SelfMPC: Automated Data-Driven MPC Design for a Class of Nonlinear Systems

Guitao Yang[1], Matteo Scandella[2], Simone Formentin[3], Thomas Parisini[4]

***Abstract*— Manual tuning required for Model Predictive Control (MPC) schemes can be labor-intensive and prone to errors due to the requisite domain expertise. In this paper, we propose a new procedure called SelfMPC: an automated, data-driven method for tuning MPC for an unknown system within a specific nonlinear class. We pursue a maximum likelihood approach using Gaussian processes to uncover system dynamics and to optimize a tracking cost function. We show the effectiveness of our approach through extensive simulations on a benchmark case study, illustrating its superior performance over traditional manual tuning techniques. Furthermore, we offer formal assurances regarding the stability and robustness of the resulting controller, ensuring its versatility across diverse operating conditions and uncertainties within the system.**

***Index Terms*— MPC, Automated cost tuning, Data-driven control, Nonlinear systems**

I. Introduction

Model Predictive Control (MPC) is recognized as a versatile control strategy [1], [2], with broad applications, spanning industrial processes [3], traffic management [4], [5], energy management systems [6], and biomedical processes [7], [8] etc. However, the practical implementation of MPC methods still faces significant challenges.

One of the fundamental obstacles lies in acquiring a reliable system model capable of accurately predicting the future trajectory based on the input and current state. The performance of the conventional MPC methods heavily depends on the accuracy of the model, prompting the development of robust MPC [9], [10] or stochastic MPC designs [11]. These approaches enhance the MPC method's ability to tolerate model imprecision. Clearly, when the system model is inaccessible, conventional MPC design would fail to achieve its objectives due to the unavailability of future system trajectory predictions. To overcome this obstacle, data-driven methodologies that assume unknown models, but leverage pre-collected data demonstrate their efficacy. Recent examples include Data-enabled Predictive Control (DeePC) [12] or Data-driven Predictive Control (DDPC) [13].

Another significant, yet often elusive challenge in the practical implementation of MPC lies in tuning or selecting critical hyperparameters, notably the weighting matrices within the cost function and the prediction horizon. Achieving the right balance among these hyperparameters is imperative to ensure that the control strategy aligns seamlessly with the specific constraints and requirements of the controlled system. Successfully overcoming these hurdles is critical in fully capitalizing on the potential of MPC, thereby unleashing its advantages across a diverse array of real-world applications.

In situations where datasets containing desired closed-loop control examples are available, alternative methodologies like inverse optimal control or inverse reinforcement learning [14] can be utilized to decipher the preferences and objectives of the controller based on observed actions. In cases where such datasets are not available, conventional practice often resorts to potentially cumbersome trial-and-error procedures. More recently, innovative black-box optimization tools [15], [16] have been introduced to improve the efficiency of this calibration process. However, employing the aforementioned iterative optimization techniques typically requires conducting closed-loop experiments, which may not always be feasible due to safety concerns related to implementing intermediate suboptimal controllers or due to the inherent costs associated with experiments.

Moreover, as we narrow our focus to model-free predictive control for nonlinear systems, the existing challenge persists. Although some recent literature has begun addressing this issue [17], [18], the tuning of costs still hinges on experiments conducted on the target system. This dependency on experimental data for cost tuning often causes suboptimal performance in certain application scenarios.

In this paper, we extend the method introduced in [19] to address the design of the MPC cost function for reference tracking. We specifically target scenarios where the system model is unknown but belongs to a certain nonlinear class, and where some dataset, collected either offline or online, is available to the designer. We assume that the system model can be learned using Gaussian process regression [20], [21].

This work has been partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement no. 739551 (KIOS CoE), by FAIR (Future Artificial Intelligence Research) project, funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, Investment 1.3, Line on Artificial Intelligence), by the Italian Ministry of Enterprises and Made in Italy in the framework of the project 4DDS (4D Drone Swarms) under grant no. F/310097/01-04/X56, and by the PRIN PNRR project P2022NB77E "A data-driven cooperative framework for the management of distributed energy and water resources" (CUP: D53D23016100001), funded by the NextGeneration EU program.

[1]Guitao Yang is with the Department of Electrical and Electronic Engineering, Imperial College London, London, United Kingdom. He is the corresponding author. Email: guitao.yang@imperial.ac.uk

[2]Matteo Scandella is with the Department of Management, Information and Production Engineering, University of Bergamo, via Marconi 5, 24044, Dalmine (BG), Italy. Email: matteo.scandella@unibg.it

[3]Simone Formentin is with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, via G. Ponzio 34/5, 20133 Milano, Italy. Email: simone.formentin@polimi.it

[4]Thomas Parisini is with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, UK, with the Department of Electronic Systems, Aalborg University, Denmark, and with the Department of Engineering and Architecture, University of Trieste, Italy. Email: t.parisini@imperial.ac.uk

Within this framework, we demonstrate that the outcomes of model learning can be utilized to formulate the task of MPC tuning by framing cost design as a Maximum Likelihood Estimation Problem. Essentially, we propose that the objective of tracking a specific reference output can be expressed as maximizing the likelihood that the model's output coincides with this given reference. This formulation naturally incorporates not only the nominal predictions produced by the model, but also the estimates of the covariance function generated by the Bayesian learning approach. The key distinction between our approach and existing data-driven methods lies in the automatic determination of weighting matrices without the need for closed-loop experiments. These automatically generated weighting matrices are "optimal" in a maximum likelihood sense, and inherently account for the accuracy of the learned model through the covariance matrix. As a result, our formulation of the MPC problem does not require additional regularization terms, which sets it apart from other methods such as stochastic implementations of DeePC [22], [23]. Furthermore, we illustrate, through a simulation example, how this formulation, which we refer to as Self cost-tuning MPC (SelfMPC), outperforms using a randomly chosen or slightly tuned weighting matrix, even with the same prediction strategy to obtain the future trajectories.

