Learning Based NMPC Adaptation for Autonomous Driving using Parallelized Digital Twin

Jean Pierre Allamaa [1,2], Panagiotis Patrinos [2], Herman Van der Auweraer [1,3], Tong Duy Son [1]

***Abstract*—In this work, we focus on the challenge of transferring an autonomous driving controller from simulation to the real world (i.e. Sim2Real). We propose a data-efficient method for online and on-the-fly adaptation of parametrizable control architectures such that the target closed-loop performance is optimized while accounting for uncertainties as model mismatches, changes in the environment, and task variations. The novelty of the approach resides in leveraging black-box optimization enabled by Executable Digital Twins (xDTs) for data-driven parameter calibration through derivative-free methods to directly adapt the controller in real-time. The xDTs are augmented with Domain Randomization (DR) for robustness and allow for safe parameter exploration. The proposed method requires a minimal amount of interaction with the real-world as it pushes the exploration towards the xDTs. We validate our approach through real-world experiments, demonstrating its effectiveness in transferring and fine-tuning a Nonlinear Model Predictive Control (NMPC) with 9 parameters, in under 10 minutes. This eliminates the need for hours-long manual tuning and lengthy machine learning training and data collection phases. Our results show that the online adapted NMPC directly compensates for the Sim2Real gap and avoids overtuning in simulation. Importantly, a 75% improvement in tracking performance is achieved and the Sim2Real gap over the target performance is reduced from a factor of 876 to 1.033.**

I. Introduction

PERFORMANCE of advanced control strategies often depends on the choice of (hyper)parameters. A powerful control strategy, such as Nonlinear Model Predictive Control (NMPC), is challenged in the context of verification and validation because many parameters need to be tuned simultaneously. This tuning process requires engineering knowledge and is often difficult to scale to new applications, new environments, and new tasks. Although heuristics and methods exist for tuning some controllers, researchers often face a second barrier when proceeding to deployment, that of the simulation to reality transfer (Sim2Real). Sim2Real is a commonly faced challenge as controllers tuned in one domain (simulation) fail to transfer to a target domain (real world), and result in deceiving performance as in Figure 1 due to domain shifts, noise and uncertainties. This raises the need to close the loop between prior expectation from simulation, with actual feedback from the real world, in the form of online adaptation to encapsulate the true uncertainties and Sim2Real gap. To tackle these challenges, a strategy with an adaptation layer learns to overcome these errors without overestimating them, by directly optimizing the target domain performance. This contrasts with traditional approaches in the automotive industry that rely on a one directional V-cycle for verification and validation. The standard V-cycle goes from Model-in-the-Loop (MiL) to Hardware-in-the-Loop (HiL) and finally to Vehicle-in-the-Loop (ViL), in a one-directional X-in-the-Loop (XiL). During ViL, the performance of the full closed-loop system is determined, and the engineer is faced with two options: tedious online calibration for end-of-line tuning or iterating back to the MiL phase. Therefore, we aim to bridge the gap between simulation and reality by enabling a bidirectional communication between them so that the simulation operates in parallel with the real counterpart. This allows for safe and cheap exploration of the parameter space in simulation, which can be exploited by the real counterpart to calibrate for the optimal parameters.

Several research directions exist, focusing on including learning in the control strategy. Reinforcement and imitation learning are two examples of those strategies used to either train a control policy from scratch, which can be unsafe for safety-critical systems such as in autonomous vehicles, or to tune a predefined safe controller such as NMPC. Although these methods showed great success and are scalable as in [1], they are often data hungry, require long training time and assume that numerical gradient methods could be employed given differentiability between the output and the parameters. Another direction for incorporating learning and adaptation in the NMPC is through online learning. In [2], the authors propose to optimize over the weighting matrices on the slack variables of the MPC's terminal constraints to improve robustness in specified critical zones. The authors of [3] propose a parameter tuning approach based on contextual Bayesian Optimization (BO). In the work of [4], a novel BO approach is used for automatic control tuning by offloading exploration towards the Digital Twin (DT). Moreover, a popular method for adapting the NMPC model is through residual dynamics learning by using neural networks [5] or Gaussian Processes [6]. Adaptation to reduce conservatism compared to robust control as in [7], and safe environment learning [8], are also relevant. However, most of these methods affect the real-time applicability and structure of the NMPC or require a large amount of data to train offline. Notably, some of those methods like vanilla BO are not suitable for online training as parameter adaptation requires exploring the parameter space on the target system. Furthermore, data-driven parameter tuning in the form

[1] Siemens Digital Industries Software, 3001 Leuven, Belgium. Email: {jean.pierre.allamaa, herman.van-der-auweraer.ext@siemens.com, son.tong}@siemens.com

[2] Dept. Electr. Eng. (ESAT) - STADIUS research group, KU Leuven, 3001 Leuven, Belgium. Email: panos.patrinos@esat.kuleuven.be

[3] Dept. Mechanical Eng. - LMSD research group, KU Leuven, 3001 Leuven, Belgium

of classical adaptive control has shown promising results [9] and was applied to tuning MPC [10]. However, most of these approaches are applied solely to linearized systems and for simple tracking problems. Therefore, they are limited in their scalability and region of validity due to potential oversimplification. Although several previous works have tackled the parameter learning aspect, most of the results were either limited to simulation or controlled lab environment.

This paper extends the previous work of [11] by including experimental validation of the automatic tuning for a Real-Time NMPC (RTNMPC) on a real road vehicle, alongside parallelization of DTs through an industrial standard known as FMU, and providing necessary convergence conditions for the algorithm.

To address the previously identified gaps, we present Data-Driven Controller Calibration with Adaptive Unscented Kalman filter and SPSA (D^2C^2-AUKS): a method that offers a fast and on-the-fly automatic controller adaptation. It employs gradient-free methods for stochastic optimization that enable learning the controller's optimal parameters in a limited amount of time with sample and data efficiency. Notably, the method avoids trial-and-error in the parameter search and allows parallelization unlike BO, rendering it efficient for controller tuning on real-systems. A key aspect of the work is to optimize over possibly non-differentiable and non-analytical performance measures by means of parameters estimation using Unscented Kalman Filter (UKF) as first presented in [12]. Unlike [12], we include Domain Randomization (DR) for robustness and more accurate parameter distribution, a sampling-based gradient estimation known as Simultaneous Perturbation Stochastic Approximation (SPSA) [13] to speed up the tuning process, and offload online exploration towards Executable Digital Twins (xDTs). Moreover, we extend the method in [11] further by carefully adapting the noise covariance matrices to provide sufficient conditions for convergence and apply the entire framework to a real vehicle[1]. As the method optimizes for the closed-loop performance without changing the structure of the NMPC, it is suitable for real-time applications. In the adaptation framework, we exploit the xDTs to explore different parameter combinations efficiently and rapidly, in a digital yet highly reliable environment. This permits to offload the exploration burden towards the xDTs, eliminating the need for costly, unsafe and suboptimal manual tuning. The main contributions are:

- a fast and safe online adaptation framework that requires little interaction with the environment and that is able to directly compensate for uncertainties;
- A parallelization approach for xDTs in the loop;
- Parameter tuning through stochastic and black-box optimization that is robust to noise through adaptive Kalman filter covariance matrices and SPSA;
- Experimental validation of the method on a real-world Autonomous Driving (AD) with a RTNMPC for end-of-line tuning, resulting in 75% tracking performance improvement within few iterations and a drop in the Sim2Real gap from a factor of 876 to 1.033.

[1]Abstract and experiments video at https://youtu.be/62PgNHciIJA.

The paper is structured as follows: Section II provides a background on relevant Sim2Real techniques and introduces the proposed parallelization of multiple xDTs. Section III presents the learning and adaptation framework through Executable Digital Twin (xDT) and Derivative-Free Optimization (DFO). Furthermore, section IV validates the method in simulation for an Autonomous Valet Parking (AVP) application and shows the benefit of DR. Moreover, we present and discuss the results of the experimental validation of the automatic NMPC calibration on the road vehicle in section V. Finally, the limitations of the work are presented in section VI before concluding the paper in section VII.

II. Preliminary on Sim2Real Techniques

Manual tuning is time-consuming, costly, and can lead to performance discrepancies among different products. Moreover, transferring controllers from simulation to the real-world is challenging due to changing environments and a domain shift known as Sim2Real gap. Automatic tuning and continuous data-driven adaptation is a promising solution to these issues as in [4], [14]. In this paper, we propose a black-box optimization-driven adaptation framework specifically designed to find suitable control parameters for the NMPC by deploying xDTs in parallel with the real vehicle as depicted in Figure 2. We first provide an overview of existing Sim2Real strategies and then introduce the concept of xDTs and highlight their parallelization capability through a standalone library instance.

A. Sim2Real through Domain randomization and adaptation

We address the challenge of transferring a controller designed in a source domain to a target domain. This concerns for example the transfer of a learned control policy from a training plant to a testing plant with different parameters. Specifically, we tackle the Sim2Real transfer which suffers from Sim2Real gap curse. Tuning parameters solely in the source domain poses a significant risk of ineffective transfer due to domain or model mismatches, uncertainties, dynamic environmental changes, and lack of robustness. Therefore, additional techniques to recover or limit the performance loss in the target domain are necessary. Two common methods to address this issue are DR and Domain Adaptation (DA). DR involves injecting uncertainties into the train domain by introducing noise and model mismatches (for e.g. mass, inertia, friction coefficients, varying tasks) to enhance the robustness of the learned policy against potential disturbances. DR has been widely used in the field of robotics and autonomy [15], [16]. On the other hand, DA involves training in a latent space using a domain invariant feature representation. An example of learning in higher-dimensional spaces by automatic domain adaptation to facilitate transfer is presented in [17]. A survey on Sim2Real techniques can be found in [18].

B. xDT and FMU parallelization

Simulations play a vital role in control applications in the development and validation processes. As safety concerns

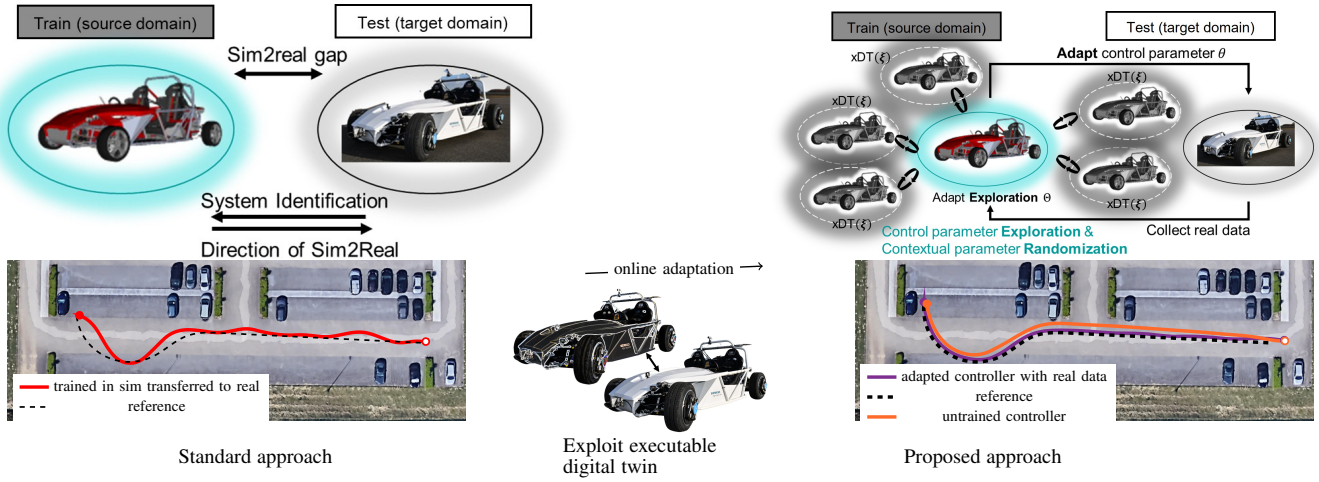

Fig. 1: Vehicle-in-the-loop performance. Left: control policy over-trained in simulation with only the nominal identified vehicle model to achieve less than 15 cm of accuracy, fails to transfer to the real-world and results in an unsafe driving style. Right: Enhancing the Sim2Real transfer through data-driven controller adaptation based on domain randomization, domain adaptation, high-fidelity simulation and with fine-tuning from real-world data. The controller is adapted in the real-world within 10 minutes.

and cost per simulation are minimal, researchers can explore various scenarios and environments using a MiL approach. When the algorithms are safe and robust, they are transitioned to HiL and ViL. The standard flow consists of first identifying an accurate high-fidelity model, then using it to train or validate the policies in closed-loop as depicted in the left framework in Figure 1. Although successful in many applications, this standard approach fails to transfer between domains and requires fine-tuning inspired by engineering expertise. Particularly, performance degradation occurs due to the simulation-optimization bias [19]. With the emergence of automated vehicles, this involves a considerable end-of-line tuning to recover the lost performance because of the Sim2Real gap. To address this, the concept xDTs has been proposed [20]: a high-fidelity twin model of the plant that can be instantiated to run in parallel with its real counterpart, on embedded hardware. The xDT captures the complex dynamics, and allows the exchange of data between the digital and real twins. Moreover, it is also possible to instantiate not one, but several xDTs to allow for real-time testing of multiple configurations while safely and efficiently exploring in the digital world. This concept has been further elaborated as a Twin-in-the-Loop (TiL) in [14] to automatically tune a compensator for the Sim2Real gap using BO, showing the benefit of employing a DT on-the-go for better prediction.

