Politecnico di Torino

Imitation Learning for Integrated Chassis Control with Road Preview

Master's Thesis

Candidate:	Cheng Yinfeng
Supervisor:	Prof. Aldo Sorniotti
Co-supervisor:	Ing. Cecilia Formento
	Ing. Davide Lazzarini

Academic Year 2025–2026

Abstract

This thesis studies integrated chassis control (ICC) for a four-wheel-steering and torque-vectoring (4WS+TV) electric vehicle, aiming to improve path tracking and stability while retaining real-time feasibility. Among ICC methods, nonlinear model predictive control (NMPC) is compelling for handling multivariable coupling, constraints, and look-ahead objectives, while the practical benefit depends on model fidelity, preview design, and computation.

We implement two teacher controllers in CasADi–acados:"NMPC-SIMP" and a dynamic-state & tracking-error augmented variant with road preview ("NMPC-DSTE")—each evaluated with/without preview (four controllers in total) and compared against classical lateral baselines (Stanley, PIDF2). The previewed DSTE configuration offers the best trade-off in cross-track/heading errors, yaw-rate behavior, and control smoothness; its preview gain is selected by sweep and it is adopted as the expert policy.

Using the teacher's closed-loop data, we train a compact dual-head residual MLP (steering/torque). The learned policy reproduces the expert's anticipative turn-in, well-damped dynamics, and small sideslip, with predictable real-time inference. The main remaining gap lies in sustained cross-track regulation in long/transition bends. Overall, the study clarifies the benefit of road preview and establishes a practical route to near-expert ICC at low runtime cost.

Keywords— Integrated chassis control; Nonlinear model predictive control; Imitation learning; Road preview; Path tracking.

Acknowledgements

I would like to express my sincere gratitude to **Professor Aldo Sorniotti** for the opportunity to work on this project and for his trust throughout. The topic proved to be of high scientific and practical value, closely aligned with my expectations and interests.

I am deeply grateful to **Cecilia Formento** and **Davide Lazzarini** for their day-to-day supervision and guidance. They provided clear technical direction, timely feedback, and a well-structured, tightly paced plan of tasks, which enabled me to make steady progress and complete the work on schedule.

Finally, I would like to thank my family and friends for their constant encouragement and patience. Their support has been a source of motivation from the beginning to the end of this project.

Contents

List of Figures

List of Tables

1 Introduction

Modern electric vehicles (EV) adopt by-wire chassis with fast, redundant actuators—electric drivetrains, brake-by-wire, and steer-by-wire—which enable torque vectoring (TV) to shape understeering, enhance yaw/sideslip damping, and reduce energy consumption; four-wheel steering (4WS) adds lateral/yaw authority for aggressive or low road surface friction coefficient μ manoeuvres [2]. Within this hardware landscape, integrated chassis control (ICC) coordinates multiple actuators to improve tracking, stability, and efficiency beyond rule-based controllers [34].

We focus on ICC with *four wheel steering (4WS) + torque vectoring (TV)*. Controllers receive a reference path and headings (no high-fidelity environment reconstruction in the loop). Because ICC is multi-objective, constrained, and over-actuated, non-linear model predictive control (NMPC) is a natural fit. Prior studies show that NMPC-based ICC can outperform baselines in yaw-rate tracking, sideslip behavior, and energy-related metrics [34]. Moreover, higher-fidelity prediction models and *road preview* further improve performance—e.g., about a 20% yaw-rate tracking gain for preview vs. non-preview under the same plant [2]. A deployment challenge, however, is the computation time and latency variability of online NMPC. *Imitation learning* (IL) addresses this by training a deep network to approximate the NMPC policy from closed-loop data, yielding comparable behavior with predictable, low-latency inference (e.g., $\sim$0.07–0.08 ms vs. NMPC spikes $>$40 ms on a scaled-car platform) [30].

In this thesis we compare two NMPC formulations—(i) a simplified model (*NMPC-SIMP*) and (ii) a dynamic-tracking-error model with road preview (*NMPC-DSTE*), against two classical baseline controllers: Stanley and PIDF2. After identifying the best NMPC (DSTE), we perform *imitation learning*: a neural controller is trained offline on the expert NMPC closed-loop data and then plugged into the same Matlab/Simulink environment for evaluation [28]. In this way, NMPC, baselines, and the learned network are compared under identical plant and reference-generation assumptions.

2 Related Work

Modern EV with by-wire actuation (e-motors, brake-by-wire, steer-by-wire) provide the redundancy and bandwidth needed for coordinated four-wheel steering (4WS) and torque vectoring (TV) [2]. Comprehensive surveys on integrated chassis control (ICC) classify multi-actuator strategies [40] and highlight the benefits of joint steering–torque control over single-actuator designs [31]. Energy-efficient TV has been demonstrated on multi-motor EVs with reductions in power losses while maintaining handling performance [11]. Further algorithmic developments optimize TV allocation under drivability and actuator limits on experimentally validated plants. [4], [7]. Within ICC, nonlinear model predictive control (NMPC) is appealing as it handles constraints, nonlinear tyre dynamics, and multi-objective trade-offs in a single framework [34]. Comparative studies show that higher-fidelity prediction models inside NMPC improve yaw-rate tracking relative to reduced-order formulations on the same vehicle and scenarios [2]. Connectivity and mapping unlock preview of curvature and future headings so the controller can plan anticipative actions over meaningful horizons [35]. Early preview controllers transmitted or estimated steering profiles but often neglected longitudinal–lateral tyre-force coupling and thus offered limited fidelity for aggressive manoeuvres [48]. Modern guidance therefore integrates preview signals directly into the predictive controller to exploit look-ahead information effectively [2].

2.1 Integrated Chassis Control (ICC) and NMPC Formulations

ICC targets the coordinated use of redundant by-wire actuators to improve tracking, stability, and efficiency beyond single–actuator designs, which naturally motivates an optimal–control formulation. [34] Nonlinear model predictive control (NMPC) is particularly suitable[12] because it embeds actuator/state constraints and multi–objective trade–offs—e.g., path/yaw tracking, sideslip damping, and energy use—within a receding–horizon optimisation. [34] On experimentally validated EV plants with multiple powertrains and steering authority, NMPC has demonstrated superior yaw–rate tracking and sideslip behaviour relative to tuned feedback or rule–based controllers under identical scenarios. [2] Energy–aware cost functions in ICC explicitly penalise sources of power loss—such as tyre slip work, friction–brake dissipation, and electrical conversion—while preserving steady–state and transient cornering response. [34] A representative integrated design solves for coordinated steering and wheel–torque commands on a 4WD EV with in–wheel motors and brake–by–wire, showing systematic improvements in stabilisation and energy saving on ramp and multi–step steer tests. [10] The same study highlights benefits in demanding transients—reduced total lateral load transfer, lower power losses, higher exit speeds—when the optimisation explicitly captures tyre nonlinearity and the coupled longitudinal–lateral dynamics. [10] From an implementation standpoint, advances in real–time iteration and automatic code generation have brought NMPC sampling times into the automotive range on embedded ECUs, enabling on–line solutions without sacrificing horizon length excessively. [19], [25] Under a common plant and reference generator, comparative evaluations further indicate that increasing the fidelity of the NMPC internal model can yield tangible gains in yaw–rate tracking relative to reduced–order variants, reinforcing the value of faithful prediction in ICC. [2] When combined with steering–torque coordination, such higher–fidelity predictive structures help achieve tighter path/yaw regulation with smoother control actions than classical baselines tuned to the same manoeuvres. [2] Overall, contemporary ICC with NMPC balances handling and energy objectives under explicit constraints, benefits from solver/ECU progress for real–time feasibility, and provides a principled baseline against which simplified or learning–based surrogates can be assessed. [34]

2.2 Prediction-Model Fidelity and Road-Preview Benefits

Systematic evaluations on an experimentally validated EV show that increasing the fidelity of the NMPC internal model—from reduced-order to higher-order—yields tangible yaw-rate tracking gains under the same plant and references. [2] In particular, comparative tests between gain-scheduled OFLQR, a 3-DoF NMPC, and an 8-DoF NMPC indicate evident advantages for the higher-order predictive structure when steering and torque actuation are coordinated. [2] Beyond model fidelity, injecting look-ahead information further improves performance, with road-curvature or heading preview delivering about a 20% yaw-rate tracking improvement when added to the higher-fidelity NMPC variant on identical scenarios. [2] The mechanism is anticipative action: preview provides future steering/yaw-rate profiles that allow the optimiser to schedule inputs over meaningful horizons instead of reacting to instantaneous references. [2] Early preview controllers in the literature often model the steering preview as a disturbance or rely on preceding-vehicle steering transmission, which restricts fidelity when longitudinal–lateral tyre-force coupling becomes relevant. [47] Connectivity and map services now enable curvature-based preview along the upcoming path, supporting the generation of realistic future heading and reference yaw-rate profiles for predictive control. [35] When such preview is embedded directly inside NMPC, the controller exploits look-ahead while simultaneously honouring state and actuator constraints, improving tracking with smoother control actions relative to non-preview designs. [2] Sensitivity analyses further suggest that the benefits of preview are most pronounced when paired with a sufficiently rich internal model, reinforcing the joint importance of fidelity and look-ahead in integrated steering–torque control. [2] These insights motivate the use of curvature/heading preview signals in NMPC formulations for 4WS+TV ICC, particularly under fast manoeuvres where anticipative coordination reduces yaw-rate error without sacrificing constraint handling. [35] Overall, the evidence indicates that faithful prediction models plus road preview constitute a complementary pair: higher-order models raise baseline accuracy, and preview shifts the controller from reactive to pre-emptive behaviour on the same validated plant. [2]

2.3 Computation and Learning-Based Surrogates

A practical limitation of on-line NMPC for integrated chassis control is the computation burden and worst-case latency variability at the small control steps typical of vehicle dynamics. [2] Imitation learning (IL) tackles this by distilling the NMPC policy into a neural controller whose inference time is short and predictable on embedded targets. [30] On a scaled-car platform, a DNN trained from NMPC closed-loop rollouts reproduced expert steering/acceleration characteristics while stabilising computation time around 0.07–0.08 ms as NMPC spiked beyond 40 ms at sharp entries. [30]

A closed-loop IL pipeline collects expert trajectories from the NMPC acting in simulation, trains a feedforward network on state or error features to predict control inputs, and then evaluates the student in identical closed-loop tests to quantify stability and tracking gaps. [1] Hierarchical designs combine learnable policies with model-based blocks, where differentiable MPC components and closed-loop state cloning reduce covariate shift and align the learned behaviour with human or expert references under comfort and safety metrics. [1] For safety–critical manoeuvres, an NMPC expert can be imitated by an offline–trained feedforward DNN paired with a lightweight LTV–LQ feedback regulator so that robustness to parameter variations is preserved during deployment. [28] Further hardening is achieved by input dimensionalisation and action limiting at the policy interface so that actuator and state bounds remain respected without on-line re-optimisation. [28]

IL-from-NMPC contrasts with end-to-end learning on human demonstrations, where datasets are

typically open-loop and prone to covariate shift once the learned policy drifts from the demonstrated manifold. [18] To mitigate this, data collection procedures deliberately perturb trajectories and record corrective actions so the network observes recovery states and maintains closed-loop stability after training. [18] Earlier non-deep approaches showed that decision trees and random forests can encode driving heuristics with active re-training triggered on control failures, though closed-loop robustness degrades more easily under distribution shift compared with expert-driven datasets. [8]

In our study, we therefore adopt the closed-loop IL route: we use the best NMPC (DSTE with road preview) as the expert to generate datasets, train a compact DNN offline, and evaluate it in the same MATLAB/SIMULINK plant and reference pipeline as the expert and classical baselines to ensure an apples-to-apples comparison. [30] This strategy leverages the predictability of learned inference while keeping the expert's multi-objective structure through dataset design and interface constraints, which is essential under the tight timing budgets of integrated 4WS+TV control. [1]

3 Simulation Framework

3.1 Overview of the Simulation Framework

Figure 1 summarizes the closed-loop simulation used throughout this thesis. A discrete *Reference Path* (polyline with headings) feeds two front–end blocks:

- **Tracking-error Generator** computes, from the ego pose (X, Y, ψ), the cross–track error e_{ct} and heading error e_h w.r.t. the nearest point and tangent on the path. These two errors are also exposed to classical baselines (Stanley/PIDF2).
- **Controller Data Interface (CDI)** consumes the reference path, the ego state and the error signals (e_{ct}, e_h) to build the controller inputs. Concretely, CDI (i) locates the current abscissa on the path, (ii) constructs the previewed reference sequence, and (iii) packs the *initial state* x_0 and the *online parameter vector* $[p]$ expected by the controllers.

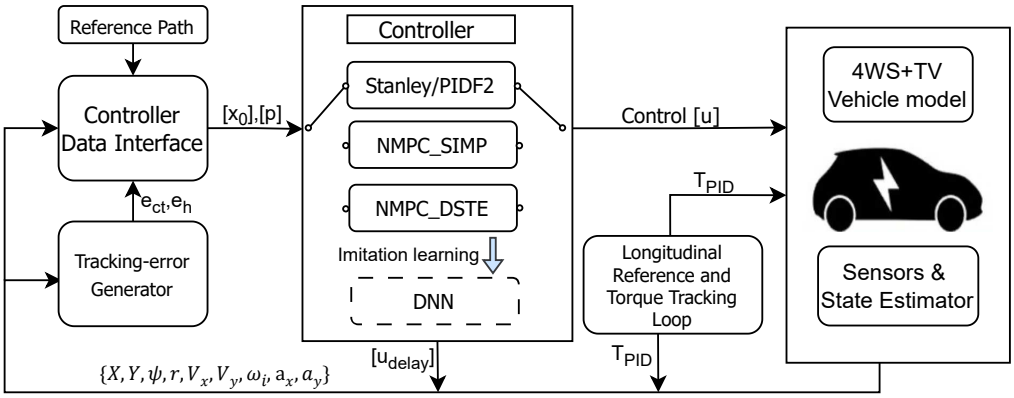

Figure 1: Simulation Framework

The **Controller** block hosts four alternatives that are mutually exclusive at run time: two predictive controllers (`NMPC_SIMP`, `NMPC_DSTE`) and two classical baselines (Stanley/PIDF2). In addition, an *imitation–learning* branch (CNN) can be swapped in to replicate the expert NMPC policy. The selected controller outputs the lateral/yaw command vector, denoted Control $[u]$.

Longitudinal motion is regulated by a dedicated **Longitudinal Reference and Torque Tracking Loop**, which generates the propulsion/braking demand T_{PID}. This demand is applied to the plant and is also routed back on a low–latency channel for logging/coordination.

The **4WS+TV Vehicle Model** represents the plant with four–wheel steering and torque–vectoring actuation. A **Sensors & State Estimator** module provides the measured/estimated signals required by CDI and by the baselines, namely

$$\{X, Y, \psi, r, V_x, V_y, \omega_i, a_x, a_y\},$$

together with any derived quantities used for preview or allocation. A delayed copy of the controller output (annotated $[u_{\mathrm{delay}}]$) is also available on the feedback bus for interface penalties or learning features. Overall, each simulation step proceeds as:

$$\text{Path} \rightarrow (\text{Errors, CDI}) \rightarrow \text{Controller} \rightarrow \text{Plant} \rightarrow \text{Estimator} \rightarrow \text{next step.}$$

3.2 Tracking-error Generator

Let the reference path be given as a polyline with headings $\mathcal{R} = \{(x_k, y_k, \psi_k)\}_{k=1}^{N}$. Given the vehicle position (x, y) at the current step, we project (x, y) onto the nearest segment $\overline{AB} = \big((x_i, y_i), (x_{i+1}, y_{i+1})\big)$ using the clamped parametric abscissa

$$t^\star = \mathrm{clip}\left(\frac{[(x,y) - A] \cdot (B - A)}{\|B - A\|^2}, 0, 1\right), \quad Q = A + t^\star(B - A).$$

The reference point is $(x_{\mathrm{ref}}, y_{\mathrm{ref}}) = Q$, and the reference heading ψ_{ref} is obtained by interpolating the segment headings at $t^\star$.