Organization. In Section II, we formally define the problem targeting a class of nonlinear systems. Section III is devoted to illustrating the data-driven method that automatically tunes the cost of the MPC within a Bayesian learning framework. Moreover, we briefly discuss the extension of our proposed control strategy to an autoregressive framework in Section IV. Subsequently, Section V showcases a simulation case study illustrating the effectiveness of our proposed algorithm. Finally, some concluding remarks are stated in Section VI.

Notation. We denote by $\mathbb{R}$ and $\mathbb{N}$ the set of real and natural numbers, respectively ($0 \in \mathbb{N}$). Given $n, m \in \mathbb{N}$, $\mathbb{R}^{n\times m}$ is the set of $n \times m$ matrices and $\mathbb{R}^n$ is the set of column vectors of dimension n. Furthermore, $\mathbf{0}_{n\times m} \in \mathbb{R}^{n\times m}$ is the zero matrix, $\mathbf{1}_{n\times m}$ is an $n \times m$ matrix of ones, and $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix. Given $n, i, j \in \mathbb{N}$ and a sequence $x : \mathbb{N} \to \mathbb{R}^n$, $x_i \in \mathbb{R}^n$ is the ith component of x and, if $j \geq i$, $x_{i:j} = (x_h)_{h=i}^{j} \in \mathbb{R}^{n(j-i+1)}$, otherwise $x_{i:j}$ is an empty tuple. With a slight abuse of notation, we use tuples of real numbers and column vectors interchangeably. The matrix $\mathrm{diag}(M_1, \ldots, M_n)$ is a block diagonal matrix composed of the matrices $M_1, \ldots, M_n$. Given two matrices A, B, $A \otimes B$ is their Kronecker product. Given a symmetric positive definite matrix $Q \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$, $\|x\|_Q^2 = x^\top Q x$ and $\|x\|^2 = x^\top x$. For the sake of brevity, we denote with "density" the probability density function. Given a random variable $x \in \mathcal{X}$ with density $\pi : \mathcal{X} \to [0, \infty]$, we write $x \sim \pi(\cdot)$. Given $p \in \mathbb{N}$, $\mu \in \mathbb{R}^p$ and $\Sigma \in \mathbb{R}^{p\times p}$, $\mathcal{N}(\cdot \,|\, \mu, \Sigma)$ denotes the density of the p-dimensional Normal distribution with mean μ and covariance matrix Σ.

II. Problem Statement

Consider a nonlinear system described with the following equations

$$\forall t \in \mathbb{N}_{\geq s-1}, \quad y_{t+1} = g(u_{t-s+1:t}) + w_t \tag{1a}$$
$$\forall t \in \mathbb{N}, \quad z_t = C y_t \tag{1b}$$

where $y_t \in \mathbb{R}^{n_\mathrm{y}}$, $u_t \in \mathbb{R}^{n_\mathrm{u}}$ and $z_t \in \mathbb{R}^{n_\mathrm{z}}$ denote the measurements, the input variables and the variables that the designer intend to control at time t, respectively. Moreover, w_t is the noise that affects y_t, and $s \in \mathbb{N}$ is the order of the model, i.e. the number of past input measurements that affect the output. Note that $g : \mathbb{R}^{sn_\mathrm{u}} \to \mathbb{R}^{n_\mathrm{y}}$ is an unknown function to the designer. Then we consider the following stochastic assumption on the stochastic process w.

Assumption 1: The stochastic process w is a white process and, for every $t \in \mathbb{N}$, $w_t \sim \mathcal{N}\big(\cdot \,\big|\, \mathbf{0}_{n_w\times 1}, W\big)$ where $W \in \mathbb{R}^{n_w\times n_w}$ is a positive definite matrix.

Since the system we intend to control is unknown, we assume that data collected from the underlying plant are available to the designer. In particular, at every time instant $t \in \mathbb{N}$, we assume to have at our disposal two datasets:

(i) Identification dataset:

$$\mathcal{D}_{\mathrm{id}}^t \coloneqq \Big\{(u_i, y_i) \,:\, t_\mathrm{a}(t) \leq i \leq t_\mathrm{b}(t)\Big\}. \tag{2}$$

This dataset contains the input/output data used for identification, which will be exploited for learning the behavior of the system (see Section III-A.1).

(ii) Prediction dataset:

$$\mathcal{D}_{\mathrm{pr}}^t \coloneqq \Big\{u_i \,:\, t-s+1 \leq i \leq t-1\Big\}. \tag{3}$$

This dataset is composed of the input data used for predicting the future system state (see Section III-A.3).

Note that $t_\mathrm{a}, t_\mathrm{b} : \mathbb{N} \to \mathbb{N}$ defines the time range of the measurements used for identification. Therefore, at time $t \in \mathbb{N}$, we have at our disposal $t_\mathrm{b}(t) - t_\mathrm{a}(t) + 1$ measurements for identification. Furthermore, the two datasets are not necessarily disjointed. If $t_\mathrm{b}(t) = t$, the algorithm is set to use newly collected samples to update the controller logic. Now, given t_a and t_b, we define the set

$$\mathbb{T} = \big\{t \in \mathbb{N} \,:\, \big(t > t_\mathrm{b}(t)\big) \wedge \big(t_\mathrm{b}(t) - t_\mathrm{a}(t) + 1 > s\big)\big\}.$$

The set $\mathbb{T}$ is the set of time instants where is possible to apply the proposed controller. In particular, the first condition ($t > t_\mathrm{b}(t)$) guarantees that we are in a time instant when all the measurements are available, and the second condition ($t_\mathrm{b}(t) - t_\mathrm{a}(t) + 1 > s$) guarantees that we have enough data to initiate the controller design (we will clarify this point later).

