Preprints, 10th IFAC International Symposium on
Advances in Automotive Control
Columbus, Ohio, USA, August 28-30, 2022

Sim2real for Autonomous Vehicle Control using Executable Digital Twin

Jean Pierre Allamaa [*,**] **Panagiotis Patrinos** [**]
Herman Van der Auweraer [*] **Tong Duy Son** [*]

[*] *Siemens Digital Industries Software, 3001, Leuven, Belgium (e-mail: jean.pierre.allamaa@siemens.com)*
[**] *Dept. Electr. Eng. (ESAT) - STADIUS research group, KU Leuven, 3001 Leuven, Belgium*

Abstract: In this work, we propose a sim2real method to transfer and adapt a nonlinear model predictive controller (NMPC) from simulation to the real target system based on executable digital twin (xDT). The xDT model is a high fidelity vehicle dynamics simulator, executable online in the control parameter randomization and learning process. The parameters are adapted to gradually improve control performance and deal with changing real-world environment. In particular, the performance metric is not required to be differentiable nor analytical with respect to the control parameters and system dynamics are not necessary linearized. Eventually, the proposed sim2real framework leverages altogether online high fidelity simulator, data-driven estimations, and simulation based optimization to transfer and adapt efficiently a controller developed in simulation environment to the real platform. Our experiment demonstrates that a high control performance is achieved without tedious time and labor consuming tuning.

Keywords: Sim2Real, ADAS, model predictive control, domain randomization

1. INTRODUCTION

With the advancement of control and planning algorithms for autonomous driving, a challenge still remains in the transfer from simulation to the real world. Manual tuning to validate the design requirements in a physical environment with various scenarios is time-consuming and expensive. The automotive industry is trying to leverage more simulation to reduce the physical tuning efforts. Digital Twin (DT) of a vehicle is an accurate and reliable representation model in a complex high degrees of freedom simulation system that supports the vehicle control algorithm development and validation processes (Van der Auweraer et al. (2018)). DTs allow us to test the autonomous driving (AD) algorithms in several configurations, traffic scenarios, and edge cases. Nevertheless, this often comes at the expense of over tuning in simulation hindering transferability to the real world. In recent advancements of control strategies for safe AD, based on reinforcement learning, neural networks, or non-linear model predictive control, transferring and embedding the development from simulation to real systems is a bottleneck. Moreover, no simulator guarantees perfect reliability given a reality gap (sim2real gap) due to erroneous sensor readings, observability of the system, actuation and process noise, and importantly an external environment different from simulation to reality.

Research from the robotics and reinforcement learning communities is currently tackling the problem of transferability. Under the umbrella of transfer learning, three main methods deal with sim2real: domain randomization, domain adaptation, and high-fidelity simulation. In Müller et al. (2018), end-to-end driving policies are transferred from simulation to reality via modularity and abstraction: the learning process becomes indifferent to the environment. Muratore et al. (2021) discuss the presence of a simulation optimization bias (SOB) that overestimates the maximum expected return in a Markovian decision process trained in simulation, in comparison with the real performance. The algorithm is then trained to compensate for the SOB by randomizing the environment. Relying more on high-fidelity simulation, Kadian et al. (2019) run parallel tests in simulation and real-world while varying the simulation parameters. The authors then measure the success rate in both worlds to find the parameters correlating both domains the most. Eventually, the reliability of the simulator is evaluated using a sim-vs-real correlation coefficient (SRCC). In general, most of the presented methods rely on a large amount of data and scenarios to train the algorithm. Aiming more towards online adaptation, Kapteyn et al. (2021) show the importance of a reliable predictive model in the shape of a DT incorporating real-world data. They use a graphical model method to describe the evolution of the DT dynamical systems. The DT is integrated for dynamic decision making in a predictive process to monitor the health and safety of the unmanned aerial vehicle.

The main contribution in this work is to suggest a framework based on an executable digital twin (xDT), combining randomization and adaptation with real data to transfer a control strategy from sim2real. The xDT is fit for control purposes and can be implemented in a vehicle ECU as it is a self-contained DT model for a specific runtime context (Hartmann and Van der Auweraer (2020)). In this work, we estimate the non-static control parameters on the real system by sampling the xDT, in

presence of noise and uncertain model parameters. We then gradually move towards parameters that improve performance, by combining stochastic gradient approximation in simulation with data-driven approaches in the form of Unscented Kalman Filter. The adaptation method requires low computational effort and low data storage on memory, making it interesting for embedded hardware deployment real-time execution. It allows the control and planning strategies to adapt to a changing environment, situations, and driving styles by automatically tuning and calibrating the parameters. The proposed framework can deal with both unseen and edge-case scenarios by learning adequate control parameters without carrying an overhead of historical data and explicit policies. By leveraging simulation-based optimization methods and data-driven approaches we reduce tuning efforts and time as we avoid trial and error tuning campaigns. This method can improve testing automation in the automotive industry for tuning and validation of a controller or planner for Advanced Driver Assistance Systems (ADAS).

The paper is organized as follows. Section II discusses some background on sim2real methods for autonomous systems, as well as two gradient-free methods used to optimize over the controller parameters in the real world. Section III presents the implementation and results of the proposed methodology in automatic tuning for an autonomous driving application. We extend the implementation with a combination of two automatic calibration methods in Section IV, and discuss their convergence.