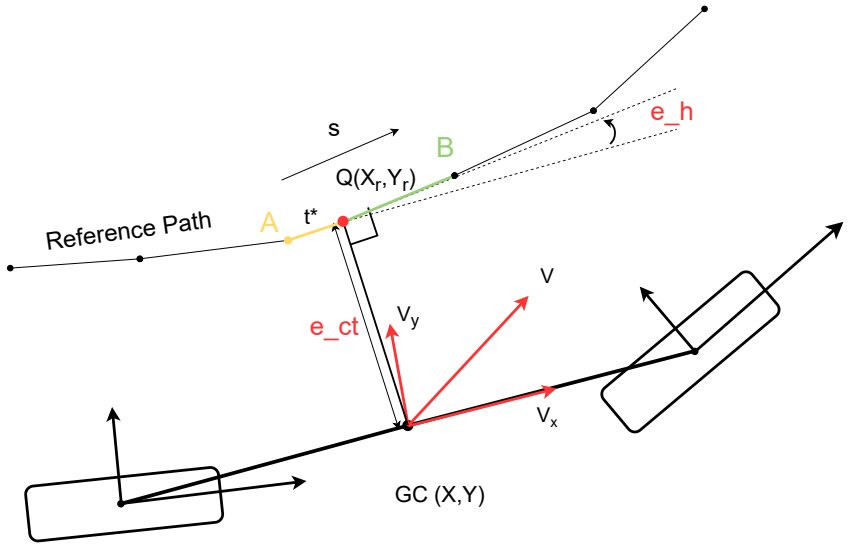

Figure 2: Definition of errors

The cross–track and heading errors are then computed as

$$e_{ct} = (y_{\mathrm{ref}} - y)\cos(\psi_{\mathrm{ref}}) - (x_{\mathrm{ref}} - x)\sin(\psi_{\mathrm{ref}}), \qquad e_h = \psi_{\mathrm{ref}} - \psi.$$

Here e_{ct} is positive when the vehicle lies to the left of the path with respect to the reference heading, and e_h is positive when the vehicle is oriented counterclockwise relative to the reference direction. These sign conventions are consistent with those used in the NMPC implementation and the DSTE error dynamics (see Section 4.3), where $e_\psi = -e_h$ following the orientation definition adopted in the dynamic single-track model.

This module outputs (e_{ct}, e_h) to the classical baselines (Stanley, PIDF2), and to the predictive controllers when error–coordinates are used in the internal state. A small search window around the previous segment index is maintained to ensure linear–time updates and numerical robustness in real time.

3.3 Controller Data Interface (CDI): reference & online-parameter packaging

The CDI computes the arc–length abscissa s at the closest point Q, then forms a vector of future abscissa

$$\mathbf{s} = \big[s,\ s + \Delta s,\ \ldots,\ s + (N_h - 1)\Delta s\big],$$

with $\Delta s \approx V_x\, T_s$ (optionally saturated by a preview distance D_{prev}). A table lookup on the path provides the reference sequences.

$$\left\{X_{\text{ref}}(s_k),\, Y_{\text{ref}}(s_k),\, \psi_{\text{ref}}(s_k)\right\}_{k=0}^{N_h-1},$$

and, when used, the heading–rate preview $\dot{\psi}_{\text{ref}}(s_k)$. These are arranged into the reference vector sequence $\mathbf{S}_y = \{y_{\text{ref},k}\}_{k=0}^{N_h-1}$ expected by the NMPC block.

In parallel, CDI assembles the online parameter sequence $\mathbf{S}_p = \{p_k\}_{k=0}^{N_h-1}$, which collects the quantities used by the controller but computed outside the optimiser: current/previewed curvature or heading profiles, measured a_y (for load–transfer–related terms), actuator/safety bounds, drivability limits, last–step inputs for rate penalties, and any precomputed maps needed by allocation or costs. Packing $(x_0, \mathbf{S}_y, \mathbf{S}_p)$ outside the NMPC is consistent with the formulation used in [2], and allows the same CDI to feed both predictive controllers and the IL policy (by logging $(\text{inputs}, \text{outputs})$ pairs from the expert for training).

4 NMPC Formulations and Baseline Controllers

4.1 Vehicle Dynamic Model

Non-linear dynamic single-track model

The proposed NMPC relies on a non-linear dynamic single-track (DST) model expressed in a curvilinear coordinate frame, balancing modeling simplicity and predictive accuracy under high-handling conditions.[38] The internal model includes the longitudinal velocity V_x, lateral velocity V_y, and yaw rate $\dot{\psi}$, forming a standard 3-DoF planar vehicle representation. The main geometric variables and force directions are illustrated in Fig. 3.

The state vector and control input are defined as

$$\mathbf{x} = [x,\, y,\, \psi,\, V_x,\, V_y,\, \dot{\psi},\, \omega_{FL},\, \omega_{FR},\, \omega_{RL},\, \omega_{RR}]^\top,$$

$$\mathbf{u} = [D_F,\, d_F,\, D_R,\, d_R,\, T_F,\, t_F,\, T_R,\, t_R]^\top,$$

where D_F and D_R are the axle steering angles, d_F and d_R are the left–right steering distribution ratios, T_F and T_R are axle torques, and t_F, t_R are torque distribution ratios. The absolute wheel commands are:

$$\delta_{FL} = D_F, \quad \delta_{FR} = D_F d_F, \quad \delta_{RL} = D_R, \quad \delta_{RR} = D_R d_R,$$
$$T_{FL} = T_F, \quad T_{FR} = T_F t_F, \quad T_{RL} = T_R, \quad T_{RR} = T_R t_R.$$

The vehicle dynamics are expressed as:

$$\begin{aligned}
\dot{V}_x &= \frac{1}{m}\sum_{ij}\big(F_{x,ij}\cos\delta_{ij} - F_{y,ij}\sin\delta_{ij}\big) - C_{rr,4}V_x^2 + V_y\dot{\psi},\\
\dot{V}_y &= \frac{1}{m}\sum_{ij}\big(F_{x,ij}\sin\delta_{ij} + F_{y,ij}\cos\delta_{ij}\big) - V_x\dot{\psi},\\
\ddot{\psi} &= \frac{1}{I_z}\sum_{ij}\Big(a_i(F_{x,ij}\sin\delta_{ij} + F_{y,ij}\cos\delta_{ij}) \pm \tfrac{b_i}{2}(F_{x,ij}\cos\delta_{ij} - F_{y,ij}\sin\delta_{ij})\Big),
\end{aligned} \tag{1}$$

where m and I_z denote the vehicle mass and yaw moment of inertia, and (a_i, b_i) are the geometric distances from the CG to the front/rear axles and to the left/right wheels.

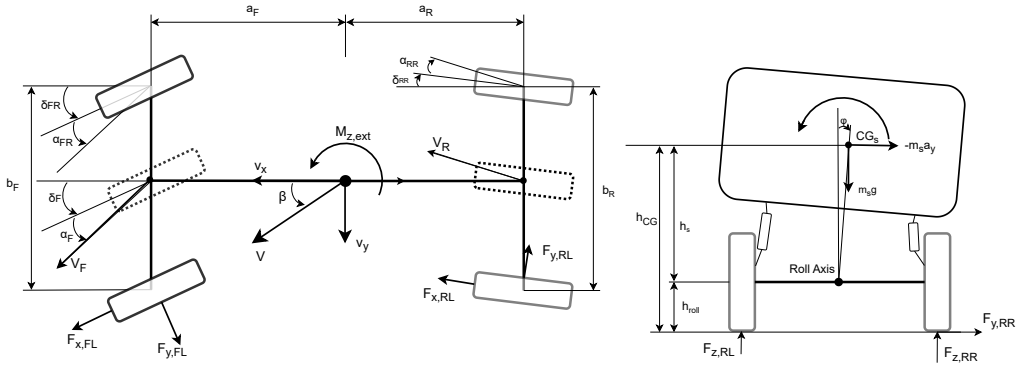

Figure 3: Geometric and force layout of the 4WS and TV model.

Tracking-error kinematics (DSTE)

In Chapter 3, the geometric cross-track and heading errors (e_{ct}, e_h) were introduced as static quantities computed from the instantaneous vehicle pose relative to the reference path. For the dynamic single-track error model (DSTE) used inside the NMPC, these errors are treated as dynamic states with continuous evolution over time.

The reference path is parametrized by the arc length s, with curvature $\kappa_{\text{ref}}(s)$. Define the orientation error e_ψ and the lateral error e_y in the curvilinear coordinate frame as

$$e_\psi = \psi - \psi_{\text{ref}}, \qquad e_y = (y - y_{\text{ref}})\cos\psi_{\text{ref}} - (x - x_{\text{ref}})\sin\psi_{\text{ref}}.$$

Note that some literature defines the orientation error as $e_\psi = \psi_{\text{ref}} - \psi$, corresponding to $e_\psi = -e_h$ under the notation adopted in Chapter 3. Both conventions represent the same physical quantity; here we retain $e_\psi = \psi - \psi_{\text{ref}}$ to match the implemented NMPC model and Simulink definition.

The time evolution of the DSTE error states follows:

$$\dot{e}_\psi = \dot{\psi} - \dot{\psi}_{\text{ref}}, \tag{2}$$

$$\dot{e}_y = V_y \cos(e_\psi) + V_x \sin(e_\psi). \tag{3}$$

The reference yaw rate is obtained from the curvature:

$$\dot{\psi}_{\text{ref}} = \kappa_{\text{ref}}(s)\,\dot{s} \approx \kappa_{\text{ref}}(s)\,V_x,$$

where the approximation holds for small cross-track deviations ($e_y \approx 0$). Equations (2)–(3) thus describe the dynamics of lateral and orientation errors within the NMPC prediction model.

Relation to static errors. The static formulation (e_{ct}, e_h) from Chapter 3 is algebraically evaluated at each sampling step and serves as input to classical controllers (Stanley, PIDF2). The DSTE-based NMPC, instead, integrates the dynamic error states (e_y, e_ψ) according to (2)–(3), which enables consistent prediction of future deviation evolution and facilitates the inclusion of preview information. [13], [14].

Preview signals and tuning

Given the path parameterized by arc length s with curvature $\kappa_{\text{ref}}(s)$, the controller employs a set of static geometric preview signals:

$$\{\psi_{\text{ahead}},\, x_{\text{ahead}},\, y_{\text{ahead}},\, \dot{\psi}_{\text{ref}}\}.$$

They are computed from the future reference point at a look-ahead distance D_{prev} ahead of the vehicle's current abscissa s_k:

$$s_{\text{ahead}} = s_k + D_{\text{prev}}, \qquad [x_{\text{ahead}},\, y_{\text{ahead}}] = [x_{\text{ref}}(s_{\text{ahead}}),\, y_{\text{ref}}(s_{\text{ahead}})],$$

$$\psi_{\text{ahead}} = \arctan\!\big(y'_{\text{ref}}(s_{\text{ahead}}),\, x'_{\text{ref}}(s_{\text{ahead}})\big)$$

The corresponding preview-based tracking errors are then

$$e_{\psi,\text{ahead}} = \psi - \psi_{\text{ahead}},$$

$$e_{y,\text{ahead}} = (y - y_{\text{ahead}})\cos(\psi_{\text{ahead}}) - (x - x_{\text{ahead}})\sin(\psi_{\text{ahead}}),$$

which provide anticipatory information on the upcoming path curvature for the NMPC cost function.

Preview distance definition. The look-ahead distance is defined as

$$D_{\text{prev}} = \alpha\, T\, V_{\text{target}},$$

where T is the NMPC horizon duration, V_{target} the nominal vehicle speed, and α a dimensionless scaling factor. Based on a sensitivity analysis, $\alpha = 0.6$ was found to yield the best compromise between anticipation and numerical stability. For brevity, the detailed sensitivity results are omitted here.

In practice, the preview sequence is discretized along the prediction grid $\{t_k\}$ and passed to the optimizer as online parameters p_k (see Section 4.3).

Implementation notes. This paper uses the preview parameters $\{\psi_{\text{ahead}}, x_{\text{ahead}}, y_{\text{ahead}}\}$ exactly as in the code (`sym_p`), and updates them at each control step.

Modeling choice and extensions

The prediction model excludes roll $(\phi, \dot{\phi})$ for two reasons: (i) roll motion is not directly controlled in this study, and its effect on tire load is captured by quasi-static transfer; (ii) including it would significantly increase model dimension and solver stiffness, reducing real-time feasibility. Roll is only present in the high-fidelity simulation plant (Sec. 4.2) to emulate realistic dynamics and model–plant mismatch.

4.2 Tire and Load Modelling

The tire, load, and roll dynamics follow the vehicle geometry in Fig. 3, which fixes the coordinate conventions (body x forward, y left) and all sign directions.

Tire and load models

Each wheel on axle $i \in \{f, r\}$ (front/rear) and side $j \in \{L, R\}$ is treated independently to capture steering geometry, slip, and load transfer. Let a_f, a_r be CG-to-axle distances ($L = a_f + a_r$), t_f, t_r the track widths ($t_f \equiv b_f$, $t_r \equiv b_r$ in code), and define $s_L = +1$, $s_R = -1$.

Wheel-center kinematics. The velocity of the wheel center (body frame) is

$$\begin{bmatrix} v_{x,\text{pt}} \\ v_{y,\text{pt}} \end{bmatrix} = \begin{bmatrix} V_x \\ V_y \end{bmatrix} + \begin{bmatrix} -r\, y_{ij} \\ r\, x_{ij} \end{bmatrix}, \quad x_{ij} = \begin{cases} a_f, & i = f \\ -a_r, & i = r \end{cases}, \quad y_{ij} = s_j \tfrac{t_i}{2},$$

where $r = \dot{\psi}$. With steering angle δ_{ij} (front: $\delta_{fL} = \delta_{fR} = \delta_f$, rear: $\delta_r = 0$), the components in the

wheel frame are obtained by a rotation of δ_{ij}:

$$v_{x,ij} = \cos\delta_{ij}\,\big(V_x - r\,y_{ij}\big) + \sin\delta_{ij}\,\big(V_y + r\,x_{ij}\big),$$
$$v_{y,ij} = -\sin\delta_{ij}\,\big(V_x - r\,y_{ij}\big) + \cos\delta_{ij}\,\big(V_y + r\,x_{ij}\big).$$

Slip definitions. Longitudinal slip ratio and slip angle are

$$s_{ij} = \frac{R_{ij}\omega_{ij} - v_{x,ij}}{\max\{\,|v_{x,ij}|,\ |R_{ij}\omega_{ij}|,\ \varepsilon_s\,\}}, \qquad \alpha_{ij} = -\arctan 2\big(v_{y,ij},\,|v_{x,ij}|\big),$$

where the minus sign follows the convention that *leftward* lateral tire force is positive.

Vertical load and load transfer

Body-frame accelerations. Rigid-body kinematics in the body frame gives the CG accelerations

$$a_x = \dot V_x - r\,V_y, \qquad a_y = \dot V_y + r\,V_x,$$

used below for quasi-static load transfer.