In this work, we propose adapting the parametrizable AD controller by sampling multiple xDTs built with Simcenter Amesim that run in parallel as in Figures 2 and 3. The prior knowledge captured by the xDTs in terms of the closed-loop controller performance is updated through additional data from real-world feedback, closing by this the Sim2Real gap. The xDTs are instantiated in the form of a Functional Mock-up Unit (FMU) for co-simulation, each with its own integration solver. FMUs follow an industry standard to interface and exchange dynamic simulation models. By setting the xDT as an FMU, it is possible to create many instances of the plant model each with its own model parameters. Thus, we are capable of parallelizing over the CPU or GPU, several rollouts of the controller in the loop with an xDT, namely a simulated oracle. For this work, we parallelize over the CPU. That is every oracle runs on a standalone core, permitting simultaneous exploration in the parameter space. For a reasonable number of control parameters, the parallelization results in significant CPU time reduction, allowing the real-time application. We utilize OpenMP, a multithreading implementation in C++ that allows shared-memory multiprocessing [21].

Unlike the TiL work of [14] which focuses on tuning an additional compensator controller such that a certain DT output resembles the real counterpart, we consider the direct performance optimization without explicit quantification of the Sim2Real gap. We address the Sim2Real gap between the DT and the real counterpart by means of DR and by accounting for the uncertainty on the Sim2Real gap within the parameter optimization. Our objective is to 1) extend the approach to multiple TiL for a better control parameter exploration, and 2) minimize the effect of Sim2Real gap by considering multiple xDTs with different contextual parameters to robustify the controller against possible real-world realizations. Moreover, the usage of vanilla BO is sample inefficient for a large number of parameters. Often with BO, tuning is either done purely in simulation then transferred to the real-world, or directly carried in the real-world in the form of online learning. That is, the target system is employed for explorative purposes as the parameters vary. We aim to combine a sensitivity guided active exploration on the xDTs with the exploitation on the real system, ensuring that the real system lies within the distribution of the simulated performances.

By executing a twin of the NMPC (cf. (3)) with carefully selected control parameters on multiple independent and parallel xDTs, we evaluate the distribution of the performance and its sensitivity with respect to the parameters. The performance of the estimated optimal parameters is evaluated in the high-

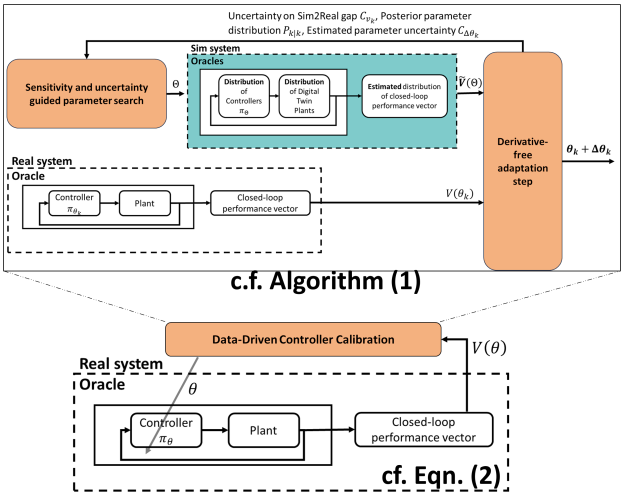

Fig. 2: D^{2}C^2-AUKS: on-the-go parametrizable controller calibration that offloads the burden of parameter search and sensitivity analysis towards multiple Digital Twins, and efficiently closes the Sim2Real gap through real-world data flow

level problem in (2) with real measurements. This process aims to automatically improve the real-world performance. Experimental results in Section V demonstrate that the framework significantly enhances performance in just few online runs with the real vehicle.

III. Learning and Adaptation Framework

The goal of the proposed work is to optimize the closed-loop performance in the target domain, while avoiding sampling the different parameter variations directly in this domain, but rather offload it to the xDT source domain. The xDT environment in this case is composed of both the vehicle model and the AD module to be tuned, namely the NMPC. This simulated oracle mimics the target closed-loop settings, i.e. the real system oracle. The resultant framework should be able to directly deal with external uncertainties ξ that affect the performance, by compensating for their effect without estimating them. On a high level, we seek to minimize the norm of the expected overall performance measure V. This is done by estimating the optimal parameters θ, after receiving data feedback from the real oracle measurements over a time window $t_0, \ldots, t_0 + T$. In this section, we first present the black-box optimization formulation for automatic controller calibration. Furthermore, we introduce the approach to push exploration towards xDTs and combine it with real-word data for safe and efficient automatic parameter calibration as in Figure 3.

Remark 1: We refer to safety as two components. First, the burden of simultaneous parameter exploration is offloaded towards simulation. This allows to explore and detect possibly unsafe regions, and deduce the sensitivity of the performance with respect to the parameters without an online learning phase. Second, as the structure of the underlying RTNMPC does not change, we consider safety as the autonomous system is driven by an underlying safe and deterministic controller. This comes in comparison with learning-based controllers that often rely on neural networks for policy generation. Therefore, NMPC acts as a safety filter for the adaptation framework.

A. Black-box optimization formulation

The target system, for which the controller is to be tuned, evolves according to the unknown dynamics f, while its xDT evolves according to f_{DT}. The map $\mathcal{F}$ defines the system evolution from time t_0 to t_0+T. The function $\mathcal{H}_i$ is an output transformation map for a specific output performance Y_i out of n_O output performance vectors. Additionally, W and O are the process and output noise. We define a closed-loop oracle:

$$X_{t_0:t_0+T} = \mathcal{F}(X_{t_0}, U_{t_0:t_0+T}(\theta_k), W_{t_0:t_0+T}), \tag{1a}$$

$$Y_{i,t_0:t_0+T} = \mathcal{H}_i(X_{t_0:t_0+T}, U_{t_0:t_0+T}(\theta_k)) + O_{t_0:t_0+T}, \tag{1b}$$

$$V(\theta_k) = \begin{pmatrix} Y_{1,t_0:t_0+T} - Y_1^r \\ \vdots \\ Y_{n_O,t_0:t_0+T} - Y_{n_O}^r \end{pmatrix}. \tag{1c}$$

The equations in (1) define a black-box oracle, or a rollout of the autonomous system with the controller tuned with the parameters θ. The first equation describes the evolution of the internal states X from t_0 to $t_0 + T$, based on the input trajectory $U(\theta)$ and under the effect of noise W. The input $U(\theta)$ results from the parametrizable controller. The second equation calculates the n_O output trajectories Y_i, using the measured or monitored signals. The third equation represents the vector $V(\theta_k)$ which stacks the errors of the individual output performances with respect to their desired or reference values Y_i^r. We consider the window $[t_0, t_0 + T]$ to contain N_T discrete elements such that $N_T = T/T_s$ where T_s is the update period of the measurements and control action. Importantly, in the case of the real system, the true $V(\theta_k)$ is captured as the system evolves with the dynamics $\mathcal{F} = f$ as in Steps 1.Real and 2.Real in Figure 3. For the xDTs, the approximate performance measure $\tilde{V}_\xi(\theta)$ is calculated and measured similarly to the real system, but with a system evolving based on $\mathcal{F} = f_{DT}$, as in Steps 1.DT and 2.DT of Figure 3. The nominal dynamics f_{DT} captured within the xDT are highly accurate in evaluating the local behavior of the cost with respect to different parameter combination, but may not perfectly capture the real system dynamics.

Furthermore, we consider three levels of system dynamics with increasing complexity and accuracy: 1) a simplified and analytical model $\hat{f}$ such as the dynamic bicycle model, used within the low-level NMPC controller for AD motion control, 2) a multi-physics data-driven high-fidelity simulator f_{DT}, namely the xDT of the car, with 15 degrees-of-freedom used for controller validation in MiL and as an xDT, and 3) the real vehicle complex and unknown dynamics f.

The inner loop affected by the change of parameters θ is the parametrizable AD controller, in this case the NMPC policy π_θ which determines the control action U for a state measurement X_{t_0}. The NMPC prediction model evolves according to $\hat{f}$, with a horizon length T_H, initial condition $\bar{x}$ and subject to constraints on the inner states $x(t)$ and control $u(t)$ defined

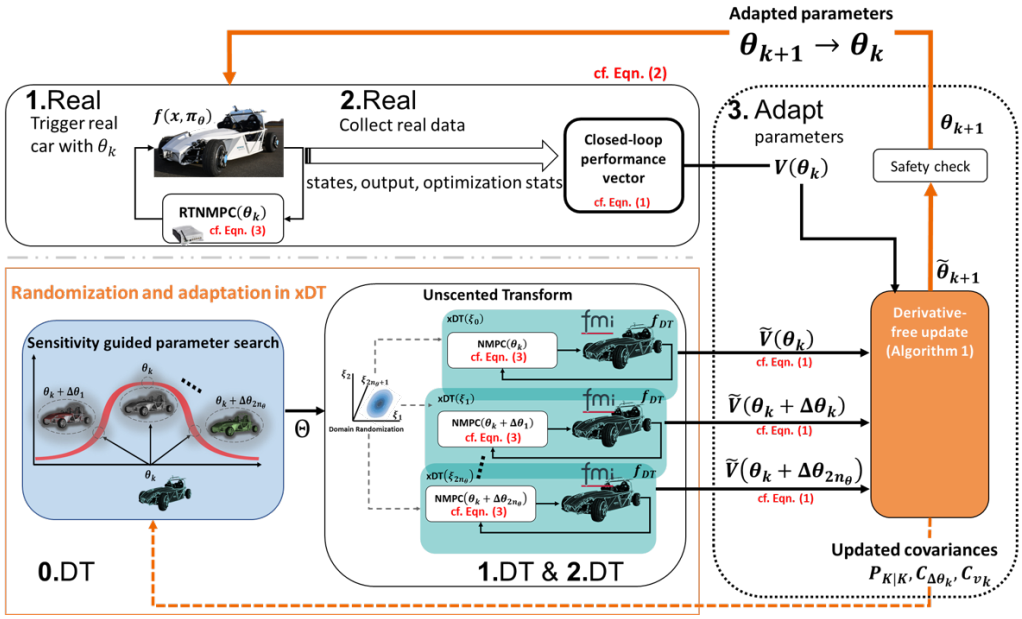

Fig. 3: Adaptation framework: optimize the target domain (real-world) closed-loop performance V safely and iteratively by sampling the different xDTs running in parallel with each other and with the real car. Framework combining Domain Randomization, Adaptation and High-fidelity simulation, and driven by real-world performance data.

