2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

From data to control: a two-stage simulation-based approach

Federico Dettù*, Braghadeesh Lakshminarayanan†, Simone Formentin* and Cristian R. Rojas†
*Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano.
Emails: federico.dettu@polimi.it (F. D.), simone.formentin@polimi.it (S. F.).
†Division of Decision and Control Systems, KTH Royal Institute of Technology.
Emails: blak@kth.se (B. L.), crro@kth.se (C. R.).

***Abstract*—For many industrial processes, a digital twin is available, which is essentially a highly complex model whose parameters may not be properly tuned for the specific process.**

By relying on the availability of such a digital twin, this paper introduces a novel approach to data-driven control, where the digital twin is used to generate samples and suitable controllers for various perturbed versions of its parameters. A supervised learning algorithm is then employed to estimate a direct mapping from the data to the best controller to use. This map consists of a model reduction step, followed by a neural network architecture whose output provides the parameters of the controller.

The data-to-controller map is pre-computed based on artificially generated data, but its execution once deployed is computationally very efficient, thus providing a simple and inexpensive way to tune and re-calibrate controllers directly from data. The benefits of this novel approach are illustrated via numerical simulations.

I. Introduction

Most industrial processes require regulation via feedback controls to operate at high efficiency. The exact dynamics of the process is often not available and practitioners come up with suitable mathematical models to incorporate the dynamics of such processes. Once a suitable mathematical model is identified, it is used to design the controller for the actual process. Nowadays, highly sophisticated *digital twins* (DT) are often used for control design [1], *e.g.* in the automotive industry. These models can be used to cheaply generate large amounts of data, and are typically used for testing and calibrating controllers.

However, although DT are extremely faithful, true parameters of a real system might change over time, and robustness or control adaptation is necessary. In this work, we leverage the availability of large amounts of available simulated data – and of the prior knowledge of some control design rule – to *learn* a law directly mapping fresh data to the control parameters. In a sense, we *meta-learn* the control design process.

The Two-Stage (TS) approach [2] is a simulation-based approach to estimate the unknown parameters of a white-box model. It is an inverse supervised learning algorithm, where the white-box model is treated as a simulator: for each fictitious choice of its parameters, one generates large number of observations, typically time domain input-output data. These observations are first compressed into a smaller set of values (first stage of the TS approach), and these "compressed samples", together with the corresponding values of the parameters, constitute the training set for a supervised learning algorithm (such as linear regression, gradient boosting or deep neural networks) to derive a mapping from the compressed samples to the estimated parameters (second stage). In this way, we leverage the availability of a DT, which is a data simulator, via TS. The TS approach has several appealing properties: (i) it is distribution-free, (ii) the estimation procedure can be posed a simple convex optimization program by a careful choice of the second stage, and (iii) once trained, the TS estimates can be efficiently computed, as they do not require solving further optimization problems.

In this paper, we leverage the TS approach for estimating the control parameters and use these estimates to design a controller for a reference tracking problem. The TS-based controller design is completely simulation (data) driven and in particular, the data compression step (first stage) helps to reduce the complexity of the model that is required to design the controller. To this end, ARMAX models are identified using the synthetically generated input-output data for several choices of the parameters of the DT. The coefficients of these ARMAX models form the compressed data. In parallel, suitable controller parameters, for example the proportional and integral constants of a Proportional-Integral (PI) controller, are computed for each parameters choice of the DT based on some control design rule. These controller parameters, along with the coefficients of ARMAX models, form the training set for a supervised algorithm in the second stage, i.e., the coefficients of the ARMAX models are the "inputs" and the corresponding controller parameters are the "targets" for the chosen supervised learning algorithm. In a nutshell, we learn a direct tuning rule for the controller from the artificially generated data via the TS approach.

Our contributions are summarized as follows:

- A TS-based approach is designed to implicitly reduce the order of our model;
- a controller tuning rule is proposed based on the TS approach; and
- the effectiveness of our approach is demonstrated on a case study via numerical simulations.

The rest of the paper is organized as follows: In Section II, we describe our problem statement. Section III outlines the procedure to use the TS approach for control design, followed by a case study for our problem in Section IV. Section V provides simulation analyses that demonstrate our controller

tuning rule. Finally, we conclude the paper in Section VI.