Static distribution. Static axle loads are

$$F_{zf,0} = mg\,\frac{a_r}{L}, \qquad F_{zr,0} = mg\,\frac{a_f}{L},$$

hence static wheel loads (left–right)

$$F_{z\mathrm{FL},0} = F_{z\mathrm{FR},0} = \tfrac{1}{2}F_{zf,0}, \quad F_{z\mathrm{RL},0} = F_{z\mathrm{RR},0} = \tfrac{1}{2}F_{zr,0}.$$

Longitudinal transfer. With CG height h_{CG},

$$\Delta F_x = \frac{m\,a_x\,h_{\mathrm{CG}}}{L},$$

which *shifts* load front→rear when $a_x > 0$ rear→front when $a_x < 0$. Assuming equal split per axle,

$$\Delta F^{(x)}_{z\mathrm{FL}} = \Delta F^{(x)}_{z\mathrm{FR}} = -\tfrac{1}{2}\Delta F_x, \quad \Delta F^{(x)}_{z\mathrm{RL}} = \Delta F^{(x)}_{z\mathrm{RR}} = +\tfrac{1}{2}\Delta F_x.$$

Lateral transfer with roll. Let h_{roll} be the CG–roll-center height. Suspension roll spring/damper moments on the two axles are $M_f^{\mathrm{spring}} = K_\phi^{(f)}\phi$, $M_r^{\mathrm{spring}} = K_\phi^{(r)}\phi$, $M_f^{\mathrm{damper}} = C_\phi^{(f)}\dot\phi$, $M_r^{\mathrm{damper}} = C_\phi^{(r)}\dot\phi$. The inertial roll moment ma_yh_{roll} is distributed by the roll-stiffness ratio $\lambda_f = \frac{K_\phi^{(f)}}{K_\phi^{(f)}+K_\phi^{(r)}} \in [0,1]$:

$$M_{\phi,f} = (M_f^{\mathrm{spring}} + M_f^{\mathrm{damper}}) + \lambda_f\,ma_yh_{\mathrm{roll}}, \quad M_{\phi,r} = (M_r^{\mathrm{spring}} + M_r^{\mathrm{damper}}) + (1-\lambda_f)\,ma_yh_{\mathrm{roll}}.$$

Axle-level$a_y > 0$

$$\Delta F_{y,f} = \frac{M_{\phi,f}}{t_f}, \qquad \Delta F_{y,r} = \frac{M_{\phi,r}}{t_r}, \qquad \begin{cases} \Delta F^{(y)}_{z\mathrm{FL}} = +\tfrac{1}{2}\Delta F_{y,f},\ \Delta F^{(y)}_{z\mathrm{FR}} = -\tfrac{1}{2}\Delta F_{y,f}, \\ \Delta F^{(y)}_{z\mathrm{RL}} = +\tfrac{1}{2}\Delta F_{y,r},\ \Delta F^{(y)}_{z\mathrm{RR}} = -\tfrac{1}{2}\Delta F_{y,r}. \end{cases}$$

Combined result. Total wheel normal loads are the superposition of

$$F_{z,ij} = F_{z,ij,0} + \Delta F_{z,ij}^{(x)} + \Delta F_{z,ij}^{(y)}, \quad F_{z,ij} \geq 0, \ \sum_{ij} F_{z,ij} \approx mg.$$

Tire forces (reduced MF96)

Longitudinal and lateral forces are computed by reduced Pacejka'96 "Magic Formula" with load sensitivity retained via the peak term $D \propto F_{z,ij}$:

$$F_{x,ij} = D_x \sin\big(C_x \arctan(B_x s_{ij} - E_x(B_x s_{ij} - \arctan(B_x s_{ij})))\big),$$

$$F_{y,ij} = D_y \sin\big(C_y \arctan(B_y \alpha_{ij} - E_y(B_y \alpha_{ij} - \arctan(B_y \alpha_{ij})))\big),$$

where (B, C, D, E) are identified per axle. Aligning moment and combined-slip coupling are neglected for efficiency; this is adequate for our controller comparisons (§5.1).

Rolling resistance

Rolling-resistance moments use a speed-dependent model

$$M_{y,ij} = \big(C_{rr1} + C_{rr2} V_x + C_{rr3} V_x^2\big)\, F_{z,ij}\, R_{ij}.$$

Roll dynamics and 4-DoF plant

The high-fidelity plant augments planar dynamics with roll $(\phi, \dot{\phi})$:

$$I_x \ddot{\phi} = m_s(h_{CG} - h_{\text{roll}})\,(\dot{V}_y + V_x \dot{\psi})\cos\phi + m_s g (h_{CG} - h_{\text{roll}}) \sin\phi - M_f^{\text{spring}} - M_r^{\text{spring}} - M_f^{\text{damper}} - M_r^{\text{damper}},$$

where m_s is the sprung mass. The resulting lateral transfer feeds back to $F_{z,ij}$ and camber, yielding a realistic roll–tire interaction.

Simulation setup

The 4-DoF roll-augmented plant is used for all closed-loop tests. The NMPC controllers (3-DoF internal models) interact with this plant to assess robustness under model–plant mismatch. Sampling times, tire parameters and actuator limits are matched across layers for consistent evaluation.

4.3 Nonlinear Optimal Control Problem (NMPC)

At each control step j_c, the nonlinear model predictive controller computes the optimal input sequence

$$\mathbf{U} = [u_0, u_1, \ldots, u_{N_c-1}]^\top$$

that minimizes the finite-horizon cost function under dynamic, input, and state constraints, following established MPC formulations for integrated vehicle dynamics control [6], [13], [14]. The discrete nonlinear optimal control problem (NOCP) is formulated as:

$$\begin{aligned} \min_{\mathbf{U}} \ J_{\text{NMPC}} &= J_{\text{stage}} + J_{\text{term}} \\ &= \frac{1}{2} \sum_{k=0}^{N_p-1} \|\mathbf{y}_k - \mathbf{y}_{\text{ref},k}\|_W^2 + \frac{1}{2} \|\mathbf{y}_{N_p} - \mathbf{y}_{\text{ref},N_p}\|_{W_e}^2, \end{aligned} \tag{4}$$

subject to the discretized nonlinear vehicle dynamics and constraints:

$$x_0 = x_{\text{in}}, \tag{5a}$$
$$x_{k+1} = f(x_k, u_k, p_k), \tag{5b}$$
$$\mathbf{y}_k = g_{\text{out}}(x_k, u_k, p_k), \tag{5c}$$
$$u_{\min} \le u_k \le u_{\max}, \tag{5d}$$
$$x_{\min} \le x_k \le x_{\max}, \tag{5e}$$
$$\Delta u_{\min} \le u_k - u_{k-1} \le \Delta u_{\max}. \tag{5f}$$

Here x_k and u_k denote the state and control vectors, respectively; $\mathbf{y}_k$ collects the outputs that enter the cost; p_k represents the online or preview parameters. N_p and N_c are the prediction and control horizons, respectively.

Discretization and solver implementation.

The NMPC problem is discretized using a multiple-shooting scheme with N_p prediction steps over the finite horizon $H_p = N_p T_s$, and implemented in the acados framework [46].

To enhance numerical accuracy near the initial stage, a non-uniform time grid is adopted:

$$t_i = T\Big(\frac{i}{N_p}\Big)^{\alpha}, \quad \alpha = 3,$$

which densifies integration nodes near the current time step. The resulting step lengths $\Delta t_i = t_{i+1} - t_i$ yield higher short-term precision while keeping computational load manageable. The continuous-time dynamics $f(x, u, p)$ are integrated via an implicit Runge–Kutta (IRK) scheme, and automatic differentiation and Jacobian evaluation are handled by CasADi[3]. The nonlinear program is solved using the acados stack with a Gauss–Newton Hessian and a full-condensing HPIPM QP solver, backed by BLASFEO for embedded linear algebra [16], [17], [46], achieving sampling times on the order of tens of milliseconds.

Stage cost J_{stage}.

The stage cost penalizes the instantaneous tracking errors, control efforts, and input-rate smoothness terms. The output vector $\mathbf{y}_k$ includes:

$$\mathbf{y}_k = [\, e_\psi,\ e_y,\ D_F,\ D_R,\ T_F,\ T_R,\ d_F,\ d_R,\ t_F,\ t_R,\ \Delta D_F,\ \Delta D_R,\ \Delta T_F,\ \Delta T_R \,]^\top.$$

The diagonal weighting matrix W scales each component to balance trajectory accuracy, actuator limits, and control smoothness.

Implementation note. In the ACADOS solver, separate weights can be defined for the initial, intermediate, and terminal shooting nodes (W_0, W, W_e). In this work, the same weighting matrix $W_0 = W$ is used for the first and intermediate stages, simplifying tuning while maintaining consistent scaling across the prediction horizon.

Terminal cost J_{term}.

The terminal cost $\|\mathbf{y}_{e,N_p}\|^2_{W_e}$ ensures convergence of the tracking error at the end of the prediction horizon. In this study, $\mathbf{y}_{e,N_p}$ includes the look-ahead heading and lateral errors:

$$\mathbf{y}_{e,N_p} = [\, e_{\psi,\text{ahead}},\ e_{y,\text{ahead}} \,]^\top,$$

which enhance the anticipatory capability of the controller in curved or high-speed scenarios.

Weight scaling and tuning.

All cost matrices W and W_e are diagonal and dimensionally consistent. Initial weights are chosen based on the physical units of each variable (radians, meters, or $\mathrm{N}\cdot\mathrm{m}$), ensuring comparable magnitudes across terms. Empirical tuning follows standard MPC practice:

- Larger weights on e_y and e_ψ improve lateral accuracy.
- Moderate weights on rate terms in $\mathbf{y}_k$ suppress oscillations.
- Terminal weights W_e promote smooth convergence near horizon boundaries.

Relative scaling may be adjusted based on experimental results or to emulate hard constraints on key states and inputs.

Summary.

Overall, the NMPC formulation in (4)–(5) achieves a consistent trade-off between trajectory accuracy, actuation effort, and input smoothness under realistic vehicle and safety constraints. By unifying the stage and rate penalties within a single weighting matrix W, and separating terminal convergence through W_e, the controller preserves tuning simplicity while ensuring numerical stability. When combined with the nonlinear vehicle dynamics and quasi-static tire–load coupling, this formulation enables real-time, high-fidelity control of the 4WS–TV platform across diverse driving conditions. Reported real-time viability of nonlinear MPC in vehicle yaw/path-control tasks has been demonstrated with continuation-based schemes and embedded implementations [20], [32].

4.4 NMPC Weight Optimization Procedure

The nominal weights of the NMPC cost function (Sec. 4.3) were further refined through an automatic optimization stage. While the structure of the stage and terminal cost remained fixed, the numerical coefficients of the weighting matrices $\mathbf{W}$ and $\mathbf{W}_e$ were tuned to balance tracking accuracy, smoothness, and numerical efficiency.

Optimization method

A *surrogate–based global optimization* was employed to search the multi–dimensional weight space without requiring gradient information.[15] At each iteration, the optimizer sampled a candidate weight vector $\mathbf{w}$, ran the full closed–loop Simulink simulation of the NMPC controller, and evaluated a scalar performance index $J(\mathbf{w})$. The process was repeated for about 150–200 iterations, which was found sufficient for convergence in both DLC and circuit cases.

Cost function definition

The objective function combined several normalized performance indicators representing lateral tracking accuracy, stability, and comfort:

$$J(\mathbf{w}) = \omega_1 \frac{\max|e_{ct}|}{0.10} + \omega_2 \frac{\mathrm{rms}(e_h)}{0.01} + \omega_3 \frac{P_{95}(|\beta|)}{0.02} + \omega_4 \frac{P_{95}(|a_y|)}{6} + \omega_5 \frac{P_{95}(|\mathrm{jerk}|)}{10}, \tag{6}$$

where each term was normalized by a typical target value (0.10 m for cross–track error, 0.01 rad for heading error, etc.) to ensure comparable magnitude among metrics. The coefficients ω_i determined the trade–off between path–tracking precision, lateral stability, and smooth control effort.

Iteration and convergence

The optimization was executed separately for the two NMPC formulations (SIMP and DSTE) under the same vehicle and scenario setup. Each iteration corresponded to one complete closed–loop simulation over the test path. The cost typically decreased rapidly during the first 50 evaluations, followed by gradual convergence after 150–200 samples, as the surrogate model accurately captured the weight–performance relation. The final optimized weights provided improved RMS tracking errors and reduced sideslip without increasing control activity.

Discussion

This surrogate–based tuning approach allowed the NMPC to self–adjust its internal weight matrix according to the actual vehicle dynamics, eliminating the need for empirical manual tuning. The resulting weight sets were then fixed for the comparative evaluations in the ISO Double Lane Change and circuit scenarios presented below.

4.5 Baseline Controllers

Stanley Lateral Controller

The Stanley controller serves as the lateral baseline, combining heading and cross-track error feedback. The control law reads:

$$\delta = k_h\, e_\psi + \arctan\Big(\frac{k_{ct}\, e_{ct}}{V_x + \epsilon}\Big),$$

where $e_\psi = \psi - \psi_{\mathrm{ref}}$ and e_{ct} are the heading and lateral errors relative to the reference path. The parameters $k_h > 0$ and $k_{ct} > 0$ tune the response sensitivity to yaw and cross-track deviations, respectively, while $\epsilon > 0$ ensures numerical robustness at low speed. In the 4WS–TV platform, the front-axle steering command δ is distributed to the individual wheel steering angles according to the same geometry defined in Fig. 3.

PIDF2 Multi-Error Controller

We use two parallel PIDF loops to regulate lateral and yaw errors, respectively. Let e_y and e_ψ denote the lateral and heading errors. Each loop is a PID controller with a *filtered derivative*. The control laws are:

$$u_y(t) = K_{p_y}\, e_y(t) + K_{i_y} \int e_y(\tau)\, d\tau + K_{d_y}\, \frac{d}{dt}\big(\mathcal{F}\{e_y\}\big),$$

$$u_\psi(t) = K_{p_\psi}\, e_\psi(t) + K_{i_\psi} \int e_\psi(\tau)\, d\tau + K_{d_\psi}\, \frac{d}{dt}\big(\mathcal{F}\{e_\psi\}\big),$$

where $\mathcal{F}\{\cdot\}$ is a first–order low-pass filter applied to the error signal before differentiation, e.g.

$$\mathcal{F}(s) = \frac{\omega_c}{s + \omega_c} \quad \text{(equivalently, } K_d \, \tfrac{\omega_d s}{s+\omega_d} \text{ on the derivative channel).}$$

The two loop outputs are blended to form the final command:

$$u = u_y + u_\psi,$$

Summary of Baseline Controllers.

The two baseline controllers, Stanley and PIDF2, serve as classical references for assessing the benefits of model predictive control. Stanley is *geometric preview*–based (look-ahead to the path tangent) but does not perform model-based prediction/optimization or explicit constraint handling. PIDF2 is purely reactive without preview, using cascaded filtered PID on instantaneous errors. Both are simple and computationally light, yet their lack of constraint awareness and actuator-coupling treatment limits performance under aggressive manoeuvres or low-friction conditions.

Taken together, the four controllers—NMPC–SIMP, NMPC–DSTE, Stanley, and PIDF2—span a spectrum from classical reactive feedback to preview-based geometric control to full optimization-based MPC. This diversity enables a focused, apples-to-apples comparison across preview capability, constraint handling, and closed-loop performance.

5 Results and Comparisons of Teacher Controller

To assess the transient and steady-state performance of the proposed controllers, two representative driving scenarios are considered: (i) the ISO double lane change (DLC) maneuver, and (ii) a closed-loop circuit trajectory. These test cases jointly evaluate agility, stability, and robustness under typical on-road maneuvers.

ISO double lane change (DLC). The DLC maneuver is a well-established benchmark for lateral control validation, defined by ISO 3888-1 [26] It consists of two consecutive lane reversals with fixed lane width and prescribed entry/exit distances. The vehicle is required to complete the maneuver at a constant target velocity of 60 km/h without hitting lane boundaries. This test highlights the controller's ability to handle high-rate yaw dynamics, steering coordination, and transient sideslip suppression.

Circuit trajectory. To further evaluate tracking robustness and long-horizon stability, a closed circuit path is used. It combines straights, medium- and large-radius curves, and gradual curvature transitions similar to a proving-ground layout. This scenario exposes each controller to sustained steering activity and repeated curvature changes, revealing potential steady-state drift, actuator saturation, or numerical drift. The circuit test therefore complements the DLC by probing long-term consistency and robustness to modeling uncertainties.

Path construction and reference data. Both paths are constructed from analytical geometric primitives (straight, circular, and clothoid segments), ensuring C^1 continuity in position and heading. The resulting centerline is sampled at fixed arc-length intervals to obtain the reference profiles

$$\{x_{\text{ref}}(s),\, y_{\text{ref}}(s),\, \psi_{\text{ref}}(s),\, \kappa_{\text{ref}}(s)\}.$$

These are stored as lookup tables used by the tracking-error generator (Sec. 3.2) and the controller data interface (Sec. 3.3).

5.1 ISO Double Lane Change Results and Discussion

This section evaluates four controllers in the ISO Double Lane Change. We first look at the geometry of the manoeuvre—trajectory and zoomed views to explain why NMPC turns earlier and follows an inside line (Sec. 5.1). We then analyse lateral dynamics (a_y, β) and tracking errors (e_{ct}, e_h), and conclude with a quantitative KPI summary and a computation-time assessment.

