Data-Driven Predictive Control for Autonomous Driving

Julius Beerwerth* Bassam Alrifaee*

Abstract: Data-driven predictive control (DPC) optimises a system's behaviour directly from measured input–output trajectories without relying on an explicit state-space model. This property is attractive for autonomous driving, where accurate models can be difficult to obtain across operating conditions. In practice, however, more data improves prediction quality at the cost of a larger optimisation problem, creating a tension between tracking accuracy and real-time feasibility. We complement the standard DPC formulation with a set of practical design choices that substantially affect numerical robustness and closed-loop performance in practice. We apply DPC to trajectory tracking on a scaled autonomous vehicle and investigate the trade-off between dataset size, tracking accuracy, and computational effort by varying the number of trajectories through random sampling. A model-based MPC controller using a known kinematic bicycle model serves as a baseline, providing context for the achievable tracking accuracy and computation time.

Keywords: Autonomous driving, data-driven control, predictive control, trajectory tracking

1 Introduction

Model predictive control (MPC) is widely used in robotics and autonomous driving because it optimises future behaviour while explicitly handling constraints [1]. Its practical performance, however, depends on the availability of an adequate model and suitable model parameters. Data-driven predictive control (DPC), also referred to as data-enabled predictive control (DeePC) [2], provides an alternative: predictions are built directly from recorded input–output trajectories organised in Hankel matrices. This avoids explicit model identification and integrates constraints into the receding-horizon optimisation. DPC builds on Willems' fundamental lemma [3], which establishes that, for a controllable linear time-invariant (LTI) system, any finite-length trajectory can be expressed as a linear combination of previously recorded trajectories, provided the input data are persistently exciting of sufficient order. Despite these benefits, applying DPC in real time remains challenging. For systems with nonlinearities, large and diverse datasets are typically required to obtain accurate predictions, but the optimisation problem size and computational effort increase with the number of available trajectory segments. Recent surveys discuss system-theoretic aspects and guarantees for data-driven predictive control [4, 5]. The original DeePC algorithm was proposed by Coulson et al. [2] and extended with regularisation and distributional robustness in [6]. For nonlinear systems,

*Department of Aerospace Engineering, University of the Bundeswehr Munich, Germany.

one research direction embeds nonlinear structure into data-driven prediction, e.g., via kernel methods or learned bases [7, 8, 9, 10]. Another direction exploits local linearity and updates or selects data to match the current operating regime [11, 12]. Generalised prediction frameworks that merge subspace and Hankel-based approaches have also been proposed [13]. Further practical investigations of DeePC on physical platforms, including quadcopter control, can be found in [14]. Data-driven MPC has also been applied to autonomous vehicle steering [15] and to connected vehicles in mixed traffic [16]. Independently of the particular nonlinear extension, a recurring practical question is how much data is required to obtain satisfactory closed-loop tracking performance at acceptable computation times. We address this question by performing a systematic empirical study on an autonomous driving platform based on our recent work on data selection for DPC [17]. We evaluate the standard DPC formulation under varying dataset sizes obtained through random sampling, while all controller parameters are kept fixed to isolate the effect of data availability. A model-based MPC controller using a kinematic bicycle model serves as a baseline throughout the experiments.

2 Preliminaries

2.1 Notation

We consider a discrete-time system with m inputs $u_k \in \mathbb{R}^m$ and p outputs $y_k \in \mathbb{R}^p$ at time step k. A trajectory of length T is a sequence $\{(u_k, y_k)\}_{k=0}^{T-1}$. For a signal $z = (z_0, z_1, \ldots, z_{T-1})$ and a window length $L \leq T$, the *Hankel matrix* of depth L is defined as

$$\mathcal{H}_L(z) \;=\; \begin{bmatrix} z_0 & z_1 & \cdots & z_{T-L} \\ z_1 & z_2 & \cdots & z_{T-L+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_{L-1} & z_L & \cdots & z_{T-1} \end{bmatrix}. \tag{1}$$

Each column of $\mathcal{H}_L(z)$ contains a contiguous sub-trajectory of length L. The matrix has $T - L + 1$ columns.

2.2 Willems' Fundamental Lemma

The theoretical foundation of DPC is the fundamental lemma of Willems et al. [3]. An input sequence $u = (u_0, \ldots, u_{T-1})$ is said to be *persistently exciting* of order L if the Hankel matrix $\mathcal{H}_L(u)$ has full row rank, i.e., $\operatorname{rank}(\mathcal{H}_L(u)) = mL$.