by the set $\mathbf{X}$. We form the controller parameter calibration problem of Figure 2 as follows:

$$\begin{aligned} &\underset{\theta}{\textbf{minimize}}\, \mathbb{E}_\xi \left[\frac{1}{2}\|V(\theta)\|^2\right] \\ &\quad \textbf{subj. to } \theta \in \mathbf{C}_\theta, \\ &X(t+1) = f\big(X(t), \pi_\theta(X(t)), \xi\big), \qquad t = t_0, \ldots, T \end{aligned} \tag{2}$$

$$\begin{aligned} \pi_\theta(x) = \underset{x(.),u(.)}{\textbf{argmin}} \Big[J_\theta^* = \int_0^{T_H} (x^\top Q_\theta x + u^\top R_\theta u) dt \Big] \\ x(0) = \bar{x} = X_{t_0}, \\ \dot{x}(t) = \hat{f}(x(t), u(t)) \;\; \forall t \in [0, T_H], \\ x(t), u(t) \in \mathbf{X}, \end{aligned} \tag{3}$$

where:

- the vector V is a stacking of performance measures such as passenger comfort, tracking performance, driving style, optimization statistics, that can only be measured once a closed-loop oracle of the system with the controller is rolled out. The vector is measured through sensor readings, user feedback, or system performance metrics.
- The vector ξ captures unmodeled dynamics, real-world uncertainties, disturbances, and domain shifts that influence the behavior of the system. The vector ξ can be seen as perturbations to the simulator's physical parameters (for e.g. mass, inertia, length, friction coefficients), measurement and control noise, and task choice. By considering those perturbations as DR, the trained policy robustifies against real-world uncertainties and avoids overfitting the simulation.
- The vector θ contains the NMPC tuning parameters (in this example the state and input weights Q_θ, R_θ).

The outer loop (2) optimizes the target domain performance measured in the real-world as shown in 1.Real and 2.Real of Figure 3. It is possible to measure $V(\theta)$ as a black-box oracle on the real plant, but it is expensive. Instead, we offload the burden towards the xDTs where we measure an approximate distribution $\tilde{\mathbf{V}}(\theta)$ as shown in Figure 2. The algorithm explores and approximates through the xDT oracles, the distribution of V and that of the parameters θ by sampling the xDTs in the loop as shown in 1.DT and 2.DT of Figure 3. The inner loop (3) of the optimization involves the parametric RTNMPC that is attached to each xDT and to the real car. Each vehicle (real, or xDT) is controlled with a clone of the NMPC but with different parameters, denoted as NMPC(θ) that leads to π_θ. For clarity, we refer to NMPC(θ) as the NMPC controller parametrized by θ and that is solved according to (3). The controller to tune or calibrate is available in both its digital and RT versions, as a C++ standalone library. Note that if a different controller architecture is used, for e.g. a Proportional-Integral-Derivative (PID), the inner loop in (3) is replaced by a simpler formulation: $\pi_\theta(x) = \theta_P e(t) + \theta_I \int e(t)dt + \theta_D \dot{e}$, where $e(t)$ is an error with respect to a state or output reference.

The vector V is computed from onboard measurements such as position tracking errors using a GPS, velocity tracking error using an IMU, comfort level using acceleration and jerk measures, as well as monitored signals such as the computation time of the NMPC controller. More precisely, $V(\theta)$ stacks the error of the measurements Y_i with respect to their desired or reference values Y_i^r as in (1). Each Y_i is performance indicator of some target output over the closed-loop data collected between time t_0 and $t_0 + T$, representing different metrics to be optimized over. We consider three metrics: Y_{path} collects the path tracking error with $Y_{path}^r = 0$, $Y_{velocity}$ collects the

velocity tracking error with $Y^r_{velocity} = 0$ and Y_{cost} stores the NMPC cost with $Y^r_{cost} = 0$. In practice, V could either contain sparse or dense measurements. For dense measurements, the instantaneous performances are measured (for e.g. tracking error, comfort) as the closed-loop system evolves. For sparse measurements, a sparse performance metric is measured at the end of a window T, for e.g. the total distance covered during T seconds. Finally, the total Key Performance Indicator (KPI) combines the individual metrics for multi-objective optimization:

$$\text{KPI} = \frac{1}{2N_T}\|V(\theta)\|^2 \tag{4}$$

The NMPC controller is implemented in a receding horizon fashion. At every instant, the states are measured (real or virtual), NMPC plans a trajectory for a horizon T_H ahead, and the first control action of the trajectory is applied as a policy. As we expose the NMPC parameters such as the weighting matrices in the cost function, the choice of θ directly affects the performance of the controller in the policy $\pi_\theta(x)$. We present in Section III-C an iterative framework to solve problem (2), by sampling a carefully chosen set of parameters in the digital world as seen in Steps 0.DT, 1.DT and 2.DT of Figure 3. Finally, the derivative-free algorithm is presented in Algorithm 1, constituting of the simultaneous exploration in the xDTs through a sensitivity guided parameter search as in 0.DT of Figure 3, and the performance improvement in the real world by feeding back the real-world data from an oracle with the candidate θ_k as in 1.Real of Figure 3.

Remark 2: Although it should be scalable to tune the parameters θ that affect the constraints $\mathbf{X}(\theta)$ and the prediction model $\hat{f}(x(t), u(t), \theta)$ of the NMPC, in the remainder of this paper we focus only on automatically calibrating the cost function parameters Q_θ, R_θ. That is the inner loop in (3) is only parametrized by $\theta = [Q_\theta, R_\theta]$, and all other components such as the constraints, the model parameters, or the horizon length T_H are fixed.

As the NMPC is limited in its forecast by the prediction horizon T_H, it is rather complicated to assess the effect of a decision taken by the NMPC at time t, on the performance of the vehicle at time $t+T$ where $T \gg T_H$. For this reason, we seek to solve (2) where V encapsulates the collected closed-loop data over a sliding time window T. The positive scalar T is the window of real-world data collection over which the algorithm tries to improve the performance. It is equivalent to the period of parameter adaptation in the target domain. There exists a trade-off between short and long adaptation period T: small values of T lead to local and short term performance improvement over the current driving area with frequent adaptation but with a forgetting factor. On the other hand, large values of T lead to overall performant controller, with less computational demand for the adaptation, but with slower response to local changes. As we do not aim to build a surrogate of the optimization function as with BO, employing the algorithm for adapting the control parameters at a fast rate improves the performance locally but has a forgetting factor when revisiting the same area. Whereas, if long adaptation period is chosen, such as $T = T_{episode}$ where $T_{episode}$ is a full scenario, the algorithm optimizes for one set of parameters that is optimal over the whole scenario. However, the proposed method allows both by exposing the adaptation rate as a user chosen hyperparameter. The experiments to be carried in this work are such that we optimize over the control parameters after collecting data for $T = 85\,\text{s}$, allowing the vehicle to launch and engage in path and velocity tracking, as well as completing the parking maneuver. The goal is to optimize over one set of control parameters that works well for a variety of references and speed profiles (i.e. not a repetitive task).

B. Research questions

The stochastic optimization problem to shape V has three hard conditions making it difficult to solve:

- **R.1** the vector V could be of any form, not necessarily differentiable nor analytical, and can only be measured by sampling a rollout.
- **R.2** The method must tolerate inexactness in the gradient approximation as V is corrupted with noise.
- **R.3** The goal is to optimize directly, but as the adaptation is completed online and the AD module operates on a real-time hardware, sample and data efficiency are a major pillar of the method. It is impossible to sample every trial from both a safety and cost perspective [3].

We seek to solve (2) iteratively, where the update iteration is denoted by the subscript k. Following a target system oracle as in (1), $V(\theta_k)$ is measured. As the output is corrupted with noise $O_{t_0:t_0+T}$, let v_k be the stacking of the noise components over the time window and over the individual outputs. The elements of v_k are assumed to be random variables following a Gaussian distribution with zero mean and a prior covariance C_v, where $v_k \sim \mathcal{N}(0,\, C_v)$. That is, let $\mathcal{H}(\theta_k)$ be the stacking of all the individual and instantaneous outputs $\mathcal{H}_i$ for the time window $[t_0, t_0 + T]$, and Y^r the stacking of the reference values for each of the instantaneous outputs. It follows that the update step is rewritten as:

$$V(\theta_k) = (\mathcal{H}(\theta_k) - Y^r) + v_k, \tag{5}$$

$$\theta_{k+1} = \theta_k + \Delta\theta_k. \tag{6}$$

Moreover, the uncertainty on θ_k results from it following the distribution $\Delta\theta_k \sim \mathcal{N}(0,\, C_{\Delta\theta})$. Note that the distribution of $\theta_k, \Delta\theta_k, V(\theta_k)$ will be approximated through an Unscented Transform (UT), which is not restricted to the assumption that the distributions of noise sources are Gaussian [22].

C. Sim2Real using xDT and derivative-free optimization

The proposed algorithm falls under the category of directional direct search methods in DFO that sample at several perturbed points to explore the local behavior of the cost, to then form a directional step. However, to avoid greedy steps by choosing the sample point with the minimum function evaluation, we consider the distribution over all θ and ξ and step in the direction of expected value improvement. For this, we adopt a zeroth order optimization framework that only requires access to the value function V. Moreover, scalability to both episodic and rapid local adaptation is sought for by the method, which is not possible by BO. As we seek to extend

the training to any desired measurable objective, differentiable or not, and to exploit parallel xDT black-box structure, we consider gradient or derivative free, zeroth order Optimization (DFO). A list of available methods for such an approach is presented in [23]. One method that falls in this category is the optimization through gradual parameter estimation in the form of iterative Kalman filtering. In fact, Extended Kalman Filter (EKF) is proven to work for such applications as asymptotically it resembles a gradient method with diminishing step-size. From [24], EKF resembles a Gauss-Newton step for least squares problems, except that it works incrementally, and for large iterations the method leads to a Gauss-Newton step with diminishing step size [24].

In our case we propose the use of iterating UKF steps, with no linearization involved, to deal with arbitrary complex V and to incorporate real-world data. In the work of [25], the Iterative UKF (IUKF) is presented as a derivative free extension to the Iterative EKF (IEKF), requiring no linearization and that can be used in optimization algorithms. As the gradient is never numerically calculated, the UT prediction and estimation steps allow to form a sort of zero-order, derivative-free update step. However, further studies are to be carried to extend the results of resemblance between IEKF and a Gauss-Newton algorithm, towards the UKF and its ability to accurately solve a nonlinear least squares type of optimization.

Exploiting the UKF for optimal parameter tuning has been presented in [12] to outperform BO given its ability to handle high-level of noise through the weighted averaging of samples. Moreover, the number of samples in a UKF framework grows linearly with the number of parameters and is model free as it does not require to build a surrogate of the performance metric. Finally, the optimal parameter search is guided through the covariance matrices eliminating the need for dangerous random sampling. In [11], the combination of two derivative-free methods (UKF and SPSA) was presented to estimate the optimal set of control parameters θ in the target domain. The derivative-free update is achieved through sampling several xDTs online by smartly perturbing the parameters around the current iterate θ_k, and measuring the performance index $\tilde{V}_\xi(\theta + \Delta\theta)$ in each xDT. We extend [11] by providing an experimental validation of the approach through a parallelizable framework, to automatically adapt the parameters of a RTNMPC controller used for AVP applications. Moreover, we complete the algorithm by adding an adaptation layer to the covariance matrices that are directly related the learning rate of the approach, and its convergence. For robustness and generalization of the controller against external uncertainties ξ, we add DR in the xDT rollouts in the form of noise and model parameter uncertainties. For online applications, both UKF and SPSA are suited for parallel computation as the sample rollouts in the xDTs are independent. The iterative DFO algorithm is presented in Algorithm 1.