2. BACKGROUND

There is a deep belief that increasing the simulator or the digital twin's accuracy alone, will not decrease the gap between simulation and reality. Muratore et al. (2021) discuss an SOB in machine learning and specially in trained models with reinforcement learning. The maximized expected return is generally overestimated in simulation. This causes the trained algorithm to be less performant, less robust, and more prone to failure in the real world. In this section, we quickly introduce the concepts of domain randomization and adaptation, and we discuss the xDT. Furthermore, we give a short technical background about gradient-free methods that could be used for automatic tuning. Finally, we formulate the path following NMPC.

2.1 Sim2Real in Autonomous Systems

To prevent overfitting and over-optimizing the algorithm in simulation, a technique called Domain Randomization (DR) is introduced. It consists of adding perturbation to the simulator's physical parameters (e.g mass, inertia, length, friction coefficients), noise to the control actions and state estimations, and disturbances to the visual properties, in training. This method regularizes the algorithm and robustifies it against real-world uncertainties. The agent trains to maximize the expected return over a distribution of uncertain parameters, delays, and noise levels. Another method of tackling sim to real problems is Domain Adaptation (DA). This transfer learning method tackles the ability to train an algorithm in one source domain and deploy it on another, possibly different, target domain. This method learns to match the distributions in the target and source domain to reach a domain-invariant feature representation, which is used as an input to the trained agent.

To our knowledge, there is no single framework that combines the benefit of the three methods (DR, DA, and high-fidelity simulations) into one, allowing a simple transfer from simulation to reality. We introduce a learning method to learn the distribution in the domain randomization and update it on the fly. This could could be beneficial when parameters are not well defined in advance (system identification) and in the framework of learning the hyperparameters of a learning algorithm (meta-learning in Parker-Holder et al. (2022)). The goal is not to find parameters solving a single task well but rather generalizing to several tasks, with automatically adapted parameters.

2.2 Executable Digital Twin

Simulations are a main pillar in an X-in-the-loop (XiL: Model-in-the-loop, Software-in-the-loop, Hardware-in-the-loop, Vehicle-hardware-in-the-loop) framework. MiL is a major requirement in the automotive industry for the development and validation processes of control and planning algorithms as they allow to minimize the risk and effort in real-life testing. Reliable simulations can contribute to reducing the real mileage needed for algorithm validation (Son et al. (2017)). There is also a trade-off between analytical models, which are often simple and differentiable, and black-box models of high-fidelity simulators. The DT integrates data and other knowledge of a physical asset and bridges the physical and the virtual world to assess performance, predict the behavior, and optimize the service (Hartmann and Van der Auweraer (2020)). We use Simcenter Amesim to design the high-fidelity DT. Sim2real addresses the transfer of an algorithm from simulation with such DT to reality, without losing its performance, but more importantly, while keeping the real system stable and safe. The xDT is an executable representation of a complex non-linear dynamical system or asset, in our case a road vehicle (Hartmann and Van der Auweraer (2020)). It can be embedded on hardware, and run in a real-time environment. We make use of xDT to evaluate online the effect of control action and exploit learning-based controllers while easily varying model and environment parameters in an efficient, quick and safe way. According to Hartmann and Van der Auweraer (2020) "[xDT] can be instantiated on the edge, on premise, or in the cloud and used autonomously by a non-expert or a machine through [...] APIs". The xDT allows us to test configurations that are rather complex to recreate in real-life, to validate in XiL process by exploiting dynamics and prediction that are often not captured well with simplistic analytical ODE car models. Moreover, the xDT allows us to update the prior knowledge about our system online.

2.3 Gradient-free randomization methods

We present two gradient-free methods that can be used to approximate the optimal set of control parameters on the real car. The adaptation schematic is shown in Figure 1. The NMPC parameters θ are adapted online to minimize the performance error $h_{ref} - \tilde{h}(\theta_k)$, by sampling the xDT. We add randomization to θ_k and the model parameters.

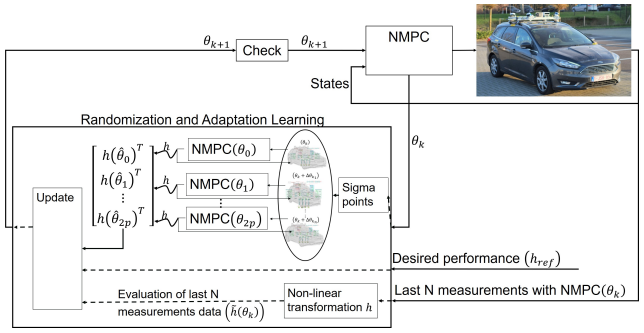


Fig. 1. Control parameter adaptation on an xDT: the main NMPC parameters are updated by sampling the xDT with NMPC(θ_j), evaluating the predicted performance h_j, and including the real measured performance $\tilde{h}$ of the last horizon N