Trajectory tracking and anticipative steering

Figure 4 compares the vehicle trajectories for the ISO double lane change (DLC), with two zoomed windows highlighting the entry and the plateau segments. The two NMPC controllers follow an *inside line* around both curvature reversals, because the optimisation looks ahead along the path and schedules steering *before* the curvature peak. [13] As a result, the cross–track and heading errors at entry/exit are reduced (e_{ct} stays small with earlier zero-crossings and e_h anticipates the path), improving peak and RMS tracking accuracy while avoiding large corrective inputs.

In contrast, PIDF is purely reactive (no preview), whereas Stanley is a geometric preview law without model-based prediction/optimization or explicit constraint handling; both lack constraint awareness, so the strongest corrections come around curvature changes [24], [41].

Between the two NMPC variants, **DSTE** augments the state with the tracking errors (e_{ct}, e_h) and roll–yaw coupling, so the errors are driven to zero *as states* across the horizon .[13] This yields the smallest tracking errors in the test. Notably, although **SIMP** sometimes appears *geometrically* more "inside" than DSTE in the global (X, Y) view, the cross–track error is defined in the Frenet frame as the orthogonal distance to the path tangent; without explicitly shaping the error dynamics in the state, SIMP can show a larger e_{ct} even when its footprint lies slightly further inside (see zoom B).

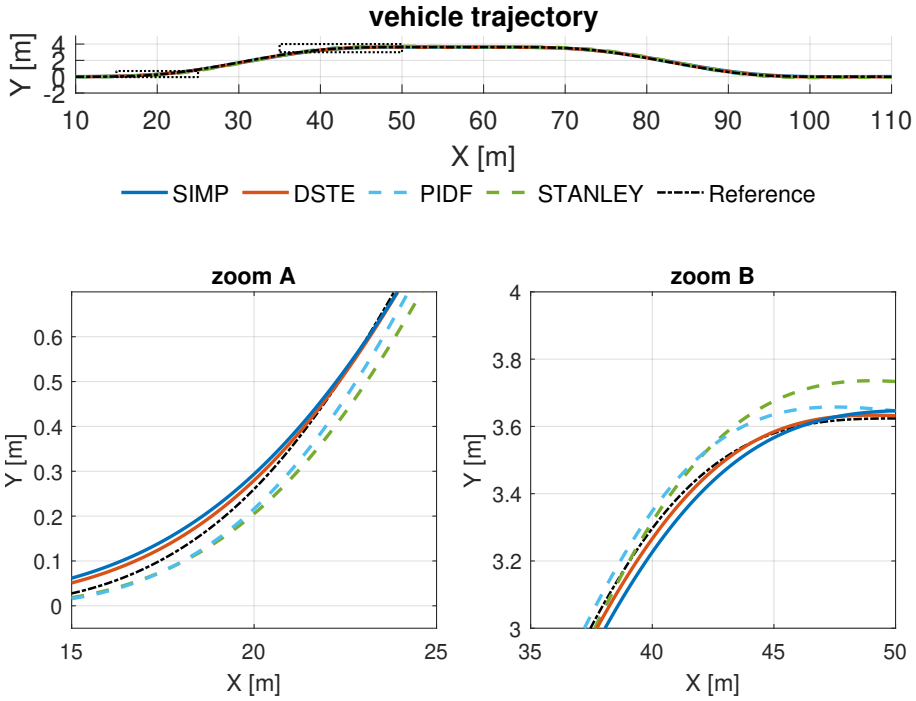

Figure 4: Vehicle trajectories in the ISO double lane change test. Zoom A: entry; Zoom B: plateau.

Lateral dynamics: a_y and sideslip β

Figure 5 reports the lateral acceleration and sideslip during the DLC. Since the test is run at a prescribed speed and over a fixed curvature profile, the lateral-acceleration envelope is essentially set by the kinematic relation $a_y \approx V_x^2\,\kappa_{\mathrm{ref}}(s)$ (for small sideslip). [38] Consequently, all controllers attain comparable peak a_y (SIMP 3.86 m/s^2, DSTE 4.11, PIDF 5.16, Stanley 5.00); the main differences appear in timing and smoothness around curvature reversals, rather than in the attainable maxima. Minor deviations from $V_x^2\kappa_{\mathrm{ref}}$ are due to transient sideslip and roll dynamics: controllers that better coordinate yaw and lateral forces exhibit less ringing and smaller β.

Sideslip is a more sensitive stability indicator. DSTE delivers the smallest values ($\beta_{\max} = 0.41°$, $\mathrm{RMS}_\beta = 0.24°$), markedly below PIDF (1.22°, 0.57°) and Stanley (1.13°, 0.54°), and also below SIMP (0.63°, 0.37°). A **smaller** β indicates better yaw–lateral force coordination and a *more stable/predictable* entry into the bend; this aligns with the inside-line behaviour enabled by NMPC's look-ahead optimisation.

Body roll is passive in this plant (no active anti-roll actuation), hence ϕ is primarily driven by lateral acceleration through the roll DOF (cf. Sec. 4.2):

$$I_x\ddot{\phi} + c_\phi\dot{\phi} + K_\phi\phi \;\approx\; m_s\,(h_{CG} - h_{\mathrm{roll}})\,a_y,$$

where $K_\phi = K_\phi^{(f)} + K_\phi^{(r)}$ and $c_\phi = c_\phi^{(f)} + c_\phi^{(r)}$.For small angles the quasi-static gain is

$$\phi_{\text{qs}} \approx \frac{m_s\,(h_{CG} - h_{\text{roll}})}{K_\phi}\,a_y,$$

so ϕ follows the low-frequency trend of a_y and differences among controllers arise mainly from the *timing and smoothness* of a_y rather than from different roll set-points. Accordingly, roll peaks remain moderate across all controllers (SIMP 0.72°, DSTE 0.78°, PIDF 0.97°, Stanley 0.88°)

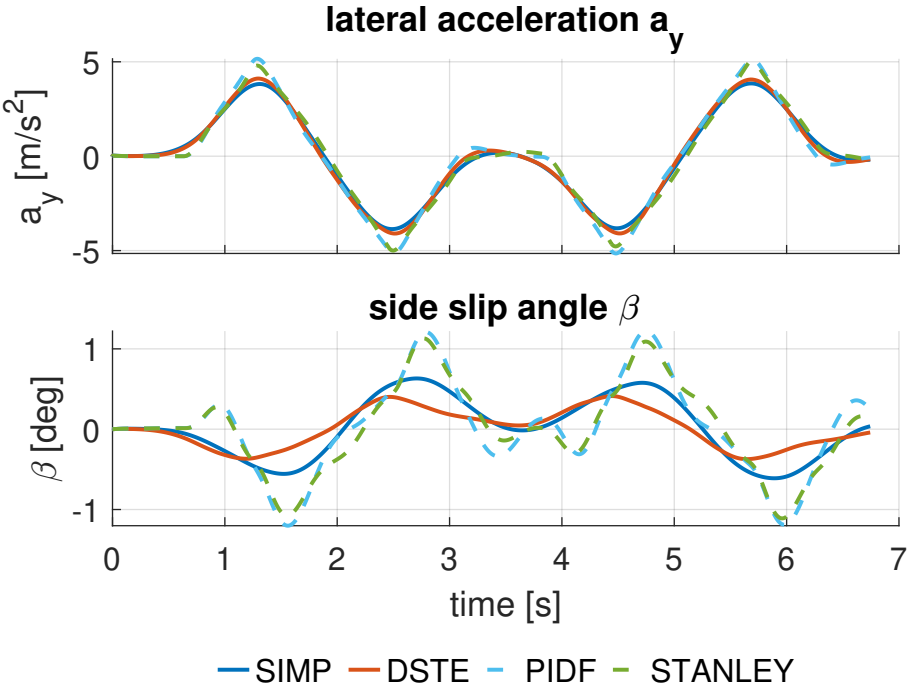

Figure 5: Lateral acceleration a_y (top) and sideslip angle β (bottom) in the DLC.

Tracking errors e_{ct} and e_h (lead–lag at corner entry)

Figure 6 shows the time histories of cross–track and heading errors. Because NMPC plans over the horizon, it turns in earlier. Consequently, at the entry of each bend the signs of e_{ct} and e_h for NMPC are often the *opposite* of those from PIDF/Stanley: NMPC leads the curvature change (an early bias with smaller e_{ct} and a heading that anticipates the path), while classical controllers lag and correct *after* the error builds up. This lead–lag pattern explains the inside-line trajectory versus the outside drift of the reactive controllers.

Quantitative analysis (KPIs)

Tables 1–2 summarise the quantitative results. DSTE achieves the smallest steady accuracy: $\text{RMS}_{e_{ct}} = 0.0205\,\text{m}$ and $\text{RMS}_{e_h} = 0.0038\,\text{rad}$, which are lower by **57%** vs. SIMP (0.0476 m), **38%** vs. PIDF (0.0330 m) and **73%** vs. Stanley (0.0764 m) for e_{ct}; and lower by **2.6%/71%/65%** vs. SIMP/PIDF/Stanley for e_h. On stability, DSTE limits sideslip to $\beta_{\max} = 0.41°$ (**66%** vs. PIDF, **64%** vs. Stanley, **35%** vs. SIMP) and $\text{RMS}_\beta = 0.24°$ (**58%/56%/35%** vs. PIDF/Stanley/SIMP). The max a_y of NMPC (3.86–4.11 m/s^2) is **17–20%** lower than classical controllers (5.00–5.16 m/s^2), with comparable roll peaks.

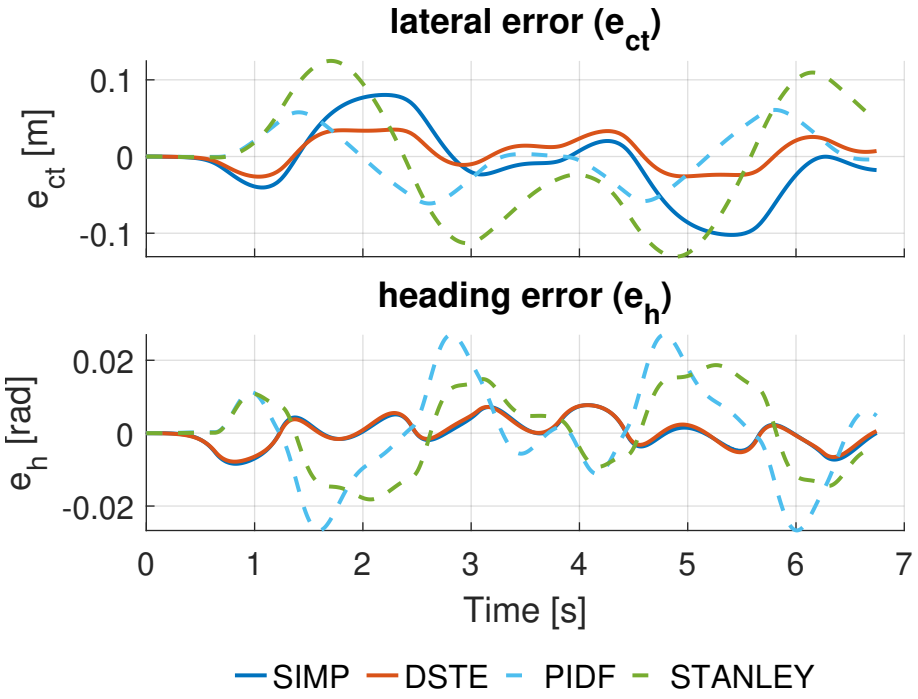

Figure 6: Cross–track (e_{ct}) and heading (e_h) errors in the DLC. NMPC leads at corner entry; classical controllers lag.

Table 1: Tracking-error KPIs in ISO DLC.

	SIMP	DSTE	PIDF	STANLEY
Max e_{ct} [m]	0.1023	**0.0351**	0.0612	0.1306
RMS e_{ct} [m]	0.0476	**0.0205**	0.0330	0.0764
Max e_h [rad]	0.0084	**0.0078**	0.0270	0.0187
RMS e_h [rad]	0.0039	**0.0038**	0.0131	0.0108

Table 2: Dynamic-response KPIs in ISO DLC.

	SIMP	DSTE	PIDF	STANLEY
Max a_y [m/s^2]	3.861	4.113	5.159	5.004
RMS a_y [m/s^2]	2.236	2.343	2.603	2.510
Max β [deg]	0.63	**0.41**	1.22	1.13
RMS β [deg]	0.37	**0.24**	0.57	0.54
Max ϕ [deg]	**0.72**	0.78	0.97	0.88

Computation cost and real-time feasibility

Table 3 lists the execution statistics. We report real-time factors as RTF $= T_{\text{sim}}/T_{\text{exec}}$; values > 1 indicate faster-than-real-time execution. PIDF and Stanley are faster than real time (RTF $>$ 1), with execution times around 5.4–5.5 s for a 6.74 s simulation (RTF 1.23–1.25). NMPC is below real time (RTF $<$ 1); however, **DSTE is** $\sim 60\%$ **faster than SIMP** (25.39 s vs. 63.58 s) while delivering higher accuracy, offering a better accuracy–cost trade-off.

Although DSTE has a slightly larger state dimension, it is numerically more efficient: by regulating the error states (e_{ct}, e_h) in its model, the stage cost is centred near zero and the local linearisation is better conditioned; consequently, the SQP/RTI scheme needs fewer outer iterations and the QP subproblems activate fewer constraints, which shortens the per-step CPU time relative to SIMP in

this scenario. Average CPU time per integration step is 0.2–0.5 ms for NMPC and negligible for classical controllers at the shown precision. In practice, real-time NMPC feasibility can be further improved by code generation, warm starts (RTI), shorter horizons, and structure-exploiting QP solvers [16], [45].

Table 3: Computation cost in ISO DLC.

	SIMP	DSTE	PIDF	STANLEY
Exec time [s]	63.584	25.390	5.410	5.499
Total time [s]	65.491	26.076	6.982	6.800
Sim time [s]	6.744	6.744	6.744	6.744
RTF (sim/exec)	0.11	0.27	**1.25**	**1.23**
Avg step [s/step]	0.0005	0.0002	0.0000*	0.0000*

* rounded at 10^{-4} s resolution.

Summary

Overall, NMPC tightens the line via anticipative steering and reduces sideslip. *DSTE* achieves the best accuracy by regulating error states directly while keeping dynamics smooth; *SIMP* can appear more "inside" geometrically but shows larger e_{ct} without explicit error-state shaping. Classical controllers are real-time capable but exhibit larger transients and sideslip around curvature reversals.

5.2 Circuit Road Results and Discussion

This section compares the four controllers on the Circuit Road. Following the same structure of the ISO DLC analysis (Sec. 5.1), we discuss: (i) trajectory geometry, (ii) lateral dynamics (a_y, β), (iii) tracking errors (e_{ct}, e_h), and (iv) quantitative KPIs and computation time.

Trajectory tracking and anticipative steering

Figure 7 shows the (X, Y) trajectories with two zoomed windows. Zoom A highlights the *first right-hand entry*; Zoom B the *final right-hand exit*. Unlike the ISO DLC, this circuit features *larger peak curvature*, so the test is run at a *lower speed*. Even so, the NMPC solutions still look ahead and schedule steering *before* the curvature peak—turn-in occurs earlier and exit is settled earlier. At this lower speed the geometric envelopes are similar, so the visible differences are modest. Between the two NMPC variants, **DSTE** stays closer to the reference centreline in both zooms, whereas **SIMP** often appears slightly more "inside" geometrically but ends up with a larger Frenet orthogonal error; this reflects SIMP's more aggressive early turn-in, which helps sideslip but biases e_{ct} around the reversals (cf. Sec. 5.2).

Lateral dynamics: a_y and sideslip β

Figure 8 shows lateral acceleration and sideslip angle. Since the speed profile is prescribed and curvature is fixed, the lateral-acceleration envelope follows the kinematic relation $a_y \approx V_x^2\, \kappa_{\mathrm{ref}}(s)$ (small sideslip), hence all controllers attain comparable a_y plateaus. **DSTE** and **SIMP** show almost identical a_y peaks (5.655 vs. 5.590 m/s^2) and RMS levels (2.831 vs. 2.796 m/s^2), with a small phase advance consistent with NMPC's earlier turn-in (Fig. 7).