II. Problem statement

Consider a discrete-time single-input single-output model $\mathcal{M}(\Theta)$

$$\mathcal{M}: \begin{cases} x_{k+1} = f(x_k, u_k, \Theta), \\ y_k = g(x_k, \Theta), \end{cases} \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}$, $y_k \in \mathbb{R}$ are the state, input and output of the model, respectively. f and g are possibly non-linear functions. $\Theta \in \mathbb{R}^p$ is a vector of uncertain parameters, $\Theta = [\theta_1, \dots, \theta_p]$, each distributed according to a known probability density function (*pdf*) $\theta_i \sim \mathcal{D}_i$. We assume that we are able to obtain input-output sequences, say, $D_1^N, \dots, D_m^N$, of length N, from an available simulator of $\mathcal{M}$, given a specific value of the parameter vector.
Further, consider a parametric controller $C(\phi)$, where $\phi \in \mathbb{R}^{n_\phi}$ is the vector of controller parameters, and assume that a control design rule $\mathcal{R}\colon \mathcal{M}(\Theta) \mapsto \phi$ is available.
The assumptions made above are realistic, and indeed are motivated by the wide spread of high-fidelity simulators (also known as *digital twins*) in many sectors [3]. These models are very good replica of the system, let apart a few key parameters that are inherently prone to variations, and cannot be known in advance. Under these considerations, we are able to generate a set of data, depicted in Table I. Each row of the table shows how a realization of the uncertain parameters is mapped onto a collection of input-output time-domain data – the pairs $(u_i^{(j)}, y_i^{(j)})$, $i = 1, \dots, N$, $j = 1, \dots, m$ – and a parametric controller $C(\phi_j)$, $j = 1, \dots, m$.

Given this framework, we are able to design a controller for a given realization of the model $\mathcal{M}(\Theta)$ which is optimal – in the sense of satisfying the tuning rule $\mathcal{R}$. In practice, the parameters in Θ are subject to variations, in which case a classical solution is to re-identify system parameters, which can be computationally intensive. Instead, we leverage the available simulator to find a *direct* solution to get the controller parameters, *de-facto* learning the tuning rule $\mathcal{R}$.

Θ_1	$D_1^N = \left\{ \left(u_1^{(1)}, y_1^{(1)}\right), \dots, \left(u_N^{(1)}, y_N^{(1)}\right) \right\}$	$C(\phi_1)$
Θ_2	$D_2^N = \left\{ \left(u_1^{(2)}, y_1^{(2)}\right), \dots, \left(u_N^{(2)}, y_N^{(2)}\right) \right\}$	$C(\phi_2)$
$\vdots$	$\vdots$	$\vdots$
Θ_M	$D_m^N = \left\{ \left(u_1^{(m}, y_1^{(m)}\right), \dots, \left(u_N^{(m)}, y_N^{(m)}\right) \right\}$	$C(\phi_m)$

TABLE I: Collection of uncertain parameters, input-output trajectories, and designed controllers.

III. Two-Stage Control Design Approach

The TS approach is a methodology for parameter estimation in uncertain white-box models, where a simulator of the plant is available; originally proposed in [2], recent developments and theoretical foundations analyses have been conducted in [4]. The goal in TS is that of estimating an unknown function $\mathcal{F}$, mapping the set of measured data to the parameter. In this case, we shift the problem from estimating an unknown model parameter to estimating the controller parameters ϕ, thus embedding the control design rule $\mathcal{R}$ within $\mathcal{F}$, that is,

$$\mathcal{F}^* = \underset{\mathcal{F}}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \left\| \phi_i - \mathcal{F}\left(u_1^{(i)}, y_1^{(i)}, \dots, u_N^{(i)}, y_N^{(i)}\right) \right\|^2. \tag{2}$$

Estimating $\mathcal{F}$ can in general be a hard problem, if we leave it in this form. The high dimensionality of the input data (N can be in the order of thousands) is not easy to treat with standard optimization techniques. The authors of [2] thus solved it in two separate stages, that we now review and extend to our specific control tuning problem.