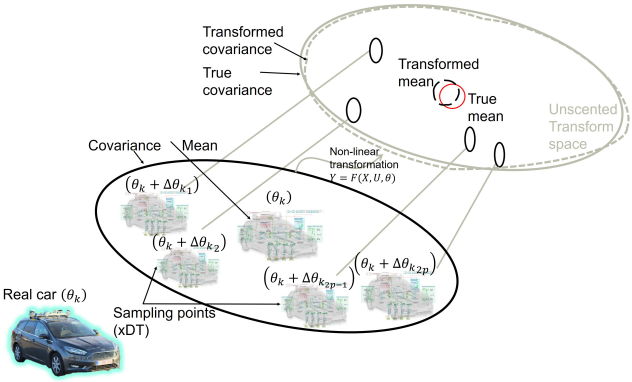


Fig. 2. Sampling points in an UKF: The mean and covariance of the transformed parameter distribution are approximated by sampling the xDT with $2p$ points around the current parameter set θ_k and measuring the respective outputs

Simultaneous Perturbation Stochastic Approximation (SPSA) Spall (1998) introduces a stochastic approximation algorithm for stochastic optimization in multivariate systems. This method approximates the gradient of a loss function with respect to a system parameter with only two measurements, real or from simulation. Regardless of the system dimensionality, the gradient in the objective function can be approximated by perturbing all the parameters to be optimized over at once. For systems where the analytical relationship between an objective function and the parameters is unknown or difficult to develop, this method could be of significant benefit. This recursive optimization method significantly cuts down the number of iterations needed to estimate a gradient. SPSA seeks to minimize a loss function $L(\theta)$ where θ is a p-dimensional vector. We approximate the gradient $g(\theta) = \nabla L(\theta)$ as direct measurements of it are assumed non viable. The simultaneous perturbation step has all p-elements of θ_k perturbed at once. At an instance k, we run two perturbed simulations out of which two measurements of L are sufficient to calculate the gradients with respect to every $j = 1, \ldots, p$ parameter such that:

$$\hat{g}_{kj}(\hat{\theta}_k) = \frac{\partial L}{\partial \theta_{kj}} = \frac{L(\theta_k + c_k\Delta_k) - L(\theta_k - c_k\Delta_k)}{2c_k\Delta_{kj}} \quad (1)$$

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. Spall (1998) suggests a random perturbation vector $\Delta_k = [\Delta_{k1}, \ldots, \Delta_{kp}]^T$, following a Bernoulli distribution symmetric about zero, with mutually independent elements. As opposed to other gradient-based methods requiring p simulations, SPSA requires only 2, leading to considerable savings when p is large. The gradient is approximated as:

$$\hat{g}_k(\hat{\theta}_k) = \frac{L(\theta_k + c_k\Delta_k) - L(\theta_k - c_k\Delta_k)}{2c_k} \begin{pmatrix} \Delta_{k1}^{-1} \\ \vdots \\ \Delta_{kp}^{-1} \end{pmatrix} \quad (2)$$

The general form in a recursive SPSA algorithm to solve for $\partial L/\partial\theta_k = 0$, with a step size a_k, is:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k\hat{g}_k(\hat{\theta}_k) \quad (3)$$

Unscented Kalman Filter In the work of Menner et al. (2021), an Unscented Kalman Filter method is employed to estimate the control parameters in PID, state-feedback, neural networks, and optimal controller,s and validate it with a vehicle simulator. An Unscented Kalman Filter propagates the parameters through non-linear dynamics and updates the estimated parameters without relying on gradient measurements or back-propagation. The method samples a set of points around the mean called sigma points, which are used to predict the output of the model. However, their sampling method predicts the system output using a simple bicycle model simulator. In our approach, we propose the closed-loop propagation of the perturbed parameters through a high-fidelity xDT. The predicted performance is closer to the real-world data, contributing to a quicker and safer convergence of the approach given a small sim2real reality gap. In addition, to add domain randomization, the model parameters in the xDT are perturbed and follow a normal distribution. Contrary to the extended Kalman filter (EKF), the UKF does not linearly approximate the dynamics and output functions around the mean of the Gaussian to predict the values. This makes it attractive in a framework containing high-fidelity black-box models and non-linear evaluation metrics. For this, we first define the governing dynamics: let $\mathcal{F}$ be a possibly non-linear transformation defining the system dynamics and stacking the states from time k to $k+N$ where N is a chosen horizon. $\mathcal{H}$ is a non-linear, possibly non-analytical and non-differentiable output transformation, serving as an evaluation metric map. V and N are the process and output noise.

$$X_{k|k+N+1} = \mathcal{F}(X_k, U_{k|k+N}, V_{k|k+N}, \theta_k) \quad (4a)$$

$$Y_{k|k+N} = \mathcal{H}(X_{k|k+N}, U_{k|k+N}, \theta_k) + N(k|k+N) \quad (4b)$$

Let $Y_{ref,k}$ be the stacking of reference evaluation metric from time $k-N$ to k. In our approach, $f(x, u, v, \theta)$ is the output of a black-box, xDT of the real vehicle. Given a p-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 sigma or sampling points Θ around the mean $\Theta = [\theta_k|\theta_k + c_kA^j|\theta_k - c_kA^j] \in \mathbb{R}^{p\times 2p+1}$:

$$\Theta^0 = \theta_k, \quad (5a)$$

$$\Theta^j = \theta_k + c_kA^j, \qquad j = 1, \ldots, p \quad (5b)$$

$$\Theta^j = \theta_k - c_kA^{j-p}, \quad j = p+1, \ldots, 2p \quad (5c)$$

where $c_k = \sqrt{p+\lambda}$, and A^j is the jth column of the matrix $A = \sqrt{P_{k|k}}$. Matrix A can be computed by performing

