Data-based predictive control with nonlinear and multi-step models

Zur Erlangung des akademischen Grades eines

Dr.-Ing.

von der Fakultät Bio- und Chemieingenieurwesen
der Technischen Universität Dortmund
genehmigte Dissertation

vorgelegt von

M.Sc. Felix Malte Hermes Fiedler

aus

Berlin, Deutschland

Tag der mündlichen Prüfung: 12.06.2024

1. Gutachter: Prof. Dr. Sergio Lucia Gil
2. Gutachter: Prof. Dr. Sebastien Gros

Dortmund 2024

Danksagung

An dieser Stelle möchte ich mich bei allen Menschen bedanken, die mich auf meinem Weg begleitet haben. Ohne eure Unterstützung wäre ich nicht so weit gekommen.

Danke Sergio, für dein Vertrauen, die kreative Freiheit und die gemeinsame Arbeit an so vielen spannenden Projekten. Du bist der beste Betreuer und Mentor, den ich mir hätte wünschen können.

Danke Prof. Sebastien Gros, für die Übernahme des Zweitgutachtens, die konstruktiven Anregungen und die spannenden Diskussionen.

Danke Benjamin und David, für eure Freundschaft, den Austausch beim gemeinsamen Essen und Kaffee, und die ständige Motivation. Geteilte Sorgen sind halbe Sorgen.

Danke an alle Kolleginnen und Kollegen vom PAS-Lehrstuhl, für die gemeinsame Arbeit, die Diskussionen und die wunderschöne Abschiedsfeier.

Danke an alle meine Freundinnen und Freunde, die bis zum Schluss mit mir mitgefiebert haben.

Danke Agathe, für deine Geduld und deine Liebe. Du hast dir aus nächster Nähe meine Zweifel, Ängste und Freuden angehört und mich immer wieder aufgebaut oder mit mir gefeiert.

Danke an meine Familie: Judith, Werner und Lena. Für eure Liebe, euer Vertrauen und eure bedingungslose Unterstützung. Ohne euch wäre ich nicht der Mensch, der ich heute bin.

Kurzzusammenfassung

In den letzten Jahrzehnten hat die modellprädiktive Regelung (MPC) herausragende Ergebnisse in verschiedenen Anwendungsbereichen erzielt. Dieses Potenzial wird jedoch häufig durch die Schwierigkeiten bei der Beschaffung eines regelungsorientierten Modells eingeschränkt. Um dieser Herausforderung zu begegnen, werden in dieser Dissertation fortgeschrittene Methoden zur datenbasierten Systemidentifikation untersucht, wobei der Schwerpunkt auf nichtlinearer State-Space-Identifikation und linearer Multi-Step-Identifikation mit deterministischen und probabilistischen Modellen liegt. Die resultierenden Modelle werden zur Formulierung von datenbasierten nichtlinearen, ökonomischen und stochastischen MPC-Reglern verwendet.

Die nichtlineare State-Space-Identifikation mit neuronalen Netzen wird für Systeme mit State-Feedback und Output-Feedback vorgestellt. Die Datenerfassung für die Identifikationsaufgabe und die Formulierung des nichtlinearen MPC-Reglers mit neuronalem Netzmodell wird mit der vorgestellten Open-Source-Software do-mpc unterstützt.

Multi-Step-Modelle prädizieren finite Sequenzen eines dynamischen Systems. Für lineare Systeme lässt sich die Multi-Step-Identifikation als lineare Regressionsaufgabe lösen. In dieser Arbeit wird ein MPC-Regler mit identifiziertem Multi-Step-Modell vorgeschlagen und der Ansatz mit der populären Methode Data-Enabled Predictive Control verglichen. Zusätzlich wird in dieser Arbeit für nicht-deterministische Systeme ein probabilistischer Multi-Step-Identifikationsansatz präsentiert. Unter ausschließlicher Verwendung von aufgezeichneten Daten eines linearen Systems mit Unsicherheiten führt die vorgeschlagene Methode zu einem stochastischen MPC-Regler mit vorteilhaften Eigenschaften. Darüber hinaus werden die inhärenten Vorteile der Multi-Step-Identifikation gegenüber der State-Space-Identifikation für lineare Systeme mit Messrauschen dargestellt. Es wird gezeigt, dass ein identifiziertes und rekursiv ausgewertetes State-Space-Modell zu fehlerhaften Multi-Step-Vorhersagen führt, während das entsprechende identifizierte Multi-Step-Modell im Erwartungswert unverfälscht ist.

Eine mögliche Erweiterung der vorgeschlagenen Methoden ist die nichtlineare probabilistische Systemidentifikation unter Verwendung von neuronalen Netzen mit Bayesian Last Layer. Als weiterer Hauptbeitrag führt der vorgeschlagene Trainingsalgorithmus zu einer verbesserten Vorhersagequalität der resultierenden Modelle. Dieser Beitrag ermöglicht den Entwurf von datenbasierten stochastischen MPC-Reglern mit neuronalen Netzmodellen für zukünftige Arbeiten.

Abstract

Over the last decades, model predictive control (MPC) has demonstrated exceptional performance in control tasks across various domains. Unfortunately, this potential is often limited by the challenge of obtaining a control-oriented model. To address this challenge, this thesis explores advanced methods for data-based system identification, specifically focusing on nonlinear state-space identification and linear multi-step identification with deterministic and probabilistic models. The resulting models are used to formulate data-based nonlinear, economic and stochastic MPC controllers.

Nonlinear state-space identification with neural networks is proposed for systems with state-feedback and output-feedback. Sampling data for the identification task and formulating the nonlinear MPC controller with neural network system model is enabled by the introduced open-source software do-mpc.

Multi-step models predict finite sequences of a dynamic system. For linear systems, multi-step identification boils down to a tractable linear regression task. This thesis proposes an MPC controller with identified multi-step model and compares the approach with the recently popularized data-enabled predictive control method. Additionally, for non-deterministic systems, this thesis introduces a probabilistic multi-step identification approach. Using only recorded data of a noise-affected linear system, the proposed method yields an output-feedback stochastic MPC controller with favorable properties. Furthermore, the inherent advantages of multi-step identification over state-space identification for linear systems with measurement noise are demonstrated. It is shown that an identified and recursively evaluated state-space model yields biased multi-step predictions, whereas the respective identified multi-step model is, in expectation, unbiased.

A possible extension of the proposed methods is nonlinear probabilistic system identification using neural networks with Bayesian last layer. Another major contribution is the proposal of a novel training algorithm that enhances the predictive distribution of the resulting models. This contribution enables the design of data-based stochastic MPC with neural network models for future work.

Contents

List of Notation

Numbers and Arrays

a	A scalar
$\boldsymbol{a}$	A vector
a	A scalar random variable
a	A vector-valued random variable
$\boldsymbol{A}$	A matrix
A	A matrix-valued random variable
$\boldsymbol{I}_n$	Identity matrix with n rows and n columns
$\boldsymbol{I}$	Identity matrix with dimensionality implied by context

Sets

$\mathbb{A}$	A set
$\mathbb{I}_{[0,n]}$	The set of all integers between 0 and n
$\mathbb{I}$	The set of integers
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^+$	The set of positive real numbers
$\{0, 1\}$	The set containing 0 and 1
$[a, b]$	The real interval including a and b
$(a, b]$	The real interval excluding a but including b

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1.
$\boldsymbol{a}_{[k,N]}$	Sequence (time-series) of elements $\boldsymbol{a}$ from k to N stacked as vector.
$\boldsymbol{a}_i$	Element i of sequence (time-series) of elements $\boldsymbol{a}$.
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$[\boldsymbol{A}]_{i,j}$	Block element i, j of matrix $\boldsymbol{A}$ (block structure implied by context)

Linear Algebra Operations

$\boldsymbol{A}^\top$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^\dagger$	Moore-Penrose pseudoinverse of $\boldsymbol{A}$
$\boldsymbol{A} \odot \boldsymbol{B}$	Element-wise (Hadamard) product of $\boldsymbol{A}$ and $\boldsymbol{B}$
$\boldsymbol{A} \otimes \boldsymbol{B}$	Kronecker product of $\boldsymbol{A}$ and $\boldsymbol{B}$
$\det(\boldsymbol{A})$	Determinant of $\boldsymbol{A}$
$\text{diag}(\boldsymbol{a})$	Diagonal matrix with the elements of $\boldsymbol{a}$

Probability

$p(\mathbf{x})$	A probability density function
$p(\mathbf{x}\|\boldsymbol{y})$	Conditional probability of random variable $\mathbf{x}$ given $\boldsymbol{y}$
$\mathbf{x} \sim p$	Random variable $\mathbf{x}$ has distribution p
$\mathbb{E}[\mathbf{x}]$	Expectation of random variable $\mathbf{x}$
$\text{Var}(\text{x})$	Variance of scalar random variable x
$\text{Cov}(\mathbf{x})$	Covariance of multivariate random variable $\mathbf{x}$
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Shorthand notation for $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f \circ g$	Composition of the functions f and g
$\log x$	Natural logarithm of x
$\|\boldsymbol{x}\|_p$	L^p norm of $\boldsymbol{x}$
$\|\boldsymbol{x}\|$	L^2 norm of $\boldsymbol{x}$
$\|\boldsymbol{x}\|$	L^1 norm of $\boldsymbol{x}$
$\|\boldsymbol{x}\|_{\boldsymbol{Q}}$	Weighted L^2 norm $\|\boldsymbol{x}\|_{\boldsymbol{Q}} = \sqrt{\boldsymbol{x}^\top \boldsymbol{Q} \boldsymbol{x}}$
$\|\boldsymbol{A}\|_F$	Frobenius norm of matrix $\boldsymbol{A}$
$\mathcal{K}$	Class of functions that are zero at zero, continuous and strictly increasing

List of Figures

List of Tables

Chapter 1

Introduction

Human-caused climate change is the major challenge of the 21st century. Approximately 3.3 to 3.6 billion people are currently highly vulnerable to the effects of climate change and all of humanity is facing adverse impacts, related losses and damages [1]. To mitigate these effects, it is imperative to rapidly reduce greenhouse gas emissions and to accelerate the decline towards net zero [2]. A significant source of global greenhouse gas emissions stems from electricity and heating (23 % in 2019) and can be mainly attributed to consumption in buildings (47 %) and industry (43 %) [2]. Reducing the consumption of energy and heating in these sectors is thus a pivotal measure to combat climate change. An important element towards this goal are advanced control systems for the efficient use of resources and energy [3]. *Model predictive control* (MPC) is a promising method for this task. In building systems, MPC can reduce energy consumption between 10 %-30 % [3], [4]. Furthermore, MPC has applications in the energy-intensive process industries [5] and power electronics [6], [7], where the resulting improved operation can yield significantly higher energy and resource efficiency.