A. First stage

The collected data $D_1^N, \dots, D_m^N$ overall form a set of $m \times 2N$ numbers, as each input and output sequence consists of N values. This number can explode in case several unknown parameters or large sample sizes are considered; unfortunately, this is a typical case in real systems.
Dimensionality reduction can be employed to obtain a manageable amount of data, retaining the necessary information with a lower number of samples. Specifically, we aim to obtain a reduced data-set $\tilde{D}_i^n$, $i = 1, \dots, m$, where $n \ll N$, by employing a compression function $g\colon (D_1^N, \dots, D_m^N) \mapsto (\tilde{D}_1^n, \dots, \tilde{D}_m^n)$. Leveraging the fact that the D_i^N's are generated by a dynamical system, [2] considered the identification of a black-box model as such compression function, which captures each time series with a few parameters. Different possibilities have been employed in the literature [5], we however consider in the following the Auto-Regressive Moving-Average with Exogenous-Input (ARMAX) polynomial model, widely versatile and easy to treat [6]. A generic ARMAX model reads

$$\begin{aligned} &y_k + \alpha_1 y_{k-1} + \dots + \alpha_{n_a} y_{k-n_a} = \alpha_{n_a+1} u_{k-n_d} + \cdots \\ &+ \alpha_{n_a+n_b} u_{k-n_d-n_b+1} + \alpha_{n_a+n_b+1} e_{k-1} + \dots \\ &+ \alpha_{n_a+n_b+n_e} e_{k-n_e} + e_k. \end{aligned} \tag{3}$$

The set of the polynomial model parameters $\tilde{D}_i^n = (\alpha_1, \dots, \alpha_n) \in \mathbb{R}^{n=n_a+n_b+n_e}$ is then collected and associated with the corresponding value of the parameter Θ_i.

B. Second stage

Having compressed the data samples, we now aim at finding some function $h^* \in \mathcal{H}$ such that $\mathcal{F} = h^* \circ g$, by solving

$$\begin{aligned} h^* &= \underset{h \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{m} \left\| \sum_{i=1}^{m} \phi_i - h\left(g\left(u_1^{(i)}, y_1^{(i)}, \dots, u_N^{(i)}, y_N^{(i)}\right)\right) \right\|_2^2 \\ &= \underset{h \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{m} \left\| \sum_{i=1}^{m} \phi_i - h\left(\tilde{D}_1^n, \dots, \tilde{D}_m^n\right) \right\|_2^2, \end{aligned} \tag{4}$$

where $\mathcal{H}$ is a prescribed space of functions, or *hypothesis class* [7]. Some commonly used classes $\mathcal{H}$ that give reliable solution to Eq. (4) are Gradient Boosted Machines (GBMs) [8]

and Feed-Forward Neural Networks (FFNNs), which we employ in the second stage of the TS approach. The functions $h \in \mathcal{H}$ can in turn be parameterized in terms of hyper-parameters $w \in \mathbb{R}^{N_w}$. In other words, $h^* = h(w^*)$, where

$$w^* = \underset{w \in \mathbb{R}^{N_w}}{\operatorname{argmin}} \frac{1}{m} \left\| \sum_{i=1}^{m} \phi_i - h(w) \left(\tilde{D}_1^n, \dots, \tilde{D}_m^n \right) \right\|_2^2. \quad (5)$$

Implementation details for GBM and FFNN are given in

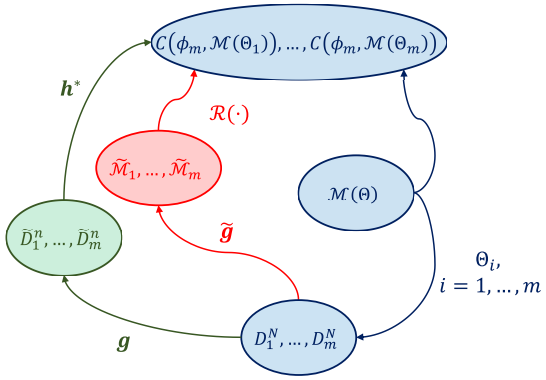

Fig. 1: Schematic representation of the Two-Stage approach considered herein, comprising the data-generation from the simulator, the estimation of function h, and the benchmark approach based on $\tilde{g}$.

Section V. The tuning rules learned by using GBM and FFNN are called TS-GBM and TS-FF respectively.

IV. Case Study: The Yaw-Rate Tracking Problem

As a case study to prove the methodology described in Section III, we consider the yaw-rate tracking problem via steer-by-wire actuation, being of high relevance within the automotive context.