Algorithm 1 Automatic parameter estimation algorithm

Require: $\Theta, w_a, C_\theta, C_n$
  function UNSCENTED TRANSFORM
    $\bar{\theta} = \sum_{j=0}^{2p} w_a^j \Theta^j$
    $P_{k+1|k} = C_\theta + \sum_{j=0}^{2p} w_a^j (\Theta^j - \bar{\theta})(\Theta^j - \bar{\theta})^T$
    $\mathcal{Y}^j = h(\theta^j, x_{k-N})$ ▷ Propagate
    $\bar{y} = \sum_{j=0}^{2p} w_a^j \mathcal{Y}^j$
  end function
  function MEASUREMENT UPDATE STEP
    $P_{\theta y} = \sum_{j=0}^{2p} w_a^j (\Theta^j - \bar{\theta})(\mathcal{Y}^j - \bar{y})^T$
    $P_y = C_n + \sum_{j=0}^{2p} w_a^j (\mathcal{Y}^j - \bar{y})(\mathcal{Y}^j - \bar{y})^T$
  end function
  $K_k = P_{\theta y} P_y^{-1}$ ▷ Kalman gain
  $P_{k+1|k+1} = P_{k+1|k} - K_k P_y K_k^T$ ▷ Posterior covariance

a Cholesky decomposition of the prior covariance matrix such that $P_{k|k} = AA^T$. The hyper-parameter λ dictates the spread of the sampling (sigma) points around the mean. In total, $2p + 1$ time evolutions are performed for one update step. The Unscented Transformation is based on the weighted mean of all the sigma points and their covariance. We form the weighting vector $\mathcal{W} = [w_a^0, w_a^1, \ldots, w_a^{2p}] \in \mathbb{R}^{1\times 2p+1}$ such that:

$$w_a^0 = \frac{\lambda}{(p+\lambda)}, \; w_a^j = \frac{1}{2(p+\lambda)} \quad \text{for } j = 1, \ldots, 2p, \tag{6}$$

where w_a^j is the weight associated with the jth point, and w_a^0 is the weight associated with the first sigma point, which is the mean θ_k. A good heuristic for choosing λ according to Julier et al. (2000), would be $p + \lambda = 3$. The step $\mathcal{Y}^j = h(\theta^j, x_{k-N})$ in algorithm 1 propagates the sigma points through the non-linear transformations (dynamics and evaluation) starting with the initial condition state x_{k-N} until time k. This method seeks to find the set of control parameters that would have improved the performance from time $k - N$ to k where N is the length of the data from the real system. The control parameters are then updated according to algorithm 1 and the law:

$$\theta_{k+1} = \theta_k + K_k(\mathcal{Y}_{ref,k} - \tilde{h}(\theta_k)), \tag{7}$$

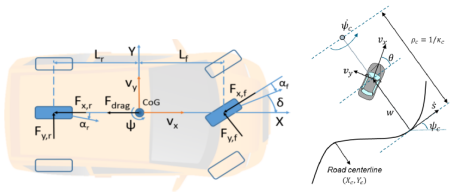
where $\tilde{h}(\theta_k)$ is the vector-valued evaluation metric from the real system (Menner et al. (2021); Wan and Van Der Merwe (2000)). C_n and C_θ are output and process noise covariance matrices.

2.4 NMPC: Path following formulation

Car dynamics used in the NMPC are represented with a real-time feasible model such as the 6 DoF single-track model described in Allamaa et al. (2022), satisfactory in path following and lane keeping scenarios.

$$\begin{aligned}
\dot{v}_x &= (F_{xf}cos\delta + F_{xr} - F_{yf}sin\delta - F_{res} + M\dot{\psi}v_y)/M,\\
\dot{v}_y &= (F_{xf}sin\delta + F_{yr} + F_{yf}cos\delta - M\dot{\psi}v_x)/M,\\
\dot{r} &= (L_f(F_{yf}cos\delta + F_{xf}sin\delta) - L_rF_{yr})/I_z,\\
\dot{s} &= (v_x\cos\theta - v_y sin\theta)/(1-\kappa_c w),\\
\dot{w} &= v_x\sin\theta + v_y\cos\theta,\\
\dot{\theta} &= \dot{\psi} - \dot{\psi}_c = \dot{\psi} - \kappa_c\dot{s}.
\end{aligned} \tag{8}$$

In the single-track vehicle model of (8) and Figure 3, the first three equations dictate the dynamics in the car body frame with x pointing forward and last three dictate the kinematics in the curvilinear frame. We add the state s to track the evolution along the center-line. Inputs to the model are the body frame steering angle δ and the front/rear axles longitudinal forces F_{xf} and F_{xr}. At moderate speeds, a linear tire model estimates the lateral forces F_{yf} and F_{yr} as proportional to the slip angles α_f, α_r, assuming small slip angles using constant cornering stiffness. For control purposes, it is beneficial to represent the car position with respect to the track and path, in the curvilinear reference frame, local and attached to the car body frame as in Figure 3(b). Each point on a path is defined by a 4-tuple $X_c, Y_c, \psi_c, \kappa_c$ to represent the position, heading, and a curvature in the Cartesian frame. The transformation to a curvilinear frame facilitates the optimal control problem (OCP) formulation as it is only function of the road curvature. Finally, track limits can be added as varying box constraints as w_l and w_r, the left and right distances from the center-line. w and θ are respectively the distance and heading deviation from the center-line, with $w > 0$ to the left of the center-line and $\theta > 0$ a counter-clockwise rotation with respect to the tangent to the center-line ψ_c.