1.1 Model predictive control

Model predictive control (MPC) is an advanced control method based on predicting and optimizing the future behavior of the controlled system [8]. To this end, MPC requires a mathematical model of the system, a control objective function that should be minimized and additional safety and performance constraints. These components form an *optimal control problem* that is solved repeatedly to yield the closed-loop control actions. Importantly, MPC can be applied to nonlinear systems with multiple inputs and outputs, and allows the introduction of constraints that naturally arise in various control tasks. Furthermore, MPC is not limited to set-point tracking or disturbance rejection, but can also directly incorporate economic control objectives. The resulting controller, coined *economic MPC* [9], [10], can be employed, for example, for building systems or in the process industries to directly minimize the energy consumption, resource usage or emissions. The combination of economic control objectives, the consideration of nonlinear system behavior and the anticipatory control actions are responsible for the desired improvements in performance and efficiency of MPC controlled systems.

The model to predict the future behavior of the controlled system is central to every form of MPC. In practice, obtaining a system model of reasonable accuracy can be a significant challenge and constitutes a major obstacle for the adoption of MPC [11], [12]. Physical modeling is time consuming, requires expert knowledge and does not benefit significantly from the economies of scale. Additionally, complex system models are often created in dedicated dynamic modeling software which is typically not suitable for optimization and MPC.

Another major challenge for the application of MPC is the presence of uncertainty. MPC does provide a certain robustness to model errors and disturbances, due to the feedback nature of the control law [8]. However, this inherent robustness is often insufficient if the system operates close to safety-critical constraints. A promising approach to handle this situation is stochastic MPC (SMPC) [13], [14]. SMPC can be applied when the uncertainty in the model follows a known probability distribution function and it relies on the formulation of chance constraints, that is, constraints with specified probability of violation. Unfortunately, the adoption of stochastic MPC is even more impaired by the complexity of obtaining a suitable probabilistic model of the controlled system.

1.2 System identification

A potential solution to mitigate the challenge of obtaining a control-oriented model is data-based system identification [15]. System identification is an established method in control engineering [16], [17] and commonly used for traditional MPC schemes [11]. In many applications, system data is available from historic measurements or simulation results of a non control-oriented model, and system identification can be applied. In contrast to modeling, system identification does hold the promise of economies of scale, at least for related systems, and is thus an essential element to foster the adoption of MPC. The characteristics of MPC, namely, the ability to consider nonlinear and uncertain systems, and the anticipatory control strategy, enable the application of advanced methods for system identification. These methods include nonlinear system identification, multi-step identification and probabilistic identification, and are introduced in the following.

1.2.1 Nonlinear identification

Traditional model-based control, including MPC, is based on linear models [11]. The assumption of linearity is often sufficient for regulation tasks, where the system is steered to a pre-determined state. However, most systems exhibit nonlinear behavior and linear models should only be used as local approximations. For economic MPC, the operating condition is not a priori known and can include a wide range of possible states. In this scenario, nonlinear system identification should be considered.

A main challenge for nonlinear system identification is model selection [11]. Popular methods are feature regression [18], [19], Gaussian processes [20]–[22] and neural networks [23]–[25]. Among those, neural networks are of particular interest in recent years, due to their versatility, expressiveness and the ability to learn from large quantities of data. While neural networks are a potent model class to capture

complex nonlinear systems, they are also prone to silent failures [26]. In other words, traditional neural networks may yield severely flawed predictions without any indication of their confidence. This can be a major obstacle for their application, especially for safety-critical control tasks and for systems subject to uncertainty.

1.2.2 Multi-step identification

To enable anticipatory control actions, MPC predicts and optimizes the future behavior of the system. Traditionally, this prediction is obtained by recursively evaluating a state-space model of the system [11]. However, a recent trend in system identification for MPC is to directly identify a multi-step model [27]–[30]. A multi-step model can simultaneously predict finite sequences of future states and omits the recursive evaluation of the state-space model. At first glance, a multi-step model appears to add unnecessary complexity to the system identification task. However, recent works have shown that multi-step identification can have a better accuracy than state-space models [29], [30], and complexity is only added to the offline identification task, not to the online control task. A closely related trend to MPC with identified multi-step model is data-enabled predictive control (DeePC) [31]–[37]. DeePC draws from the behavioral systems theory [38] and combines an implicit multi-step identification and control task in a single optimization problem. Both methods, MPC with identified multi-step model and DeePC, have promising extensions to tackle the challenge of uncertain systems [30], [39].

1.3 Aim, scope and outline

The aim of this thesis is to explore advanced data-based system identification, specifically nonlinear and multi-step identification, for nonlinear, economic and stochastic MPC. Ultimately, this combination seeks to facilitate the design and foster the widespread adoption of MPC. This endeavor has been carried out in multiple publications, which are the scope of this thesis.

1.3.1 Publications covered in this thesis

The publications covered in this thesis are listed below and their content is summarized successively.

[12] F. Fiedler, B. Karg, L. Lüken, D. Brandner, M. Heinlein, F. Brabender, and S. Lucia, "Do-mpc: Towards FAIR nonlinear and robust model predictive control", *Control Engineering Practice*, vol. 140, p. 105 676, Nov. 1, 2023 (Discussed in Chapter 2 and provided in Appendix C.1)

[40] F. Fiedler, A. Cominola, and S. Lucia, "Economic nonlinear predictive control of water distribution networks based on surrogate modeling and automatic clustering", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 16 636–16 643, 2020 (Discussed in Chapter 2 and provided in Appendix C.2)

[41] F. Fiedler and S. Lucia, "On the relationship between data-enabled predictive control and subspace predictive control", in *European Control Conference*, 2021, pp. 222–229 (Discussed in Chapter 3 and provided in Appendix C.3)

[42] F. Fiedler and S. Lucia, "Probabilistic Multi-Step Identification With Implicit State Estimation for Stochastic MPC", *IEEE Access*, vol. 11, pp. 117 018–117 029, 2023 (Discussed in Chapter 4 and provided in Appendix C.4)

[43] F. Fiedler and S. Lucia, "Improved uncertainty quantification for neural networks with Bayesian last layer", *IEEE Access*, vol. 11, pp. 123 149–123 160, 2023 (Discussed in Chapter 5 and provided in Appendix C.5)

Nonlinear system identification with neural networks is investigated in [12], [40]. In both works, an existing system model is approximated with a neural network for the purpose of formulating an MPC controller. The main objective in [12] is to motivate the importance of nonlinear system identification with neural networks for the widespread adoption of MPC. It is demonstrated in this work that an MPC controller with neural network system model can achieve near-identical performance to an MPC controller with exact model. Sampling data for the identification task, and formulating the MPC controller with neural network system model is enabled by the introduced open-source software package *do-mpc*, which is another main contribution in [12]. Neural network system identification is further investigated in [40]. In this work, economic MPC with neural network system model is applied to a large-scale water distribution network. This is a concrete example of a system for which a non control-oriented model is available, and where neural network system identification enables the application of MPC.

Multi-step identification for model predictive control is investigated in [41], [42]. As a preliminary for both works, a linear multi-step model is derived and the parameter estimation task is presented. With the identified multi-step model, an optimal control problem is proposed in [41] and the relationship to data-enabled predictive control is investigated. In [42], multi-step identification is further applied to systems subject to uncertainty. The proposed probabilistic multi-step identification allows the formulation of a stochastic MPC controller and enables safe control-decisions. As a main contribution in [42], it is shown that the identified multi-step model yields, in expectation, the same distribution as the true multi-step model, including the effect of estimating the initial state with a Kalman filter. Importantly, these favorable theoretical properties are not shared with an identified and recursively evaluated state-space model. The advantages of multi-step identification over state-space identification is further investigated in two simulation studies in [42]. While the proposed method is originally derived for linear systems, its applicability for the control of nonlinear systems is demonstrated.

Nonlinear probabilistic identification with neural networks, either with multi-step or state-space models, is a promising field for future work. In combination with stochastic MPC, this holds the potential to safely control complex nonlinear systems. As a prerequisite, uncertainty quantification with neural networks is investigated. Bayesian neural networks are a theoretical concept to obtain a distributed prediction, as required for stochastic MPC. Unfortunately, most practical implementations of Bayesian neural networks require sampling for training and inference [26], [44] and,

therefore, have only limited application for optimization-based control. To overcome this limitation, neural networks with Bayesian last layer are discussed in [43]. Neural networks with Bayesian last layer are a simplified Bayesian neural network and yield tractable Gaussian distributions [45], [46]. In [43], we propose an efficient training algorithm for neural networks with Bayesian last layer and an approach to tune the extrapolative uncertainty quantification. This enables their future application for safe data-based predictive control.

1.3.2 Other publications not covered in this thesis

Further publications that have been authored by the author of this thesis, but are not covered, are listed below.

- F. Fiedler and S. Lucia, "Model predictive control with neural network system model and Bayesian last layer trust regions", in *IEEE International Conference on Control & Automation*, Jun. 2022, pp. 141–147
- C. Döpmann, F. Fiedler, S. Lucia, and F. Tschorsch, "Optimization-Based Predictive Congestion Control for the Tor Network: Opportunities and Challenges", *ACM Transactions on Internet Technology*, vol. 22, no. 4, pp. 1–30, Nov. 30, 2022
- P. Guillén, F. Fiedler, H. Sarnago, S. Lucía, and O. Lucía, "Deep Learning Implementation of Model Predictive Control for Multioutput Resonant Converters", *IEEE Access*, vol. 10, pp. 65 228–65 237, 2022
- C. Döpmann, F. Fiedler, S. Lucia, and F. Tschorsch, "Towards Optimization-Based Predictive Congestion Control for the Tor Network", *Electronic Communications of the EASST*, vol. 80, 2021
- F. Fiedler, C. Döpmann, F. Tschorsch, and S. Lucia, "PredicTor: Predictive Congestion Control for the Tor Network", in *IEEE Conference on Control Technology and Applications*, Aug. 2020, pp. 863–870
- N. Krausch, S. Hans, F. Fiedler, S. Lucia, P. Neubauer, and M. N. C. Bournazou, "From screening to production: A holistic approach of high-throughput model-based screening for recombinant protein production", in *Computer Aided Chemical Engineering*, vol. 48, Elsevier, 2020, pp. 1723–1728
- F. Fiedler, D. Baumbach, A. Börner, and S. Lucia, "A Probabilistic Moving Horizon Estimation Framework Applied to the Visual-Inertial Sensor Fusion Problem", in *European Control Conference*, May 2020, pp. 1009–1016

1.3.3 Outline

This thesis is structured as follows. In Chapter 2 nonlinear system identification with neural networks is presented. Chapter 3 introduces linear multi-step identification and compares multi-step model predictive control with data-enabled predictive control. Probabilistic multi-step identification for stochastic MPC is presented in Chapter 4. Finally, neural networks with Bayesian last layer are presented as a tractable method for uncertainty quantification in Chapter 5. Conclusions and an outlook of this thesis are presented in Chapter 6.