> **Fundamental Lemma** [3]**.** Consider a controllable LTI system with m inputs and p outputs, and lag ℓ. Let $(u^{\mathrm{d}}, y^{\mathrm{d}})$ be a measured input–output trajectory of length T such that u^{d} is persistently exciting of order $L + \ell$. Then, a sequence $(\bar{u}, \bar{y})$ of length L is a trajectory of the system if and only if there exists a vector $g \in \mathbb{R}^{T-L+1}$ such that
>
> $$\begin{bmatrix} \mathcal{H}_L(u^{\mathrm{d}}) \\ \mathcal{H}_L(y^{\mathrm{d}}) \end{bmatrix} g \;=\; \begin{bmatrix} \bar{u} \\ \bar{y} \end{bmatrix}. \tag{2}$$

In other words, any valid trajectory can be written as a linear combination of the columns of the Hankel matrices. The coefficient vector g parametrises the trajectory space non-parametrically—no explicit system matrices (A, B, C, D) are required.

3 Data-Driven Predictive Control

3.1 Hankel Matrix Construction

Given a dataset of input–output measurements $(u^{\mathrm{d}}, y^{\mathrm{d}})$ of length T, we form Hankel matrices of depth $L = T_{\mathrm{ini}} + N$, where T_{ini} is the length of the initial window (used to implicitly estimate the current state) and N is the prediction horizon. We partition the input and output Hankel matrices into *past* and *future* blocks:

$$\begin{bmatrix} U_{\mathrm{p}} \\ U_{\mathrm{f}} \end{bmatrix} = \mathcal{H}_{T_{\mathrm{ini}}+N}(u^{\mathrm{d}}), \qquad \begin{bmatrix} Y_{\mathrm{p}} \\ Y_{\mathrm{f}} \end{bmatrix} = \mathcal{H}_{T_{\mathrm{ini}}+N}(y^{\mathrm{d}}), \tag{3}$$

where $U_{\mathrm{p}} \in \mathbb{R}^{mT_{\mathrm{ini}} \times N_{\mathrm{col}}}$ contains the first T_{ini} block-rows of the input Hankel matrix and $U_{\mathrm{f}} \in \mathbb{R}^{mN \times N_{\mathrm{col}}}$ contains the remaining N block-rows. The output matrices Y_{p} and Y_{f} are partitioned analogously, and $N_{\mathrm{col}} = T - T_{\mathrm{ini}} - N + 1$ denotes the number of columns.

When the dataset consists of multiple shorter trajectories rather than a single long one, the Hankel matrices can be constructed by concatenating the individual Hankel matrices column-wise, consistent with recent extensions of the fundamental lemma to multiple experiments [18].

3.2 Optimisation Problem

At each control step k, DPC solves the following constrained optimisation problem:

$$\min_{g,\,\sigma_y} \quad \sum_{i=0}^{N-1} \left\| y_{k+i|k} - r_{k+i} \right\|_Q^2 + \sum_{i=0}^{N-1} \left\| u_{k+i|k} \right\|_R^2 + \lambda_g \left\| g \right\|_2^2 + \lambda_\sigma \left\| \sigma_y \right\|_1 \tag{4a}$$

$$\text{s.t.} \quad \begin{bmatrix} U_{\mathrm{p}} \\ Y_{\mathrm{p}} \end{bmatrix} g = \begin{bmatrix} u_{\mathrm{ini}} \\ y_{\mathrm{ini}} \end{bmatrix} + \begin{bmatrix} 0 \\ \sigma_y \end{bmatrix}, \tag{4b}$$

$$U_{\mathrm{f}}\, g = \bar{u}, \qquad Y_{\mathrm{f}}\, g = \bar{y}, \tag{4c}$$

$$u_{\min} \le u_{k+i|k} \le u_{\max}, \quad i = 0, \ldots, N-1, \tag{4d}$$

$$y_{\min} \le y_{k+i|k} \le y_{\max}, \quad i = 0, \ldots, N-1. \tag{4e}$$

where $\bar{u} = (u_{k|k}, \ldots, u_{k+N-1|k})$ and $\bar{y} = (y_{k|k}, \ldots, y_{k+N-1|k})$ denote the predicted future input and output trajectories, r_{k+i} is the reference at step $k+i$, and $Q \succeq 0$, $R \succeq 0$ are the output and input weighting matrices. The vectors u_{ini} and y_{ini} collect the most recent T_{ini} measured inputs and outputs, respectively.