For sideslip we adopt $\beta = atan2(V_y, V_x)$ (left positive), so a steady right/left turn yields $\beta < 0/\beta >$

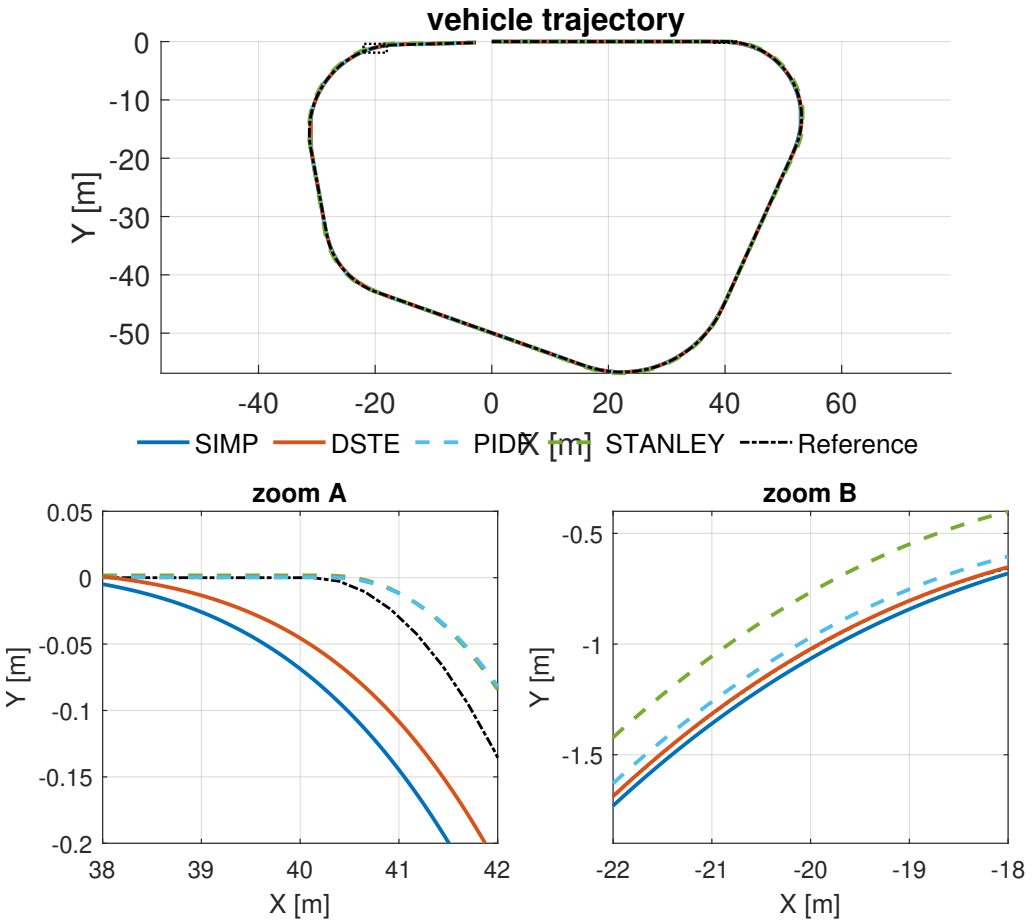

Figure 7: Circuit Road trajectories. Zoom A: first right-hand entry; Zoom B: final right-hand exit.

0. At corner entry, the transient evolution obeys

$$\dot{\beta} \approx \frac{a_y}{V_x} - r,$$

so the sign is set by the balance between $\frac{a_y}{V_x}$ and yaw rate r. Classical baselines (PIDF/Stanley) build r and a_y nearly in lockstep and quickly take the conventional sign; NMPC can briefly decouple them using 4WS+TV (rear-steer and yaw moment), yielding a short pre-alignment in r with modest $|a_y|$ and a temporary sign difference, before settling to the steady sign. In this low-speed case the effect is mild and the overall $|\beta|$ remains small for both NMPC controllers. Quantitatively, **SIMP** yields the smallest sideslip ($\beta_{\max} = 0.46°$, $\mathrm{RMS}_\beta = 0.14°$), while **DSTE** is slightly larger (0.54°, 0.23°), consistent with a tuning that prioritises tracking accuracy over minimal β.

Tracking errors e_{ct} and e_h

Figure 9 shows e_{ct} and e_h. Because NMPC plans over the horizon, it produces a short *lead* at each entry (earlier zero-crossings of e_{ct} and anticipative e_h), then returns towards zero sooner than the classical controllers. With the re-optimised weights in this run, **DSTE** achieves the smallest RMS errors (RMS e_{ct} = 0.0280 m, RMS e_h = 0.0107 rad), improving over **SIMP** (0.0483 m, 0.0108 rad). This indicates that **DSTE** *aligns more closely with the tracking-oriented optimisation objective* under the current weighting scheme—by regulating the error states in its model, it steers the optimisation towards reducing e_{ct}/e_h—whereas **SIMP**'s stronger early turn-in yields slightly smaller β at the expense of larger e_{ct}.

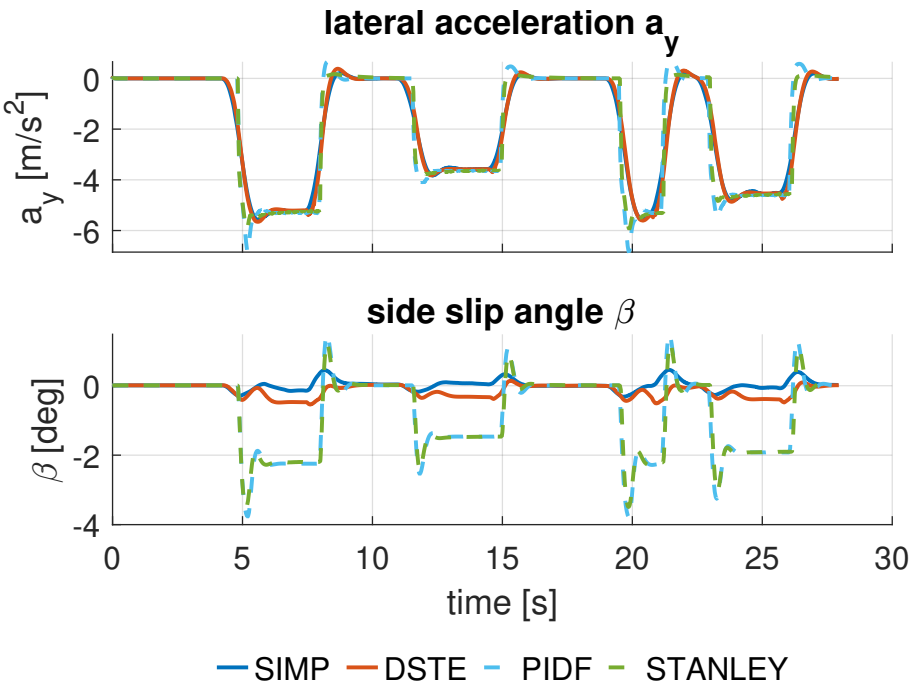

Figure 8: Lateral acceleration a_y (top) and sideslip angle β.

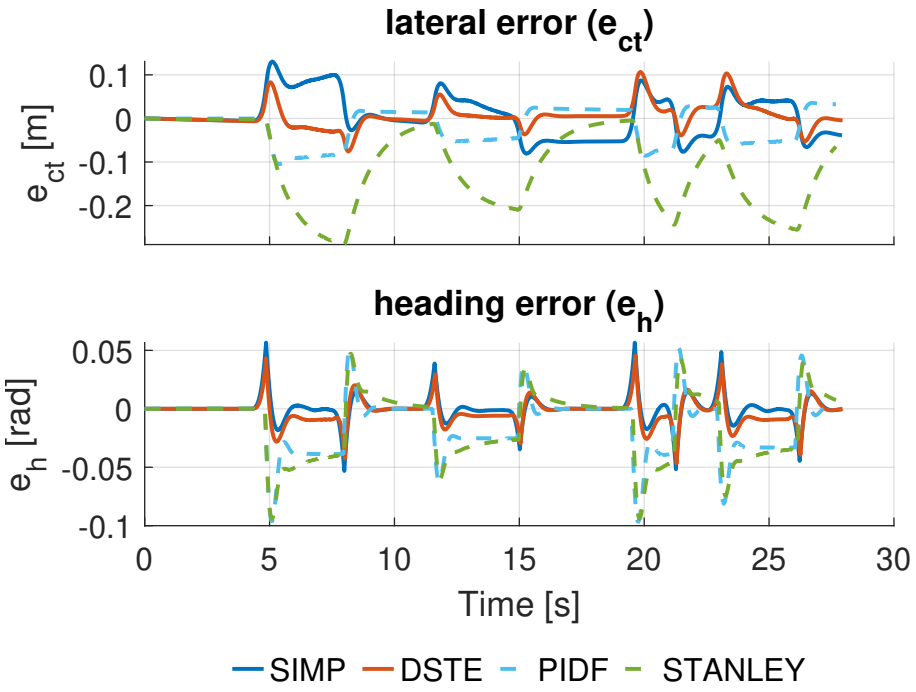

Figure 9: Cross–track (e_{ct}) and heading (e_h) errors.

Quantitative analysis (KPIs)

Tables 4–5 summarise the metrics for this run. **DSTE** attains the best steady tracking accuracy (RMS $e_{ct} = 0.0280\,\text{m}$; RMS $e_h = 0.0107\,\text{rad}$), while **SIMP** shows the lowest sideslip (RMS $\beta = 0.14°$). Classical controllers exhibit much larger sideslip spikes and higher RMS errors.

Table 4: Tracking-error KPIs on the Circuit Road.

	SIMP	DSTE	PIDF	STANLEY
Max e_{ct} [m]	0.1307	0.1067	0.1043	0.2894
RMS e_{ct} [m]	0.0483	**0.0280**	0.0442	0.1439
Max e_h [rad]	0.0566	0.0467	0.0969	0.0944
RMS e_h [rad]	0.0108	**0.0107**	0.0280	0.0314

Table 5: Dynamic-response KPIs on the Circuit Road.

	SIMP	DSTE	PIDF	STANLEY
Max a_y [m/s^2]	5.590	5.655	6.850	5.944
RMS a_y [m/s^2]	2.796	2.831	3.022	2.989
Max β [deg]	**0.46**	0.54	3.78	3.50
RMS β [deg]	**0.14**	0.23	1.36	1.32

Computation cost and real-time feasibility

Table 6 lists the execution statistics. Both NMPC controllers run below real time on this platform, with similar average step times, whereas the classical controllers are faster than real time. In this run, **DSTE** is only about **2.5%** faster than **SIMP** (196.87 s vs. 201.81 s exec time; both RTF ≈ 0.14 and average step 0.4 ms), so the computational gap is negligible here. At the *lower speed* profile of the Circuit Road, the optimisation problem is less demanding (weaker nonlinearities and fewer active constraints), and both NMPC formulations converge with comparable per-step effort; accordingly, their overall execution times become very close.

Table 6: Computation cost on the Circuit Road.

	SIMP	DSTE	PIDF	STANLEY
Exec time [s]	201.810	196.871	15.765	15.904
Total time [s]	203.316	197.741	22.549	17.827
Sim time [s]	27.920	27.944	27.679	27.679
RTF (sim/exec)	0.14	0.14	**1.76**	**1.74**
Avg step [s/step]	0.0004	0.0004	0.0000*	0.0000*

* rounded at 10^{-4} s resolution.

5.3 Chapter Summary and Outlook

This chapter compared four controllers on ISO DLC and the Circuit Road, analysing trajectory geometry, lateral dynamics (a_y, β), tracking errors (e_{ct}, e_h), and computation time. NMPC consistently turned in *before* curvature peaks and followed an inside line, reducing entry/exit errors; classical PIDF/Stanley corrected *after* errors built up. Among NMPC variants, **DSTE** achieved the smallest RMS e_{ct}/e_h and the lowest sideslip in ISO DLC (Tables 1–2), and remained the most accurate on the Circuit Road (Table 4); **SIMP** sometimes showed slightly smaller β but at the cost of larger e_{ct} around reversals. In computation, NMPC ran below real time; **DSTE was** $\sim 60\%$ **faster than SIMP** in ISO DLC (Table 3) due to better-conditioned linearisations from error-state regulation, while on the Circuit Road the gap was negligible (Table 6). Overall, **DSTE** offers the best accuracy–stability–compute trade-off across scenarios. These findings motivate Chapter 6, where we distill NMPC via imitation learning to retain small (e_{ct}, e_h, β) with real-time execution.

6 Imitation Learning of NMPC for Real-Time Path Following

Chapter 5.1 showed that the NMPC-DSTE achieves the best tracking and sideslip suppression but falls short of real-time on our platform. This chapter investigates *supervised imitation learning* (behavior cloning) [9], [33]. to mimic the teacher's state–action mapping and enable real-time deployment in Simulink while retaining the closed-loop benefits of NMPC.[23], [27]

We first formulate the learning target (Sec. 6.1), then describe the dataset generation and preprocessing pipeline that matches our Simulink setup (Sec. 6.2). The neural network, training details, and deployment are given in Secs. 6.4 6.5; closed-loop results follow in Chapter 7.

6.1 Problem formulation and learning target

Teacher and goal. Let $\pi^\star$ denote the NMPC–DSTE used in Chapter 5.1. At controller step $T_s = 0.008\,\mathrm{s}$ (horizon $N = 20$), the teacher maps the online feature vector $\mathbf{x}_k \in \mathbb{R}^8$ to the *absolute* control $\mathbf{u}_k^\star \in \mathbb{R}^8$. We learn a student $\hat{\pi}_\theta : \mathbb{R}^8 \to \mathbb{R}^8$ that can replace $\pi^\star$ in real time with comparable closed-loop behavior.[23], [27]

Inputs and outputs (what we use). The policy input is the 8-D, centerline-referenced local state

$$\mathbf{x}_k = \{V_x, V_y, \dot{\psi},\ e_{ct}, e_h,\ \kappa_{\mathrm{ref}},\ e_{ct}^{\mathrm{ahead}}, e_h^{\mathrm{ahead}}\},$$

z-score normalized using *training-split* statistics (frozen thereafter). The target is the teacher's absolute command

$$\mathbf{u}_k^\star = \{D_F, d_F, D_R, d_R,\ T_F, t_F, T_R, t_R\}.$$

Loss. Given $\mathcal{D} = \{(\mathbf{x}_k, \mathbf{u}_k^\star)\}$, we minimize the MSE on *absolute* outputs with weight decay:

$$\mathcal{L}(\theta) = \sum_k \left\| \hat{\pi}_\theta(\tilde{\mathbf{x}}_k) - \tilde{\mathbf{u}}_k^\star \right\|_2^2 \ +\ \lambda \|\theta\|_2^2, \tag{7}$$

where tildes denote the fixed, training-split transforms used throughout (§6.2, §6.4).

Safety layer (deployment). At deployment we enforce the same box and rate limits as the teacher:

$$\mathbf{u}_{\min} \le \hat{\mathbf{u}}_k \le \mathbf{u}_{\max}, \qquad \|\hat{\mathbf{u}}_k - \hat{\mathbf{u}}_{k-1}\|_\infty \le \dot{\mathbf{u}}_{\max} T_s,$$

implemented by Saturation and Rate Limiter blocks.

6.2 Dataset generation and preprocessing

We fix the longitudinal speed command during data acquisition and focus on lateral policy imitation. To make the learned policy robust beyond the teacher's near-perfect regimes, we deliberately diversify training rollouts by random initialization and stratified sampling on curvature. We then normalize inputs (and optionally scale labels) using statistics computed on the *training* split only. Quality filtering keeps only successful NMPC solves. The following subsections detail each step.

6.2.1 Randomized initial conditions and deliberately non-optimal teacher rollouts

Even though the NMPC teacher can keep the tracking error tiny when started near the centerline and heading-aligned, such clean rollouts concentrate almost all samples in a small-error manifold

and provide little supervision for large-error recovery.