(a) Single track model (b) Curvilinear frame transformation

Fig. 3. Single track curvilinear model for path following

$$\begin{aligned}
w &= (Y - Y_c)\cos(\psi_c) - (X - X_c)\sin(\psi_c),\\
\theta &= \psi - \psi_c.
\end{aligned} \tag{9}$$

To summarize, the single-track dynamics between the state vector x and input u, in the Curvilinear frame is:

$$\dot{x} = f(x, u), \; x = [v_x, v_y, r, s, w, \theta], \; u = [\delta, t_r]. \tag{10}$$

NMPC controls the car by computing the normalized throttle t_r, and the steering angle δ. We use a receding horizon scheme of N_H steps, and we augment the dynamics with the input rates variables $\dot{u}$ for smoother driving. Moreover, we obtain the nonlinear difference equations f_d by applying a 4 step Runge-Kutta 4^{th} order method to the dynamics $\dot{x} = f(x, u)$ in (10). Finally, the NLP optimizes over the discrete-time OCP for path following as in (11).

$$\min_{x(0),\ldots,x(N),u(0),\ldots,u(N-1)} \sum_{k=0}^{N-1} l_k(x_k - x_{ref,k}, u_k) + V_N(x_N) \tag{11}$$
$$\begin{aligned}
\text{subject to: } & x_0 = x(0) && \text{(Initial condition)}\\
& x(k+1) = f_d([x(k), u(k)], \dot{u}(k)) && \text{(Dynamic equations)}\\
& x_{min} \le x(k) \le x_{max} && \text{(State constraints)}\\
& u_{min} \le u(k) \le u_{max} && \text{(Input constraints)}\\
& \dot{u}_{min} \le \dot{u}(k) \le \dot{u}_{max} && \text{(Input rate constraints)}
\end{aligned}$$

$V_N(x_N)$ is a terminal cost and the stage cost $l_k(x_k, u_k)$ is:

$$l_k(x_k, u_k) = x(k)^T Q x(k) + u(k)^T R u(k) + \dot{u}(k)^T S \dot{u}(k)^T, \tag{12}$$

with $Q \in \mathbb{R}^{6\times6} \succeq 0, R \in \mathbb{R}^{2\times2} \succ 0, S \in \mathbb{R}^{2\times2} \succ 0$. The path following problem reduces to a regulation about a zero reference for all states except the velocity.

3. IMPLEMENTATION FOR AUTONOMOUS DRIVING APPLICATIONS

In order to demonstrate the benefit of an automatic adaptation sampling an xDT, we implement our methodology on an autonomous driving application. While our approach is not restricted to tuning controllers, or to a specific type of controllers, we demonstrate it on a path following NMPC. This section formulates a performance-improver automatic calibrator. We build on the XiL verification process described in the work of Allamaa et al. (2022) for the deployment of a real-time NMPC on embedded platforms. The NMPC is solved to full convergence using a Sequential Quadratic Programming (SQP) method with an active-set QP solver. We optimize over the resulting NLP using a multiple-shooting framework, with a sampling time $T_s = 40ms$ and an NMPC horizon $N_H = 30$.

3.1 Problem Formulation

The proposed automatic adaptation method can calibrate controllers of an unknown dynamical system online, rendering it attractive for embedded deployment in a real-world system as a sim2real method. For our implementation, the controller learns to track a parameterized reference path and velocity profile, in 4 successive lane changes. The function $\mathcal{H}$ in (4) converts the measured output over the past time window of size N, to a vector-valued performance metric as shown in the framework of Figures 1 and 2. As the single-track dynamics are formulated in a curvilinear frame with respect to the path, deviation from the path center-line is the state w. We then form the performance vector $h(\theta_k)$ such that:

$$h(\theta_k) = \begin{bmatrix} 10(\mathcal{V}_x - \mathcal{V}_{ref}) \\ 10(w_{k-N|k}) \\ \mathcal{J}^*_{NMPC} \end{bmatrix}, \mathcal{Y}_{ref,k} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad (13)$$

where $\mathcal{V}_x - \mathcal{V}_{ref} = v_{x,k-N|k} - v_{ref,k-N|k}$ is a vector stacking of velocity tracking errors over the past time window. Similarly for the deviation from the path $w_{k-N|k}$. In addition, we include the stacking of the NMPC's optimal cost $\mathcal{J}^*_{NMPC}$ such that the algorithm minimizes it to keep the energy bounded. The factor of 10 in the tracking errors is added for normalization. The objective performance in this implementation leads to a tradeoff between reducing Q, R, S such that $\mathcal{J}^*_{NMPC} \to 0$, and increasing Q, R, S to minimize the velocity and path tracking errors. Another way of implementing this, is to set an activation function:

$$r(J^*) = \max(J^*, \underline{\mathrm{J}}) - \underline{\mathrm{J}} \quad (14)$$

where $\underline{\mathrm{J}}$ is a threshold for an allowable optimal cost that keeps the system stable. For this particular implementation, we are interested in tuning the controller parameters, specifically the Q, R, and S matrices in (12). This method can be also used to optimize offline over the horizon length N_H of the NMPC resulting in the best performance and computation time. Increasing the weights on the OCP variables could render the behavior slightly more aggressive and dynamic, and potentially increase the number of SQP and QP iterations needed to solve for the primal and dual estimates. Therefore, we can augment the vector-valued objective metric by the NMPC computation time, to reach a trade-off between tracking performance and real-time feasibility. Adding a performance metric such as the computation time helps demonstrate the benefit of this method in dealing with black-box, non-analytical, and non-differentiable by design, choice of measurements. We choose a starting hyperparameter $\lambda = 1$, $w_0 = 1/3$ and a horizon length of 3 seconds to update the controller.

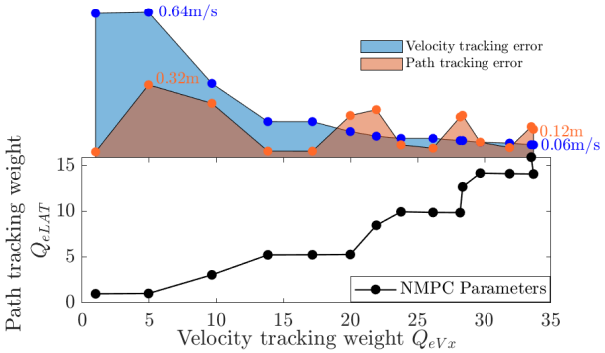


Fig. 4. Automatic tuning without noise: Control parameter evolution and Infinity norm of tracking error

Regularization heuristic As the sampling points are calculated in the Unscented Transform step, it could occur that the algorithm tries to run a simulation with indefinite Q, R, S matrices resulting from negative parameters in θ. While Menner et al. (2021) argues that the method learns to stay away from indefinite matrices, we add a regularization process that guarantees the positive definiteness of the parameters. As we sample with an xDT, this allows avoiding infeasible and unstable roll-outs. The regularization finds the smallest step size c that causes the sampling point to activate the high or low boundary constraints $\bar{\theta}$ and $\underline{\theta}$. This regularization method does not clip the parameters individually to their limits, but rather finds a coefficient that conserves the shape of Gaussian distribution around the mean. The coefficient is calculated such that: $c_k = \min(c_0, c|\underline{\theta} \leq \theta_k + cA^j \leq \bar{\theta}\ \&\ \underline{\theta} \leq \theta_k - cA^j \leq \bar{\theta})$, where $c_0 = \sqrt{p+\lambda}$. We also update the process noise covariance matrix using real-world data. C_n is updated using the stacking of the measured slack variables $\tilde{N} = \mathcal{Y}_{ref,k} - h(\theta_k)$ from real measurements.

3.2 Example of successive double lane changes

We validate our methodology with 4 successive ISO 3888-1 standard double lane change scenarios at 80kph. We show how such a formulation could facilitate the task of controller adaptation. We start with a unit and diagonal Q, R, and S matrices, with no prior knowledge. For sake of simplicity, we visualize two elements of the Q matrix, namely Q_{eLAT} and Q_{eVx} which are the weights on the lateral deviation error and velocity tracking error respectively, as seen in Figure 4 along with the infinity norm of the tracking errors. Every dot represents newly updated parameter set, evolving with time from left to right. At the end of the scenario, with only one run, the algorithm adapts the control parameters such that the

velocity tracking error drops by almost 91% and the path tracking error by 63%. From the second update step, the algorithm detects the model mismatch in the longitudinal dynamics between the single-track model and the xDT and increases the weight on the velocity to 10. This plot allows us to visualize the sensitivity of the tracking error norm to the change in parameters. In those sections with little to no steering (Figure 5), the algorithm does not learn about the lateral deviation. The automatic tuning updates Q_{eLAT} mainly during the lane change, as seen in the peaks of the path tracking error in Figure 4. Every peak corresponds to a double lane change, and it is clear that the error peak norm decreases as the scenario evolves.

Figure 5 shows the closed-loop result of the scenario with and without automatic tuning using an xDT. Green dots correspond to an update instance (every 3 seconds). In the first three seconds, as the gains are set to unity, the velocity drops considerably, to which the algorithm reacts by increasing the weight on the velocity tracking error. This results in an increase in throttle, all while still satisfying the constraints of the NMPC. The switch leads to a peak in the optimal cost but it is followed by a quick decrease. This shows the potential of this method, as it improves the tracking performance and keeps the NMPC optimal cost bounded.