*1) **Unscented Kalman Filter**:* A UKF propagates the parameters through nonlinear dynamics and updates the estimated parameters without relying on gradient measurements. The method samples a set of points around the current estimate and can be used for estimating the optimal controller parameters as in [12]. The UKF is chosen over an EKF, as the former samples several points simultaneously. In an iterative scheme, UKF forms a more accurate estimate of the stochastic directional derivative between the parameters to tune and the performance index to optimize over. This is possible as UKF captures the uncertainty by using a set of sigma points, instead of an explicit linearization around the mean estimate. This generates a set of simulation models instead of a single one, allowing it to better capture nonlinear systems and a wider range of distributions [22]. In the presence of nonlinearities and inaccuracies, UKF tends to be more robust than EKF as it avoids linearization around the mean estimate of the Gaussian and can better propagate the uncertainties. Moreover, at the cost of computational overhead, UKF provides generally better accuracy in parameter estimation. It utilizes not only the mean estimate for a rollout, but rather a set of carefully chosen sigma points to capture the mean and covariance of the distribution. This makes it attractive for frameworks combining black-box plant models and possibly non-analytical, nonlinear evaluation metrics. In [11], we proposed to propagate the UT through the DT as the predicted performance is closer to the target domain than simplistic models and allows the algorithm to converge faster.

Given an n_θ-sized parameter vector θ following a normal Gaussian distribution of the form $\theta \sim \mathcal{N}(\theta_k, P_{k|k})$ at an instance k, we obtain a set of candidate sampling points Θ around the mean θ_k. The sampling or sigma points are concatenated in a matrix $\Theta = [\theta_k | \theta_k + c_k A^j | \theta_k - c_k A^j] \in \mathbb{R}^{n_\theta \times 2n_\theta + 1}$, where $c_k = c_0 = \sqrt{n_\theta + \lambda}$, and A^j is the jth column of the matrix $A = \sqrt{P_{k|k}}$. The matrix A is computed by performing a Cholesky decomposition of the prior covariance matrix $P_{k|k}$. The scalar λ dictates the spread of the sampling points around the mean and contributes to fine-tuning the higher order moments of approximation [22]. From [22], one good heuristic for choosing λ is $n_\theta + \lambda = 3$ which minimizes the difference between the moments of the standard Gaussian and the sigma points up the fourth order. To form Θ, the sampling points are generated from the step size c_k in the direction of the uncertainty represented by A^j. In total, $2n_\theta + 1$ rollouts are performed for one update step k to $k+1$. As it is inefficient to evaluate every sample point Θ^j in the target domain, we propagate the dynamics and performance evaluation through the xDTs. Then, we form the distribution in the form of mean and covariance as a weighted sum of the independent and parallel xDT closed-loop performances, through the weighting parameters w_a. The weighting vector is formed as $\mathcal{W} = [w_a^0, w_a^1, \ldots, w_a^{2n_\theta}] \in \mathbb{R}^{1 \times 2n_\theta + 1}$, where:

$$w_a^0 = \frac{\lambda}{(n_\theta + \lambda)}, \; w_a^j = \frac{1}{2(n_\theta + \lambda)} \quad \text{for } j = 1, \ldots, 2n_\theta. \tag{7}$$

The scalar w_a^0 is the weight associated with the first sample point, i.e. the mean θ_k, and w_a^j is the weight associated with the jth sigma point. As the closed-loop dynamics in the target domain evolve according to $f(x, \pi_\theta)$ from time t_0 to $t_0 + T$, all the xDTs are spawned and run in parallel with it starting from the initial condition $x(t_0)$. At time $t_0 + T$, the algorithm calculates the stochastic directional derivative in the form of a Kalman gain K_k which contains first and second order information about the step direction for performance

Algorithm 1 Stochastic DFO algorithm

Require: $\Theta, w_a, C_{\Delta\theta,0}, C_{v,0}, P_{0|0}$
Step 1: Unscented Transform
$\bar{\theta} = \theta_k = \sum_{j=0}^{2n_\theta} w_a^j \Theta^j$
$P_{k+1|k} = C_{\Delta\theta,k} + \sum_{j=0}^{2p} w_a^j (\Theta^j - \bar{\theta})(\Theta^j - \bar{\theta})^T$
for $j = 1, \ldots, 2n_\theta$ **do** ▷ Parallel xDT rollouts
$y^j = \tilde{V}_\xi(\Theta^j)$ ▷ Propagate
end for
$\bar{y} = \sum_{j=0}^{2n_\theta} w_a^j y^j$
Step 2: Cross covariance update
$P_{\theta y} = \sum_{j=0}^{2n_\theta} w_a^j (\Theta^j - \bar{\theta})(y^j - \bar{y})^T$
$C_{yy,k} = \sum_{j=0}^{2n_\theta} w_a^j (y^j - \bar{y})(y^j - \bar{y})^\top$
$P_{yy} = C_{v,k} + C_{yy,k}$
Step 3: Calculate gain and curvature
$K_k = P_{\theta y} P_{yy}^{-1}$ ▷ Kalman gain
$P_{k+1|k+1} = P_{k+1|k} - K_k P_{yy} K_k^\top$ ▷ Posterior covariance
Step 4: Calculate the optimization step
$\delta\theta_{SPSA} = -a_k \hat{g}_{SPSA}$ from (9) and (10) ▷ SPSA step
$\delta\theta_{UKF} = -K_k V(\theta_k)$ ▷ UKF step
$\Delta\theta_k = w\delta\theta_{UKF} + (1-w)\delta\theta_{SPSA}$ ▷ UKF+SPSA
$\tilde{\theta}_{k+1} \leftarrow \theta_k + \Delta\theta_k$
$a_k \leftarrow a/(\|y^{j=0}\|^2 + k^{0.602})$
Step 5: Adapt noise covariances from [26]
$\epsilon_k = V(\theta_k) - \bar{y}$
$C_{\Delta\theta,k+1} \leftarrow \alpha C_{\Delta\theta,k} + (1-\alpha)(\Delta\theta_k \Delta\theta_k^\top)/(k^2)$
$C_{v,k+1} \leftarrow \alpha C_{v,k} + (1-\alpha)(C_{yy,k} + \epsilon_k^\top \epsilon_k)/(k^2)$

improvement. Furthermore, the gain K_k incorporates the covariance of the parameters, and cross-covariance between the parameters and the predicted output. Drawing resemblance to the equivalence between IEKF and Gauss-Newton, the spread of the covariance matrix can be seen as the step size in the Gauss-Newton algorithm. Finally, as we aim for a bidirectional communication between real and virtual, the target domain performance data is received in the adaptation framework. The parameters are updated by combining the Kalman gain K_k from the xDTs exploration with the target domain performance $V(\theta_k)$, to generate the next possible iterate $\tilde{\theta}_{k+1}$:

$$\tilde{\theta}_{k+1} = \theta_k - K_k V(\theta_k). \tag{8}$$

Note that $\tilde{\theta}_{k+1}$ undergoes a safety check to be discussed in III-E before being applied as θ_{k+1}.

*2) **Simultaneous Perturbation Stochastic Approximation**:* There exist several approaches for estimating the stochastic gradient of a certain performance index with respect to n_θ parameters, out of which the SPSA method that was first proposed in [13] for stochastic optimization. By perturbing all parameters simultaneously, SPSA requires exactly only 2 evaluations of $L(\theta_k) = \|V(\theta_k)\|^2$ for the stochastic approximation of the gradient, regardless of n_θ. The scalar $L(\theta_k)$ represents the total key performance index to minimize in (2). The method has proven to be beneficial in case no analytical relationship between V and θ is present and for online deployment purposes, as it is memory and data efficient. Moreover, as sampling V with the perturbed θ directly in the target (real world) is unsafe and non-viable, we perform a simulation based optimization to solve for $\delta L/\delta\theta_k = 0$ by estimating the gradient $g(\theta) = \nabla L(\theta)$ while simultaneously perturbing all parameters:

$$\hat{g}_{SPSA}(\theta_k) = \frac{L(\theta_k + c_k\Delta_k) - L(\theta_k - c_k\Delta_k)}{2c_k} \begin{pmatrix} \Delta_{k1}^{-1} \\ \vdots \\ \Delta_{kn_\theta}^{-1} \end{pmatrix} \tag{9}$$

The gradient estimate in (9) differs from a usual finite difference approximation in that it employs only 2 perturbed measurements instead of $2n_\theta$. Every parameter is independently perturbed with a magnitude of $c_k\Delta_{kj}$ where c_k is the differential step size hyperparameter of the SPSA algorithm. [13] suggests a random perturbation vector $\Delta_k = [\Delta_{k1}, \ldots, \Delta_{kn\theta}]^T$, following a Bernoulli distribution symmetric about zero, with mutually independent elements. The recursive SPSA algorithm updates the parameter estimate using a step size a_k as:

$$\tilde{\theta}_{k+1} = \theta_k - a_k \hat{g}_{SPSA}(\theta_k) \tag{10}$$

From [13], a necessary condition for the convergence of the algorithm is that both the update and sampling steps (a_k, $c_k\Delta_k$) converge to zero. In other words, $a_k, c_k > 0 \ \forall \ k$, and $a_k \to 0, c_k \to 0$ as $k \to \infty$. Therefore, we propose to combine UKF and SPSA, by setting the SPSA sampling to be guided by the real data-driven covariance of the parameters $P_{k|k}$ from the UKF counterpart:

$$c_k\Delta_k = (c_0 \times \sqrt{P_{k|k}}) \times \text{Bernoulli}(1), \tag{11}$$

where Δ_k uses a Bernoulli ± 1 distribution. The parameter search will then be fully guided by the parameters' uncertainty, to generate sampling points for SPSA. Starting from an initial scalar c_0, we can now ensure a diminishing sampling step condition, if the uncertainty diminishes as will be explained in the next section. Moreover, the update step size a_k is set to diminish as per Step 5 of Algorithm 1. The division by $\|y^{j=0}\|^2$ scales the step size, and ensures a continuous parameter adaptation, if the performance index is non-zero. The factor $k^{0.602}$ follows the guidelines in [27]. Note that since SPSA does not require calculating gradients analytically, only relies on zero-order information of the cost to optimize $V(\theta_k)$, and can handle noisy and complex objective functions, it is a derivative-free method and is suitable to address our research questions **R.1**, **R.2**.

*3) **Fused adaptive UKF with SPSA**:* An extensive convergence theory for SPSA exists and pertains to both local and global optimization as in [28], [29]. SPSA has shown to be an effective, fast and efficient method for derivative-free global optimization methods and has been extensively used for numerous applications as in [30]. However, SPSA can also be sensitive to the noise realizations over the random perturbation of the parameters. Furthermore, vanilla SPSA does not follow a sensitivity guided parameter search. For these reasons, we propose to combine UKF and SPSA. Given that the UKF algorithm generates the $2n_\theta$ rollouts, we exploit the measurements to perform an aggregate SPSA step. Moreover, as we aim to transfer the performance from Sim2Real, SPSA does not embed by default the characteristic of mixing

simulation and real data. On the other hand, the iterative UKF with the adaptive noise covariance captures the Sim2Real gap. It introduces the gap as a bias in the iterative framework, and smartly explores interesting parameter regions, through the sensitivity guided parameter search. Thus, we can exploit the parameter search and rollouts generated by the iterative UKF to estimate more accurate SPSA stochastic gradients. On the other hand, SPSA improves the UKF performance: UKF tends to get stuck in local minima as it captures the noise and samples several sigma points around the current mean. The efficient use of SPSA for global optimization as in [29] allows to escape those minima by injecting noise on the gradient. Therefore, we combine the strengths from SPSA and UKF in the parameter calibration: UKF allows generating the sample points and rollouts, and SPSA allows escaping local minima in the optimization routine. Moreover, we fuse the adaptation steps on the parameter estimate from UKF and SPSA (c.f. (8), (10)) in Step 4 of Algorithm 1 through a weighted mean using the hyperparameter $0 \leq w \leq 1$. For the remainder of the work, we set $w = 0.5$. However, this hyperparameter can be tuned more carefully if more weight is to be assigned to the step using the current real-world performance with UKF, or the expected simulation improvement from SPSA.