Therefore, for every simulation we uniformly sample the start arc-length on the track and randomize the initial yaw (and, optionally, a small lateral offset), subject to feasibility guards. This widens the distribution of $(e_{ct}, e_h, \kappa_{\text{ref}}, \ldots)$ and the resulting teacher commands $u_k^\star$ cover a wider range , exposing the learner to corrective strategies rather than only steady-state refinements. [33], [39] This design choice follows prior NMPC→DNN imitation that also randomized the initial pose to enrich data diversity. [30]

6.2.2 Signals logged and feature selection

At the controller rate we record the NMPC state, parameters, and control in your implementation:

$$\begin{aligned}
\textbf{State } (\texttt{sym_x}) &: [x,\ y,\ \psi,\ V_x,\ V_y,\ \dot{\psi},\ \omega_{FL}, \omega_{FR}, \omega_{RL}, \omega_{RR},\ e_y,\ e_\psi]^\top, \\
\textbf{Parameters } (\texttt{sym_p}) &: [x_{\text{ref}}, y_{\text{ref}}, \psi_{\text{ref}}, \dot{\psi}_{\text{ref}},\ \psi_{\text{ahead}}, x_{\text{ahead}}, y_{\text{ahead}}, \\
&\qquad \dot{V}_x, \dot{V}_y,\ T_{\text{pid}},\ d_F^{\text{prev}}, d_R^{\text{prev}}, T_F^{\text{prev}}, T_R^{\text{prev}},\ T_d]^\top, \\
\textbf{Teacher input } (\texttt{sym_u}) &: [D_F,\ d_F,\ D_R,\ d_R,\ T_F,\ t_F,\ T_R,\ t_R]^\top.
\end{aligned}$$

Feature selection

- **Global pose** $[x, y, \psi]$: removed to enforce *translation/rotation invariance*. Absolute map-frame coordinates encourage track memorization and harm cross-track/generalization; local errors already encode the needed geometry.
- **Wheel speeds** $[\omega_{FL}, \omega_{FR}, \omega_{RL}, \omega_{RR}]$: removed due to (i) strong collinearity with longitudinal states/commands in closed loop, (ii) risk of label leakage under wheel-speed/torque constraints.
- **Previous control** $[d_F^{\text{prev}}, d_R^{\text{prev}}, T_F^{\text{prev}}, T_R^{\text{prev}}]$: *excluded in the main model.* We train and deploy *absolute* commands with plant-side saturation/rate limits; feeding the previous input turns the mapping into a quasi-autoregressive smoother and entangles actuator-rate dynamics with the policy, reducing portability across different limit settings. (We still log them for ablation and diagnostics.)
- **Accelerations** $[\dot{V}_x, \dot{V}_y]$: not used as inputs (high-frequency derivatives add noise and are largely redundant with $[V_x, V_y, \dot{\psi}]$ for lateral policy).

Derived signals and what we *keep*. We keep local, centerline-referenced quantities and look-ahead cues:

$e_y \equiv e_{ct}, \quad e_\psi \equiv e_h, \quad (e_{ct}^{\text{ahead}}, e_h^{\text{ahead}})$ are computed from $(x_{\text{ahead}}, y_{\text{ahead}}, \psi_{\text{ahead}})$ w.r.t. the reference frame.

The curvature reference κ_{ref} is obtained from the path generator.

Policy input (8-D).

$$\mathbf{x}_k^{\text{in}} = \{V_x, V_y, \dot{\psi},\ e_{ct}, e_h,\ \kappa_{\text{ref}},\ e_{ct}^{\text{ahead}}, e_h^{\text{ahead}}\} \in \mathbb{R}^8.$$

All inputs are z-score standardized using *training-split* statistics and then frozen for validation, test, and deployment.

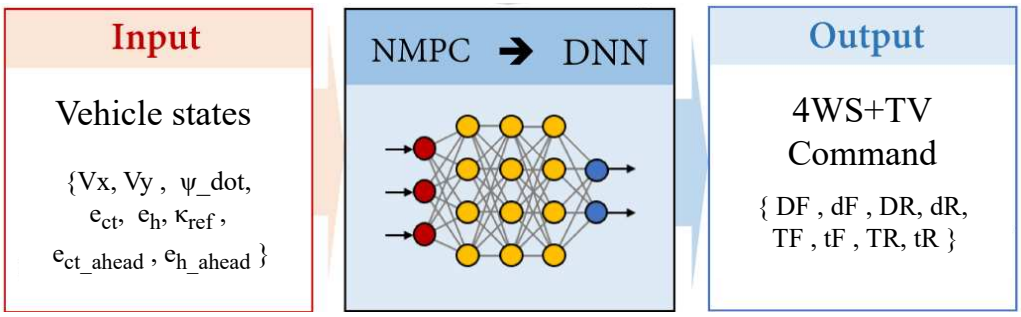

Figure 10: Imitation learning architecture

Outputs . The network predicts the teacher's *absolute* control $\hat{\mathbf{y}} = [D_F, d_F, D_R, d_R, T_F, t_F, T_R, t_R]^\top \in \mathbb{R}^8$, implemented as $\hat{\mathbf{y}} = \mathrm{diag}(\mathbf{s})\,\tanh(\mathbf{z}) + \mathbf{q}_{50}$ with $(\mathbf{q}_{50}, \mathbf{s})$ computed on the training split; plant-side saturation and rate limits are applied downstream.

6.2.3 Labels, trimming, split, and quality filtering

We use the teacher's *absolute* commands as labels, $\mathbf{y}_k = \mathbf{u}_k^\star \in \mathbb{R}^8$. Rate limits are enforced downstream at deployment, so no target differencing is used. For each rollout we trim the first/last 0.2 s to remove start/stop transients. Quality filtering keeps only successful NMPC solves (`solver_status` $= 0$). Optional filters for top-cost outliers or excessive SQP iterations are disabled by default.

We then shuffle with a fixed seed and split **80/10/10** into train/val/test, enforcing a minimum absolute size per split when available. To avoid a straight-dominated learner, *curvature-aware resampling is applied to the* ***training*** *split only*, while validation/test remain untouched to preserve natural statistics. The resampling scheme (percentile-based buckets with fixed duplication multipliers and an additional transition boost) follows our code exactly; thresholds and multipliers are detailed in §6.2 "Curvature-balanced upsampling".

6.2.4 Curvature-balanced upsampling

Natural laps are imbalanced: low-curvature (near-straight) segments dominate, whereas medium and large bends and curvature *transitions* are scarce. To ensure the learner sees enough cornering and transition cases [21],we *pre-duplicate* training samples by curvature buckets with fixed multipliers (validation/test unchanged).

Let $k_i = |\kappa_{\mathrm{ref}}(i)|$ and $K_{\max} = \max_{i \in \mathcal{I}_{\mathrm{train}}} k_i$. We form four exclusive classes in the *training* split:

$$\text{Small: } 0 \le k_i < 0.2K_{\max}, \quad \text{Medium: } 0.2K_{\max} \le k_i < 0.5K_{\max}, \quad \text{Large: } 0.5K_{\max} \le k_i \le K_{\max},$$

$$\text{Transition: } \left|\tfrac{d\kappa}{ds}(i)\right| \ge \tau_g \text{ (top 20\% on } \mathcal{I}_{\mathrm{train}}),$$

where "Transition" overrides curvature bins to avoid overlap.

Duplication multipliers & transition boost. We duplicate each bucket by fixed factors

$$R_{\mathrm{Flat}} = 1, \quad R_{\mathrm{Small}} = 4, \quad R_{\mathrm{Medium}} = 4, \quad R_{\mathrm{Large}} = 7.$$

Let $\Delta\kappa(i)$ be the one-step curvature difference; define a transition threshold

$$T_\kappa = \max\big(Q_{0.92}(|\Delta\kappa|),\, 5\times 10^{-5}\big), \quad \text{and mark Transition if } |\Delta\kappa(i)| \geq T_\kappa.$$

Transition samples receive an additional boost $R_{\text{Trans}} = 6$ (i.e., add $R_{\text{Trans}} - 1 = 5$ extra copies). Thus a *Large*+Transition sample appears 7+5 = 12 times in the training pool; a *Small/Medium*+Transition sample appears $4 + 5 = 9$ times. Flat segments typically do not trigger the transition boost.

On the **Flat** bucket we optionally add mild Gaussian jitter to a small subset. Finally, we concatenate all duplicated (and jittered) samples and shuffle to form the training set; validation and test splits remain untouched.

6.2.5 Normalization and scaling

We apply per-feature z-score standardization using *training-split* statistics and keep them frozen thereafter:

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \boldsymbol{\mu}_{\text{train}}}{\boldsymbol{\sigma}_{\text{train}} + 10^{-8}}.$$

Labels / network outputs We train on the *absolute* labels $\mathbf{y}$ using a robust, per-dimension percentile scaling centered at the median: for each output component $y^{(j)}$ on the training split, compute

$$q_1^{(j)} = \text{perc}_{1\%}\big(y^{(j)}\big), \quad q_{50}^{(j)} = \text{median}\big(y^{(j)}\big), \quad q_{99}^{(j)} = \text{perc}_{99\%}\big(y^{(j)}\big),$$

and define the half-range scale

$$s^{(j)} = \max\Big(10^{-6},\, \tfrac{q_{99}^{(j)} - q_1^{(j)}}{2}\Big).$$

We then map

$$\tilde{y}^{(j)} = \frac{y^{(j)} - q_{50}^{(j)}}{s^{(j)}} \quad \text{for all } j,$$

without mandatory clipping (clipping to $[-1, 1]$ is optional and was not used in our main runs). At inference we invert the scaling

$$y^{(j)} = \tilde{y}^{(j)}\, s^{(j)} + q_{50}^{(j)},$$

and finally apply the outer saturation / rate-limiter using $(\mathbf{u}_{\min}, \mathbf{u}_{\max})$ as in deployment. All (q_1, q_{50}, q_{99}, s) are computed on the *training* split and reused for validation, test, and deployment.[5]

6.3 Network architecture

We adopt a dual-head residual MLP for 4WS+TV *absolute* control. The trunk uses residual MLP blocks at width 1024; shared projection compresses to 512; each head is a 3-layer MLP at 512 with a 4-D output; the two heads concatenate to 8-D and pass through a `tanh`+fixed de-scaling map.11(a).

6.3.1 Input layer (z-score, frozen)

We normalize each input feature with training-split statistics and *freeze* them for validation, test and deployment:

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \boldsymbol{\mu}_{\text{train}}}{\boldsymbol{\sigma}_{\text{train}} + 10^{-8}}.$$

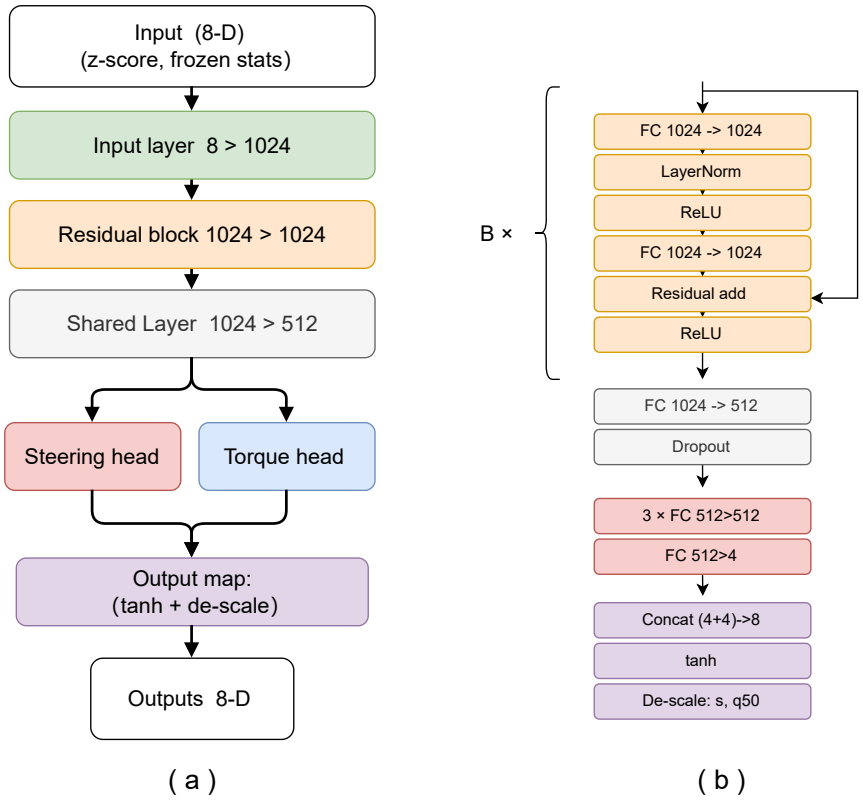

Figure 11: MLP Architecture

Freezing avoids leakage from non-training splits, keeps the Simulink interface deterministic, and makes the input layer non-trainable (only storing $\boldsymbol{\mu}_{\text{train}}, \boldsymbol{\sigma}_{\text{train}}$). The input dimensionality is 8-D. See §6.2 for the exact feature set and exclusions.

6.3.2 Residual trunk

Each residual block maintains a width of 1024 and learns a residual mapping with a skip connection:

$$\mathbf{a} = \mathrm{ReLU}\big(\mathrm{LN}(\mathbf{W}_a\mathbf{h}_{\text{in}} + \mathbf{b}_a)\big), \quad \mathbf{b} = \mathbf{W}_b\mathbf{a} + \mathbf{b}_b, \quad \mathbf{h}_{\text{out}} = \mathrm{ReLU}\big(\mathbf{b} + \mathbf{h}_{\text{in}}\big).$$

This *FC–LN–ReLU–FC + skip-add (post-activation)* block (Fig. 11(b)) keeps the channel width fixed (1024→1024) by using element-wise *addition* rather than concatenation, thereby avoiding parameter growth. It reframes learning as a small correction on top of the identity: the skip path supplies an identity gradient path that stabilizes optimization in deep/wide MLPs and accelerates convergence[44]. These choices follow the residual-learning principle of He et al. [22] and mirror the residual-style sublayers used in purely MLP architectures such as ResMLP and MLP-Mixer, which enable scalable training and strong generalization [43].

6.3.3 Shared projection (1024 → 512) with dropout

After the residual trunk, we compress to a shared 512-D representation via

$$\text{FC } 1024\to512 \xrightarrow{\text{LN+ReLU}} \text{Dropout}(p = 0.12).$$

Table 7: Network summary (dual-head residual MLP; 8-D input, 8-D absolute output).

Block	**Shape / width**	**Depth**	**Norm./act.**
Input layer	$8 \to 8$	–	z-score (frozen)
Input projection	$8 \to 1024$	1 FC	LN + ReLU
Residual trunk	$1024 \to 1024$	B=6 blocks	LN (pre), ReLU, skip-add
Shared projection	$1024 \to 512$	1 FC	LN + ReLU + Dropout(0.12)
Steering head	$512 \to 4$	3 FC + out	LN + ReLU (hidden)
Torque head	$512 \to 4$	3 FC + out	LN + ReLU (hidden)
Concat & output	$4{+}4 \to 8$	–	`tanh` + de-scale + clamp
Total FC layers (with B=6): $1 + 2B + 1 + 8 = 22$ (excluding output map/LN).			

This bottleneck reduces head parameters and computation, regularizes the shared features, and serves as a clean interface between the task-agnostic trunk and task-specific heads[42].

6.3.4 Dual heads and output map

We separate actuator families to reduce cross-task interference and let each head specialize. Each head is a conventional 3-layer MLP at width 512 (hidden layers: FC+LN+ReLU), followed by a 4-D linear output(Figure 11):

$$\text{Head}: \quad 512 \to 512 \to 512 \to 512 \to 4.$$

Concatenating the two 4-D outputs yields an 8-D latent vector $\mathbf{z} = [\mathbf{z}^{\text{st}}; \mathbf{z}^{\text{tq}}]$. The final output map enforces bounded latent values and restores physical units:

$$\hat{\tilde{\mathbf{y}}} = \tanh(\mathbf{z}) \in [-1, 1]^8, \qquad \hat{\mathbf{y}} = \text{diag}(\mathbf{s})\, \hat{\tilde{\mathbf{y}}} + \mathbf{q}_{50},$$

optionally followed by a clamp to $[\mathbf{u}_{\min}, \mathbf{u}_{\max}]$ at the plant interface. The $(\mathbf{q}_{50}, \mathbf{s})$ statistics are computed on the training split and reused for validation/test/deployment (see §6.2).