Throughout this paper, we aim to devise an MPC-fashioned controller to drive the unknown system described in (1) to a desired trajectory $r(t)$ by only using $\mathcal{D}_{\mathrm{id}}^t$ and $\mathcal{D}_{\mathrm{pr}}^t$. Moreover, the proposed method is capable of tuning the MPC gains automatically, which will be illustrated explicitly in the next section.

III. Data-driven SelfMPC design for nonlinear FIR models

In this section, we provide a comprehensive overview of the design of the proposed data-driven SelfMPC applied to the nonlinear system (1). As introduced in [19], the SelfMPC design consists of two key stages: **(i)** Firstly, we detail the method for predicting the future output distribution utilizing the available data. This step is crucial for effectively steering the system towards the desired reference trajectory. **(ii)** Subsequently, we elucidate the auto-tuning procedure for the cost function based on the derived output distribution. This procedure ensures that the MPC controller adapts dynamically to changes in the system dynamics, enhancing its robustness and performance.

A. Identification and Prediction

In order to define the distribution of the unknown function g in (1) given the available dataset $\mathcal{D}_{\mathrm{id}}^t$ at time $t \in \mathbb{T}$, we need to define the distribution of the data given g and assign a priori distribution on g.

1) Data Distribution: Firstly, for all $t \in \mathbb{T}$, we use $\mathcal{D}_{\mathrm{id}}^t$ to define the following vectors:

$$\begin{aligned}\mathbf{Y}_t &\coloneqq \Big[y_{i+1} \,:\, i \in \mathbb{I}_{\mathrm{id}}\Big] \in \mathbb{R}^{N_t n_{\mathrm{y}}},\\ \mathbf{G}_t &\coloneqq \Big[g(u_{i-s+1:i}) \,:\, i \in \mathbb{I}_{\mathrm{id}}\Big] \in \mathbb{R}^{N_t n_{\mathrm{y}}},\\ \mathbf{W}_t &\coloneqq \Big[w_i \,:\, i \in \mathbb{I}_{\mathrm{id}}\Big] \in \mathbb{R}^{N_t n_{\mathrm{y}}},\end{aligned}$$

where $\mathbb{I}_{\mathrm{id}} \coloneqq \big\{t_{\mathrm{a}}(t)+s-1,\dots,t_{\mathrm{b}}(t)-1\big\}$ and $N_t \coloneqq t_{\mathrm{b}}(t) - t_{\mathrm{a}}(t) - s + 1$ denote the index set and the number of data that we desire to use at time t, respectively. These vectors are always well-defined since $N_t \geq 1$ for all $t \in \mathbb{T}$. With the definitions of $\mathbf{Y}_t$, $\mathbf{G}_t$, and $\mathbf{W}_t$, we have

$$\forall t \in \mathbb{N}, \quad \mathbf{Y}_t = \mathbf{G}_t + \mathbf{W}_t \tag{4}$$

from the system described in (1). Since $\mathbf{G}_t$ is a deterministic vector and $\mathbf{W}_t$ is a vector of Normal random variable, we have

$$\mathbf{Y}_t \big| \mathbf{G}_t \sim \mathcal{N}\big(\cdot \,\big|\, \mathbf{G}_t, \bar{W}_{N_t}\big)$$

where

$$\bar{W}_{N_t} \coloneqq \operatorname{diag}(\underbrace{W, W, \dots, W}_{N_t}).$$

Note that the variance of $\mathbf{W}_t$ equals the variance of $\mathbf{Y}_t \big| \mathbf{G}_t$.

2) Prior Distribution: Before defining the prior on the unknown function, it is useful to introduce the concept of reproducing kernel for vector-valued Gaussian Processes.

Definition 1: Let $n_{\mathrm{a}}, n_{\mathrm{b}} \in \mathbb{N}$. Then, a function $k : \mathbb{R}^{n_{\mathrm{a}}} \times \mathbb{R}^{n_{\mathrm{a}}} \to \mathbb{R}^{n_{\mathrm{b}} \times n_{\mathrm{b}}}$ is called reproducing kernel if and only if:

(i) given $a, b \in \mathbb{R}^{n_{\mathrm{a}}}$, $k(a,b) = k(b,a)^\top$;
(ii) given $a \in \mathbb{R}^{n_{\mathrm{a}}}$, $k(a,a)$ is positive semi-definite;
(iii) given $n \in \mathbb{N}$, $\{a_j \,:\, 1 \leq j \leq n\} \subset \mathbb{R}^{n_{\mathrm{a}}}$ and $\{c_j \,:\, 1 \leq j \leq n\} \subset \mathbb{R}^{n_{\mathrm{b}}}$,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i^\top k(a_i, a_j) c_j \geq 0.$$