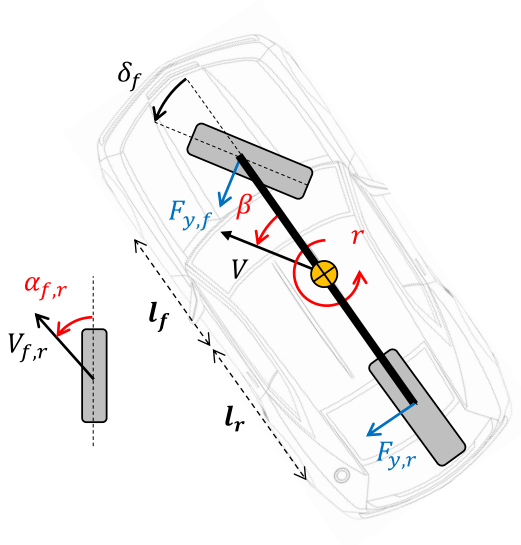

Fig. 2: A cartoon describing the dynamics of a vehicle.

1) System modelling: Upon the assumptions of small steering angle δ_f and front/rear wheels side-slip angles $\alpha_{f,r}$, the following set of equations describes the side-slip β and yaw-rate r dynamics:

$$\begin{aligned} \dot{\beta} &= -r - \frac{C_f \alpha_f}{M_{veh} v_x} - \frac{C_r \alpha_r}{M_{veh} v_x}, \\ \dot{r} &= -\frac{l_f C_f \alpha_f}{J_z} + \frac{l_r C_r \alpha_r}{J_z}, \end{aligned} \quad (6)$$

where l_f amd l_r are the distances between vehicle center-of-gravity and front/rear wheels respectively, C_f and C_r are the cornering stiffnesses, M_{veh} is the vehicle mass, and J_z the yaw-axis vehicle inertia. The differential equations above can be simply obtained by balances of moments and forces in Figure 2, and they represent the well known single-track or bicycle vehicle model.

The tyre-road contact forces are approximated as $F_{y,f,r} = -C_{f,r}\alpha_{f,r}$, for small slip angles.

In Equation (6), v_x is the vehicle longitudinal speed: it is indeed not a physical parameter, but rather a time-varying state of the system. However, it is common practice to consider it as a constant in the yaw-rate control problem [9], [10].

The wheel lateral slip angles $\alpha_{f,r}$ are computed via the kinematic equations

$$\alpha_f^{kin} = -\delta_f + \beta + \frac{l_f}{v_x} r, \;\; \alpha_r^{kin} = \beta - \frac{l_r}{v_x} r, \quad (7)$$

However, it is well known that the tyre lag phenomenon introduces a phase shift between theoretical slips and actual ones [11], [12]. The tyre dynamics is thus introduced:

$$\dot{\alpha}_f = -\frac{v_x}{l_{rel,f}} \left(\alpha_f - \alpha_f^{kin} \right), \;\; \dot{\alpha}_r = -\frac{v_x}{l_{rel,r}} \left(\alpha_f - \alpha_r^{kin} \right), \quad (8)$$

where $l_{rel,f/r}$ governs the lag phenomenon, and is known as *relaxation length*. Finally, the steer-by-wire actuator dynamics could be well within the range of the system dynamics. It is thus necessary to model it. A second order system is here considered, with two complex-conjugate poles (typical for this kind of actuator [13]):

$$\begin{aligned} \dot{z} &= -\omega_n^2 \delta_f + \omega_n^2 \delta_f^{cmd}, \\ \dot{\delta}_f &= z - 2\xi\omega_n \delta_f. \end{aligned} \quad (9)$$

Combining Eqs. (6) to (9) eventually yields a 6-th order Linear and Time-Invariant System, with state vector $x = \begin{bmatrix} \beta & r & \alpha_f & \alpha_r & z & \delta_f \end{bmatrix}$, input $u = \delta_f^{cmd}$, output $y = r$.

The cornering stiffness coefficient is highly dependent on the type of terrain, the tyre characteristics, temperature and so on [14]. For this reason, it is to be considered as an uncertain parameter, prone to significant variation throughout the vehicle functioning:

$$C_{f,r} = C_{f,r}^{nom} \cdot \mu_s, \quad (10)$$

where μ_s is considered as a scaling coefficient for the tyre-road friction.