Remark 3: Although the parameter search resembles a Monte Carlo method, the sampling points are not random, but are rather carefully chosen to capture information about the distribution in the exploration [22]. This results in the sampling efficiency when running the multiple simulations, and addresses **R.3**. That is, the parallel xDTs rollouts follow the propagate step of Algorithm 1. We augment the algorithm with DR in the form of ξ to robustify the controller against external uncertainties when transferring to the real world. In its simplest form, ξ is an additive input and output noise, and some mismatch on model parameters such as the total vehicle mass, the cornering stiffness, etc. The xDTs are sufficient to capture the vehicle behavior under the varying uncertainties, but more importantly serve to capture the sensitivity between the output and the parameters.

D. Adaptive covariance matrix

As the data-driven UKF learns the optimal parameter set in a recursive approach, research has proven that the learning rate is directly correlated to the choice of process and output noise covariances $C_{\Delta\theta}, C_v$ [26]. Moreover, if those two matrices are kept constant, the parameters' covariance matrix $P_{k|k}$ does not necessarily diminish. This leads to unnecessary sampling in the parameter space. Moreover, as we combine UKF with SPSA, it is imperative that both the sampling and update steps (a_k, $c_k\Delta_k$) diminish as the performance stops improving. For this, we follow the adaptive noise covariance matrices approach developed in [26] for an EKF in a state estimation framework and introduce it to our UKF approach as seen in Step 5 of Algorithm 1. It consists of adapting the parameter step and output covariances $C_{\Delta\Theta}, C_v$ through a forgetting factor α. We set $\alpha = 3$. Furthermore, the Sim2Real gap is captured through ϵ_k. It allows the algorithm to account for the Sim2Real gap in the form of a bias with respect to the simulated distribution of oracles. By adapting C_v on-the-go, we integrate the uncertainty on the Sim2Real gap into the update step. Therefore, the objective is not to develop an xDT that mimics the real counterpart, but rather have the real-world closed-loop performance contained within the distribution of virtual closed-loop performances. Through the adaptive covariance matrices, the algorithm leads to an uncertainty minimization as $C_{\Delta\theta}, C_v$ and $P_{k|k}$ diminish. This uncertainty minimization contributes to the convergence of the algorithm as the search space shrinks. Eventually, the real-world performance is mimicked by its digital counterpart, hence achieving a Sim2Real transfer. Thus, the adaptive covariances lead to a converging algorithm with shrinking update and exploration step sizes, which are necessary conditions for the convergence of SPSA.

Combining all the components together, the overall framework of D^{2}C^2-AUKS is shown in Figure 2, and the particular example of NMPC adaptation in Figure 3. The underlying iterative and data-driven calibration scheme is presented in Algorithm 1. The black-box optimization method is not one shot, but rather iterative: first, data is collected from the target domain. Second, the local properties of the optimization function (nonlinearities, curvature) are discovered or learned in the digital domain. Third, the parameter iterate is applied on the target system. Finally fourth, the exploration space is adapted through feedback from the real output and a sensitivity analysis. The update step is according to Step 4 of Algorithm 1. A summary of the algorithms hyperparameters is found in Table I.

E. Safety check

As we deal with noisy data, performance improvement is sought over a global optimization. For this, we incorporate a simple safety check on the updated parameters. The safety check consists of rolling out an additional oracle with the nominal xDT and the controller with the new parameter set $\tilde{\theta}_{k+1}$. This digital oracle aims to verify that 1) the closed-loop system is not unstable, 2) the parameter set is within its bounds C_θ, 3) the NMPC cost function or energy measure is not increasing in comparison with the rollout with θ_k. For applications with highly nonlinear dynamics and in presence of noise, strict Lyapunov stability is hard to prove. Therefore, the third check consists of checking that the total closed-loop NMPC energy or cost function $H^\theta_{cost} = \sqrt{\frac{1}{N_T}\sum_{i=1}^{N_T}(J^*_\theta(i))^2}$ is at least contained. The instantaneous optimal cost $J^*(\theta)$ is calculated as in (3), and N_T is the number of discrete measurements in $[t_0, t_0 + T]$. For this, we exploit the xDTs to check that starting from the same initial conditions $\bar{x}$ at t_0, an xDT rollout with $\tilde{\theta}_{k+1} > 0$ would have kept the total energy contained as $H_{cost}(\tilde{\theta}_{k+1}) \leq (1 + R)H_{cost}(\theta_k)$, where $R \geq 0$. Eventually, $\theta_{k+1} \leftarrow \text{Check}(\tilde{\theta}_{k+1})$, otherwise $\theta_{k+1} = \theta_k$. Note that in all the results presented below, $\tilde{\theta}_{k+1}$ were safe enough to be directly exploited on the real system. Finally, we exploit the prior parameter covariance $P_{k|k}$ to modify the step size c_k as in [11]. When computing Θ, the scalar c_k is adjusted such that the parameters θ are within the safe bounding box $C_\theta = [\theta_{\min}, \theta_{\max}]$ provided

by the user. That is, we add a regularization heuristic to choose the smallest c_k such that none of the parameters violates their limits. This is important in the case of NMPC where $Q_\theta \succeq 0$, $R_\theta \succ 0$, to avoid sampling the xDTs with indefinite matrices. The regularization method avoids clipping the parameters individually to their limits which breaks the Gaussian distribution. Instead, we find the suitable exploration step size c_k that forms the sigma points $\Theta = \theta_k \pm c_k\sqrt{P_{k|k}}$ such that $c_k = \min(c_k, c|\theta_k \pm c\sqrt{P_{k|k}} \in C_\theta)$. This heuristic ensures that the Gaussian distribution of $\Delta\theta_k$ around 0 is preserved.

IV. NMPC ADAPTATION IN SIMULATION

The objective of the data-driven calibration framework is not to replace online learning with offline learning, nor the opposite. We seek to exploit heavily the digital world to generate a prior on the distribution of the parameters and their uncertainties. Then, we complement with the real-world data as to close the Sim2Real gap with minimal real-world effort. We first validate D^{2}C^2-AUKS for an AVP application in simulation, where NMPC acts as the low-level vehicle motion controller.

A. Autonomous valet parking formulation

The underlying vehicle model $\hat{f}$ used within the NMPC formulation should be able to capture well the dynamics of the vehicle, without over-complicating it such that it remains real-time feasible. For this, we employ the fused kinematic and dynamic single-track model in the Curvilinear (Frenet) frame as presented in [31]. This frame transformation from a Cartesian to an error frame with respect to the path, allows to be domain independent and to avoid carrying reference trajectories within the NMPC. The vehicle dynamics are over the longitudinal and lateral velocities (v_x, v_y) and the yaw rate r. The state s tracks the evolution along the centerline, and w and θ track the distance and heading deviation from the path centerline. The formulation is parametrized by the curvature $\kappa_c(s)$ which, alongside the initial conditions, is enough to capture the evolution of the deviations. Moreover, the Curvilinear formulation allows to convexify position constraints of the vehicle on curved roads as they can be directly cast as a tube constraint of the form $w_l \leq w(t) \leq w_r$. Here, w_l, w_r are the left and right limit deviations from the centerline. We augment the curvilinear single-track model with the throttle $(\dot{t}_r)$ and steering $(\dot{\delta})$ rates as control input in the Optimal Control Problem (OCP) for smoother driving. Therefore, the state vector in the NMPC is $x = [v_x, v_y, r, s, w, \theta, \delta, t_r]$ and the input vector is $u = [\dot{\delta}, \dot{t}_r]$. The dynamics are represented by $\dot{x} = (\lambda)f_{dyn}(x,u) + (1-\lambda)f_{kyn}(x,u)$. We fuse the kinematic f_{kyn} and dynamic f_{dyn} single-track models to allow smooth and differentiable transition from low to high speed and to permit reaching a zero velocity. This is possible through the smooth activation value $\lambda(v_x)$, generating a continuous switching between the models inside the optimization routine. For an AVP application, the NMPC optimizes as in (3) over $J^*_\theta = \min \int_{t_0}^{t_0+T_H} (x-x^r)^\top Q_\theta (x-x^r) + u^\top R_\theta u$, where the reference $x^r = [v^r, \mathbf{0}]$. That is, all states (except for s) and control inputs are regulated to zero, and for the velocity tracks the reference v^r. Moreover, the path evolution s is not penalized in the NMPC cost (i.e. $Q_s = 0$). Therefore, $Q_\theta \in \mathbb{R}^{7\times7}$, $R_\theta \in \mathbb{R}^{2\times2}$. The NMPC policy $\pi_\theta(x_{t_0})$ is parametrized by θ to tune, and where the first control action is applied in a receding horizon scheme. As multiple and single shooting methods were non-viable for our real-time NMPC application, we resort to the Spectral Orthogonal Collocation with Safety Envelope (SOCSE) method in [31] to cast the OCP into an NonLinear Programming (NLP) problem. Only box constraints are imposed on all optimization variables x, u. A horizon of $T_H = 3$ s, and splines of order 5 are used within SOCSE.

Remark 4: The detailed NMPC formulation has been omitted, as the goal is to automatically calibrate a black-box controller, in a black-box oracle, without particular and detailed knowledge on the controller and its parameters. A detailed NMPC formulation is found in [31].

B. Parameter tuning with domain randomization

The objective is to shape the optimal NMPC policy such that it compensates for the finite horizon assumption, by feeding back the closed-loop performance over a larger time window $T > T_H$. We optimize over the vector θ that stacks the elements of the diagonal matrices Q_θ, R_θ such that $\theta = [\text{diag}(Q_\theta), \text{diag}(R_\theta)]$. The optimization objective is to improve the real-world performance that has been deteriorated by the Sim2Real transfer. The main Sim2Real uncertainties are in the low-level actuation system such as steering delays, simplified throttle and brake models, and the road grade of approximately 4% that was not accounted for in the NMPC model. For this, we set V in (1) to be composed of the measurements over 3 outputs $Y_{velocity}$, Y_{path} and Y_{cost}:

$$Y_{velocity} = [v_x(1) - v^r(1), \dots, v_x(N_T) - v^r(N_T)], \quad (12a)$$

$$Y_{path} = [w(1), \dots, w(N_T)], \quad (12b)$$

$$Y_{cost} = [J^*_\theta(1), \dots, J^*_\theta(N_T)]. \quad (12c)$$

The objective is to simultaneously improve velocity and path tracking performances, while keeping the energy as low as possible through the inclusion of the NMPC cost in the objective to minimize. Each metric H^θ is a Root-Mean-Square (RMS) over the closed-loop data collected during $[t_0, t_0+T]$ through an oracle with parameter θ. Considering the N_T discrete elements in the rolling window, we calculate the RMS of the individual outputs:

$$H^\theta_{velocity} = \sqrt{\frac{1}{N_T}\sum_{i=1}^{N_T}(v_x(i) - v^r(i))^2} \quad (13a)$$

$$H^\theta_{path} = \sqrt{\frac{1}{N_T}\sum_{i=1}^{N_T}(w(i))^2} \quad (13b)$$

$$H^\theta_{cost} = \sqrt{\frac{1}{N_T}\sum_{i=1}^{N_T}(J^*_\theta(i))^2} \quad (13c)$$

Importantly, the transformation to the Curvilinear formulation allows to directly penalize the lateral path tracking error w. It

TABLE I: Summary of hyperparameters for Algorithm 1

Parameter	Description	Initial Value
n_θ	Number of parameters to tune	9
n_O	Number of user-defined performance metrics	3
$C_{\Delta\theta}$	Parameter perturbation covariance	$\mathbf{I}$
C_v	Noise covariance	$\mathbf{I}$
$P_{0\|0}$	Parameter covariance	$\mathbf{I}$
w_a^0	Weight on first sample	-2.0
dt	Sample time of measurement [in seconds]	0.05
N_T	Number of steps in adaptation window	1700
T	Adaptation window [in seconds]	85

measures the instantaneous deviation from the path, regardless of the global reference system, in a domain invariant fashion. This transformation serves as a latent invariant space in DA.