6.4 Training

We train on *absolute* labels with the output de-scaling embedded in the graph (Fig. 11). Hence the loss is computed in physical units.

Our recipe combines:

(i) a residual MLP trunk;

(ii) curvature-balanced data augmentation (Sec. 6.2);

(iii) Adam with piecewise–decayed learning rate;

(iv) strong early stopping on validation RMSE;

(v) L2 weight decay and gradient clipping;

(vi) frozen input normalization statistics and fixed output de-scaling $(\mathbf{s}, \mathbf{q}_{50})$ computed on the training split.

We use Adam with a piecewise learning-rate decay [29] to couple fast early progress with controlled late-stage refinement. A comparatively higher initial step size quickly reaches a good basin; sched-

uled $\times 0.5$ drops at fixed intervals then anneal the updates, improving stability and final RMSE at the chosen batch size (1024) and model capacity.

Early stopping functions as an explicit regularize. With high capacity, patience-based stopping using infrequent validation (large validation interval with large patience) avoids reacting to short-term noise while preventing overfitting[37]. The best-validation checkpoint is retained to minimize selection bias.

L2 weight decay and absolute-value gradient clipping stabilize optimization in wide fully connected layers. Weight decay controls weight norms and improves generalization; clipping at a fixed threshold prevents rare exploding gradients from producing destabilizing updates, yielding smoother training curves and more reproducible endpoints.[36]

Finally, freezing the input z-score statistics and the output de-scaling $(\mathbf{s}, \mathbf{q}_{50})$ prevents train/validation leakage and guarantees parity across splits and deployment. Losses and metrics remain in physical units throughout, which matches the downstream Simulink interface and simplifies the application of external saturation and rate limits.

We do not rely on Bayesian hyper-parameter search. Instead, we adopt a simple model-scaling procedure: begin with a smaller network and gradually increase width and the number of residual blocks while monitoring validation RMSE; the final configuration is the smallest one at which the validation curve saturates.

6.4.1 Coarse training

We first train longer with a relatively higher learning rate to reach a good basin.

- **Objective / metrics.** MSE on absolute outputs $\hat{\mathbf{y}}$; we report RMSE on validation/test.
- **Optimizer.** Adam (`Epsilon`=10^{-8}), batch size 1024, shuffle every epoch.
- **LR schedule.** Initial LR 8×10^{-4} with piecewise decay: $\times 0.5$ every 10 epochs.
- **Epochs / early stopping.** Max 220 epochs; validation every 200 iterations; patience 80 (keep `best-validation` checkpoint).
- **Regularization & stability.** L2 weight decay 4×10^{-4} on all FC layers; absolute gradient threshold 7.5.
- **Sampling.** Mini-batches are drawn uniformly from the augmented training pool; validation/test are never altered.

6.4.2 Fine tuning

We then reduce the learning rate and relax the decay period to refine the solution around the best checkpoint.

- **Initialization.** Load the best validation checkpoint from Stage A.
- **LR schedule.** Initial LR 2×10^{-4}; piecewise decay $\times 0.5$ every 20 epochs.
- **Epochs / early stopping.** Max 200 epochs; validation every 200 iterations; patience 80 (again keep `best-validation`).

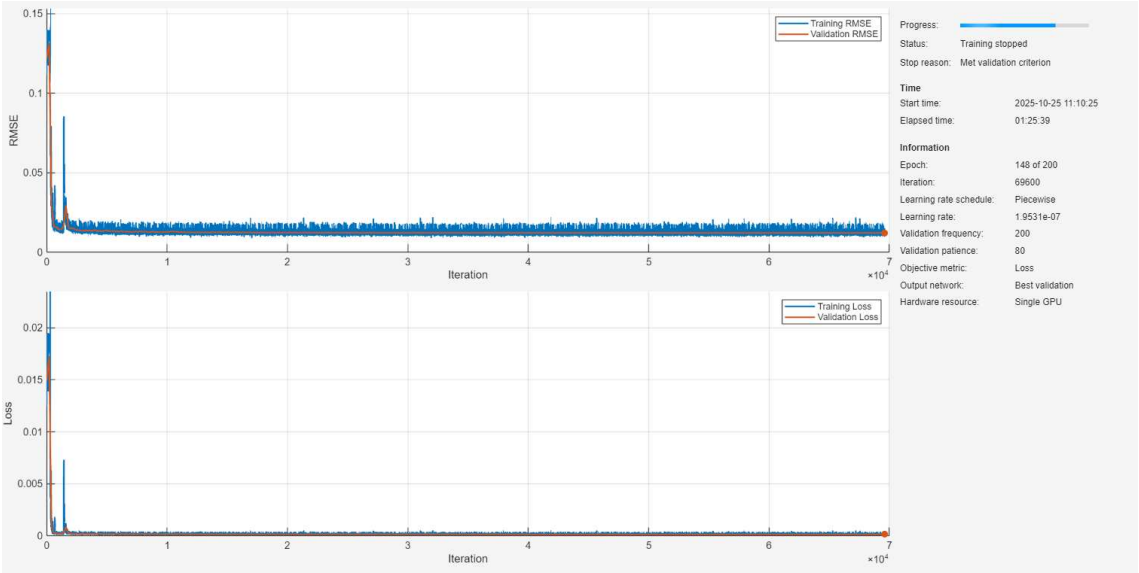

Figure 12: Training/validation curves

- **Regularization & stability.** L2 weight decay 3×10^{-4}; absolute gradient threshold 7.5 (same as Stage A).

6.4.3 Validation and test

We evaluate the *best-validation* checkpoint (kept by early stopping) on the validation and test splits. No augmentation or curvature re-sampling is applied at evaluation; metrics are computed in *physical units* using the same frozen transforms as at training time (Sec. 6.4). We report per-head RMSE (steering vs. torque) and an overall RMSE.

Metrics. For a split $\mathcal{I}$ and output component j,

$$\mathrm{MSE}^{(j)} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \left(\hat{y}_i^{(j)} - y_i^{(j)}\right)^2, \quad \mathrm{RMSE}_{\mathrm{steer}} = \sqrt{\frac{1}{4} \sum_{j \in \mathrm{steer}} \mathrm{MSE}^{(j)}}, \; \mathrm{RMSE}_{\mathrm{torque}} = \sqrt{\frac{1}{4} \sum_{j \in \mathrm{torque}} \mathrm{MSE}^{(j)}},$$

$$\mathrm{RMSE}_{\mathrm{overall}} = \sqrt{\frac{1}{8} \sum_{j=1}^{8} \mathrm{MSE}^{(j)}}.$$

(Equivalently, $\mathrm{RMSE}_{\mathrm{overall}} = \sqrt{\frac{1}{|\mathcal{I}|} \sum_i \frac{1}{8} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2}$.)

6.5 Deployment in Simulink

The trained `dlnetwork` is loaded by a `Predict` block (*Network from MAT-file*). Upstream we mux the **8** features in the same order as in training ($[V_x, V_y, \dot{\psi}, e_{ct}, e_h, \kappa_{\mathrm{ref}}, e_{ct}^{\mathrm{ahead}}, e_h^{\mathrm{ahead}}]$). No pre/post scaling is done in Simulink: input z-score and output de-scaling are *embedded* in the network as frozen layers (Sec. 6.4).

Output path. The network already implements

$$\hat{\mathbf{y}}_k = \mathrm{diag}(\mathbf{s}) \, \tanh(\mathbf{z}_k) + \mathbf{q}_{50} \xrightarrow{\text{internal clamp}} \mathrm{clip}(y; \mathbf{u}_{\min}, \mathbf{u}_{\max}),$$

so amplitudes are bounded inside the model. Downstream we still place a *Saturation* (same

$[\mathbf{u}_{\min}, \mathbf{u}_{\max}]$) and a *Rate Limiter* ($\|\Delta\mathbf{u}\|_\infty \le \dot{\mathbf{u}}_{\max} T_s$) as defensive guards.

In nominal operation, both guards are effectively pass-through (the internal clamp and the learned smoothness keep outputs inside range and below slew limits); they only activate if limits or sampling time are mismatched, or under rare numerical glitches.

Timing and types. The policy runs at $T_s = 0.008\,\text{s}$ (same as NMPC) and executes in single precision (float32). Upstream `double` signals are cast *once* to `single` before `Predict`; if the plant expects `double`, the network output is cast back *once* after it. Avoiding back-and-forth conversions reduces noise and load. The policy outputs *absolute* commands, so no previous-control tap is needed; actuator slew is enforced by a downstream Rate Limiter.

Parity. Input z-score ($\boldsymbol{\mu}, \boldsymbol{\sigma}$) and output de-scaling ($\mathbf{s}, \mathbf{q}_{50}$) are embedded in the network, so training/validation/deployment share identical transforms: $\hat{\mathbf{x}} = (\mathbf{x} - \boldsymbol{\mu}_{\text{train}})/(\boldsymbol{\sigma}_{\text{train}} + 10^{-8})$, $\hat{\mathbf{y}} = \text{diag}(\mathbf{s})\tanh(\mathbf{z}) + \mathbf{q}_{50}$. Feature order is the same fixed 8-D vector; sampling time is identical ($T_s = 0.008\,\text{s}$); no extra scaling is applied in Simulink. After `Predict` only a Saturation (amplitude) and a Rate Limiter (slew-rate) are used, normally inactive. Inference runs with dropout disabled and stateless LayerNorm, matching MATLAB `predict` up to float32 rounding.

7 Imitation Learning Results

We compare the student policy (DNN) against the NMPC–DSTE teacher in closed loop on the *same* Simulink plant, reference, bounds, and rate limits, at $T_s = 0.008\,\mathrm{s}$. The DNN runs in single precision inside a `Predict` block (see §6.5). We use the *best-validation* checkpoint from §6.4, evaluate on the original (un-resampled) validation/test splits, and keep the frozen input z-score and output de-scaling embedded in the network so that all metrics are reported in physical units. To assess generalization (rather than mere distribution matching), we also evaluate on a *held-out* track generated by the same tool chain but never seen during training or validation, not involved in resampling or early-stopping decisions; no augmentation is applied at evaluation time. Figures report trajectories, tracking errors, and lateral dynamics; tables summarize KPIs and execution statistics.

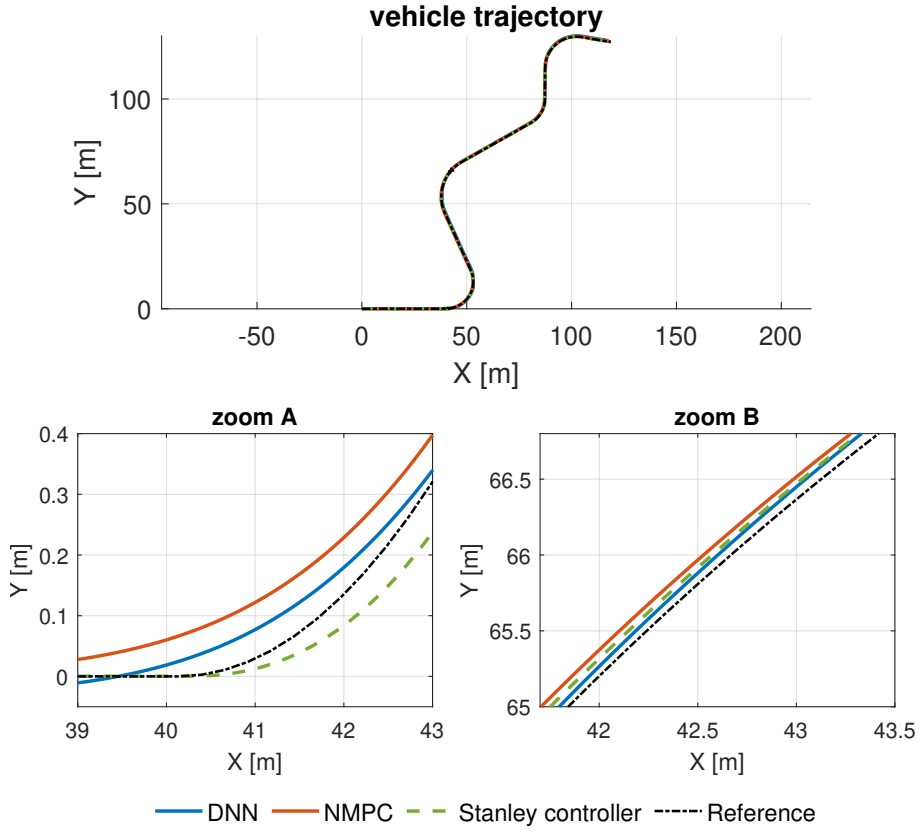

Figure 13: Circuit trajectories with two zoomed windows (A: entry, B: exit).

7.1 Trajectory geometry

Figure 13 overlays the paths with two zoom windows (**A**: first *left–hand* entry; **B**: *right–hand* exit). All controllers complete the lap and remain close to the centreline. The **DNN** clearly *learns the NMPC strategy*: it turns in before the curvature peak and follows a smooth centreline–referenced path on gentle curvature. In the bends, its behaviour is nuanced. In mid–corner the DNN can be slightly *tighter* than the teacher—see Zoom A, where its inside line sits closer to the reference than NMPC and Stanley. This is consistent with the bias of MSE regression with a bounded output layer (`tanh`+affine de-scaling): small, high-frequency corrections in the teacher labels are attenuated, yielding smoother actuation in steady cornering and, at times, a slightly tighter mid-corner line. Around the exits, the DNN reacts a bit slower: in Zoom B the *right–hand exit* shows a mild outward bias and a delayed return to the centreline, whereas **NMPC** settles

earlier thanks to horizon–based look–ahead. The **Stanley** trajectory exhibits piecewise–linear segments; although its global cross–track envelope can be small, the step–like local transitions align with larger heading/sideslip excursions (see the error and dynamics figures). Overall, the student reproduces the teacher's qualitative policy and is competitive in mid–corner geometry, with most of the gap concentrated at entry/exit transients.

7.2 Tracking errors e_{ct} and e_h

Figure 14 shows cross–track and heading errors. On straights and mild curvature the **DNN** matches the teacher; in long right–hand bends it develops a sustained positive e_{ct} and exhibits a light underdamped "S–shape" at curvature transitions, while **NMPC** returns to zero sooner. The **Stanley** controller achieves small cross–track error by construction but at the cost of larger heading error spikes; Table 8 reflects this trade–off.

Heading error e_h In the centreline frame, $\dot{e}_h \approx r - \kappa_{\text{ref}} V_x$. Steering (with 4WS+TV) changes yaw rate r almost immediately through tire lateral forces and yaw moment, so e_h has a short, well–conditioned control path. The teacher exploits look–ahead to schedule pre–alignment; the DNN inherits this behaviour and attains a close second in RMS e_h, indicating it has learned the teacher's fast heading regulation reasonably well.

Cross–track error e_{ct} Cross–track evolves as $\dot{e}_{ct} \approx V_x \sin(e_h) + V_y \approx V_x (e_h + \beta)$ for small angles, hence depends on the *integrated* effect of heading bias and sideslip. This makes e_{ct} slower and more sensitive to constraints and speed. On long straights where $|\kappa_{\text{ref}}| \approx 0$ and $|e_{ct}|$ is already small, the NMPC often issues only tiny "maintenance" tweaks (limited near-zero labels). Trained with MSE and a bounded output layer (`tanh`+affine de-scaling), the DNN tends to attenuate such micro-corrections and behaves more conservatively, so small e_{ct} can drift until curvature builds up again. This explains the larger straight–line e_{ct} visible in Figs. 13 and 14. The gap is concentrated in these regimes and suggests straightforward remedies (e.g., modest up-sampling of straight segments with persistent e_{ct}, or adding short temporal context) without affecting real-time inference.

