thesis/mojuco_sim/plot_model_fidelity.py`	Model-fidelity figure	`fig:rms_cte_vs_model_level`	`metrics_summary.csv`	`Thesis/comparison_results/fig_rms_cte_vs_model_level.{png,pdf}`
`Thesis/mojuco_sim/plot_speed_tracking.py`	Matched speed-tracking figures	`fig:speed_tracking_comparison`, `fig:effective_speed_vs_cte`	`results/mpc_mppi_benchmark.csv`	`Thesis/comparison_results/fig_speed_tracking_comparison.png`
`run_mpc_mppi_benchmark.py`	Matched Stanley/inverse/MPC/MPPI benchmark	`tab:mpc_mppi_bench`	ellipse/circle reference trajectories	`results/mpc_mppi_benchmark.csv`
`model_comparison_four_wheel.py`	Four-wheel normal-load benchmark	`tab:fw_rmse`, `fig:fw_deep_dive`	MuJoCo four-wheel load datasets	`comparison_results/four_wheel/benchmark_v8/summary.csv`
`plot_four_wheel_comparison.py`	Four-wheel RMSE grouped plot	`fig:fw_rmse_bars`	`comparison_results/four_wheel/benchmark_v8/summary.csv`	`comparison_results/four_wheel/benchmark_v8/fig_model_rmse_bars.png`
`suspension_step_response.py`	Suspension calibration figure	`fig:susp_validation`	calibrated spring–damper parameters	`comparison_results/four_wheel/suspension_validation.png`
`comparison_results/divergence_analysis/summary_divergence.csv`	Residual-model error summary	`tab:residual_rmse`	sampled residual predictions	RMSE/MAE/bias table
`Thesis/car_data_logs/lidar_mocap_benchmark.py`	LiDAR odometry benchmark	Table 1 in `mecc_paper`	MoCap + LiDAR logs	`metrics.csv`

Statement on Use of AI Tools

Generative AI tools were used in a limited supporting role during the preparation of this thesis. Their use included editorial revision, LaTeX formatting support, code-debugging assistance, benchmark scripting support, and literature categorization for contextualizing related work. Generated suggestions were treated as draft support. The author reviewed, modified, tested, and validated all code, scripts, analysis outputs, and written material before including them in the thesis.

The research questions, controller design, mathematical derivations, simulation studies, analysis, interpretation of results, novelty claims, and conclusions are the author's own. The author takes full responsibility for the final content of this thesis.