We first run the calibration process in a MiL approach, where the target vehicle is an xDT with different parameters from the ones used for sampling. This is to verify the adaptation from one model to the other (Sim2Sim). We train on one single dynamic path that has been recorded by a human driver as shown in Figure 4 in dashed-black line. The calibration is started by setting Q_θ and R_θ as unit matrices $\mathbf{I}$, that is $\theta = \mathbf{1} \in \mathbb{R}^{9\times 1}$. The choice of a unity vector is an uneducated initial guess for the control parameters. It leads to a stable but under-performant oracle. 19 xDTs are parallelized and run independently each with a perturbed $\theta_0 + \Delta\theta$. From Figure 4 on top, the exploration process of the xDTs is clear, and they perform with a significant variance as also depicted in Table II; Some xDTs are not able to complete the scenario. However, after just one tuning iteration with a D^{2}C^2-AUKS update and with feedback from the target vehicle performance, the next iterate θ_1 is applied. The tracking performance is immediately improved as the RMS on path tracking drops from $0.589\,\text{m}$ to $0.206\,\text{m}$. The RMS on velocity error, NMPC cost as well as the total KPI are also decreased. The total KPI in (2) improves by 16% within one iteration and 70% within 4 iterations. Moreover, the sampled xDTs within the first iteration only have converged to a quasi-similar performance level. All the xDTs are superimposed, due to a shrinking uncertainty on the parameter set. Moreover, the performance of the target vehicle is contained within the distribution of the xDTs, within one standard deviation. The results of the iterations are indicated in Table II. The uncertainty on the tracking performance output drops from $0.296\,\text{m}$ to $0.015\,\text{m}$ within one iteration. From the first iteration, the identified sensitivity between the controller parameters and the performance is not only simulation (source xDT) based, but rather it is enriched with the target system (ego xDT) data. The uncertainty on the parameter set shrank directly, and the method sampled more efficiently than with trial and error. Given that all the xDTs run in parallel with the target vehicle using an FMU, the adaptation took place in real-time.

Keeping the training in simulation only, we show the benefit of injecting DR in the form of noise, on the adaptation process as shown in Figure 5. Due to the awareness that the target domain contains input and output noise, we inject the sample xDTs with noise. This helps to avoid overfitting the trained controller to a noiseless environment. From Figure 5, one can see that in the presence of noise, the parameter update on path

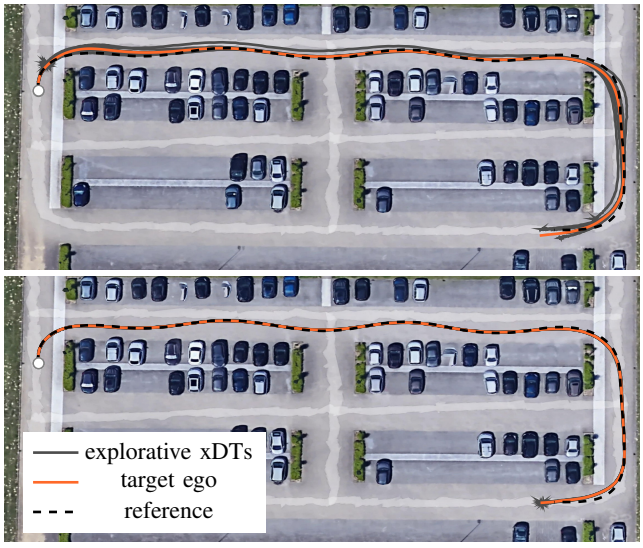

Fig. 4: MiL training on dynamic path: Top: Iteration 0: the ego vehicle follows the path with suboptimal performance. The xDTs have a high variance in their performance; Bottom: Iteration 1: after one single training iteration, the ego performance has improved and the xDTs are overlapped

TABLE II: Performance evolution through the automatic tuning iterations in Figure 4: 70% KPI improvement in 4 iterations and minimization of the performance uncertainty

	Iteration 0	Iteration 1	Iteration 4
Ego car H_{path}[m]	0.589	0.206	0.146
Ego car $H_{velocity}$[m/s]	0.799	0.373	0.309
Ego car H_{cost}	6.226	5.736	3.415
Ego car Total KPI	19.874	16.542	5.89
xDT cars H_{path}[m]	0.66±0.296	0.209±0.015	0.159±0.036
xDT cars $H_{velocity}$[m/s]	0.9432±0.416	0.3699±0.031	0.3178±0.021

tracking is more conservative than in the noiseless case. This is due to the higher weight on the path deviation that causes the controller to react rapidly to path error deviation which are corrupted with noise. Therefore, it is clear that the controller adapts to the present uncertainties directly without estimating them by sampling the 19 parallel xDTs. The training process is shown in the lower plot of Figure 5. The importance of the method resides in its sampling and data efficiency, as it takes few iterations to reach an acceptable performance level. The NMPC cost immediately drops to a plateau from the first iteration with an improvement of 87%. The tracking performance improves by 60% in just 4 iterations.

To determine the scalability of the method, we consider several paths with different driving styles (aggressive human

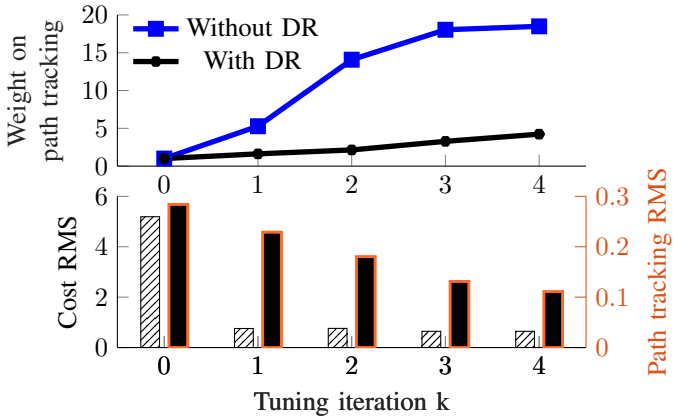

Fig. 5: Adaptation process: evolution of path tracking and NMPC cost RMS on path 1, alongside the path tracking NMPC parameter Q_w. The learning rate on cost improvement is large, and the performance is enhanced within few iterations

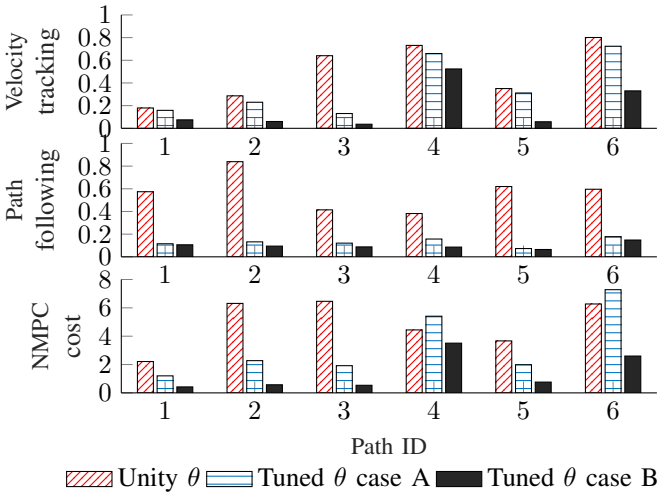

Fig. 6: Validation process: RMS for the closed-loop NMPC cost, velocity tracking and path following errors over the different paths in the library. Red: unit θ. Blue: tuned θ only on path #1. Black: tuned θ with path domain randomization

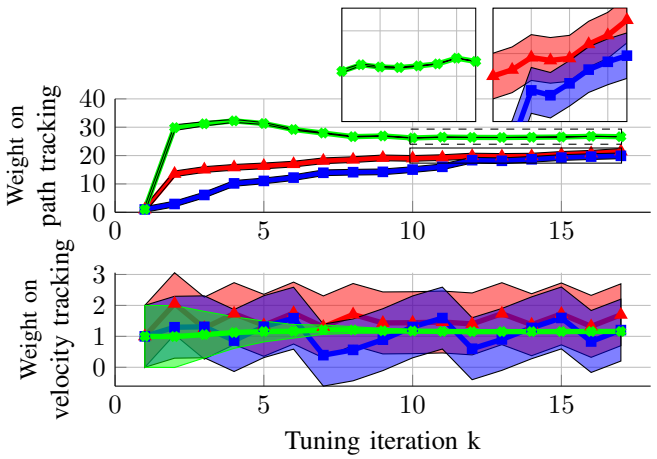

Fig. 7: Diminishing parameter uncertainty (shaded areas) with adaptive covariance matrices proposed in this work (green), in comparison with [11] (red triangles) and [12] (blue squares)

driving, smooth optimal path) and different curvatures. From a random set of initial position and target parking spots in the parking lot, an A* hybrid planner generates a library of reference paths, with different velocity profiles, different curvatures (sharp turns, straight lines, sinusoidal driving). Some of those paths will lead the car to drive on the uphill part of the parking lot, and others will trigger an aggressive driving style. For the sake of clarity, we focus on 6 of those chosen paths to study the performance transferability between varying paths. In comparison with the TiL work in [14], the objective is not to tune the DT to capture the real world dynamics for every path on its own, but rather generate a controller that is robust enough to a variation of scenarios. Let *case A* be the result of tuning using only the first path. When applied to the other unseen trajectories, the trained controller is generally better in velocity and path tracking than the unit θ initial weight matrix, as in Figure 6. However, on dynamic trajectories (path 4 and 6), the controller results in a higher cost and computational burden. This indicates that the controller failed to generalize to other trajectories by over-tuning for path 1 only. Therefore, we include DR on the path by using 80% of the trajectories' dataset for training and generate the tuned matrix of *case B* presented in Figure 6. This second tuned matrix generalizes better to other tasks by finding a local optima for the NMPC parameters improving all the performance metrics. The overall RMS of the NMPC cost time-series drops from 5.1481 (unity) to 4.0107 (case A) to 1.846 (case B). For path following, the RMS on the tracking error drops from 0.59031 (unity) to 0.13344 (case A) to 0.10111 (case B). Finally, the RMS on velocity tracking drops from 0.55126 (unity) to 0.438 (case A) to 0.25719 (case B).

Finally, we benchmark our proposed approach against [12] and [11] where the covariance matrices were kept constant. It is worth noting that tuning the covariance matrices is by itself a bottleneck in UKF and affects the results directly. This is alleviated by the data-driven covariance adaptation presented in D^{2}C^2-AUKS. In Figure 7, we perform a tuning campaign on only two parameters: the weights on path deviation Q_w and velocity tracking Q_{vx}. The work of [12] is outperformed by [11] in terms of adaptation speed given the inclusion of the SPSA step. However, a closer look at the parameter posterior covariance indicates that both approaches maintain a relatively large uncertainty in the parameter set. This causes unnecessary sampling of the xDTs even after the performance index has converged. In comparison, by including the adaptive covariance as in D^{2}C^2-AUKS, the guided search directions are more reflective of the true sensitivity of the parameters with respect to the real-world performance. This leads to a faster convergence of the method, in terms of both step size and sampling direction.

C. Stochastic black-box optimization

The proposed framework helps to simultaneously optimize all controller parameters, for a given user defined performance.