Definition 2: Let $n_{\mathrm{a}}, n_{\mathrm{b}} \in \mathbb{N}$, $\mu : \mathbb{R}^{n_{\mathrm{a}}} \to \mathbb{R}^{n_{\mathrm{b}}}$, and a reproducing kernel $k : \mathbb{R}^{n_{\mathrm{a}}} \times \mathbb{R}^{n_{\mathrm{a}}} \to \mathbb{R}^{n_{\mathrm{b}} \times n_{\mathrm{b}}}$. Then a random function $f : \mathbb{R}^{n_{\mathrm{a}}} \to \mathbb{R}^{n_{\mathrm{b}}}$ is distributed according to a Gaussian Process with mean μ and variance k if and only if

$$\forall n \in \mathbb{N}, \quad \forall \{a_j \,:\, 1 \leq j \leq n\} \subset \mathbb{R}^{n_{\mathrm{a}}},$$
$$\begin{bmatrix} f(a_1) \\ \vdots \\ f(a_n) \end{bmatrix} \sim \mathcal{N}\left(\cdot \,\middle|\, \begin{bmatrix} \mu(a_1) \\ \vdots \\ \mu(a_n) \end{bmatrix}, \begin{bmatrix} k(a_1,a_1) & \cdots & k(a_1,a_n) \\ \vdots & \ddots & \vdots \\ k(a_n,a_1) & \cdots & k(a_n,a_n) \end{bmatrix}\right).$$

Then, we write $f \sim \mathcal{GP}(\mu, k)$.
For a good reviews on Gaussian Processes for vector-valued process refer to [24], [25]. With these definitions in mind, we assume that

$$g \sim \mathcal{GP}(\mu, k)$$

where $\mu : \mathbb{R}^{s n_{\mathrm{u}}} \to \mathbb{R}^{n_{\mathrm{y}}}$ and $k : \mathbb{R}^{s n_{\mathrm{u}}} \times \mathbb{R}^{s n_{\mathrm{u}}} \to \mathbb{R}^{n_{\mathrm{y}} \times n_{\mathrm{y}}}$ is a reproducing kernel. Therefore, using the definition of Gaussian Process, we obtain

$$\mathbf{G}_t \sim \mathcal{N}\big(\cdot \,\big|\, \mathbf{m}_t, \mathbf{K}_t\big)$$

where

$$\begin{aligned}\mathbf{m}_t &\coloneqq [m(u_{i-s+1:i}) \,:\, i \in \mathbb{I}_{\mathrm{id}}] \in \mathbb{R}^{N_t n_{\mathrm{y}}},\\ \mathbf{K}_t &\coloneqq \left[k(u_{i-s+1:i}, u_{j-s+1:j}) : \begin{matrix} i \in \mathbb{I}_{\mathrm{id}} \\ j \in \mathbb{I}_{\mathrm{id}} \end{matrix}\right] \in \mathbb{R}^{N_t n_{\mathrm{y}} \times N_t n_{\mathrm{y}}}.\end{aligned}$$

3) Posterior Distribution: Firstly, for all $t \in \mathbb{T}$, we define the following vectors that contain the "future" data:

$$\begin{aligned}\overline{\mathbf{Y}}_t &\coloneqq \Big[y_{i+1} \,:\, i \in \mathbb{I}_{\mathrm{pr}}\Big] \in \mathbb{R}^{m n_{\mathrm{y}}},\\ \overline{\mathbf{G}}_t &\coloneqq \Big[g(u_{i-s+1:i}) \,:\, i \in \mathbb{I}_{\mathrm{pr}}\Big] \in \mathbb{R}^{m n_{\mathrm{y}}},\\ \overline{\mathbf{W}}_t &\coloneqq \Big[w_i \,:\, i \in \mathbb{I}_{\mathrm{pr}}\Big] \in \mathbb{R}^{m n_{\mathrm{y}}}\end{aligned}$$

where $\mathbb{I}_{\mathrm{pr}} \coloneqq \big\{t, \dots, t+m-1\big\}$. Notice that $\overline{\mathbf{G}}_t$ depends on the inputs $u_{t-s+1:t+m-1}$. This vector can be divided into two parts: the past input measurements $u_{t-s+1:t-1}$ that are already available, and the control variable $u_{t+1:t+m-1}$ that we need to select to steer the system. Then, by considering (4) we write

$$\begin{bmatrix} \mathbf{Y}_t \\ \overline{\mathbf{Y}}_t \end{bmatrix} = \begin{bmatrix} \mathbf{G}_t \\ \overline{\mathbf{G}}_t \end{bmatrix} + \begin{bmatrix} \mathbf{W}_t \\ \overline{\mathbf{W}}_t \end{bmatrix}$$

Since $\big[\mathbf{G}_t, \overline{\mathbf{G}}_t\big]^\top$ is a deterministic vector and $\big[\mathbf{W}_t, \overline{\mathbf{W}}_t\big]^\top$ is a vector of Normal random variable, we have

$$\begin{bmatrix} \mathbf{Y}_t \\ \overline{\mathbf{Y}}_t \end{bmatrix} \Bigg| \begin{bmatrix} \mathbf{G}_t \\ \overline{\mathbf{G}}_t \end{bmatrix} \sim \mathcal{N}\left(\cdot \,\middle|\, \begin{bmatrix} \mathbf{G}_t \\ \overline{\mathbf{G}}_t \end{bmatrix}, \begin{bmatrix} \bar{W}_{N_t} & \mathbf{0}_{n_{\mathrm{w}} \times m} \\ \mathbf{0}_{m \times n_{\mathrm{w}}} & \bar{W}_m \end{bmatrix}\right) \tag{5}$$

where the variance of $\overline{\mathbf{W}}_t$ is given by

$$\bar{W}_m \coloneqq \operatorname{diag}(\underbrace{W, W, \dots, W}_{m}).$$

Furthermore, from the definition of Gaussian process, we obtain

$$\begin{bmatrix} \mathbf{G}_t \\ \overline{\mathbf{G}}_t \end{bmatrix} \sim \mathcal{N}\left(\cdot \,\middle|\, \begin{bmatrix} \mathbf{m}_t \\ \overline{\mathbf{m}}_t \end{bmatrix}, \begin{bmatrix} \mathbf{K}_t & \mathbf{K}_t^\star \\ (\mathbf{K}_t^\star)^\top & \overline{\mathbf{K}}_t \end{bmatrix}\right) \tag{6}$$