4. RESULTS AND DISCUSSION

As we deal with an adaptive controller, it is important to add safety checks to ensure that the system is not diverging, and to verify the applicability of the updated control parameter set. In this section, we discuss an energy-based study to validate the controller. Moreover, we compare the UKF-data-driven approach to a simulation-based-optimization using SPSA. Finally, we combine both methods and show potential results.

4.1 Convergence analysis

For applications with highly non-linear dynamics and in presence of noise, theoretical Lyapunov stability is hard to prove. However, the xDT allows us to quickly verify the new controller and calculate its associated benefit in energy, if we can find a function $V(x) = x^T P x$ where $P \succ 0$ such that the $V(x(k+1)) \leq V(x(k))$. In a less strict sense, given the noise sources, we aim to find a non-monotonically decreasing energy function contained within a ball of radius R. The first controller check is such that $V(P_{k+1}, x_{k-N|k}) \leq V(P_k, x_{k-N|k})$ where $P_k = diag(\theta_k) \succ 0$, is the diagonal matrix constructed from the entries of the tuned controller parameters: the estimated θ_{k+1} would have reduced the energy function if it were applied instead of θ_k starting $x(k-N)$. The second validation is through applying the updated controller parameter to the real vehicle and verifying that the resulting energy function in $[k, \ldots, k+N]$ is non-increasing.

SPSA perturbs the parameter (sampling points) with $\Delta_{k,i}$ following a Bernoulli distribution with $p = 0.5$, with all p-parameters being mutually independent. We set $a_k = 0.05, c_k = 0.1$ as we try to estimate non-static parameters. The parameters are perturbed and propagated through the xDT, similar to the proposed tuning method in the

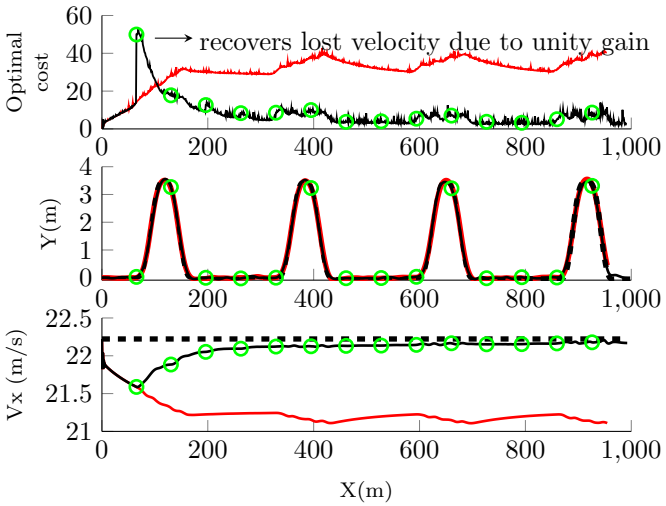


Fig. 5. Convergence of NMPC cost function (with automatic tuning: solid-black, without automatic tuning: red, reference: dashed-black, tuning instances: green)

previous section. We add an additive output noise with a signal-to-noise ratio of 2dB. To enhance the gradient approximation in presence of noise, we run $2p$ sets of perturbed parameter simulations, symmetric with respect to 0 such that $\Delta_{k,i} = \pm 1$ and $\Delta_{k,i} = \pm 2$. We then average the approximated gradients from simulation and update the control parameter according to (3). However, as this method performs a simulation-based optimization, it does not take into consideration real world data and can perform at best to compensate for noise and mismatch between the NMPC's single-track model and the xDT. We add it to our approach as a comparison. To leverage simulation and real-world data, we combine both methods. The UKF technique propagates the dynamic with $2p+1$ set of parameters, sampled according to the prior knowledge on distribution $P_{k|k}$ and noise level. We make use of the measured performance metric $\mathcal{Y}$ to approximate the gradient around the current θ_k using SPSA. We then average the steps $\Delta\theta_k$ from both methods. The SPSA method optimizes over a scalar loss function L. We set $L(\theta_k) = \|h(\theta_k)_{1:N}\|_2^2 + \|h(\theta_k)_{N:2N}\|_2^2$ where $\|h(\theta_k)_{1:N}\|_2$ is the 2-norm of the h vector containing the velocity tracking error and $\|h(\theta_k)_{N:2N}\|_2$ contains the path tracking errors. The SPSA method is quicker to react to the path tracking error compared to the UKF, and is less sensitive to the velocity tracking error as seen in Figure 7. It also results in a smaller tracking error. However, combining both methods results in a quicker reaction, from the first iteration, and simultaneously to both tracking errors showing the benefit of updating the estimations with real data, all while employing the DT to optimize over the parameters. Overall, velocity tracking error drops from $1m/s$ to around $0.1m/s$ for the SPSA, UKF and UKF+SPSA. Infinity norm on the path tracking drops from 0.26m to 0.12m with UKF+SPSA. Both methods have similar trends and effects on the NMPC optimal cost. Another important aspect to consider is the noise effect as we compare the UKF in Figures 4 and 6. As the algorithm has prior and posterior knowledge about the noise level, it adapts less aggressively to tracking error measurements which are mainly due to noise. Q_{eLAT} is 15 without noise