Constraint (4b) enforces consistency with the measured initial conditions, and constraint (4c) defines the predicted future trajectory through the same coefficient vector g, ensuring that the predicted trajectory is a valid system trajectory according to the fundamental lemma.

3.3 Regularisation

Two regularisation terms appear in the cost (4a). The term $\lambda_g \|g\|_2^2$ penalises the norm of the coefficient vector, which tends to produce smoother and more robust predictions in the presence of noise or nonlinearities [6]. The slack variable σ_y softens the initial output constraint to accommodate measurement noise and model mismatch; without it, noise in y_{ini} can render the problem infeasible. A 1-norm penalty $\lambda_\sigma \|\sigma_y\|_1$ is preferred over a 2-norm, as it satisfies the exact penalty property: above a finite threshold on λ_σ, the slack is driven to exactly zero whenever the original hard-constrained problem is feasible [19].

The resulting optimisation (4) is a convex QP and can be solved efficiently with standard QP solvers.

3.4 Receding-Horizon Implementation

Following the receding-horizon principle, only the first element of the optimal input sequence $u^\star_{k|k}$ is applied to the system. At the next time step, the initial window is shifted by one sample, incorporating the latest measurements, and the optimisation is solved again. This feedback mechanism enables DPC to reject disturbances and adapt to changing conditions, analogous to standard MPC.

4 Dataset Sizing and Implementation

4.1 Random Sampling to Vary Dataset Size

Let the full Hankel matrices contain N_{col} columns. To study the effect of dataset size, we construct reduced Hankel matrices by selecting a subset of $N_{\text{s}} \ll N_{\text{col}}$ columns uniformly at random. The DPC optimisation problem (4) is then solved using only the reduced matrices. This random sampling provides a simple mechanism to control the number of decision variables: the dimension of g reduces from N_{col} to N_{s}, directly affecting both the QP size and the solve time.

4.2 Practical Design Choices

Several preprocessing steps improve numerical conditioning and practical performance.

Normalisation. Inputs and outputs are normalised using z-score statistics (mean and standard deviation) computed from the offline dataset. The same statistics are reused during online operation. This ensures that all signals have comparable scales, which benefits both the QP solver and the relative weighting of the cost terms.

Local reference frame. Recorded trajectories are expressed in a local reference frame by shifting the final pose of the initial output window to a common origin $(x, y, \psi) = (0, 0, 0)$. This reduces variability due to absolute position and heading, making trajectory segments more interchangeable across different regions of the track.

Incremental inputs. Control inputs can be expressed as increments $\Delta u_k = u_k - u_{k-1}$ relative to the previous control action. This formulation naturally penalises abrupt changes and can improve local linearity of the input–output mapping.

Output augmentation. Outputs may be augmented with the current control inputs to retain control-affine structure in the lifted input–output behaviour.

Regularisation scaling. When changing N_s, the regularisation weight on g is scaled proportionally to the number of selected columns, i.e., $\lambda_g(N_\mathrm{s}) = \lambda_g^0 \cdot N_\mathrm{s}/N_\mathrm{col}$, where λ_g^0 is the base weight. This keeps the regularisation effect comparable across dataset sizes.

5 Model Predictive Control Baseline

To provide a meaningful comparison, we implement a model-based MPC controller using a kinematic bicycle model,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos(\psi) \\ v\sin(\psi) \\ \frac{v}{l}\tan(\delta) \\ a \end{bmatrix}, \tag{5}$$

where (x, y) is the vehicle position, ψ is the heading angle, v is the longitudinal velocity, l is the wheelbase, δ is the steering angle, and a is the longitudinal acceleration. The model is discretised at the same sampling time T_s used for DPC and linearised around the current operating point at each time step. The MPC controller solves a QP of the same structure as (4a)–(4e), but replaces the Hankel-based prediction with the linearised state-space model. The same cost matrices Q and R, prediction horizon N, and constraints are used so that any performance differences reflect the prediction method rather than tuning choices.