where

$$\overline{\mathbf{m}}_t \coloneqq [m(u_{i-s+1:i}) \,:\, i \in \mathbb{I}_{\rm pr}] \in \mathbb{R}^{mn_{\rm y}},$$

$$\overline{\mathbf{K}}_t \coloneqq \begin{bmatrix} k(u_{i-s+1:i}, u_{j-s+1:j}) : \begin{matrix} i \in \mathbb{I}_{\rm pr} \\ j \in \mathbb{I}_{\rm pr} \end{matrix} \end{bmatrix} \in \mathbb{R}^{mn_{\rm y} \times mn_{\rm y}},$$

$$\mathbf{K}_t^\star \coloneqq \begin{bmatrix} k(u_{i-s+1:i}, u_{j-s+1:j}) : \begin{matrix} i \in \mathbb{I}_{\rm id} \\ j \in \mathbb{I}_{\rm pr} \end{matrix} \end{bmatrix} \in \mathbb{R}^{N_t n_{\rm y} \times mn_{\rm y}}.$$

From the Normal distribution given in (5) and (6), we obtain

$$\begin{bmatrix} \mathbf{Y}_t \\ \overline{\mathbf{Y}}_t \end{bmatrix} \sim \mathcal{N}\Big(\cdot \,\Big|\, \begin{bmatrix} \mathbf{m}_t \\ \overline{\mathbf{m}}_t \end{bmatrix}, \begin{bmatrix} \mathbf{K}_t + \bar{W}_{N_t} & \mathbf{K}_t^\star \\ (\mathbf{K}_t^\star)^\top & \overline{\mathbf{K}}_t + \bar{W}_m \end{bmatrix}\Big).$$

Finally, from the conjugacy properties of the Normal distribution, we have

$$\overline{\mathbf{Y}}_t \big| \mathbf{Y}_t \sim \mathcal{N}\big(\cdot \,\big|\, \widehat{\mathbf{m}}_t, \widehat{\mathbf{\Sigma}}_t\big) \tag{7}$$

where

$$\widehat{\mathbf{m}}_t = \overline{\mathbf{m}}_t + (\mathbf{K}_t^\star)^\top \big(\mathbf{K}_t + \bar{W}_{N_t}\big)^{-1}\big(\mathbf{Y}_t - \mathbf{m}_t\big) \in \mathbb{R}^{mn_{\rm y}}$$

$$\widehat{\mathbf{\Sigma}}_t = \overline{\mathbf{K}}_t + \bar{W}_m - (\mathbf{K}_t^\star)^\top \big(\mathbf{K}_t + \bar{W}_{N_t}\big)^{-1} \mathbf{K}_t^\star \in \mathbb{R}^{mn_{\rm y} \times mn_{\rm y}}$$

B. Automated Data-driven MPC Design

From (1b), we know that

$$\forall t \in \mathbb{N},\ \forall m \in \mathbb{N}, \quad z_{t+1:t+m} = \bar{C}_m\, y_{t+1:t+m} = \bar{C}_m \overline{\mathbf{Y}}_t$$

where

$$\bar{C}_m \coloneqq \operatorname{diag}(\underbrace{C, C, \ldots, C}_{m}) \in \mathbb{R}^{mn_{\rm z} \times mn_{\rm y}}.$$

Therefore, using (7), we have

$$z_{t+1:t+m} | \mathbf{Y}_t \sim \mathcal{N}\Big(\cdot \,\Big|\, \bar{C}_m \widehat{\mathbf{m}}_t, \bar{C}_m \widehat{\mathbf{\Sigma}}_t \bar{C}_m^\top\Big).$$

Naturally, the future control variable is selected by solving the optimization problem

$$\underset{u_{t:t+m-1}}{\operatorname{argmax}} \quad \mathcal{N}\Big(r_{t+1:t+m} \,\Big|\, \bar{C}_m \widehat{\mathbf{m}}_t, \bar{C}_m \widehat{\mathbf{\Sigma}}_t \bar{C}_m^\top\Big). \tag{8}$$

Instead of solving this optimization problem, it is convenient to consider the equivalent problem

$$\underset{u_{t:t+m-1}}{\operatorname{argmin}} \quad -2 \log \mathcal{N}\Big(r_{t+1:t+m} \,\Big|\, \bar{C}_m \widehat{\mathbf{m}}_t, \bar{C}_m \widehat{\mathbf{\Sigma}}_t \bar{C}_m^\top\Big).$$

By logarithm the cost, we preserve the monotonicity while simplifying the calculation, which gives

$$\underset{u_{t:t+m-1}}{\operatorname{argmin}} \quad \delta_t(u_{t:t+m})^\top \Big(\bar{C}_m \widehat{\mathbf{\Sigma}}_t \bar{C}_m^\top\Big)^{-1} \delta_t(u_{t:t+m}) + \\ + \log\det\Big(\bar{C}_m \widehat{\mathbf{\Sigma}}_t \bar{C}_m^\top\Big) \tag{9}$$

where $\delta_t(u_{t:t+m}) \coloneqq r_{t+1:t+m} - \bar{C}_m \widehat{\mathbf{m}}_t$.