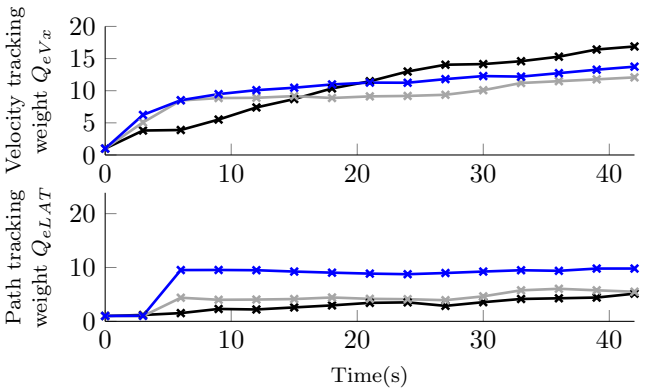


Fig. 6. Automatic tuning: control parameters evolution (UKF: black, SPSA: grey, UKF+SPSA: blue)

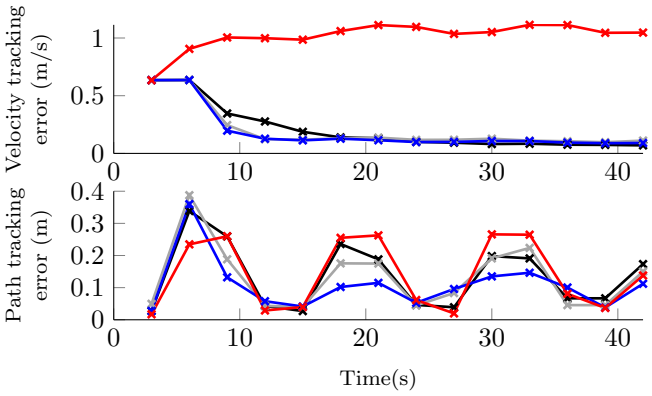


Fig. 7. Automatic tuning: infinity-norm tracking error evolution (UKF: black, SPSA: grey, UKF+SPSA: blue, No tuning: red)

and 5 with noise at the end of the scenario. The control parameters adaptation is shown in Figure 6.

5. CONCLUSION

This paper presents a learning-based adaptation framework for transferring control and planning strategies from simulation to the real setup without manual tuning. The adaptation occurs online and on-the-go as the optimal parameters are estimated to minimize an objective performance error. The proposed algorithm automatically tunes the parameters, to unseen noise levels, edge-case situations and environment, by closing the loop on the prediction using an executable digital twin with real-world data. Unscented Kalman Filter is used to sample a set of perturbed parameters and propagate them through complex non-linear predictive systems. SPSA minimizes the performance error by approximating the performance gradient with respect to the parameters using only 2 simulations regardless of the dimension of the parameters. We validate the proposed algorithm on 4 successive lane changes and show that it estimates the parameters in only one online run, improving the performance up to 91%.

ACKNOWLEDGEMENTS

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

Allamaa, J.P., Listov, P., Van der Auweraer, H., Jones, C., and Son, T.D. (2022). Real-time nonlinear MPC strategy with full vehicle validation for autonomous driving. In *2022 American Control Conference (ACC)*, 1982–1987.

Hartmann, D. and Van der Auweraer, H. (2020). Digital twins. *CoRR*, abs/2001.09747.

Julier, S., Uhlmann, J., and Durrant-Whyte, H. (2000). A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3), 477–482. doi: 10.1109/9.847726.

Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva, M., Chernova, S., and Batra, D. (2019). Are we making real progress in simulated environments? measuring the sim2real gap in embodied visual navigation. *CoRR*, abs/1912.06321.

Kapteyn, M.G., Pretorius, J.V.R., and Willcox, K.E. (2021). A probabilistic graphical model foundation for enabling predictive digital twins at scale. *Nature Computational Science*, 1, 337–347. doi:10.1038/s43588-021-00069-0.

Menner, M., Berntorp, K., and Di Cairano, S. (2021). Automated controller calibration by Kalman filtering.

Müller, M., Dosovitskiy, A., Ghanem, B., and Koltun, V. (2018). Driving policy transfer via modularity and abstraction. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, 1–15. PMLR.

Muratore, F., Gienger, M., and Peters, J. (2021). Assessing transferability from simulation to reality for reinforcement learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(4), 1172–1183. doi:10.1109/TPAMI.2019.2952353.

Parker-Holder, J., Rajan, R., Song, X., Biedenkapp, A., Miao, Y., Eimer, T., Zhang, B., Nguyen, V., Calandra, R., Faust, A., Hutter, F., and Lindauer, M. (2022). Automated reinforcement learning (autorl): A survey and open problems. *CoRR*, abs/2201.03916.

Son, T.D., Hubrechts, J., Awatsu, L., Bhave, A., and Van der Auweraer, H. (2017). A simulation-based testing and validation framework for ADAS development. In *Transport Research Arena*.

Spall, J.C. (1998). An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins Apl Technical Digest*, 19, 482–492.

Van der Auweraer, H., Donders, S., Hartmann, D., and Desmet, W. (2018). Simulation and digital twin for mechatronic product design. 3547–3565. Desmet, W, KATHOLIEKE UNIV LEUVEN, DEPT WERKTUIGKUNDE.

Wan, E. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 153–158. doi: 10.1109/ASSPCC.2000.882463.