Another typical source of uncertainty is the vehicle mass, which usually changes about some nominal value, typically

including the chassis and tyre payload, as well as the driver one, i.e.,

$$M_{veh} = M_0 + M_\delta, \tag{11}$$

where M_δ is the mass variation.
For the considered model, we have identified two typical sources of uncertainty, the mass and the tyre-road friction, which can be collected in the vector $\Theta \in \mathbb{R}^2$, and are both uniformly distributed:

$$\Theta = \begin{bmatrix} \mu_s & M_\delta \end{bmatrix}, \ \mu_s \sim \mathcal{U}(0.3, 1.1), \ M_\delta \sim \mathcal{U}(0, 300). \tag{12}$$

Finally, the system defined above is discretized with sampling period $T_s = 0.01$ s (*e.g.* also used in [9]). The physical parameters used in the simulations are given in Table II. The

M_{veh} [kg]	$l_{f,r}$ [m]	J_z [kg· m^2]	$l_{rel,f,r}$ [m]
1895	1.18, 1.53	2400	0.4, 0.8

$C_{f,r}$ [N/(rad $\times 10^3$)]	ω_n [rad/s]	ξ
124.9, 166	$2\pi \cdot 5$	0.9

TABLE II: Vehicle model physical parameters.

vehicle and actuator parameters are taken from [13], and the tyre relaxation lengths from [12].

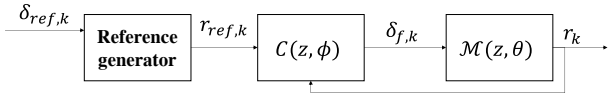

Fig. 3: Control scheme for the yaw-rate tracking problem.

2) Control design: The objective in yaw-rate tracking problems is to regulate a reference behavior, which is generated using a driver's steering request based on ad-hoc logic. In practice, electronic systems track the driver's intent, thereby preventing unsafe situations [15].
One common method for controlling a linear time-invariant system is the loop shaping approach. This method ensures the desired phase margin ϕ_m and cutoff frequency ω_c. These two parameters subsequently define the closed-loop behavior of the system, in accordance with established principles of control theory. A Proportional-Integral controller is sufficient and is represented as follows:

$$C(z, \phi) = \frac{k_p}{1 + 2T_i/T_s} \frac{z+1}{z + \dfrac{T_s - 2T_i}{T_s + 2T_i}}. \tag{13}$$

The controller is discretized using the Tustin method at a sampling time of T_s, with the tuning parameters being the proportional gain and integral time, denoted as $\phi = \begin{bmatrix} k_p & T_i \end{bmatrix}$. By appropriately choosing the regulator parameters, it is possible to achieve the desired frequency-domain performance. We aim for $\varphi_m \geq 60°$ as a fundamental threshold for ensuring an asymptotically stable and well-damped closed-loop response, and cutoff frequency $\omega_c \geq 1.5$ Hz. To implement the control tuning strategy, the loop shaping approach is employed, denoted as $\mathcal{R}(\mathcal{M}(\Theta))$, and this is carried out using the MATLAB function `pidtune`.

V. Simulation analysis

In this section, we apply the Two-Stage approach described in Section III to the case study of Section IV, showing its performance in terms of learning the control design policy.

A. Dataset generation

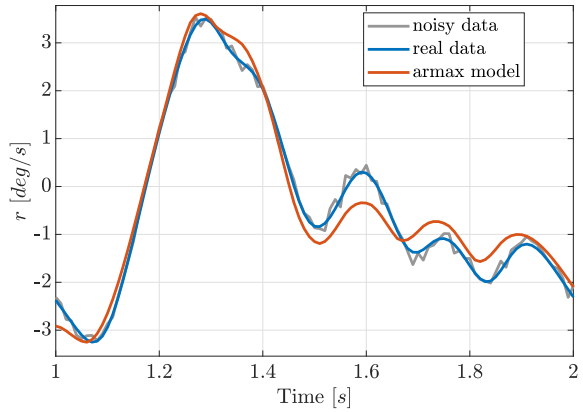

Fig. 4: Time-domain representation of noiseless data samples, noisy ones, and ARMAX fitting performance.