Note that the terms $\widehat{\mathbf{m}}_t$ and $\widehat{\mathbf{\Sigma}}_t$ depend on $u_{t:t+m-1}$. This implies that we are reshaping the posterior distribution by optimally choosing the future control variable following (8). It is worth mentioning that if we use the offline data, we are able to compute almost all the parameters composing $\widehat{\mathbf{m}}_t$ and $\widehat{\mathbf{\Sigma}}_t$ except $\mathbf{K}_t^\star$ beforehand. It is well known that this method heavily depends on the choice of kernel for constructing $\mathbf{K}_t$, $\mathbf{K}_t^\star$ and $\overline{\mathbf{K}}_t$.

Remark 1: The structure of System (1) inherently implies the stability of our design. Thus, for nonlinear FIR systems the proposed approach does not impact the stability properties of the model.

The optimization (9) is non-convex, and it can be challenging to solve. However, it shares a similar structure of the well-known model selection problem described in [20, Sec. 5.4.1].

IV. AUTOREGRESSIVE SYSTEM ANALYSIS

In this section, we discuss the challenges and potential methods for extending our SelfMPC algorithm to an unknown autoregressive system.

Consider an autoregressive system driven by input with Gaussian noise, described by the equation:

$$\forall t \in \mathbb{N} \qquad y_t = g_t(y_{0:t-1}, u_{0:t}) + w_t$$

where $y_t \in \mathbb{R}^{n_{\rm y}}$ and $u_t \in \mathbb{R}^{n_{\rm u}}$ represent the measurable output and input variables at time t, respectively, w_t denotes the noise affecting y_t, and, for every $t \in \mathbb{N}$, $g_t : \mathbb{R}^{tn_{\rm y}} \times \mathbb{R}^{(t+1)n_{\rm u}}$ is an unknown function to the designer. We maintain the same stochastic assumption on the process w as outlined in Assumption 1.

Given the general nonlinear nature of $(g_t)_{t\in\mathbb{N}}$, the distribution of the prediction of $y_{t:t+m}$ given the past available measurements cannot be assessed analytically. Consequently, estimating the distribution of future measurements based on current measurements can only be achieved numerically. As a result, the cost function following the SelfMPC algorithm cannot have an analytical expression, rendering analysis challenging.

In the linear scenario where the function g_t is linear for every $t \in \mathbb{N}$, we have

$$\forall t \in \mathbb{N} \qquad y_t = a_t y_{0:t-1} + b_t u_{0:t} + w_t$$

where $a_t \in \mathbb{R}^{n_{\rm y} t}$ and $b_t \in \mathbb{R}^{n_{\rm u}(t-1)}$ for all $t \in \mathbb{N}$. For compactness, we define $\theta = (a_t, b_t)_{t\in\mathbb{T}}$. Here, the prediction of $y_{t:t+m}$ given the past available measurements is distributed according to a Normal distribution with variance that depends on W and mean as a nonlinear function of a_t and b_t. Hence, it is not possible to define a prior distribution on θ such that the joint distribution of $y_{t:t+m}$, θ can be determined analytically. Thus, even in the linear case, retrieving the cost function of SelfMPC is difficult.

Alternatively, frequentist methods for identifying θ could be considered as an approach to tackle this challenge. Assuming we collect a sufficient amount of data to estimate the true parameter θ°, with the frequentist perspective in mind, we treat the parameter as deterministic values and aim to minimize the gap between our estimate of the parameter and their true values. According to the central limit theorem, we have

$$\sqrt{N_t}\Big(\theta^\circ - \hat{\theta}\Big) \sim \mathcal{N}\big(\cdot \,\big|\, \mathbf{0}_{(n_a+n_b)\times 1}, \Sigma_\theta\big) \tag{10}$$

where $\hat{\theta}$ denotes the estimate of θ° and N_t is the number of independent observations. It is worth noting that the process or measurement noise is not requrired to be Gaussian for (10) to hold. Thus, we approximate the parameter of the system as:

$$\theta \sim \mathcal{N}\left(\cdot \,\middle|\, \hat{\theta}, \frac{1}{N_t}\Sigma_\theta\right).$$

We then proceed to derive the probability density of the controlled variable z_t with future m steps:

$$p(z_{t+1:t+m}|\mathcal{D}) = \int p(z_{t+1:t+m}|\theta) \cdot \mathcal{N}\left(\cdot \,\middle|\, \hat{\theta}, \frac{1}{N_t}\Sigma_\theta\right) d\theta.$$

It is important to note that $p(z_{t+1:t+m}|\mathcal{D})$ is also modified based on the choice of $u_{t:t+m-1}$, allowing us to select future inputs to maximize the likelihood of $z_{t+1:t+m}$ aligning with the reference signal.

V. Numerical Example

To evaluate the efficacy of the proposed SelfMPC scheme, we present a numerical example in the form of (1). The unknown nonlinear model under consideration is described by the following representation

$$\begin{aligned} y_{t+1} &= \frac{-u_{t-2}}{\sqrt[3]{1+u_{t-1}}} + \frac{(u_t+10)(1-u_{t-1})}{10e^{\frac{u_{t-2}}{10}+1}-1} + \\ &\quad + u_{t-2}e^{\frac{u_t}{8}} + w_t \\ z_t &= y_t \end{aligned} \tag{11}$$

where the stochastic influence w_t is represented by Gaussian noise, with a standard deviation of $5 \cdot 10^{-2}$.