6 Case Study: Scaled Autonomous Vehicle

6.1 Experimental Setup

We evaluate the DPC and MPC controllers in a simulated autonomous driving scenario using the RoboRacer (formerly known as F1TENTH) autonomous racing framework [20]. The simulator uses a single-track dynamic bicycle model from the CommonRoad framework [21], which includes tyre slip and yaw rate dynamics. The kinematic model used by MPC neglects these effects, introducing a model-plant mismatch that becomes more pronounced at higher speeds and cornering forces — an effect that DPC can compensate for implicitly through the richness of the recorded trajectories.

An offline dataset of approximately 300 s of input–output measurements is collected using persistently exciting inputs. The inputs are the steering angle δ and longitudinal acceleration a ($m = 2$); the outputs are the position (x, y), heading ψ, and velocity v ($p = 4$). The controller is then evaluated on a fixed reference trajectory. To reflect realistic sensing conditions, Gaussian measurement noise with standard deviations $\sigma_{x,y} = 0.05\,\mathrm{m}$, $\sigma_v = 0.05\,\mathrm{m\,s^{-1}}$, and $\sigma_\psi = 0.01\,\mathrm{rad}$ is added to all outputs throughout.

6.2 Controller Parameters

Table 1 lists the controller parameters used in the experiments. The same parameter values are used for all DPC runs and the MPC baseline.

Table 1: Controller parameters for the vehicle case study.

Parameter	Value
Sampling time T_s	0.1 s
Initial window T_ini	5
Prediction horizon N	10
Output weighting Q	$\mathrm{diag}(1, 1, 0.1, 0.1)$
Input weighting R	$\mathrm{diag}(0.1, 1)$
Dataset length	300 s
Total Hankel columns N_col	2987

6.3 Evaluation Setup

To illustrate the effect of dataset size, we vary the number of randomly selected trajectory segments $N_\mathrm{s} \in \{30, 60, 90, 120\}$ while keeping all remaining controller parameters unchanged. The no-sampling case uses all $N_\mathrm{col} = 2987$ columns available in the collected dataset. For each N_s, five independent runs with different random seeds are performed and the results are aggregated to reduce variance induced by sampling.

The MPC controller is evaluated on the same reference trajectory under the same conditions, providing a model-based baseline.

We report two performance metrics: (i) the weighted closed-loop tracking error $\|y_k - r_k\|_Q$, summarised as box plots across all time steps and seeds, and (ii) the per-step computation time, characterised by the 99th percentile (p99) and maximum values.

7 Results and Discussion

7.1 Tracking Performance

Figure 1 shows the distribution of the weighted tracking error for each dataset size and for the MPC baseline. When N_s is small ($N_\mathrm{s} = 30$), the tracking error is large and exhibits high variance: the limited number of trajectory segments does not sufficiently cover the operating regime, leading to inaccurate predictions and poor closed-loop performance. As N_s increases to 60 and 90, both the median and the spread of the tracking error decrease substantially, reflecting improved prediction quality from a more representative dataset. Further increasing N_s to 120 yields only marginal additional improvement, indicating diminishing returns. Using all 2987 columns (no sampling) achieves the lowest median tracking error and the smallest variance.

The MPC baseline achieves a median tracking error comparable to DPC with intermediate dataset sizes, but is outperformed by DPC with larger datasets. This can be