In order to test the Two-Stage approach, we generate a set of data from the yaw-rate dynamics model. We set $m = 1500$, so as to obtain a meaningful number of examples; they are randomly split between 80% for training and 20% for validation. Each dataset lasts 10 seconds, and it is sampled at the frequency of 1000 Hz, so $N = 10000$ data points in the time domain are available for each $D_i^N, i = 1, \ldots, 1500$.
Further, we generate $m = 1500$ more scenarios, to evaluate the model on a different dataset (testing set). The testing set is also used to evaluate the benchmark identification and control tuning approach that we are going to describe in the following. The steer input $\delta_{f,k}$, $k = 1, \ldots, N$, is selected to be a Pseudo-Random-Binary signal, so as to provide a meaningful excitation to the system; the output yaw-rate r_k, $k = 1, \ldots, N$, is perturbed with Gaussian white noise, so as to achieve a signal-to-noise ratio of ≈ 30 dB.

B. Gray-box identification benchmark

As a benchmark to compare TS approach to, we consider gray-box (GB) identification. This benchmark is possibly the best solution to be carried out when the model structure and some initial value of the parameters are approximately known, and only some refinement is necessary. Given the initial model $\mathcal{M}(\Theta_0)$, we refine it using available data, obtaining $\mathcal{M}_{gb}$. The prediction error minimization approach can be utilized to adjust the model parameters (we consider here the MATLAB implementation `pem`).
Once the refined model has been identified, one can apply the loop shaping tuning rule described in Section III.

C. Model compression details

For the model compression phase, we consider an ARMAX model of order $(2, 2, 2, 1)$. Such an ARMAX model is a 2nd order system. Compared to the data generator, which is a 6-th order one, we significantly reduce the model dimensionality. An example of the ARMAX fitting performance is shown in Fig. 4, where noise-perturbed data are shown too. As one can notice, the reduced order model exhibits some prediction error; nevertheless, we will show that the extracted ARMAX features are enough to obtain a good learning of the loop-shaping rule.

D. Second stage training details

We try GBM and FFNN in the second stage to estimate the controller parameters k_p and T_i. The reason to choose these methods as function approximators is due to their capabilities to capture non-linearities in the data (corresponding to the ARMAX coefficients that we obtain from the first stage of the TS approach). One can simply use FFNN alone as a function approximator, but more recently it has been demonstrated empirically that GBMs outperform neural networks for tabular data and the training time is remarkably fast in GBMs [16]. Hence, for the sake of fair comparison, we use GBM in the second stage.

To implement GBM, we use XGBoost, a Python package developed by [17]. More details on using the XGBoost package and the implications of its hyper-parameters can be found in [17]. FFNN is implemented in the MATLAB Deep Learning Toolbox [18]. Their training details are as follows:

- XGBoost: We have selected, via cross-validation, `n_estimators` = 2000, `max_depth` = 7, `max_leaves` = 10, `learning_rate` = 0.1, `reg_lambda` = 4, and `min_child_weight` = 25 for the tuning parameters of XGBoost. We run two separate XGBoost algorithms in parallel with the stated choices of parameters to predict k_p and T_i.
- FFNN: Model using `feedforwardnet` function, with 10 neurons in the hidden layer, $tanh$ activation, and a linear output layer. The Levenberg-Marquardt (`trainlm`) algorithm is considered for training. `epochs` is set to 1000; `mu` is set to 0.001 (a parameter regulating gradient descent in `trainlm`); `Mu Decrease Ratio` and `Mu Increase Ratio` are respectively set to 0.1 and 10. `max_fail` – the maximum number of failed validation checks before stopping– is set to 5. The other parameters are left at MATLAB default.