Chapter 2

Nonlinear system identification

A main challenge for the application of model predictive control (MPC) is the availability of a control-oriented system model with reasonable accuracy. *System identification*, where a model is obtained from observed data of the system, is a commonly applied solution to overcome this challenge [15]–[17]. However, most traditional system identification approaches yield linear models and can be a poor approximation for highly nonlinear systems. Furthermore, most systems exhibit nonlinear behavior if they are operated in a broad range of conditions, as required for economic MPC.

To unlock the potential of nonlinear and economic MPC, this chapter investigates nonlinear system identification with neural networks. As preliminaries, this chapter introduces a general nonlinear dynamic system, the resulting nonlinear MPC formulation, fundamentals of system identification and neural networks for function approximation. Furthermore, the open-source software *do-mpc* is introduced, which, among other features, facilitates the formulation and application of nonlinear and economic MPC. Neural network system identification for MPC is investigated in two case studies, demonstrating the effectiveness of the approach.

This chapter summarizes the main findings of the publications [12] and [40]. The full publications are provided in Appendices C.1 and C.2.

2.1 Nonlinear model predictive control

Model predictive control is based on the formulation of an optimal control problem where a dynamic system model is employed to predict the future behavior of the system. A general nonlinear system and the resulting nonlinear MPC problem are introduced in the following.

2.1.1 Dynamic system

A discrete-time nonlinear dynamic system is introduced as:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{2.1a}$$

$$\boldsymbol{y}_k = h(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{2.1b}$$

with states $\boldsymbol{x} \in \mathbb{R}^{n_x}$, measurements $\boldsymbol{y} \in \mathbb{R}^{n_y}$, and inputs $\boldsymbol{u} \in \mathbb{R}^{n_u}$. The time-index is denoted with $k \in \mathbb{I}$ and the time is obtained as $t_{k+1} = t_k + \Delta t$ with timestep $\Delta t \in \mathbb{R}^+$. For ease of notation, further dependencies in (2.1) are omitted, such as algebraic states or uncontrolled parameters. The functions $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ are assumed to be $\mathcal{K}$-continuous [8, Def. 4.15].

2.1.2 Optimal control problem

Model predictive control (MPC) is based on the formulation of a constrained *optimal control problem* for a finite horizon $N \in \mathbb{I}$:

$$\min_{\boldsymbol{x}_{[0,N+1]}, \boldsymbol{u}_{[0,N]}} \sum_{k=0}^{N} l_k(\boldsymbol{x}_k, \boldsymbol{u}_k) + v(\boldsymbol{x}_{N+1}) \tag{2.2a}$$

$$\text{s.t.:} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}, \tag{2.2b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \qquad \forall k \in \mathbb{I}_{[0,N]}, \tag{2.2c}$$

$$g_k(\boldsymbol{x}_k, \boldsymbol{u}_k) \leq \boldsymbol{0} \qquad \forall k \in \mathbb{I}_{[0,N]}, \tag{2.2d}$$

$$g_N(\boldsymbol{x}_N) \leq \boldsymbol{0}. \tag{2.2e}$$

with stage cost $l_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$, terminal cost $v : \mathbb{R}^{n_x} \to \mathbb{R}$, stage constraints $g_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_{g,k}}$ and terminal constraints $g_N : \mathbb{R}^{n_x} \to \mathbb{R}^{n_{g,N}}$. Importantly, the sequence of states $\boldsymbol{x}_{[0,N+1]}$ is constrained by the initial state $\boldsymbol{x}_{\text{init}}$ and the dynamic system (2.1a). The optimal control problem (2.2) is solved for the initial state $\boldsymbol{x}_{\text{init}}$, yielding the optimal control sequence $\boldsymbol{u}^*_{[0,N]}$ and the predicted optimal state sequence $\boldsymbol{x}^*_{[0,N+1]}$. For the control task, the first element $\boldsymbol{u}^*_0$ is then applied to the system. Feedback control is achieved by repeatedly solving (2.2) at each time k for the current state $\boldsymbol{x}_{\text{init}} = \boldsymbol{x}_k$. This yields the MPC control law:

$$\boldsymbol{u}_k = \kappa_{\text{MPC}}(\boldsymbol{x}_k). \tag{2.3}$$

Further details on verifying the theoretical properties of the control law (2.3), such as closed-loop stability and recursive feasibility, can be found in the literature [8] and are not within the scope of this thesis.

2.2 System identification

The core of the model predictive control formulation (2.2) is the dynamic system model (2.1). System identification denotes the task of identifying the dynamic system model from observed data and is introduced in the following. To simplify the discussion, it is initially assumed that the system exhibits state-feedback. The setting in which the system exhibits output-feedback is discussed in Subsection 2.2.2.

Assumption 2.1. *The system* (2.1) *exhibits state-feedback, that is,* $h(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{x}_k$.

For a system with state-feedback, the observed data of system (2.1) is composed of a recorded sequence of states and inputs, that is:

$$\boldsymbol{x}_{[1,m+1]} = \left[\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_{m+1}^\top\right]^\top, \; \boldsymbol{u}_{[1,m]} = \left[\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_m^\top\right]^\top. \tag{2.4}$$

In practice, data in the form of (2.4) may be obtained either from measurements of a real-world system or from simulation results of a high-fidelity model. If Assumption 2.1 holds, and with system data in the form of (2.4), we seek to identify:

$$\boldsymbol{x}_{k+1} = \hat{f}(\boldsymbol{x}_k, \boldsymbol{u}_k; \boldsymbol{W}), \tag{2.5}$$

where $\hat{f}(\cdot, \cdot; \boldsymbol{W})$ is a model structure with parameters $\boldsymbol{W}$. System identification involves, among others, the task of determining a suitable model structure and estimating the model parameters [25]. Commonly used structures for control-oriented models are linear models [16], [17], models with nonlinear features [18], [19], Gaussian processes [20]–[22] and neural networks [23]–[25]. Additionally, each of these structures can be further subdivided, for example by selecting the type of nonlinear features, the kernel function of a Gaussian process, or the structural hyperparameters of the neural network. While all of these model structures have distinct advantages and disadvantages, especially neural networks have shown promising results in recent years [23]–[25]. These models can capture complex nonlinear system behavior from large datasets and are suitable for gradient-based optimization, as required for efficient MPC implementations.

2.2.1 Neural network system identification

A general feed-forward neural network with L hidden layers, input $\boldsymbol{v} \in \mathbb{R}^{n_v}$ and predicted outputs $\hat{\boldsymbol{t}} \in \mathbb{R}^{n_t}$ is introduced as the function:

$$\hat{\boldsymbol{t}} = NN(\boldsymbol{v}; \mathbb{W}_{L+1}) = g_{L+1} \circ h_{L+1} \circ \cdots \circ g_1 \circ h_1(\boldsymbol{v}), \tag{2.6}$$

where $\circ$ denotes function composition. Each layer consists of a linear mapping $h_l(\cdot)$ followed by a nonlinear activation function $g_l(\cdot)$, that is:

$$\boldsymbol{h}_l^\top = h_l(\boldsymbol{a}_{l-1}) = \left[\boldsymbol{a}_{l-1}^\top, \, 1\right] \boldsymbol{W}_l \qquad l \in \mathbb{I}_{[1,L+1]}, \tag{2.7a}$$

$$\boldsymbol{a}_l = g_l(\boldsymbol{h}_l) \qquad l \in \mathbb{I}_{[1,L+1]}, \tag{2.7b}$$

where $\boldsymbol{a}_0 = \boldsymbol{v}$, $\boldsymbol{a}_{L+1} = \hat{\boldsymbol{t}}$, and $\boldsymbol{a}_l \in \mathbb{R}^{n_{a,l}}$ for all $l \in \mathbb{I}_{[1,L]}$. The number of neurons in layer l is denoted as $n_{a,l}$, and the weights $\boldsymbol{W}_l \in \mathbb{R}^{n_{a,l-1}+1 \times n_{a,l}}$ include the bias term. The set of weight matrices with cardinality $L+1$ is denoted

$$\mathbb{W}_{L+1} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_{L+1}\}. \tag{2.8}$$

The model structure is thus defined by the number of layers L and the number of neurons $n_{a,l}$ in each layer. Next, the parameters of the model must be estimated using the observed data of the system (2.1). To this end, the dataset $\mathcal{D} = \left\{\boldsymbol{v}^{(i)}, \boldsymbol{t}^{(i)}\right\}_{i=1}^m$ is

introduced, where $\boldsymbol{v}^{(i)\top} = \left[\boldsymbol{x}_i^\top,\ \boldsymbol{u}_i^\top\right]$ and $\boldsymbol{t}^{(i)} = \boldsymbol{x}_{i+1}$. The neural network function (2.6) is fitted to the data $\mathcal{D}$ by minimizing the mean-squared-error (MSE) loss function:

$$J_{\text{MSE}}(\mathbb{W}_{L+1}; \mathcal{D}) = \frac{1}{m} \sum_{i=1}^{m} \left\| NN\left(\boldsymbol{v}^{(i)}; \mathbb{W}_{L+1}\right) - \boldsymbol{t}^{(i)} \right\|_2^2, \tag{2.9}$$

where $\|\cdot\|_2$ denotes the 2-norm. Alternative training losses and additional regularization terms may be employed depending on the specific problem. The optimal set of weights is obtained as:

$$\mathbb{W}_{L+1}^* = \arg\min_{\mathbb{W}_{L+1}} J_{\text{MSE}}\left(\mathbb{W}_{L+1}; \mathcal{D}\right). \tag{2.10}$$

With the resulting optimal weights $\mathbb{W}_{L+1}^*$ from (2.10) the neural network function (2.6) approximates (2.1a) as:

$$\hat{\boldsymbol{x}}_{k+1} = NN(\boldsymbol{x}_k, \boldsymbol{u}_k; \mathbb{W}_{L+1}^*). \tag{2.11}$$

The identified model can then be used to formulate the optimal control problem in (2.2).