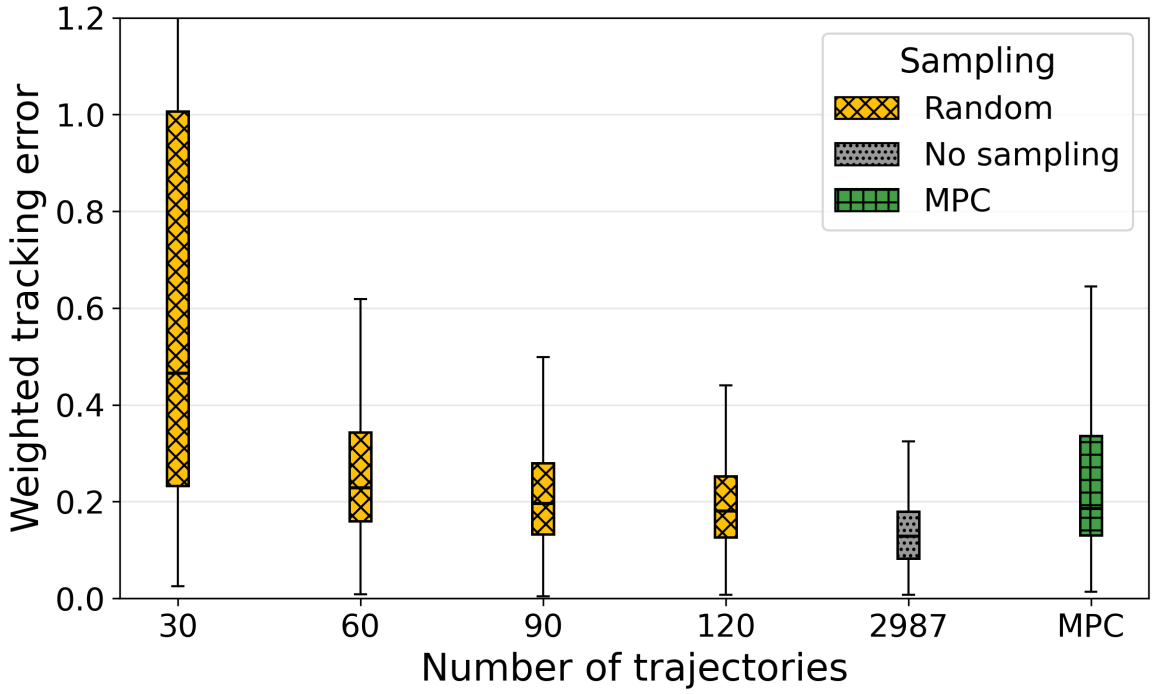

Figure 1: Weighted closed-loop tracking error distributions across different numbers of trajectories N_s, the full-data DPC (2987 columns), and the model-based MPC baseline.

attributed to the model-plant mismatch between the kinematic bicycle model and the dynamic single-track simulator: at higher speeds and cornering forces, linearisation errors accumulate, whereas DPC implicitly captures these nonlinear effects through the richness of the recorded trajectories.

7.2 Computation Time

Table 2 summarises the per-step computation time for each configuration. As expected, the computation time increases with N_s because the QP grows in dimension. For $N_s \leq 90$, the p99 computation time remains below 13 ms, well within a 100 ms sampling interval. At $N_s = 120$, the p99 computation time increases substantially to 48 ms, and the maximum per-step time exceeds 200 ms. Using all 2987 columns leads to a p99 of 193 ms and a maximum of 538 ms, which would violate real-time constraints in many practical deployments.

The MPC baseline achieves the lowest computation times (p99 of 1.92 ms, maximum of 26.58 ms), which is expected because the QP dimension depends only on the fixed horizon length N and not on the dataset size.

Notably, the p99 computation time does not increase linearly with N_s between 90 and 120, whereas the maximum solve time exhibits a clear upward trend. This suggests that occasional solver difficulties (e.g., due to near-singular Hankel sub-matrices) become more pronounced at larger problem sizes, even if the typical solve time remains manageable.

Table 2: Per-step computation time percentiles (milliseconds).

	Computation time	
N_s	p99 (ms)	Max (ms)
30	8.33	66.20
60	10.66	67.22
90	12.24	68.07
120	48.00	213.38
No sampling	192.67	538.37
MPC	1.92	26.58

7.3 Trade-off Analysis

The results reveal a clear trade-off between tracking accuracy, computational cost, and robustness to random seed selection. For intermediate values of N_s (60–90), the DPC controller achieves substantial tracking improvements over the smallest dataset while keeping computation times well within the sampling interval. Further increases beyond $N_s = 120$ yield diminishing accuracy gains at disproportionately higher computational cost.

Compared to MPC, DPC with $N_s \geq 90$ achieves competitive or even superior median tracking accuracy without requiring an explicit model. The MPC controller remains advantageous in computation time, making it preferable when a sufficiently accurate model is available and real-time constraints are tight. However, when model identification is difficult—for example, due to complex tyre dynamics, varying road surfaces, or coupling effects in multi-vehicle scenarios—DPC offers a viable alternative that trades computational overhead for modelling simplicity.

8 Conclusion

Data-driven predictive control was applied to trajectory tracking on a scaled autonomous vehicle platform, demonstrating that competitive closed-loop performance can be achieved without an explicit system model.