E. Controller parameters identification performance

Figures 5 and 6 respectively depict the proportional gain k_p and integral time T_i estimation performance, for the three different methods. As one can notice, good performance is achieved for all methods, and the average value is properly estimated, although TS-GBM exhibits some variance, especially for high values of T_i. The figures also give the $rms_{\%}$ error metric, which is defined – for a generic parameter ν and its estimated $\hat{\nu}$ – as

$$rms_{\%}\left(\nu, \hat{\nu}\right) = \sqrt{\frac{1}{N_s}\sum_{i=1}^{N_s}\left(100 \cdot \frac{\nu_i - \hat{\nu}_i}{\nu_i}\right)^2}. \qquad (14)$$

Fitting the controller parameters is one way to assess the estimators goodness: however, it is interesting to look at some closed-loop system metrics to see what the impact of an error in fitting the integral reset time is. Table III provides statistics about the phase margin and cutoff frequency of the open-loop transfer function $C(z)\cdot\mathcal{M}(z)$, obtained by using the estimated controller parameters. One can note that all methods satisfy the

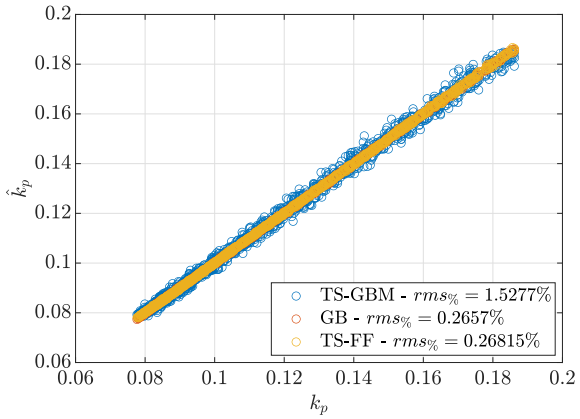

Fig. 5: Regression plot showing the fitting of k_p in the testing case, for both considered methods.

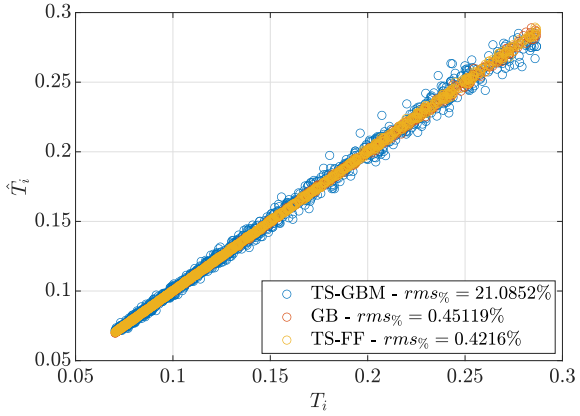

Fig. 6: Regression plot showing the fitting of T_i in the testing case, for both considered methods.

	φ_m [deg]		ω_c [Hz]	
	Mean	Std.	Mean	Std.
GB	59.98	0.2385	1.50	0.0037
TS-FF	59.99	0.1714	1.50	0.0032
TS-GBM	60.01	0.69	1.50	0.0143

TABLE III: Loop-shaping metrics statistics. "Mean" and "Std." respectively stand for average value and standard deviation, over a set of experiments.

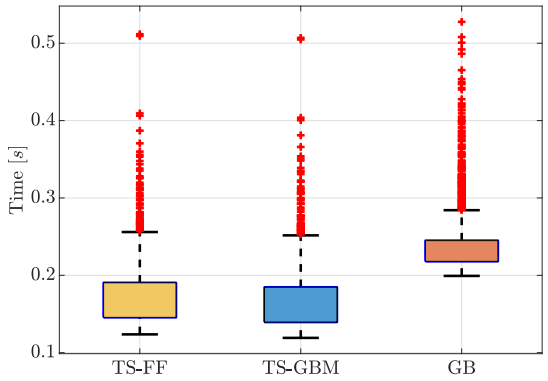

Fig. 7: Analysis of computational times necessary for executing both methods.

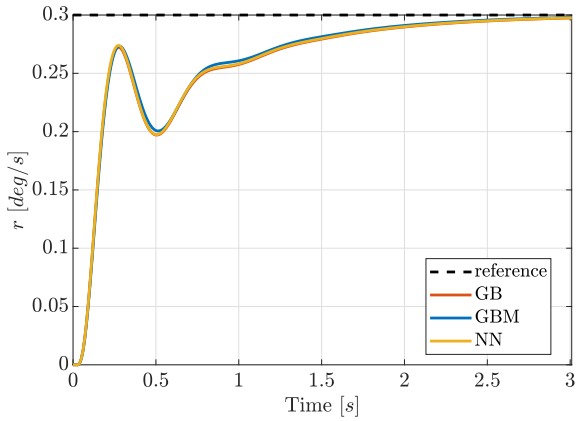

Fig. 8: Time domain closed-loop behaviour of the system, for $\Theta = \begin{bmatrix} 0.60 & 44.14 \end{bmatrix}$.

control performance requirements of Section IV-2, yielding a phase margin $\varphi_m \approx 60$ deg and $\omega_c \approx 1.5$ Hz. In fact, the phase margin achieved with TS-GBM reflects the increased variance of the estimator, but the impact on closed-loop performance is absolutely irrelevant.