2.2.2 Output-feedback

For most practical applications Assumption 2.1 does not hold, implying that the states of the system are not directly measured. Instead, the observed data of system (2.1) is composed of:

$$\boldsymbol{y}_{[1,m]} = \left[\boldsymbol{y}_1^\top, \ldots, \boldsymbol{y}_{m+1}^\top\right]^\top,\ \boldsymbol{u}_{[1,m]} = \left[\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_m^\top\right]^\top. \tag{2.12}$$

To identify the system described in Subsection 2.2.1, the unknown states can be identified from the available measurements. For the nonlinear system in (2.1) a *moving horizon estimator* (MHE) [8] is a suitable choice and can be employed if the system is *observable* [8, Def. 4.13].

Definition 2.1 (Observability)**.** *System* (2.1) *is observable if there exist* $l \in \mathbb{I}$, $\gamma_u(\cdot) \in \mathcal{K}$ *and* $\gamma_y(\cdot) \in \mathcal{K}$ *[8, Def. B.3] such that for every two initial states* $\boldsymbol{x}_0^1$ *and* $\boldsymbol{x}_0^2$ *and every two input sequences* $\boldsymbol{u}_{[0,l-1]}^1$ *and* $\boldsymbol{u}_{[0,l-1]}^2$ *and the resulting output sequences* $\boldsymbol{y}_{[0,l-1]}^1$ *and* $\boldsymbol{y}_{[0,l-1]}^2$*:*

$$|\boldsymbol{x}_0^1 - \boldsymbol{x}_0^2| \leq \gamma_u\left(\max_{i \in \mathbb{I}_{[0,l-1]}} |\boldsymbol{u}_i^1 - \boldsymbol{u}_i^2|\right) + \gamma_y\left(\max_{i \in \mathbb{I}_{[0,l-1]}} |\boldsymbol{y}_i^1 - \boldsymbol{y}_i^2|\right). \tag{2.13}$$

As an intuitive consequence of Definition 2.1, an observable system that is subject to identical input sequences of length l and yields identical output sequences must have identical initial states. The sequence length l is of particular importance and formalized in the following definition.

Definition 2.2 (Observability index / system lag)**.** *The smallest integer* l *for which the system* (2.1) *is observable, according to Definition 2.1, is called the observability index or system lag.*

For an observable system (2.1), according to Definition 2.1, an MHE with $t \geq l$ can be formulated as shown in [8]. The MHE is a parametric optimization problem which is stated as the function:

$$\boldsymbol{x}_{k-1} = \kappa_{\text{MHE}}(\boldsymbol{y}_{[k-t,k-1]}, \boldsymbol{u}_{[k-t,k-1]}). \tag{2.14}$$

As a consequence of [8, Theo. 4.18], the MHE (2.14) yields the exact estimate $\boldsymbol{x}_{k-1}$ for the deterministic nonlinear dynamic system (2.1). The MHE (2.14) does not depend on a previous estimate of the state $\boldsymbol{x}_{k-t}$ if the system is observable and the finite sequence length t is chosen such that $t \geq l$.

Unfortunately, the design of the MHE (2.14) requires knowledge of the dynamic system (2.1) which is sought-after in the first place. However, it is sufficient to assume the existence of the MHE (2.14). This allows us to substitute (2.14) into the nonlinear system (2.1a), which yields:

$$\boldsymbol{x}_k = f(\kappa_{\text{MHE}}(\boldsymbol{y}_{[k-t,k-1]}, \boldsymbol{u}_{[k-t,k-1]}), \boldsymbol{u}_{k-1}). \tag{2.15}$$

Further substituting (2.15) into (2.1b) yields:

$$\boldsymbol{y}_k = h(f(\kappa_{\text{MHE}}(\boldsymbol{y}_{[k-t,k-1]}, \boldsymbol{u}_{[k-t,k-1]}), \boldsymbol{u}_{k-1}), \boldsymbol{u}_k), \tag{2.16}$$

where the function composition is then summarized as:

$$\boldsymbol{y}_k = f_{\text{NARX}}(\boldsymbol{y}_{[k-t,k-1]}, \boldsymbol{u}_{[k-t,k]}). \tag{2.17}$$

The newly introduced function $f_{\text{NARX}}(\cdot)$ is a nonlinear autoregressive exogenous (NARX) model [54] that implicitly combines the state estimation (2.14), the system dynamics (2.1a) and the output function (2.1b). Function (2.17) relates only known quantities, that is, the sequences of measured outputs and inputs, and can be identified from the available data. To this end, the dataset $\mathcal{D} = \left\{\boldsymbol{v}^{(i)}, \boldsymbol{t}^{(i)}\right\}_{i=1}^{m}$ is introduced, where $\boldsymbol{v}^{(i)\top} = [\boldsymbol{y}_{[i-t,i-1]}^{\top},\ \boldsymbol{u}_{[i-t,i]}^{\top}]$ and $\boldsymbol{t}^{(i)} = \boldsymbol{y}_i$. A neural network system model is then identified as shown in the previous subsection. The optimal weights $\mathbb{W}_{L+1}$ are obtained from (2.10) and the identified model approximates (2.17) as:

$$\hat{\boldsymbol{y}}_k = \hat{f}_{\text{NARX}} = NN(\boldsymbol{y}_{[k-t,k-1]}, \boldsymbol{u}_{[k-t,k]}; \mathbb{W}_{L+1}^{*}). \tag{2.18}$$

Finally, it is possible to recover a dynamic model of the system in the form of (2.1). To this end, the following state is introduced, as shown in [54][1]:

$$\tilde{\boldsymbol{x}}_k^{\top} = \begin{bmatrix} \boldsymbol{y}_{[k-t,k-1]}^{\top} & \boldsymbol{u}_{[k-t,k-1]}^{\top} \end{bmatrix}. \tag{2.19}$$

The system dynamics (2.1a), with the state (2.19), are then formulated as:

$$\begin{aligned} \tilde{\boldsymbol{x}}_{k+1} &= \tilde{f}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) \\ &= \begin{bmatrix} \boldsymbol{y}_{k-t+1}^{\top} & \dots & f_{\text{NARX}}(\boldsymbol{y}_{[k-t,k-1]}, \boldsymbol{u}_{[k-t,k]})^{\top} & \boldsymbol{u}_{k-t+1}^{\top} & \dots & \boldsymbol{u}_k^{\top} \end{bmatrix}^{\top}. \end{aligned} \tag{2.20}$$

[1] Equation (2.19) defines the state at time k because with sequences $\boldsymbol{u}_{k-1}$ and $\boldsymbol{x}_{k-1}$ until time $k-1$, $\boldsymbol{x}_k$ is uniquely determined with (2.1a).

To obtain the same input-output behavior as the unknown system (2.1) the measurement equation $h(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) = \boldsymbol{y}_{k-1}$ can be introduced. Alternatively, it is possible to satisfy Assumption 2.1 by introducing $h(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) = \tilde{\boldsymbol{x}}_k$. Finally, the identified model from a system with output-feedback allows the formulation of an optimal control problem (2.2) which yields the desired feedback controller in the form of:

$$\boldsymbol{u}_k = \kappa_{\text{MPC}}(\tilde{\boldsymbol{x}}_k) = \kappa_{\text{MPC}}(\boldsymbol{y}_{[k-t,k-1]}, \boldsymbol{u}_{[k-t,k]}).$$

2.3 Implementation and case studies

Two case studies combining nonlinear system identification with neural networks and model predictive control are presented in the following. The first case study, published in our work [12], motivates this combination by demonstrating that a data-based MPC controller can achieve near-identical performance to an MPC controller with exact system model. In the context of this investigation, we also propose an excitation strategy to sample data from the original system.

In the second case study, published in [40], we investigate a data-based predictive controller for a water distribution network. In contrast to the first case study, the available system model is not control-oriented and does not permit the formulation of a model predictive controller. The case study thus represents a common challenge for model-based control. Furthermore, it shows the potential of data-based MPC to lead to significant energy savings and performance improvements in a relevant real-world example.

2.3.1 Implementation

The investigated MPC controllers are implemented using the open-source software do-mpc, which is introduced as another main contribution in [12]. At its core, do-mpc offers a high-level interface to the CasADi [55] optimization library and facilitates the formulation of optimal control problems (2.2). Through CasADi, it leverages the interior-point optimization algorithm IPOPT [56] to efficiently solve the resulting optimization problems. The core features of do-mpc are the formulation of robust multi-stage [57], nonlinear and economic MPC problems, moving horizon estimation problems, and a simulation environment for rapid control prototyping [58]. Additionally, do-mpc integrates an OPC UA interface [59] for the communication with physical systems and facilitates the integration of neural network systems by supporting the Open Neural Network Exchange format (ONNX)[2]. In the following case study, the neural network system model is trained with Tensorflow [60], converted to the ONNX format and then imported into do-mpc to develop the controller under investigation.

2.3.2 Continuously stirred tank reactor

We investigate the identification of a continuously stirred tank reactor (CSTR) with a neural network system model in [12].

[2] `htttps://onnx.ai`

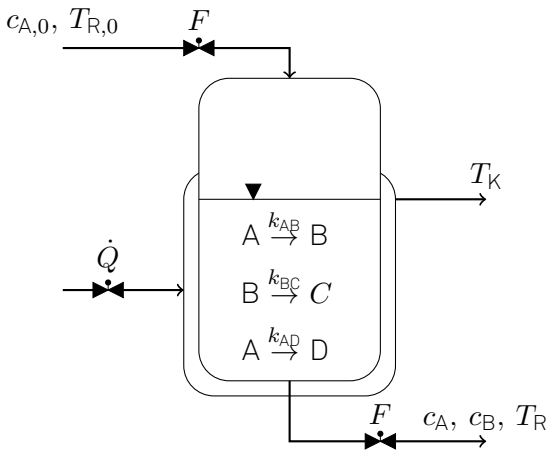

Figure 2.1: Continuously stirred tank reactor (CSTR) producing product B from educt A. The inflow $F = \dot{V}/V_R$ and heat flow $\dot{Q}$ removed from the coolant are used to control the system [12, Figure 2].