The empirical study confirmed the expected trade-off: increasing the number of trajectory segments improves tracking performance but raises computational cost. For intermediate dataset sizes, the DPC controller achieved tracking accuracy comparable to a model-based MPC baseline that has access to the true system model, while maintaining real-time-feasible computation times. The MPC baseline provided the fastest solve times but relies on an accurate model, which may not always be available.

Future work includes structured dataset management strategies, such as coverage-aware or relevance-based sampling, to improve the accuracy–computation trade-off beyond random selection. Understanding the interplay between prediction accuracy, robustness, and real-time feasibility will be crucial for scalable deployment of DPC on resource-constrained autonomous platforms. Efficient data selection thus remains a key challenge for practical DPC deployment.

References

[1] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing, 2017.

[2] J. Coulson, J. Lygeros, and F. Dörfler, "Data-enabled predictive control: In the shallows of the DeePC," in *Proc. European Control Conference*, Naples, Italy, 2019.

[3] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. M. De Moor, "A note on persistency of excitation," *Systems & Control Letters*, vol. 54, no. 4, 2005.

[4] J. Berberich and F. Allgöwer, "An overview of systems-theoretic guarantees in data-driven model predictive control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 8, 2025.

[5] I. Markovsky, L. Huang, and F. Dörfler, "Data-driven control based on the behavioral approach: From theory to applications in power systems," *IEEE Control Systems*, vol. 43, no. 5, 2023.

[6] J. Coulson, J. Lygeros, and F. Dörfler, "Regularized and distributionally robust data-enabled predictive control," in *Proc. IEEE Conference on Decision and Control*, Nice, France, 2019.

[7] L. Huang, J. Lygeros, and F. Dörfler, "Robust and kernelized data-enabled predictive control for nonlinear systems," *IEEE Transactions on Control Systems Technology*, vol. 32, no. 2, 2024.

[8] M. Lazar, "Basis-functions nonlinear data-enabled predictive control: Consistent and computationally efficient formulations," in *Proc. European Control Conference*, Stockholm, Sweden, 2024.

[9] M. Lazar, "Neural data-enabled predictive control," *IFAC PapersOnLine*, vol. 58, no. 15, 2024.

[10] Y. Lian and C. N. Jones, "Nonlinear data-enabled prediction and control," *Proceedings of Machine Learning Research*, vol. 144, 2021.

[11] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Linear tracking MPC for nonlinear systems—Part II: The data-driven case," *IEEE Transactions on Automatic Control*, vol. 67, no. 9, 2022.

[12] M. Alsalti, V. G. Lopez, J. Berberich, F. Allgöwer, and M. A. Müller, "Data-driven nonlinear predictive control for feedback linearizable systems," *IFAC PapersOnLine*, vol. 56, no. 2, 2023.

[13] M. Lazar and P. C. N. Verheijen, "Generalized data-driven predictive control: Merging subspace and Hankel predictors," *Mathematics*, vol. 11, no. 9, 2023.

[14] E. Elokda, J. Coulson, P. N. Beuchat, J. Lygeros, and F. Dörfler, "Data-enabled predictive control for quadcopters," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, 2021.

[15] J. Zhang, A. Kong, Y. Tang, Z. Lv, L. Guo, and P. Hang, "Application of data-driven model predictive control for autonomous vehicle steering," *arXiv preprint arXiv:2407.08401*, 2024.

[16] J. Wang, Y. Zheng, Q. Xu, and K. Li, "Data-driven predictive control for connected and autonomous vehicles in mixed traffic," in *Proc. American Control Conference*, Atlanta, GA, 2022.

[17] J. Beerwerth and B. Alrifaee, "Less is more: Contextual sampling for nonlinear data-enabled predictive control," accepted to *European Control Conference (ECC)*, 2026.

[18] H. J. van Waarde, C. De Persis, M. K. Camlibel, and P. Tesi, "Willems' fundamental lemma for state-space systems and its extension to multiple datasets," *IEEE Control Systems Letters*, vol. 4, no. 3, 2020.

[19] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[20] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1TENTH: An open-source evaluation environment for continuous control and reinforcement learning," *Proceedings of Machine Learning Research*, vol. 123, 2020.

[21] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. IEEE Intelligent Vehicles Symposium*, Los Angeles, CA, 2017.