Figure 7 provides computational time analysis for the proposed approach and the benchmark. It can be seen that the TS approaches (TS-FF and TS-GBM) take less time in computing the controller estimates compared to the benchmark. Among the proposed approaches, TS-GBM has slightly better computational advantage over TS-FF.

Figures 8 shows the controller performance in the time-domain. The three controllers yield practically the same results, and this confirms the almost equivalence between the proposed approaches. Indeed, the reference tracking per-se can be improved, *e.g.* by selecting a more complex controller, which is however not in the scope of this work.

VI. Conclusions

This paper presented a novel methodology for meta-learning in control systems: we employ a Two-Stage approach, with a first stage consisting in model order reduction, and a second one consisting in direct estimation via black-box models. Specifically, we see how a possibly unknown control tuning rule can be learned from simulated data, and then applied to generate new suitable controllers just from fresh data. Simulation analyses showed that the presented approach can achieve performance similar to an identification-for-control method, while providing an improvement in computational time.

Future research will be devoted to the extension of the methodology to other applications.

VII. Acknowledgement

This work has been partially supported by the Swedish Research Council under contract number 2016-06079 (NewLEADS) and by the Digital Futures project EXTREMUM.

References

[1] F. Dettù, S. Formentin, and S. M. Savaresi, "The twin-in-the-loop approach for vehicle dynamics control," *IEEE/ASME Transactions on Mechatronics*, 2023.

[2] S. Garatti and S. Bittanti, "A new paradigm for parameter estimation in system modeling," *International Journal of adaptive control and signal processing*, vol. 27, no. 8, pp. 667–687, 2013.

[3] E. VanDerHorn and S. Mahadevan, "Digital twin: Generalization, characterization and implementation," *Decision support systems*, vol. 145, p. 113524, 2021.

[4] B. Lakshminarayanan and C. R. Rojas, "A statistical decision-theoretical perspective on the two-stage approach to parameter estimation," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 5369–5374.

[5] S. Formentin, K. Van Heusden, and A. Karimi, "A comparison of model-based and data-driven controller tuning," *International Journal of Adaptive Control and Signal Processing*, vol. 28, no. 10, pp. 882–897, 2014.

[6] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Prentice-Hall, 1999.

[7] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2012.

[8] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[9] A. Lucchini, S. Formentin, M. Corno, D. Piga, and S. M. Savaresi, "Torque vectoring for high-performance electric vehicles: an efficient mpc calibration," *IEEE Control Systems Letters*, vol. 4, no. 3, pp. 725–730, 2020.

[10] A. Lucchini, F. P. Azza, M. Corno, S. Formentin, and S. M. Savaresi, "Design and implementation of a mpc-based rear-wheel steering controller for sports cars," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2021, pp. 802–807.

[11] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Science robotics*, vol. 4, no. 28, p. eaaw1975, 2019.

[12] J. S. Loeb, D. A. Guenther, H.-H. F. Chen, and J. R. Ellis, "Lateral stiffness, cornering stiffness and relaxation length of the pneumatic tire," *SAE transactions*, pp. 147–155, 1990.

[13] M. Corno, G. Panzani, F. Roselli, M. Giorelli, D. Azzolini, and S. M. Savaresi, "An lpv approach to autonomous vehicle path tracking in the presence of steering actuation nonlinearities," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 4, pp. 1766–1774, 2020.

[14] C. E. Beal and J. C. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1258–1269, 2012.

[15] R. Rajamani, "Electronic stability control," *Vehicle Dynamics and Control*, pp. 201–240, 2012.

[16] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022.

[17] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," ser. KDD'16, 2016, p. 785–794.

[18] "MATLAB deep learning toolbox," https://se.mathworks.com/products/deep-learning.html, accessed: October 4th 2023.