The CSTR model was previously introduced in [61] and is depicted in Figure 2.1. The full description of the dynamic model is provided in Appendix B.1. The CSTR has $n_x = 4$ states, that is, the concentration of the educt c_A, the concentration of the product c_B, the reactor temperature T_R and the temperature of the coolant T_K. In addition to the main reaction, two additional reactions form the byproducts C and D. The system is controlled with $\boldsymbol{u} = [F,\ \dot{Q}]^\top \in \mathbb{R}^2$, that is, the normalized inlet flow $F = \dot{V}/V_R$ and the heat removed by the coolant $\dot{Q}$. The control task in this case study is to regulate the concentration of the product c_B to a desired set-point c_B^{set}, while complying with the operational constraints of the system, shown in Appendix B.1.

In [12], we consider that the system exhibits state-feedback and Assumption 2.1 is satisfied. An important focus of the investigation is the data-generation process. To this end, we suggest to sample m closed-loop trajectories of length N of the system, where each sample $i \in \mathbb{I}_m$ is obtained with a random initial state $\boldsymbol{x}_0^{(i)}$ and an input strategy $\kappa^{(i)}(\boldsymbol{x})$. The input strategy is the superposition of inputs from an existing set-point tracking controller $\kappa_p(\boldsymbol{x}, \boldsymbol{x}^{\text{set}})$, and a random input $\mathbf{e}_u \in \mathbb{R}^{n_u}$:

$$\kappa^{(i)}(\boldsymbol{x}) = \left(1 - \alpha^{(i)}\right) \kappa_p(\boldsymbol{x}, \boldsymbol{x}^{\text{set}}) + \alpha^{(i)} \mathbf{e}_u, \tag{2.21}$$

where $\alpha \in [0, 1]$ is a weighting factor. By combining the set-point tracking with random inputs, the data contains plausible input sequences around the desired set-point as well as more exploratory sequences. The proposed software do-mpc facilitates the data-generation process by providing a reproducible sampling framework.

With the collected data, a neural network system model in the form of (2.11) is trained and employed in the MPC formulation (2.2). Finally, the data-based MPC controller is evaluated in comparison to an MPC controller with exact system model. Apart from the system model, both MPC formulations are identical. In Figure 2.2, the results of the comparison are shown in terms of the closed-loop cost and maximum constraint violation, both depicted in relationship to the set-point c_b^{set}. This shows that the MPC controller with data-based model performs nearly-identical to the variant

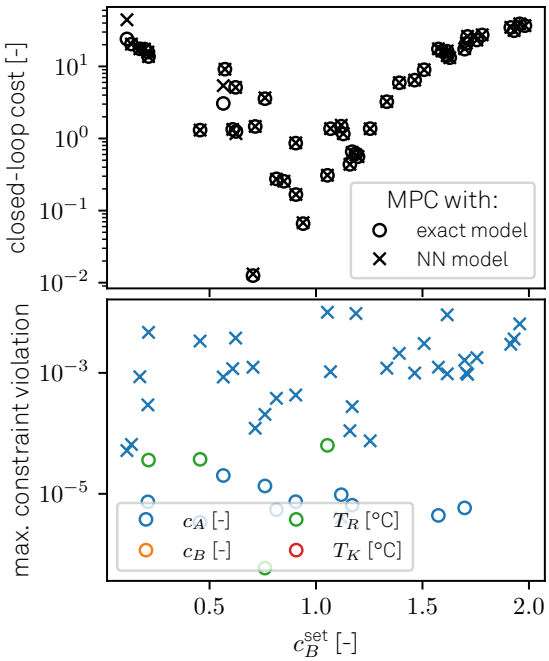

Figure 2.2: Closed-loop control performance for the CSTR in Figure 2.1. Comparison of MPC with exact system model and with identified NN system model, considering the closed-loop cost and maximum constraint violation over $N = 50$ steps for different set-points c_B^{set}. [12, Figure 6].

with exact system model. The closed-loop cost is nearly indistinguishable with minor exceptions in two out of the $p = 50$ investigated cases. Minor constraint violations, which are possible due to the formulation of soft-constraints, are observed with both controllers and slightly more pronounced for the variant with neural network system model.

2.3.3 Water distribution network

As the second case study for MPC with identified neural network system model, we investigate the important task of controlling a water distribution network in [40]. Water distribution networks are large-scale interconnected systems that must be operated reliably, while complying with operational constraints. Water resources have the highest energy intensity in the urban metabolism of cities [62], with a significant share attributed to the water distribution [63]. The energy optimal control of water distribution networks with MPC is therefore an important step towards the green transformation of cities, and has been investigated in previous works [64]–[66].

As a concrete example, we investigate the C-Town water distribution network [67], with $n_{\text{junc}} = 378$ junctions, seven tanks, five pump stations and four valves. A high-fidelity model of the system is available in the form:

$$\boldsymbol{b}_{k+1} = f_{\text{WDN}}(\boldsymbol{b}_k, \boldsymbol{v}_k, \boldsymbol{q}_k, \boldsymbol{d}_k), \tag{2.22}$$

$$\boldsymbol{p}_k,\ \boldsymbol{b}_k = h_{\text{WDN}}(\boldsymbol{b}_k, \boldsymbol{v}_k, \boldsymbol{q}_k, \boldsymbol{d}_k), \tag{2.23}$$

with tank level $\boldsymbol{b} \in \mathbb{R}^7$, valve setting $\boldsymbol{v} \in \mathbb{R}^4$, pump speed $\boldsymbol{q} \in \mathbb{R}^5$, junction demand $\boldsymbol{d} \in \mathbb{R}^{378}$ and junction pressure $\boldsymbol{p} \in \mathbb{R}^{378}$. The model in (2.22) is in the form of (2.1), where the tank levels are states, valve settings and pump speed are inputs, and the junction pressures and tank levels are measured outputs. As a straightforward extension of (2.1), we have the junction demand as additional uncontrolled parameters of the nonlinear model.

The existence of high-fidelity models of water distribution networks is common in practice, owing to popular tools such as the EPANET [68] software application, developed by the United States Environmental Protection Agency. Unfortunately, EPANET models are not tailored towards optimization due to discrete switching behavior and the lack of gradient information. The problem is further intensified by the size of real-world networks, rendering the application of MPC with these models infeasible. To mitigate these problems, we propose in [40] the utilization of neural network system identification to obtain a surrogate model from simulation data, that is:

$$[\boldsymbol{b}_{k+1},\ \boldsymbol{p}_k]^\top = NN(\boldsymbol{b}_k, \boldsymbol{v}_k, \boldsymbol{q}_k, \boldsymbol{d}_k). \tag{2.24}$$

To reduce the size of the model, we combine the neural network system identification with an additional model reduction approach based on clustering. This approach reduces the number of nodes in the water distribution network and, successively, yields a more compact neural network system model. Similar node reduction techniques are also applied in related works and typically based on heuristics [69] or system knowledge [70]. As a main novelty of our work [40], we employ a data-driven hierarchical clustering approach with connectivity constraints [71], which is implemented using *Scikit-learn* [72].

The neural network system model with clustered nodes is suitable to formulate a tractable optimal control problem in the form of (2.2). We propose an economic control objective, that is, we minimize the energy consumption of the pumps while satisfying water pressure requirements at all nodes. Furthermore, we anticipate imperfections of the data-based model which may lead to unsatisfied demand at the junctions. To avoid this scenario, we tighten the constraints of the tank levels with a lower bound $b_{\text{min}} \geq 0$ and formulate both them and the junction pressure constraints as soft constraints [8, Sec. 1.2.5]. The full optimal control problem is stated in [40], and yields the control law:

$$[\boldsymbol{v}_k,\ \boldsymbol{q}_k]^\top = \kappa_{\text{WDN}}(\boldsymbol{b}_k, \boldsymbol{d}_{[k,k+N-1]}). \tag{2.25}$$

Importantly, we require a prediction of the future demand of the network to evaluate (2.25). In practice, this prediction may be inferred from past data and will be subject to uncertainty.

The resulting controller (2.25) is investigated in a closed-loop simulation using the original model implemented in EPANET. The focus of the investigations is to evaluate the energy savings in comparison to the default rule-based control approach, the satisfaction of node pressure demand, and the robustness to uncertain demand predictions. We present an excerpt from the results in [40] in Table 2.1. Compared to the default rule-based control approach, the proposed MPC controller leads to energy savings as high as 10.39 % if no tightening of constraints is used ($b_{\text{min}} = 0$). Constraint violations can be reduced by tightening the lower bound of the tank level as shown in Table 2.1, thereby achieving a trade-off with energy savings. We further

Table 2.1: Control performance for the water distribution network: Comparison of energy consumption and constraint violations between MPC controllers with different lower bounds for the tank levels and a rule-based reference controller. Violations of satisfying the water demand at the junctions are expressed as a percentage of the operating time and the average violation during these instances [40, Table 2].

	MPC for different b_{min}			**Ref.**
	1.5	0.5	0.0	
energy [MWh]	106.49	105.56	99.97	111.57
energy saved [%]	4.55	5.39	10.39	-
cons. viol. [%]	0.05	0.13	0.51	0.0
avg. cons. viol. [bar]	0.21	0.50	0.51	0.00

show in [40] that the proposed controller is robust to significant uncertainty of the demand predictions at the nodes. This is found to be an important contribution of the proposed clustering approach. Due to the aggregation, fluctuations at individual nodes of the network are averaged and result in predictable behavior.

The results motivate further work with the objective of using economic MPC with data-based system model for real-world water distribution networks. The availability of data for these systems and the potential energy savings, both as a percentage and in absolute numbers, make this an attractive application of data-based MPC.

2.4 Conclusions

In this chapter, nonlinear system identification with neural networks is proposed to obtain a nonlinear and economic MPC controller in the absence of a control-oriented model. A prerequisite for system identification with neural networks, and other identification methods, is the existence of an informative set of training samples. Ideally, the data contains sequences that explore the state-space and sequences that approximate the optimal control behavior. Clearly, this is difficult to achieve in practice and makes data-generation an important element of nonlinear system identification. However, with sufficiently informative data, as demonstrated in Subsection 2.3.2, an MPC controller with neural network system model can achieve near-identical performance to a variant with exact system model. Neural network system identification for MPC is further demonstrated for the control of a water distribution network. It is found that the proposed controller achieves significant energy savings while satisfying water demand in the network. The study also demonstrates that neural network system identification can be combined with data pre-processing methods, specifically a clustering approach, to reduce the complexity of the neural network architecture. Similar combinations, for example using principal component analysis [73, Sec. 12.1], are conceivable for future work.