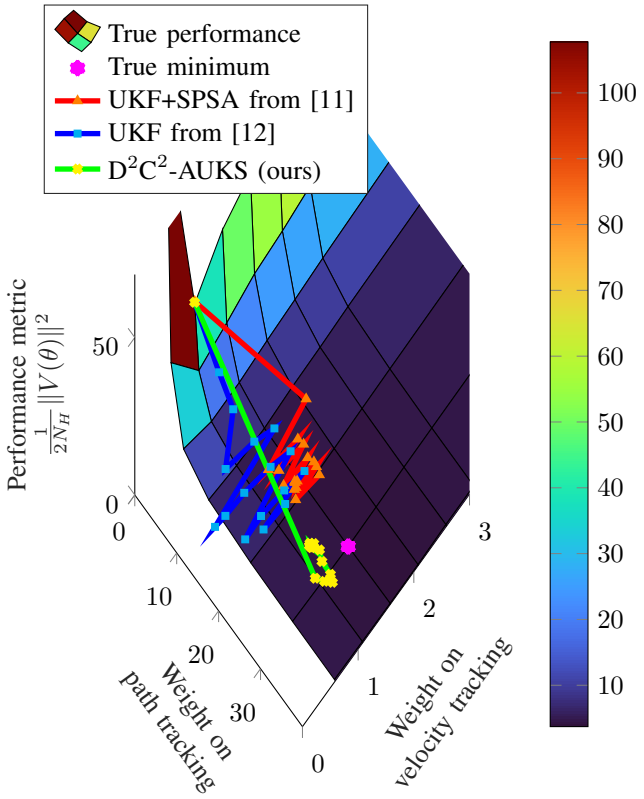

Fig. 8: Iterative adaptation framework and comparison with the true performance metric KPI on a fine grid

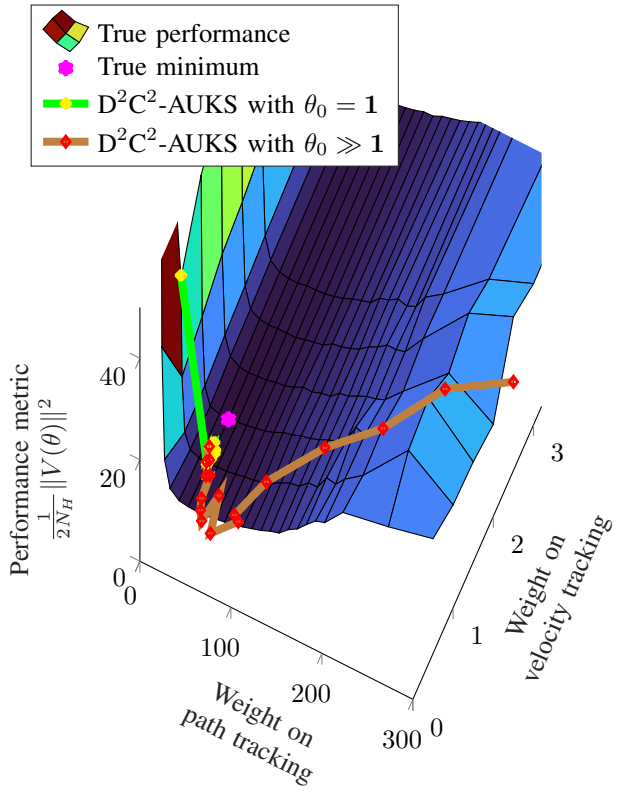

Fig. 9: Iterative adaptation framework solving for the stochastic black-box optimization problem in (2). For an initial $\theta_0 \gg \mathbf{1}$, the iterative framework recovers from the local minima to reach the optimal KPI value

As the adaptation takes place in the target domain in a closed-loop fashion, under several sources of noise, delays, and uncertainties, we seek to validate that the algorithm does indeed solve iteratively an optimization problem. Therefore, we run oracles with randomization in the high-fidelity simulator, on a grid of the combined parameters to tune $\theta = [\text{diag}(Q_\theta), \text{diag}(R_\theta)]$. We collect for each scenario the total performance metric $\frac{1}{2N_T}\|V(\theta)\|^2$ which is proportional to (2). For the sake of visualization, we consider the two most dominant parameters: the weight on path tracking error Q_w and the velocity tracking error Q_{vx}. The results of a parameter adaptation campaign are shown in Figure 8. Starting from a unit set of parameters, our D^{2}C^2-AUKS algorithm reaches a minimal cost within few iterations, as the parameters converge and the uncertainties shrink. In comparison with [12] and [11] which employ constant noise covariance matrices, the proposed approach exhibits no oscillation around the optimal set of parameters. D^{2}C^2-AUKS benefits from the data-driven adaptation of the parameter perturbation and noise covariance matrices $C_{\Delta\theta}, C_v$. In fact, the two covariance matrices adapt to the actual noise level in the target domain through uncertainties on the parameters and on the Sim2Real gap respectively. Thus, the algorithm compensates for those uncertainties.

Moreover, we initiate the parameter set in the neighborhood of a local minimum as seen in Figure 9, with $\theta_0 \gg \mathbf{1}$, indicating that every element of θ is larger than 1. This is in comparison to the case with $\theta = \mathbf{1}$ where the individual elements of θ are equal to 1. Starting from $Q_w = 300, Q_{vx} = 3$, the algorithm escapes the local minimum by exploring the local behavior of the KPI, and converges to the minimum value of the total performance metric, or equivalently reaches the maximum performance. This is due to the explorative nature of the framework in sampling several parallel xDTs in order to form the derivative-free update and moving along the average performance improving direction.

V. Sim2Real Experimental Validation with Real-Time Automatic Adaptation

One of the novelties in this work resides in the experimental validation of the adaptation and parameter tuning on road vehicles. Our platform is a SimRod drive-by-wire vehicle as seen in Figure 1. The RTNMPC runs on a dSPACE MicroAutobox III with a real-time operating system for embedded applications. The NMPC is sampled at 20 Hz as a low-level controller, commanding the steering and acceleration/deceleration rates. This section presents the results of the Sim2Real automatic NMPC adaptation using D^{2}C^2-AUKS.

A. Importance of Sim2Real transfer

First, the NMPC calibrated in simulation is transferred to the real-world, and the closed-loop performance is shown in Figure 1. It is a direct observation, that the NMPC parameters were overfitted in simulation to generate an optimal performance level in NMPC cost and tracking errors, which

motivates our work. Similar to most applications, including learning and classical control methods, engineers employ this Sim2Real methodology that consists of parameter tuning in simulation until a satisfying performance is reached. However, this approach often fails to transfer to the real world. This raises the need for 1) an adaptive scheme that can learn from the data collected in the real world, 2) closing the loop between the prior expectation (simulation) and the actual (real-world) closed-loop performance, and 3) a method to rapidly recalibrate the parameters and avoid manual tuning.

On the right-hand side of Figure 1, we present an alternative methodology for the Sim2Real transfer. The xDTs are spawned in parallel with the real vehicle, to smartly explore the parameters to tune and randomize the disturbances ξ. Exploration takes place rapidly and safely so that the parameters update on the target vehicle is facilitated. The performance of our approach is validated, and the results are shown in the Figure 1 and Table III. The path and parking tracking errors dropped to below 30 cm and 15 cm respectively, and the NMPC cost is minimal. We start the training with $\theta = \mathbf{1}$, which could represent a case of complete absence of engineering expertise. The closed-loop performance is shown in orange. After 4 iterations (purple curve), lasting a total of approximately 10 minutes including the repositioning time, the NMPC is tuned to optimize for this target configuration that is corrupted with steering actuator delays. The performance improvement is significant, and the end-of-line tuning time is cut down from hours to just few minutes.

B. Generalization to different paths

Depending on the intended application, the presented algorithm could be used to either reach an optimal parameter tuning for one specific task and environment condition, or to generalize for a set of tasks and conditions. In other words, if the vehicle is to repeatedly follow the same path, then a task specific parameter optimization could be reached. However, if the vehicle is set to drive on different curvatures, with varying velocity profiles and road conditions, then D^2C^2-AUKS is employed to adapt the parameters such that the performance is conserved across the tasks. To tackle the latter case, we train our algorithm starting from a unit $\theta = \mathbf{1}$ with DR on the chosen path and the vehicle's initial conditions. This helps to avoid overfitting for one specific task. The ViL, real-world adaptation of a RTNMPC is shown in Figure 10 and Table III. In particular, the blue path, encounters a road grade of approximately 4%. By tuning on the flat orange path solely, then validating on the uphill blue path, the vehicle fails to climb the slope. This is due a to a calibrated controller with low weight on the velocity tracking component in the NMPC cost Q_{vx}. However, we combine trajectories of different driving styles (human driven and smooth A* planner generated), with a multitude of target parking spots and perturbed initial conditions. Thus, the learning-based optimal controller adaptation generalizes for a variety of scenarios with improved performance overall. Three examples of tracked paths are shown in Figure 10, where the real vehicle is controlled to park within $15\,\text{cm}$ of accuracy using a RTNMPC as in Figure 10. The total RMS of the path tracking error, velocity tracking error and closed NMPC cost over a multitude of trajectories are $0.295\,\text{m}$, $0.642\,\text{m/s}$, and 2.417 respectively as represented in Table III.

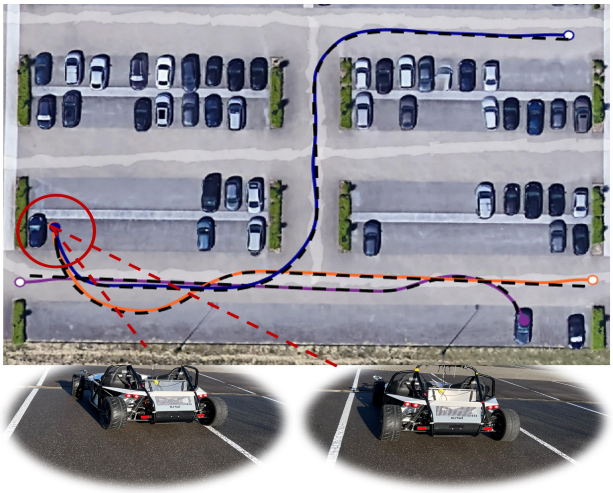

Fig. 10: ViL: policy trained by in closed-loop by randomizing over the paths. Reference path (dashed line), closed-loop ego position (colored). The controller is tuned to park properly (right) in few iterations ($\sim$10 minutes) after starting from a suboptimal performance (left)

TABLE III: RMS of the individual outputs, Sim2Real performance degradation and recovery with automatic controller calibration over different paths from Figures 1 and 10

Tuning	Tracking error [m]	Velocity error [m/s]	NMPC cost	Total KPI	Sim2Real Deterioration
SimOptimal in Sim	0.1327	0.2148	2.4645	3.0678	0%
SimOptimal in Real	1.2015	0.8587	73.4097	2.69e3	8.77e4%
Unit θ in Real	1.365	0.39133	8.933	40.906	1.21e3%
Autotuned in Real	**0.295**	**0.642**	**2.417**	**3.17**	**3.34%**

The necessity of the automatic calibration is to recover the expected performance from simulation, when transferring to the real-world. That is, the parameter search and optimization seek to directly compensate for model mismatches and noises with the single objective of optimizing the performance in (2). For this reason, we summarize the results of the method in Table III: the total performance multi-objective V is composed of the path tracking error Y_{path}, the velocity tracking error $Y_{velocity}$ and the closed-loop NMPC cost Y_{cost}. They are measured online on the real system, and we report their RMS over the time window T. We compare three tuning combinations to a controller tuned perfectly in simulation (*SimOptimal in Sim*). First, *SimOptimal in Real* is the evaluation of the optimal tuning obtained in simulation that is transferred to the real world. This is the typical case of a one-directional XiL testing with a controller that is over optimized in simulation, as reported in the left part of Figure 1. Second, controller tuning *Unit in Real* is an initial uneducated guess on the parameters with $\theta = \mathbf{1}$. It serves as a starting point for our automatic calibration, visualized in orange in

Figure 1. Third, the last controller tuning *Autotune in Real* is automatically tuned in the real-world using D^2C^2-AUKS, as reported in the purple plot in Figure 1 and the closed-loop trajectories of Figure 10. *SimOptimal in Real* suffers from a catastrophic performance degradation that is 8.77e4% higher than the one expected in simulation. All components of the KPI are worsened, specifically the tracking error and NMPC cost. This is caused by the delays in the low-level actuation and a high weight on the tracking error that causes the controller to fail in tracking the trajectories. Next, for *Unit in Real*, a similar bad performance in terms of path tracking is noticed, with an RMS of $1.37\,\mathrm{m}$ because of the low weight on the tracking error. Most importantly with *Autotuned in Real*, in under 10 minutes we are able to automatically calibrate the controller on-the-go. By feeding with the real-world data, the expected performance from simulation is recovered with little testing effort on the target system. That is, we recover the total KPI value of 3.17, which is 3.34% higher than the simulated performance, showcasing the ability to close the Sim2Real gap. However, as the individual metrics contributing to V are optimized simultaneously, and given the complex nature of the optimization problem, the tracking errors and the NMPC cost are not identically matched to their simulated performances. Nevertheless, the path tracking error is enhanced by dropping from $1.365\,\mathrm{m}$ down to $0.295\,\mathrm{m}$ over the multitude of trajectories. This accounts for the noise in estimation, as well as the initial condition of the vehicle that was off the path. In terms of energy, the NMPC cost is dropped by a factor of 3.7 from 8.933 down to 2.417, in 4 iterations.