In this simulation example, we incorporate the SelfMPC algorithm, as detailed in Section III, to control the unknown nonlinear model specified in (1). We implement the SelfMPC algorithm with two fixed prediction horizons $m = 5$ or $m = 10$ and impose input bounds of $3 \leq u \leq 5$ to generate the control input sequence. We gather a dataset of $N = 500$ measurements from the plant, introducing white noise uniformly distributed within the input bounds. Note that this collected dataset is also influenced by the additive noise w_t as outlined in (11). In all the simulation, we use the kernel $k(a,b) = \lambda \exp -\beta\|a-b\|^2$ where $\lambda, \beta \in (0,\infty)$ are selected using the empirical Bayes approach [20, Section 5.4.1] based on the collected data. All the optimization problems are solved using the Matlab optimization toolbox. Furthermore, the variance of the noise W is selected using the same procedure. Subsequently, we assess the performance of the proposed SelfMPC method by steering the unknown system described in (11) to track a reference signal which is a square wave ranging from 2.5 to 4 with different amplitude levels as shown in Fig. 2. The evaluation is conducted using the performance index $T^{-1}\|z_t - r_t\|$ where $T = 120$ is the length of the considered time window. The oscillatory performance in Fig. 2 can be improved by increasing the prediction horizon m, allowing the controller to better anticipate future dynamics.

To compare the performance of our design with conventional approaches, we adopt a predictive control strategy

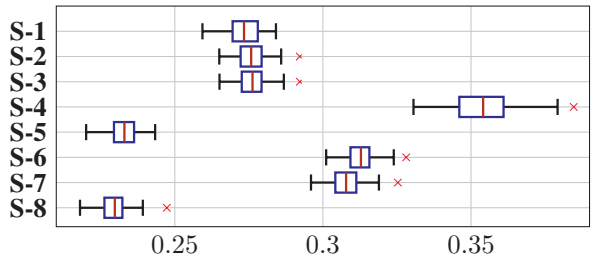

Fig. 1. Box plots of the control performance index on the controlled variable z_t over 100 Monte Carlo simulations.

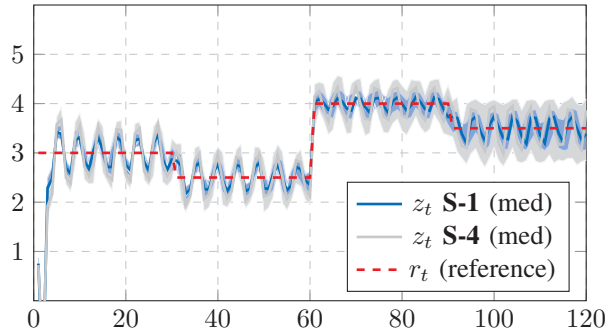

Fig. 2. Performance of the proposed SelfMPC tracking the reference signal r_t with $m = 5$ over 100 Monte Carlo simulations (shaded area).

using the same identification and prediction strategy outlined in Section III-A. However, we employ a fixed set of cost weights instead. In other words, instead of considering the cost function in (9), we minimize

$$\underset{u_{t:t+m}}{\operatorname{argmin}} \quad \left\|\delta_t(u_{t:t+m})\right\|_Q^2 + \left\|u_{t+1:t+m}\right\|_R^2 \tag{12}$$

where $Q, R \in \mathbb{R}^{m\times m}$ are weight matrices. We consider three different choices for Q and R to emulate a trial and error tuning procedure. In particular, we selected the following three choices:

$$Q = 0.1I_m, \qquad R = 0.01I_m, \tag{13}$$
$$Q = 5I_m, \qquad R = 0.01I_m, \tag{14}$$
$$Q = I_m, \qquad R = 0.01I_m. \tag{15}$$

Moreover, we investigate the impact of different prediction horizons, considering both a shorter horizon ($m = 5$) and a longer horizon ($m = 10$). The comprehensive evaluation of the controller performance using the index described before is visualized in Fig. 1 where the examined control strategies with different settings are listed as

- **S-1.** SelfMPC with $m = 5$.
- **S-2.** MPC of (12) with weights in (13) with $m = 5$.
- **S-3.** MPC of (12) with weights in (14) with $m = 5$.
- **S-4.** MPC of (12) with weights in (15) with $m = 5$.
- **S-5.** SelfMPC with $m = 10$.
- **S-6.** MPC of (12) with weights in (13) with $m = 10$.
- **S-7.** MPC of (12) with weights in (14) with $m = 10$.
- **S-8.** MPC of (12) with weights in (15) with $m = 10$.

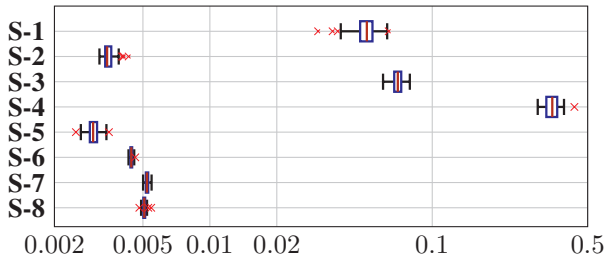

Fig. 3. Box plots (x-axis with logarithm scale) of the variance of the input variable u_t when tracking a constant (from $t = 61$ to $t = 90$) over 100 Monte Carlo simulations.

From the results depicted in Fig. 1, it is evident that the performance of predictive control with fixed cost gains is influenced by the choice of gain values. However, tuning these gains, even for simulation purposes, can be prohibitively costly. In contrast, the data-driven SelfMPC method exhibits decent performance compared to using pre-selected fixed cost gains. Particularly noteworthy is the observation that the performance of fixed cost gains may degrade as the prediction horizon increases, whereas the proposed SelfMPC method consistently enhances its performance. This discrepancy in performance can be attributed to the fact that fixed gain selection fails to account for the uncertainty inherent in predictions, often relying too heavily on unreliable predictions, especially those further into the future. In contrast, our proposed SelfMPC approach automatically adjusts the cost weights, taking into consideration the reliability of predictions, thereby yielding significantly improved performance. Moreover, we illustrate the variance of the input signal when tracking a constant reference (from $t = 61$ to $t = 90$) for all the listed control strategies in Fig. 3.