Although neural networks are a potent model class for capturing complex nonlinear system behavior, they also encounter specific challenges that must be considered for their successful application. Most importantly, traditional neural networks are

prone to *silent failures* [26], that is, they may yield arbitrarily poor predictions without indication of their confidence. Typically, poor predictions are the result of extrapolation, which often occurs if the training data is not sufficiently explorative. Furthermore, economic MPC may exploit poor predictions of the underlying neural network system model [47]. This behavior occurs if the model predicts implausible system behavior, for example, energy generation instead of energy consumption.

The challenge of silent failures is mitigated in this chapter with a focus on data-generation and careful model validation. While these are important steps, they are not sufficient to guarantee accurate model behavior under all circumstances. As a consequence, neural networks are revisited in Chapter 5, where an approach to quantify the uncertainty of their prediction is presented. In combination with stochastic model predictive control, introduced in Chapter 4, this is a promising solution to tackle the discussed challenges.

Chapter 3

Linear multi-step identification

A fundamental concept of MPC is to predict and optimize the behavior of the system for a finite future horizon. Traditionally, this prediction is obtained by recursively evaluating a state-space model (2.1). However, there is a recent trend in system identification to directly identify multi-step prediction models [27]–[30]. A multi-step model condenses the recursive evaluation of the state-space model into a single function. Multi-step identification then seeks to identify this function directly from data. In comparison to state-space models, a multi-step model requires significantly more parameters to describe the same system. At first glance, this appears to be a disadvantage. However, recent work has shown that identified multi-step models achieve a better prediction accuracy than identified state-space models [29], [30]. The reason for this advantage will be further explored in Chapter 4.

Most recent works [27]–[30], and this thesis, focus on linear multi-step identification. Linear systems have the attractive property that their recursive evaluation can be condensed into a single linear function. Consequently, the identification task boils down to a tractable linear regression problem and the linear multi-step model can be efficiently evaluated, enabling fast MPC implementations. Nonlinear multi-step identification remains an attractive goal and is discussed in Chapters 5 and 6.

In addition to the introduction and motivation of linear multi-step identification for MPC, its relationship to *data-enabled predictive control* (DeePC) [31]–[37] is explored in this chapter.

This chapter summarizes the main findings of the publication [41]. The full publication is provided in Appendix C.3.

3.1 Multi-step model

A linear dynamic system is introduced as a special case of the general nonlinear dynamic system (2.1):

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{3.1a}$$

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k, \tag{3.1b}$$

with the system matrices $\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{B} \in \mathbb{R}^{n_x \times n_u}$, $\boldsymbol{C} \in \mathbb{R}^{n_y \times n_x}$ and $\boldsymbol{D} \in \mathbb{R}^{n_y \times n_u}$. A multi-step prediction can be obtained by recursively evaluating the dynamic

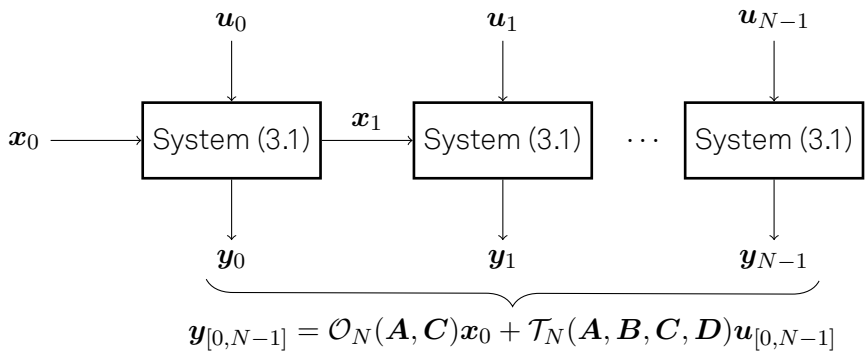

Figure 3.1: Computing the multi-step prediction from $\boldsymbol{x}_0$ by recursively evaluating the dynamic system.

system (3.1), as shown in Figure 3.1. For linear systems, this recursive evaluation can be condensed into a single linear function. To predict the sequence of length N, that is, from $k = 0$ to $k = N - 1$, the recursive evaluation yields:

$$\begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{N-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \vdots \\ \boldsymbol{CA}^{N-1} \end{bmatrix} \boldsymbol{x}_0 + \begin{bmatrix} \boldsymbol{D} & \dots & \dots & \boldsymbol{0} \\ \boldsymbol{CB} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{CA}^{N-2}\boldsymbol{B} & \dots & \boldsymbol{CB} & \boldsymbol{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{bmatrix}. \tag{3.2}$$

For conciseness, the *observability matrix* $\mathcal{O}_N(\boldsymbol{A}, \boldsymbol{C})$ and $\mathcal{T}_N(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ are introduced, of which block i, j is obtained as follows:

$$[\mathcal{O}_N(\boldsymbol{A}, \boldsymbol{C})]_{i,1} = \boldsymbol{CA}^{i-1} \qquad \forall i \in \mathbb{I}_{[1,N]}, \tag{3.3a}$$

$$[\mathcal{T}_N(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})]_{i,j} = \begin{cases} \boldsymbol{CA}^{i-j-1}\boldsymbol{B}, & \text{if } i > j \\ \boldsymbol{D}, & \text{if } i = j \\ \boldsymbol{0}, & \text{otherwise} \end{cases} \qquad \forall i, j \in \mathbb{I}_{[1,N]}. \tag{3.3b}$$

With (3.3), the *multi-step model* in (3.2) can then be stated compactly as:

$$\boldsymbol{y}_{[0,N-1]} = \mathcal{O}_N(\boldsymbol{A}, \boldsymbol{C})\boldsymbol{x}_0 + \mathcal{T}_N(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})\boldsymbol{u}_{[0,N-1]}. \tag{3.4}$$

The resulting expression (3.4) is also shown in Figure 3.1.

3.1.1 Output-feedback

The multi-step model in (3.4) can be evaluated as a function of the initial state $\boldsymbol{x}_0$ and the sequence of inputs $\boldsymbol{u}_{[0,N-1]}$. However, in most applications, the states of the system are not directly observed. To address this case, a similar approach as presented in Subsection 2.2.2 is employed. Again, the system is required to be observable, as introduced in Definition 2.1. For the linear system (3.1), observability can be shown with the following lemma [8, Sec. 1.4.5].

Lemma 3.1 (Observability)**.** *System* (3.1) *is observable if there exists a finite* $l \in \mathbb{I}$ *such that the observability matrix* $\mathcal{O}_l(\boldsymbol{A}, \boldsymbol{C})$ *in* (3.3a) *has full rank.*

To formulate the multi-step model with output-feedback the following assumption is required.

Assumption 3.1. *There exists* t *greater than or equal to the system lag* $l(\boldsymbol{A}, \boldsymbol{C})$*, according to Definition 2.2, that is,* $t \geq l(\boldsymbol{A}, \boldsymbol{C})$*.*

This allows to introduce the following lemma.

Lemma 3.2. *With past sequence length* t*, satisfying Assumption 3.1, the future sequence of outputs of length* N *can be computed as a linear function of* $\boldsymbol{y}_{[0,t-1]}$ *and* $\boldsymbol{u}_{[0,L-1]}$*, where* $L = N + t$*:*

$$\boldsymbol{y}_{[t,L-1]} = \boldsymbol{W} \begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,t-1]} \\ \boldsymbol{u}_{[t,L-1]} \end{bmatrix}. \tag{3.5}$$

The parameter matrix $\boldsymbol{W}$ *is obtained from the system matrices in* (3.1)*, as shown in* (A.6)*.*

Proof. The proof is shown in Appendix A.1. □

Furthermore, an observable system with output-feedback can be reformulated as an equivalent system representation with state-feedback using (3.5). To this end, the new state vector:

$$\tilde{\boldsymbol{x}}_t^\top = \begin{bmatrix} \boldsymbol{y}_{[0,t-1]}^\top & \boldsymbol{u}_{[0,t-1]}^\top \end{bmatrix}, \tag{3.6}$$

is introduced. As shown in Appendix A.2, this allows to obtain the equivalent dynamic system:

$$\tilde{\boldsymbol{x}}_{t+1} = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}_t + \tilde{\boldsymbol{B}}\boldsymbol{u}_t, \tag{3.7}$$

which exhibits state-feedback.

3.1.2 Multi-step identification

The multi-step model in the form of (3.5) can be derived with known system matrices from (3.1). More importantly, however, it can be directly identified from input-output data of the dynamic system. To this end, m sequences of length $L = N+t$ are collected and arranged in the matrices:

$$\boldsymbol{U}_L = \begin{bmatrix} \boldsymbol{u}_{[0,L-1]}^{(1)} & \dots & \boldsymbol{u}_{[0,L-1]}^{(m)} \end{bmatrix}, \quad \boldsymbol{Y}_L = \begin{bmatrix} \boldsymbol{y}_{[0,L-1]}^{(1)} & \dots & \boldsymbol{u}_{[0,L-1]}^{(m)} \end{bmatrix}. \tag{3.8}$$

The matrices (3.8) are further partitioned into two blocks, holding the m sequences of length t and N, respectively:

$$\boldsymbol{U}_L^\top = \begin{bmatrix} \boldsymbol{U}_t^\top & \boldsymbol{U}_N^\top \end{bmatrix}, \quad \boldsymbol{Y}_L^\top = \begin{bmatrix} \boldsymbol{Y}_t^\top & \boldsymbol{Y}_N^\top \end{bmatrix}, \tag{3.9}$$

and have the dimensions $\boldsymbol{U}_t \in \mathbb{R}^{tn_u \times m}$, $\boldsymbol{U}_N \in \mathbb{R}^{Nn_u \times m}$, $\boldsymbol{Y}_t \in \mathbb{R}^{tn_y \times m}$ and $\boldsymbol{Y}_N \in \mathbb{R}^{Nn_y \times m}$. To estimate the parameters of the linear model (3.5), the data of the independent variables is stacked in the matrix [1]:

$$\boldsymbol{V}^\top = \begin{bmatrix} \boldsymbol{Y}_t^\top & \boldsymbol{U}_t^\top & \boldsymbol{U}_L^\top \end{bmatrix}. \tag{3.10}$$

As a final preliminary for the identification task, the following assumption is introduced.

Assumption 3.2. *We have $m \geq Ln_u + n_x$ samples from the dynamic system and the matrix $\boldsymbol{V}$ in* (3.10) *has rank*$(\boldsymbol{V}) \geq (Ln_u + n_x)$.