VI. Limitations

The presented framework is to be used as a closed-loop black-box optimizer for the performance of a parametrizable controller. The following limitations are to be noted:

- Further extensions include tuning the constraints of the NMPC such that the real system meets those constraints, or tuning the prediction model in the NMPC to close the Sim2Real gap on the output responses. However, only the work concerning the automatic and on-the-fly calibration of a parametrized cost function in an NMPC for real-world AD has been validated.
- We assume that the employed NMPC under the hood maintains the safety requirements and acts as a safety filter for the data-driven automatic calibrator. The usage of xDTs to explore the control parameter space adds layers of safety and speed-up, as not all the variation of parameters are sampled on the real system to form the Kalman gain and stochastic gradient steps.
- We do not provide any guarantees on the effectiveness of combining SPSA with UKF for a derivative-free optimization. However, the necessary convergence guarantees with diminishing step size and exploration step size, support the claim that the overall adaptation framework is at least converging to a local minimum.
- Given the highly non-convex nature of the optimization problem, global optimum might not be reached, and it is possible to settle on a local minimum as we optimize by sampling several xDTs simultaneously and step in the direction of average expected improvement. This work is characterized by an automatic calibrator that improves the performance, with the least amount of real-world sampling and maximal digital world exploration, in a limited amount of time.

VII. Conclusion

In this work, we have presented a learning-based adaptation scheme for parametric controllers that allows a faster, safer, and cheaper transfer from one domain to another, through exploration in the executable digital twin domain and exploitation in the target domain. In particular, we validate the methodology on a parameter calibration and adaptation for a real-time NMPC controller in an autonomous valet parking framework. The proposed method is data and sampling efficient, and directly optimizes the closed-loop performance in the target domain with few iterations. We combine iterative Kalman filtering techniques with SPSA to solve a derivative-free optimization problem with possibly non-differentiable objectives. Experimental validation shows that the NMPC can be tuned in the matter of few minutes to significantly improve performance and reduce the incurred NMPC cost, while generalizing for different tasks. Moreover, we recover a Sim2Real gap of almost a factor of 1 through careful combination of simulated exploration and real data exploitation. The method is sample efficient as it gathers useful information with fewer interactions with the target system, thus reducing the spent time and resources on end-of-line tuning.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement ELO-X No 953348.

References

[1] F. S. Acerbo, J. Swevers, T. Tuytelaars, and T. D. Son, "Evaluation of MPC-based Imitation Learning for Human-like Autonomous Driving," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 4871–4876, 2023, 22nd IFAC World Congress.

[2] A. An, X. Hao, Q. Wang, and C. Ren, "Adaptive Weights Robust Predictive Zone Control and Its Application for Distillation Column Control," in *2009 Second International Conference on Future Information Technology and Management Engineering*. Sanya, China: IEEE, Dec. 2009, pp. 471–475.

[3] L. P. Frohlich, C. Kuttel, E. Arcari, L. Hewing, M. N. Zeilinger, and A. Carron, "Contextual Tuning of Model Predictive Control for Autonomous Racing," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: IEEE, Oct. 2022, pp. 10 555–10 562.

[4] M. Nobar, J. Keller, A. Rupenyan, M. Khosravi, and J. Lygeros, "Guided Bayesian Optimization: Data-Efficient Controller Tuning with Digital Twin," *ArXiv*, vol. 2403.16619, 2024.

[5] T. Z. Jiahao, K. Y. Chee, and M. A. Hsieh, "Online Dynamics Learning for Predictive Control with an Application to Aerial Robots," in *6th Annual Conference on Robot Learning*, 2022.

[6] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-Based Model Predictive Control: Toward Safe Learning in Control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, May 2020.

[7] M. Bujarbaruah, X. Zhang, U. Rosolia, and F. Borrelli, "Adaptive MPC for Iterative Tasks," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6322–6327.

[8] M. Schuurmans and P. Patrinos, "A General Framework for Learning-Based Distributionally Robust MPC of Markov Jump Systems," *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 2950–2965, 2023.

[9] R. Marino, "Adaptive control of nonlinear systems: Basic results and applications," *Annual Reviews in Control*, vol. 21, pp. 55–66, 1997.

[10] K. Pereida and A. P. Schoellig, "Adaptive Model Predictive Control for High-Accuracy Trajectory Tracking in Changing Conditions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 7831–7837.

[11] J. P. Allamaa, P. Patrinos, H. Van der Auweraer, and T. D. Son, "Sim2real for Autonomous Vehicle Control using Executable Digital Twin," *IFAC-PapersOnLine*, vol. 55, no. 24, pp. 385–391, 2022, 10th IFAC Symposium on Advances in Automotive Control AAC 2022.

[12] M. Menner, K. Berntorp, and S. D. Cairano, "Automated Controller Calibration by Kalman Filtering," *IEEE Transactions on Control Systems Technology*, pp. 1–15, 2023.

[13] J. C. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins Apl Technical Digest*, vol. 19, pp. 482–492, 1998.

[14] F. Dettù, S. Formentin, and S. M. Savaresi, "The Twin-in-the-Loop Approach for Vehicle Dynamics Control," *IEEE/ASME Transactions on Mechatronics*, pp. 1–12, 2023.

[15] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[16] K. L. Voogd, J. P. Allamaa, J. Alonso-Mora, and T. D. Son, "Reinforcement Learning from Simulation to Real World Autonomous Driving using Digital Twin," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1510–1515, 2023, 22nd IFAC World Congress.

[17] L. P. Fröhlich, E. D. Klenske, C. Daniel, and M. N. Zeilinger, "Bayesian Optimization for Policy Search in High-Dimensional Systems via Automatic Domain Selection," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 757–764, 2019.

[18] X. Hu, S. Li, T. Huang, B. Tang, R. Huai, and L. Chen, "How Simulation Helps Autonomous Driving: A Survey of Sim2real, Digital Twins, and Parallel Intelligence," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 593–612, 2024.

[19] F. Muratore, M. Gienger, and J. Peters, "Assessing Transferability From Simulation to Reality for Reinforcement Learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 4, pp. 1172–1183, 2021.

[20] D. Hartmann and H. V. der Auweraer, "The Executable Digital Twin: merging the digital and the physics worlds," *ArXiv*, vol. abs/2210.17402, 2022.

[21] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald, *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[22] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 477–482, 2000.

[23] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, p. 287–404, 2019.

[24] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.

[25] M. A. Skoglund, F. Gustafsson, and G. Hendeby, "On Iterative Unscented Kalman Filter using Optimization," in *2019 22th International Conference on Information Fusion (FUSION)*, 2019, pp. 1–8.

[26] S. Akhlaghi, N. Zhou, and Z. Huang, "Adaptive adjustment of noise covariance in Kalman filter for dynamic state estimation," in *2017 IEEE Power & Energy Society General Meeting*, 2017, pp. 1–5.

[27] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 3, pp. 817–823, 1998.

[28] ——, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.

[29] J. Maryak and D. Chin, "Efficient global optimization using SPSA," in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, vol. 2, 1999, pp. 890–894 vol.2.

[30] J. C. Spall. (2024) Simultaneous Perturbation Stochastic Optimization selected references on theory, applications, and numerical analysis. [Online]. Available: https://www.jhuapl.edu/spsa/Pages/References-List_Ref.htm

[31] J. P. Allamaa, P. Patrinos, H. Van Der Auweraer, and T. D. Son, "Safety Envelope for Orthogonal Collocation Methods in Embedded Optimal Control," in *2023 European Control Conference (ECC)*, 2023, pp. 1–7.



Jean Pierre Allamaa obtained the B.Eng. degree in Mechanical Engineering from the American University of Beirut, Beirut, Lebanon and the M.Sc. degree in Mechanical Engineering with a specialization in Automatic and Control from École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland in 2018 and 2020 respectively. He is currently a Marie-Curie fellow, working towards his industrial Ph.D. in Electrical Engineering at Siemens and in collaboration with the Department of Electrical Engineering (ESAT) at KU Leuven, Leuven, Belgium.

His research focuses on embedded model predictive control and its intersection with learning, in particular with applications in motion planning and control for autonomous driving and advanced driver assistance systems, in virtual and real environments.



Panagiotis (Panos) Patrinos received the M.Eng. degree in chemical engineering, M.S. degree in applied mathematics, and the Ph.D. degree in control and optimization from the National Technical University of Athens, Athens, Greece, in 2003, 2005, and 2010, respectively. He is currently an Associate Professor with the Department of Electrical Engineering (ESAT), KU Leuven, Leuven, Belgium. In 2014, he was a Visiting Professor with Stanford University, Stanford, CA, USA. After his Ph.D., he was a Postdoc with the University of Trento, Trento, Italy, and IMT Lucca, Lucca, Italy, where he became an Assistant Professor in 2012. His current research interests include the intersection of optimization, control and learning, theory and algorithms for structured nonconvex optimization, and learning-based, model predictive control with a wide range of applications including autonomous vehicles, machine learning, and signal processing.

Dr. Patrinos was the corecipient of the 2020 Best Paper Award in the International Journal of Circuit Theory and Applications.



Herman Van der Auweraer received the M.Sc. degree in electronic engineering (1980) and the Ph.D. degree in engineering science (1987) from the KU Leuven, Belgium. In 1986, he joined LMS International, Leuven, one of the earliest KU Leuven spin-offs, developing advanced testing and simulation tools for mechatronic product design engineering. LMS became part of Siemens in 2013. His research focus is acoustics, sound quality, and system identification. He was Director Research and Technology Innovation until his retirement in 2023. He continues to support the company's innovation strategy as Senior Advisor. Furthermore, he is affiliated to KU Leuven as guest professor.



Tong Duy Son received the PhD as a Marie-Curie fellow from Department of Mechanical Engineering, KU Leuven, Belgium in 2016. Since then, he is a senior researcher and an R&D ADAS Manager at Siemens Digital Industries Software, active in European Union and Belgian research programs and supervision of industrial PhDs. His research focuses on autonomous driving control, AI algorithms development, testing and validation methodologies in both virtual and physical environment.

Dr. Tong was the recipient of the Siemens DF PL Invention of the Year Award.