VI. Conclusions

In this paper, we address the challenges associated with the robust control method Model Predictive Control (MPC). Particularly, when faced with an unknown system and the tuning of hyperparameters becomes costly or even impractical, conventional MPC methods are inadequate for handling such scenarios. To overcome these challenges, we introduce Maximum Likelihood MPC (SelfMPC) tailored for a class of unknown nonlinear systems. This innovative approach selects the control sequence by maximizing the likelihood that the system output aligns with the desired reference trajectory. We demonstrate that this algorithm automatically tunes the gains of the cost function while providing a guarantee of stability. In this sense, SelfMPC becomes a valuable tool for data-driven control, utilizing Gaussian process regression for model learning and eliminating the need for additional tuning. As part of our future work, we aim to extend the application of the SelfMPC method to autoregressive systems with unknown parameters.

References

[1] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publ. LLC., 2020.

[2] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Camb. Univ. Press, 2017.

[3] M. L. Darby and M. Nikolaou, "MPC: Current practice and challenges," *Control Eng. Pract.*, vol. 20, no. 4, pp. 328–342, 2012.

[4] M. Scandella, A. Ghosh, M. Bin, and T. Parisini, "Traffic-light control in urban environment exploiting drivers' reaction to the expected red lights duration," *Transp. Res. C: Emerg. Technol.*, vol. 145, p. 103910, 2022.

[5] A. Ghosh, M. Scandella, M. Bin, and T. Parisini, "Traffic-Light Control at Urban Intersections Using Expected Waiting-Time Information," in *Proc. 60th IEEE Conf. Decis. Control (CDC)*, 2021, pp. 1953–1959.

[6] G. Pozzato, M. Müller, S. Formentin, and S. M. Savaresi, "Economic MPC for online least costly energy management of hybrid electric vehicles," *Control Eng. Pract.*, vol. 102, p. 104534, 2020.

[7] B. Sonzogni, J. M. Manzano, M. Polver, F. Previdi, and A. Ferramosca, "CHoKI-based MPC for blood glucose regulation in artificial pancreas," in *Proc. 22nd IFAC World Congr.*, vol. 56, no. 2, 2023, pp. 9672–9677.

[8] F. Cairoli, G. Fenu, F. A. Pellegrino, and E. Salvato, "Model Predictive Control of Glucose Concentration Based on Signal Temporal Logic Specifications with Unknown-Meals Occurrence," *Cybern. Syst.*, vol. 51, no. 4, pp. 426–441, 2020.

[9] W. Langson, I. Chryssochoos, S. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.

[10] D. Q. Mayne, M. M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[11] D. Chatterjee and J. Lygeros, "On stability and performance of stochastic predictive control techniques," *IEEE Trans. Autom. Control*, vol. 60, no. 2, pp. 509–514, 2014.

[12] J. Coulson, J. Lygeros, and F. Dörfler, "Data-enabled predictive control: In the shallows of the DeePC," in *Proc. 18th Eur. Control Conf. (ECC)*, 2019, pp. 307–312.

[13] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Trans. Autom. Control*, vol. 66, no. 4, pp. 1702–1717, 2020.

[14] N. Ab Azar, A. Shahmansoorian, and M. Davoudi, "From inverse optimal control to inverse reinforcement learning: A historical review," *Annu. Rev. Control*, vol. 50, pp. 119–138, 2020.

[15] F. Sorourifar, G. Makrygirgos, A. Mesbah, and J. A. Paulson, "A Data-Driven Automatic Tuning Method for MPC under Uncertainty using Constrained Bayesian Optimization," in *Proc. 16th IFAC Symp. Adv. Control Chem. Process. (ADCHEM)*, vol. 54, no. 3, 2021, pp. 243–250.

[16] F. Abbracciavento, F. Zinnari, S. Formentin, A. G. Bianchessi, and S. M. Savaresi, "Multi-intersection traffic signal control: A decentralized MPC-based approach," *IFAC J. Syst. Control*, vol. 23, 2023.

[17] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Trans. Autom. Control*, vol. 65, no. 2, pp. 636–648, 2019.

[18] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 3, pp. 269–296, 2020.

[19] G. Yang, M. Scandella, S. Formentin, and T. Parisini, "Automated Data-Driven Tuning of Learning-Based Model Predictive Control (SelfMPC): A Maximum-Likelihood Approach," in *Proc. 22nd Eur. Control Conf.* IEEE, 2024, pp. 104–109.

[20] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press Ltd, 2006.

[21] M. Mazzoleni, A. Chiuso, M. Scandella, S. Formentin, and F. Previdi, "Kernel-based system identification with manifold regularization: A Bayesian perspective," *Automatica*, vol. 142, p. 110419, 2022.

[22] F. Dörfler, J. Coulson, and I. Markovsky, "Bridging direct and indirect data-driven control formulations via regularizations and relaxations," *IEEE Trans. Autom. Control*, vol. 68, no. 2, pp. 883–897, 2022.

[23] V. Breschi, A. Chiuso, and S. Formentin, "Data-driven predictive control in a stochastic setting: A unified framework," *Automatica*, vol. 152, p. 110961, 2023.

[24] M. A. Álvarez, L. Rosasco, and N. D. Lawrence, "Kernels for Vector-Valued Functions: A Review," *Found. Trends Mach. Learn.*, vol. 4, no. 3, pp. 195–266, 2012.

[25] C. A. Micchelli and M. Pontil, "On Learning Vector-Valued Functions," *Neural Comput.*, vol. 17, no. 1, pp. 177–204, 2005.