Remark 3.1. If the data in matrix (3.10) satisfies Assumption 3.2 it is implied that the inputs are *persistently exciting*. The definition of *persistance of excitation* [38] requires $\boldsymbol{U}_L$ to have full rank [31] and is commonly employed in behavioral systems theory. Data-enabled predictive control, which is introduced in Section 3.3, is based on the behavioral systems perspective.

If Assumptions 3.1 and 3.2 hold, the parameters of matrix $\boldsymbol{W}$ in (3.5) can be identified with the least-squares problem [41, Lemma 2]:

$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \|\boldsymbol{W}\boldsymbol{V} - \boldsymbol{Y}_N\|_F^2, \tag{3.11}$$

where $\|\cdot\|_F$ denotes the Frobenius norm [74, Def. 4.63]. Problem (3.11) has the explicit solution:

$$\boldsymbol{W}^* = \boldsymbol{Y}_N \boldsymbol{V}^\dagger, \tag{3.12}$$

where $\boldsymbol{V}^\dagger$ denotes the Moore-Penrose pseudo-inverse of $\boldsymbol{V}$ [73, Prop. 7.66]. In the deterministic setting, that is, if the data is unaffected by noise, the identified parameters are equivalent to the true multi-step parameters, that is $\boldsymbol{W}^* = \boldsymbol{W}$ [41, Lemma 2].

3.2 Multi-step model predictive control

With the identified parameters from (3.12) for the multi-step model in (3.5), the multi-step model predictive control problem (MS-MPC) [41] is introduced:

$$\begin{aligned} \min_{\boldsymbol{u}_{[t,L-1]},\, \boldsymbol{y}_{[t,L-1]}} \quad & \|\boldsymbol{y}_{[t,L-1]}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}_{[t,L-1]}\|_{\boldsymbol{R}}^2 \\ \text{s.t.} \quad & (3.5) \\ & \boldsymbol{u}_k \in \mathbb{U}\ \forall k \in \mathbb{I}_{[t,L-1]}, \\ & \boldsymbol{y}_k \in \mathbb{Y}\ \forall k \in \mathbb{I}_{[t,L-1]}, \end{aligned} \tag{3.13}$$

with positive definite matrices $\boldsymbol{Q} \succ 0$ and $\boldsymbol{R} \succ 0$ and convex constraint sets $\mathbb{U}$ and $\mathbb{Y}$. Problem (3.13) is a quadratic parametric optimization problem which depends on the past sequences $\boldsymbol{y}_{[0,t-1]}$ and $\boldsymbol{u}_{[0,t-1]}$. The multi-step predictive controller is obtained

[1] In [41] the matrix $\boldsymbol{V}$ is introduced as $\boldsymbol{M}$. The change is employed to harmonize the notation with [42].

by repeatedly solving (3.13) at each time step k with the past sequences $\boldsymbol{y}_{[k-t,k-1]}$ and $\boldsymbol{u}_{[k-t,k-1]}$. This yields the following control law:

$$\boldsymbol{u}_k = \kappa_{\text{MS-MPC}}\left(\boldsymbol{y}_{[k-t,k-1]}, \boldsymbol{u}_{[k-t,k-1]}\right). \tag{3.14}$$

Remark 3.2 (Subspace predictive control). The multi-step identification and MPC approach was originally introduced as *subspace predictive control* [27]. This term originates from *subspace identification* [75], which is a popular approach to identify linear state-space models from input-output data in the form of (3.8). To emphasize that we identify multi-step models (instead of state-space models) we use the term *multi-step model predictive control*. This is in contrast to our previous work [41], and recent work on data-enabled predictive control [34], where the term subspace predictive control is used to describe (3.13).

3.3 Data-enabled predictive control

The optimal control problem (3.13) can be formulated after the multi-step model (3.5) is identified from input-output data. An alternative approach is given by the recently proposed *data-enabled predictive control* (DeePC) method [31]–[34]. The method is based on a behavioral systems perspective [38] and combines the identification and control tasks into a single optimal control problem. In the following, the data-enabled predictive control problem and the required fundamental lemma of behavioral systems are introduced. Subsequently, the relationship between the presented MS-MPC (3.13) approach and DeePC is investigated.

3.3.1 Fundamental lemma of behavioral systems

The formulation of DeePC is based on the fundamental lemma of behavioral systems [38]. As shown in [41, Lemma 1], any sequence $\boldsymbol{y}_{[0,L-1]}$ and $\boldsymbol{u}_{[0,L-1]}$ is a trajectory of system (3.1) if and only if there exists a $\boldsymbol{g} \in \mathbb{R}^m$, such that:

$$\begin{bmatrix} \boldsymbol{y}_{[0,L-1]} \\ \boldsymbol{u}_{[0,L-1]} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y}_L \\ \boldsymbol{U}_L \end{bmatrix} \boldsymbol{g}, \tag{3.15}$$

where the matrices $\boldsymbol{Y}_L$ and $\boldsymbol{U}_L$ are obtained from (3.8) and satisfy Assumption 3.2. In other words, any linear combination of the columns of $\boldsymbol{Y}_L$ and $\boldsymbol{U}_L$ is a trajectory of the system (3.1). Similarly to the multi-step model (3.5), the fundamental lemma (3.15) can be used to predict future outputs $\boldsymbol{y}_{[t,L-1]}$ from the sequences $\boldsymbol{y}_{[0,t-1]}$ and $\boldsymbol{u}_{[0,L-1]}$. To this end, the following linear system of equations must be solved for $\boldsymbol{g}^*$:

$$\begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,L-1]} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y}_t \\ \boldsymbol{U}_L \end{bmatrix} \boldsymbol{g}^*.$$

The predicted future sequence $\boldsymbol{y}_{[t,L-1]}$ is then obtained as:

$$\boldsymbol{y}_{[t,L-1]} = \boldsymbol{Y}_N \boldsymbol{g}^*.$$

As a reminder, the matrices $\boldsymbol{Y}_N$ and $\boldsymbol{Y}_t$ are partitions of $\boldsymbol{Y}_L$, according to (3.9).

3.3.2 DeePC formulation

To obtain a predictive control formulation from the fundamental lemma, the authors in [31] propose the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{u}_{[t,L-1]},\, \boldsymbol{y}_{[t,L-1]},\, \boldsymbol{g}} \quad & \|\boldsymbol{y}_{[t,L-1]}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}_{[t,L-1]}\|_{\boldsymbol{R}}^2 \\ \text{s.t.} \quad & (3.15) \\ & \boldsymbol{u}_k \in \mathbb{U} \quad \forall k \in \mathbb{I}_{[t,L-1]}, \\ & \boldsymbol{y}_k \in \mathbb{Y} \quad \forall k \in \mathbb{I}_{[t,L-1]}. \end{aligned} \tag{3.16}$$

Similarly to (3.13), the problem is formulated with positive definite matrices $\boldsymbol{Q} \succ 0$ and $\boldsymbol{R} \succ 0$ and convex constraint sets $\mathbb{U}$ and $\mathbb{Y}$. The most prominent difference, in comparison to (3.13), is the inclusion of the vector $\boldsymbol{g} \in \mathbb{R}^m$ and the data matrices $\boldsymbol{Y}_L$, $\boldsymbol{U}_L$ in the optimization problem. Problem (3.16) also depends on the past sequences $\boldsymbol{y}_{[0,t]}$ and $\boldsymbol{u}_{[0,t]}$, and can be stated as the feedback control law:

$$\boldsymbol{u}_k = \kappa_{\text{DeePC}}\left(\boldsymbol{y}_{[k-t,k-1]}, \boldsymbol{u}_{[k-t,k-1]}\right). \tag{3.17}$$

The authors in [31], [34] also refer to this approach as *direct data-driven* as it does not require the identification of model parameters. In contrast, they denote the multi-step predictive control problem (3.13) as *indirect data-driven*.

3.4 Relationship of DeePC and multi-step predictive control

In this chapter, two data-based predictive control formulations are presented. First, the multi-step model predictive control formulation (3.13) and, second, the DeePC formulation (3.16). Importantly, both formulations are based on the same premise, that is, system data in the form of (3.8) is required, and both control laws are functions of finite sequences of past outputs $\boldsymbol{y}_{[k-t,k-1]}$ and inputs $\boldsymbol{u}_{[k-t,k-1]}$.

In the following, the relationship between both approaches is investigated, distinguishing between two important cases. First, the deterministic case, where the system is not subject to random disturbances. This corresponds to the formulation of the dynamic system in (3.1). Second, the non-deterministic case, where random disturbances are present.

3.4.1 Deterministic case

In the deterministic case, with the linear system in the form of (3.1), the DeePC problem (3.16) has the same optimal solution as the multi-step predictive control problem (3.13), as shown in the following theorem.

Theorem 3.1. *The matrices $\boldsymbol{U}_L$, $\boldsymbol{Y}_L$ in* (3.8) *with data from system* (3.1) *satisfy Assumptions 3.1 and 3.2. The optimal control policy $\boldsymbol{u}^*_{[t,L-1]}$ and optimal output trajectory $\boldsymbol{y}^*_{[t,L-1]}$, as obtained from the DeePC problem* (3.16)*, is equivalent to those obtained from the multi-step predictive control problem* (3.13)*.*

Proof. The proof is shown in our work [41, Theorem 1]. □

As a consequence of the equivalence between problems (3.16) and (3.13), it can be argued that the DeePC optimization problem implicitly estimates the parameters of the multi-step model (3.12). Since the underlying data matrices (3.8) are collected prior to the control application and are unchanged afterwards, there appears to be no reason for this repeated estimation in the deterministic case. Furthermore, the DeePC problem introduces a significant number of additional optimization variables in comparison to the multi-step predictive control problem [41]. This results in an increased computational cost for DeePC, as demonstrated by the simulation study presented in Section 3.5

3.4.2 Non-deterministic case

The relationship between DeePC and MS-MPC in the non-deterministic case is more intricate and requires reformulations of both problems, which are introduced in the following. First, a variant of the linear system (3.1) with non-deterministic measurement noise is introduced:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{3.18a}$$
$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k + \mathbf{e}_{y,k}. \tag{3.18b}$$