7.3 Lateral dynamics: a_y and sideslip β

Figure 15 reports lateral acceleration and sideslip. With V_x fixed, $a_y \simeq V_x^2 \kappa_{\text{ref}}$, so all methods reach similar plateaus in the long corners (Table 9). Around curvature steps, **NMPC** ramps slightly earlier due to horizon look-ahead and settles with small over/undershoot, helped by its stage cost that explicitly penalizes input rates (diff–u, e.g., $\Delta d_{F/R}$, $\Delta T_{F/R}$) and, implicitly, lateral-jerk. [13]

The **DNN** tracks the same envelope but shows rounder transitions—i.e., a modest lag at ramp-up/ramp-down—and a faint ripple on the a_y plateaus. This is expected because the student is trained with pointwise MSE on *absolute* commands and does not inherit the teacher's diff–u penalty; small step-to-step variations therefore pass through the rate limiter (which rarely binds) and appear as minor a_y oscillations. The amplitude remains small and within bounds; incorporating a light smoothness term on $\Delta\mathbf{u}$ or a short temporal context is expected to reduce this effect without affecting real-time inference.

For sideslip β, both **DNN** and **NMPC** keep $|\beta|$ small throughout (peaks $\lesssim 1.2°$ in Table 9). At corner entries the DNN sometimes builds yaw rate r a touch later than a_y, yielding a brief opposite-sign bump; at exits it returns to zero slightly after NMPC. These effects are short and remain well damped. The **Stanley** baseline exhibits the largest spikes (2–4°) and more step-like changes in both

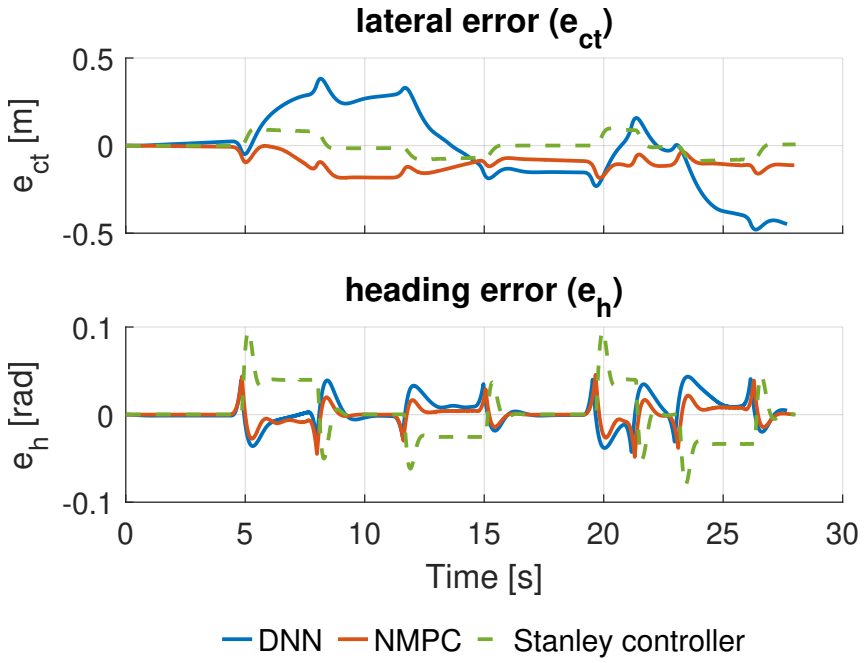

Figure 14: Tracking errors on the circuit: cross–track e_{ct} (top) and heading e_h.

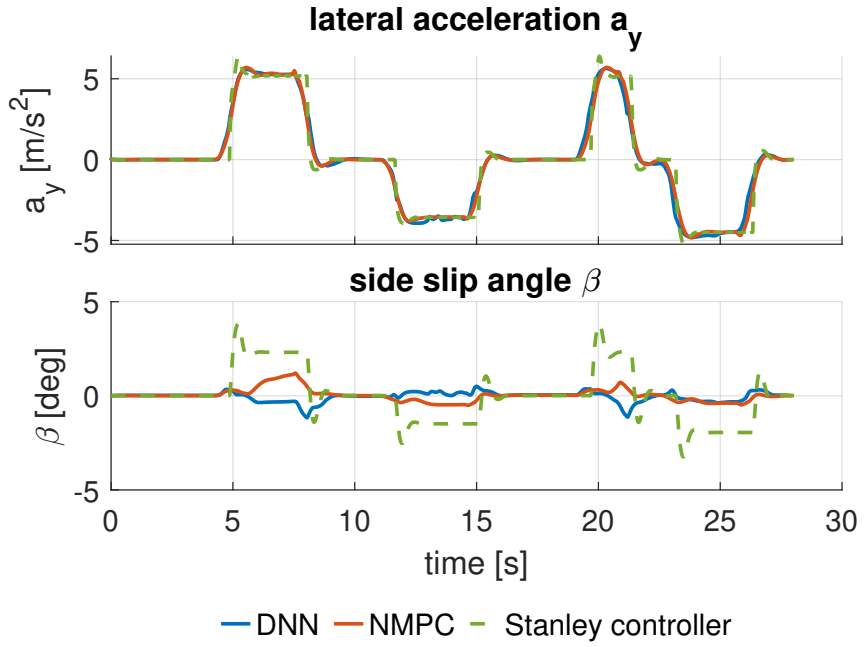

Figure 15: Lateral acceleration a_y (top) and sideslip angle β on the circuit.

a_y and β, indicating weaker damping of yaw/side-slip dynamics without rear-steer/yaw-moment authority. Overall, the student reproduces the teacher's dynamic behavior (smooth a_y tracking and small sideslip) with a small transient lag that is a clear target for future improvement (e.g., curvature-weighted loss or short temporal context).

7.4 Quantitative KPIs

Tables 8–10 report tracking, dynamics, and execution statistics on the held-out circuit.

Tracking (Table 8). The **NMPC** attains the best steady tracking: RMS $e_{ct} = 0.1064\,\mathrm{m}$ and RMS $e_h = 0.0105\,\mathrm{rad}$. The **DNN** preserves most of the teacher's behaviour but lags on cross–track (RMS $e_{ct} = 0.2144\,\mathrm{m}$), while remaining competitive on heading (RMS $e_h = 0.0162\,\mathrm{rad}$). The **Stanley** controller yields the smallest RMS e_{ct} by design, at the cost of markedly larger heading excursions (RMS $e_h = 0.0281\,\mathrm{rad}$).

Lateral dynamics (Table 9). With fixed V_x, all methods achieve similar a_y plateaus (RMS a_y within $\approx 0.04\,\mathrm{m/s^2}$ between DNN and NMPC). Sideslip remains small for both learning and MPC controllers; the **DNN** shows slightly lower β (Max 1.17°, RMS 0.27°) than **NMPC** (Max 1.21°, RMS 0.33°). **Stanley** exhibits the largest β spikes (Max 3.83°; RMS 1.38°), consistent with weaker transient damping.

Execution time (Table 10). The **DNN** runs in real time (Exec 27.75 s for a 27.68 s scenario; RTF ≈ 1.00), whereas **NMPC** is sub–real-time (Exec 179.22 s; RTF = 0.16). **Stanley** is trivially fast (RTF = 1.11) but trails in closed-loop quality.

Table 8: Tracking-error KPIs on the unseen circuit (Max and RMS).

Metric	DNN	NMPC (DSTE)	Stanley controller
Max e_{ct} [m]	0.4795	0.1850	**0.0982**
RMS e_{ct} [m]	0.2144	0.1064	**0.0521**
Max e_h [rad]	**0.0434**	0.0483	0.0953
RMS e_h [rad]	0.0162	**0.0105**	0.0281

Table 9: Dynamic-response KPIs on the unseen circuit.

Metric	DNN	NMPC (DSTE)	Stanley controller
Max a_y [m/s^2]	5.637	**5.703**	6.409
RMS a_y [m/s^2]	2.871	**2.831**	2.944
Max β [°]	**1.17**	1.21	3.83
RMS β [°]	**0.27**	0.33	1.38

Table 10: Execution statistics on the unseen circuit.

Metric	DNN	NMPC (DSTE)	Stanley controller
Exec time [s]	27.746	179.218	25.355
Total time [s]	36.287	180.725	27.551
Sim time [s]	27.679	27.976	28.032
RTF (sim/exec)	1.00	0.16	1.11

8 Conclusions and Outlook

We presented a supervised imitation pipeline that distills a high-performance NMPC (DSTE) policy for 4WS+TV path following into a compact residual MLP and validated it in closed loop.

Closed-loop performance

Across held–out evaluations, the student reproduces the teacher's qualitative strategy—anticipative turn–in, smooth centreline–referenced tracking, and well–damped transients. Heading regulation closely follows NMPC, sideslip remains small, and the lateral–acceleration envelope matches the kinematic target. The main residual gap is concentrated in sustained cross–track regulation on long/transition bends, where training supervision is relatively sparse.

Computation and real-time feasibility

The learned policy executes in real time on a laptop–class platform using single–precision inference inside a `Predict` block. In contrast, the NMPC solve remains sub–real–time on the same setup. Thus, when embedded compute budgets cannot sustain online NMPC, the DNN surrogate offers a pragmatic deployment path: it retains much of the NMPC behavior at an inference cost comparable to classical controllers and clearly outperforms them in closed–loop quality.

Limitations and remedies

The remaining gap is mainly due to two factors: (i) data imbalance toward low curvature with sparse coverage of corner entry/exit transitions; and (ii) a memoryless, pointwise loss that does not inherit the teacher's input-rate regularization. Practical fixes include curvature-balanced sampling, modest reweighting of near-straight segments that sustain nonzero e_{ct}, adding a very short temporal context or a previous-control tap, and a small penalty on $\Delta\mathbf{u}$—all compatible with real-time inference.

Two further limitations are open: feature sufficiency has not been quantified (some inputs may be redundant), and extreme scenarios (e.g., low-μ/split-μ, large-κ transitions, actuator saturation, gusts) were not included, so robustness there remains to be established.

Outlook

We will (a) run systematic feature ablations to identify a minimal, high-impact subset and co-design a lighter network (width/depth/heads) for tighter real-time margins; (b) explore light temporal/contextual cues (e.g., [prev u] or a 2–4 step micro-history) and small $\Delta\mathbf{u}$ regularization to close entry/exit transients; (c) expand training data with high-quality, balanced coverage of extreme scenarios (low-μ, split-μ, large-κ transitions, saturation events), using targeted rollouts and filtering; and (d) validate on embedded hardware and additional unseen tracks for generalization and robustness. The goal is NMPC-like behavior with the fewest features and the smallest network that still meets accuracy and stability targets under challenging conditions.

References

[1] F. S. Acerbo et al., "Evaluation of mpc-based imitation learning for human-like autonomous driving," in *IFAC World Congress*, 2023.

[2] N. H. Amer et al., "Integrated torque-vectoring ... preview," *Vehicle System Dynamics*, 2024.

[3] J. A. E. Andersson et al., "Casadi: A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[4] M. Asperti, M. Vignati, and E. Sabbioni, "On torque vectoring control: Review and comparison of state-of-the-art approaches," *Machines*, vol. 12, p. 160, Feb. 2024.

[5] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv:1607.06450*, 2016.

[6] D. Bernardini et al., "Drive-by-wire vehicle stabilization and yaw regulation: A hybrid model predictive control design," in *Proceedings of the 48th IEEE Conference on Decision and Control*, 2009.

[7] C. Chatzikomis, M. Zanchetta, and P. Gruber, "Energy-efficient tv allocation for multi-motor evs," *Mechanical Systems and Signal Processing*, 2019.

[8] P. Cichosz and Ł. Pawełczak, "Imitation learning of car driving skills with decision trees and random forests," *Int. J. Appl. Math. Comput. Sci.*, 2014.

[9] F. Codevilla et al., "End-to-end driving via conditional imitation learning," in *ICRA*, 2018.

[10] M. Dalboni, ., et al., "Nonlinear model predictive control for integrated energy-efficient torque-vectoring and anti-roll moment distribution," *IEEE/ASME Transactions on Mechatronics*, 2021.

[11] G. De Filippis, B. Lenzo, and A. Sorniotti, "Energy-efficient torque-vectoring for multi-motor evs," *IEEE Trans. Vehicular Technology*, 2018.

[12] S. Di Cairano et al., "Vehicle yaw stability control by coordinated active front steering and differential braking in the tire sideslip angles domain," *IEEE Transactions on Control Systems Technology*, 2013.

[13] P. Falcone et al., "A linear time-varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems," in *Proceedings of the 46th IEEE Conference on Decision and Control (CDC)*, 2007, pp. 2980–2985.

[14] P. Falcone et al., "A model predictive control approach for combined braking and steering in autonomous vehicles," in *MED 2007 - 15th Mediterranean Conference on Control and Automation*, 2007.

[15] A. I. J. Forrester, A. Sobester, and A. J. Keane, *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, 2008.

[16] G. Frison and M. Diehl, "Hpipm: A high-performance quadratic programming framework for model predictive control," in *IFAC-PapersOnLine*, vol. 53, 2020, pp. 6563–6569.

[17] G. Frison et al., "Blasfeo: Basic linear algebra subroutines for embedded optimization," *ACM Transactions on Mathematical Software*, vol. 44, no. 4, 42:1–42:30, 2018.

[18] A. R. Geist et al., "Data collection for robust end-to-end lateral vehicle control," in *ASME DSCC*, 2017.

[19] N. Guo et al., "A real-time nonlinear model predictive controller for yaw motion optimization of distributed drive electric vehicles," *IEEE Transactions on Vehicular Technology*, 2020.

[20] N. Guo et al., "A real-time nonlinear model predictive controller for yaw motion optimization of distributed drive electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 4935–4946, 2020.

[21] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[22] K. He, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] M. Hertneck et al., "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, 2018.

[24] G. M. Hoffmann et al., "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *Proceedings of the 2007 American Control Conference*, 2007.

[25] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration for nonlinear mpc," *Automatica*, 2011.

[26] *Iso 3888-1:2018 passenger cars—test track for a severe lane-change manoeuvre—part 1: Double lane-change*, International Organization for Standardization, 2018.

[27] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *arXiv:1806.10644*, 2018.

[28] S. Kim, K. Han, and S. B. Choi, "Imitation learning of nmpc for collision avoidance," *IEEE Trans. Intelligent Vehicles*, 2024.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[30] T. Lee and Y. Kang, "Dnn controller learning from nmpc," *Electronics*, 2021.

[31] M. Mazzilli et al., "Integrated chassis control: Classification, analysis and future trends," *Annual Reviews in Control*, 2021.

[32] T. Ohtsuka, "A continuation/gmres method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, no. 4, pp. 563–574, 2004.

[33] T. Osa et al., "An algorithmic perspective on imitation learning," *Foundations and Trends in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[34] A. Parra et al., "On nonlinear model predictive control for energy-efficient torque-vectoring," *IEEE Trans. Vehicular Technology*, 2021.

[35] A. Parra et al., "Pre-emptive stability control using curvature preview," *Vehicle System Dynamics*, 2022.

[36] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013.

[37] L. Prechelt, "Early stopping—but when?" In *Neural Networks: Tricks of the Trade*, Springer, 1998, pp. 55–69.

[38] R. Rajamani, *Vehicle Dynamics and Control* (Mechanical Engineering Series), 2nd ed. Springer, 2011.

[39] S. Ross, G. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.

[40] V. Skrickij et al., "Review of integrated chassis control techniques for automated ground vehicles," *Sensors (Basel, Switzerland)*, p. 600, 2024.

[41] J. M. Snider, "Automatic steering methods for autonomous automobile path tracking," Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-09-08, 2009.

[42] N. Srivastava et al., "Dropout: A simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, pp. 1929–1958, 2014.

[43] I. Tolstikhin et al., *Mlp-mixer: An all-mlp architecture for vision*, 2021.

[44] H. Touvron et al., "Resmlp: Feedforward networks for image classification with data-efficient training," *arXiv:2105.03404*, 2021.

[45] R. Verschueren and G. Frison, "Acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2022.

[46] R. Verschueren et al., "Acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, pp. 147–183, 2021.

[47] S. Yim, "Preview rollover prevention via yaw-moment and steering," *IEEE Trans. Vehicular Technology*, 2011.

[48] S. Yim, "V2v-based steering preview for stability control," *IEEE Trans. Intelligent Transportation Systems*, 2016.