With data from system (3.18) in the form of (3.8), the DeePC formulation (3.16) must be adapted because the fundamental lemma (3.15) can generally not be satisfied anymore. In our work [41], we introduce the following non-deterministic DeePC formulation:

$$\begin{aligned}
\min_{\boldsymbol{g},\boldsymbol{u}_{[t,L-1]},\boldsymbol{y}_{[t,L-1]},\boldsymbol{\sigma}_y,\boldsymbol{\sigma}_u} \quad & \|\boldsymbol{y}_{[t,L-1]}\|^2_{\boldsymbol{Q}} + \|\boldsymbol{u}_{[t,L-1]}\|^2_{\boldsymbol{R}} \\
& + \lambda_g\|\boldsymbol{g}\|_2^2 + \lambda_y\|\boldsymbol{\sigma}_y\|_2^2 + \lambda_u\|\boldsymbol{\sigma}_u\|_2^2 \\
\text{s.t.} \quad & \begin{bmatrix} \boldsymbol{Y}_t \\ \boldsymbol{U}_t \\ \boldsymbol{U}_N \\ \boldsymbol{Y}_N \end{bmatrix} \boldsymbol{g} = \begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,t-1]} \\ \boldsymbol{u}_{[t,L-1]} \\ \boldsymbol{y}_{[t,L-1]} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\sigma}_y \\ \boldsymbol{\sigma}_u \\ 0 \\ 0 \end{bmatrix}, \\
& \boldsymbol{u}_k \in \mathbb{U} \quad \forall k \in \mathbb{I}_{[t,L-1]}, \\
& \boldsymbol{y}_k \in \mathbb{Y} \quad \forall k \in \mathbb{I}_{[t,L-1]}.
\end{aligned} \tag{3.19}$$

Notable changes to (3.16) are the introduction of slack variables $\boldsymbol{\sigma}_y \in \mathbb{R}^{tn_y}$ and $\boldsymbol{\sigma}_u \in \mathbb{R}^{tn_u}$ with their weighted ℓ_2-norms added to the cost function. In this way, the slack variables can be used for minor corrections to satisfy the fundamental

lemma (3.15). Furthermore, the ℓ_2-norm of $\boldsymbol{g}$ is added to the cost function. Comparable modifications are also employed in the original work on DeePC [31]. More recent work on DeePC [34] proposes an equivalent formulation to (3.19) with a general penalty term $h(\boldsymbol{g})$ instead of the ℓ_2-norm.

For the sake of comparison, similar adaptations are proposed for the multi-step predictive control formulation (3.13):

$$\begin{aligned} \min_{\boldsymbol{u}_{[t,L-1]},\boldsymbol{y}_{[t,L-1]},\boldsymbol{\sigma}_y,\boldsymbol{\sigma}_u} \quad & \|\boldsymbol{y}_{[t,L-1]}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}_{[t,L-1]}\|_{\boldsymbol{R}}^2 + \lambda_y \|\boldsymbol{\sigma}_y\|_2^2 + \lambda_u \|\boldsymbol{\sigma}_u\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{y}_{[t,L-1]} = \boldsymbol{W}^* \left(\begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,t-1]} \\ \boldsymbol{u}_{[t,L-1]} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\sigma}_y \\ \boldsymbol{\sigma}_u \\ 0 \end{bmatrix} \right), \\ & \boldsymbol{u}_k \in \mathbb{U} \quad \forall k \in \mathbb{I}_{[t,L-1]}, \\ & \boldsymbol{y}_k \in \mathbb{Y} \quad \forall k \in \mathbb{I}_{[t,L-1]}. \end{aligned} \tag{3.20}$$

The parameter matrix $\boldsymbol{W}^*$ is identified with (3.12) as in the deterministic case. The only difference to the deterministic formulation (3.13) is the introduction of the slack variables $\boldsymbol{\sigma}_y$ and $\boldsymbol{\sigma}_u$. However, in contrast to the DeePC formulation (3.19), these modifications are not strictly necessary, and are only employed to obtain a fair comparison between the two formulations.

As the second main contribution of [41], we find that the DeePC formulation (3.19) and the multi-step predictive control formulation (3.20) yield the same optimal solution under certain conditions.

Theorem 3.2. *We have $\boldsymbol{U}_L$, $\boldsymbol{Y}_L$ in the form of* (3.8) *from system* (3.18)*, t satisfying Assumption 3.1, and $m = Ln_u + n_x$. Let $\boldsymbol{V}$ in* (3.10) *and $\boldsymbol{Y}_N$ from* (3.9) *have full rank. The DeePC problem* (3.19)*, without inequality constraints, and the multi-step predictive control problem* (3.20)*, without inequality constraint, yield the same optimal solution $\boldsymbol{u}^*_{[t,L-1]}$, if $\lambda_g = 0$.*

Proof. The proof is shown in our work [41, Theorem 2]. □

Clearly, the equivalence established in Theorem 3.2 holds only in specific cases (i.e. $m = Ln_u + n_x$ and $\lambda_g = 0$) and is shown for the unconstrained formulation of DeePC (3.19) and multi-step predictive control (3.20). However, the result still shows that DeePC and multi-step model predictive control are closely related, even in their non-deterministic formulation. The main reason for deviations between the DeePC and MS-MPC solution is due to the regularization of the optimization variable g in DeePC with $\lambda_g > 0$. We show in [41] that this regularization is necessary to obtain a well-posed optimization problem if $m > Ln_u + n_x$.

The relationship between the two formulations has been further explored in recent work [35]. With suitably selected regularization term $h(\boldsymbol{g})$ and weighting λ_g in the cost function of (3.19), the same optimal solution as the multi-step predictive control problem (3.20) can be obtained.

3.5 Simulation study

The relationship between DeePC and the multi-step predictive controller is further explored in a simulation study. The investigated system in the form of (3.1) is a triple-mass-spring system with rotating discs. The disc angles are the measured outputs, and two motors, attached to the outermost discs, are used to control the system. The full system description is provided in Appendix B.2. Considering the system matrices, Lemma 3.1 shows that the system is observable and the system lag, according to Definition 2.2, is $l = 2$.

A past horizon of $t = 4$ is determined to satisfy Assumption 3.1 and the prediction horizon is chosen as $N = 40$. To satisfy Assumption 3.2, a minimum of $m > Ln_u + n_x = 96$ sequences must be collected and arranged in in the data matrices (3.8). In particular, data is collected from independent experiments with random initial states. All investigated controllers are formulated with a regulation objective and stage cost:

$$l(\boldsymbol{y}, \boldsymbol{u}) = \|\boldsymbol{y}\|_2^2 + 0.1\|\boldsymbol{u}\|_2^2,$$

and are implemented using CasADi [55] with IPOPT [56].

3.5.1 Deterministic case

In the first investigation, the described system is simulated without measurement noise and $m = 150$ sequences are captured. The focus of the investigation is to compare the deterministic DeePC algorithm (3.16) with the MS-MPC controller (3.13). In accordance with Theorem 3.2, both methods yield identical control performance in this investigation. The cumulated cost:

$$c = \sum_{k=1}^{N_{\text{sim}}} l(\boldsymbol{y}_k, \boldsymbol{u}_k), \tag{3.21}$$

differs between both controller variants on the order of 10^{-11} and is equal to 5.617. This value is considered as the baseline performance for the non-deterministic case.

3.5.2 Non-deterministic system

For the second investigation, the linear system (3.18) is subject to non-deterministic measurement noise. It is assumed that the noise follows the normal distribution $\mathbf{e}_{y,k} \sim \mathcal{N}(0, 10^{-4}\boldsymbol{I})$. The non-deterministic DeePC (3.19) and MS-MPC (3.20) controller are formulated with tuning parameters $\lambda_y = 10^4$ and $\lambda_u = 10^4$.

A main objective of the investigation in [41] is to compare the closed-loop performance of both approaches and study the effect of the number of recorded sequences m on the cost (3.21) and computation time. The results of this investigation are displayed in Table 3.1 for DeePC (3.19) with $\lambda_g = 1$ and MS-MPC (3.20). All results in Table 3.1 are computed as averages over ten simulation experiments, with independently sampled data matrices and measurement noise. The results show that MS-MPC outperforms DeePC in the investigated scenario in terms of closed-loop cost and computation time. While the advantage of MS-MPC with respect to the cost is minor,

Table 3.1: Comparison of cost and computation time between the non-deterministic formulation of DeePC (3.19) and MS-MPC (3.20) for the triple mass spring system [41, Table 1].

		Number of samples m			**Ref.**
		100	150	200	
DeePC	cost (3.21)	5.924	5.674	5.662	5.617
$(\lambda_g = 1)$	comp. time [ms]	25.272	39.419	59.291	-
MS-MPC	cost (3.21)	5.810	5.64	5.632	5.617
	comp. time [ms]	17.30	16.25	16.13	-

the difference in computation time is significant. As expected, the computational complexity of DeePC increases with the number of sequences which can render the method infeasible for complex systems.

3.6 Conclusions

In this chapter, multi-step identification for model predictive control (MS-MPC) and the related data-enabled predictive control (DeePC) are introduced. Both methods yield an MPC controller from input-output data of a linear system. The main difference between both approaches is that DeePC combines the sequential multi-step identification and control tasks in a single optimization problem.

As a major contribution, the equivalence between both approaches is established for a deterministic linear system. Furthermore, it is shown that both approaches are equivalent for linear systems with non-deterministic measurement noise if specific conditions are met. The equivalence and close relationship is further illustrated in a simulation study. This study also reveals that DeePC has a significantly increased computational cost, in comparison to MS-MPC. This is the result of introducing additional optimization variables, depending on the number of samples in the data matrices. In the presented study, the additional complexity of DeePC does not yield an improvement in control performance over MS-MPC.

DeePC has potential advantages, however, that remain to be investigated in future work. In particular, it permits to update the data matrices online to obtain an adaptive controller [34], [36]. This extension does not impact the computational cost of DeePC but would require repeating the parameter estimation for the MS-MPC controller at each time-step. Additionally, the DeePC problem is typically formulated with a general regularization term $h(\boldsymbol{g})$, to achieve different de-noising and robustification effects [34]. This may be advantageous in practice, but also introduces the challenge of choosing appropriate values for the tuning parameters λ_g [34]. It is concluded that the proposed multi-step predictive control approach in (3.13) is preferable for many applications.

In the next chapter, the investigation of multi-step models is extended to the probabilistic setting. This also allows to analyze the advantages of multi-step identification over the traditional state-space identification.

Chapter 4

Probabilistic multi-step identification for stochastic MPC

Multi-step models predict finite sequences of a dynamic system and can be readily employed in a model predictive control formulation. Linear multi-step models can be efficiently identified from data of the investigated system and thus facilitate the application of MPC. However, it remains a challenge that the identified multi-step model predicts a deterministic future behavior, whereas most real-world systems are subject to uncertainty. Applying an MPC controller with identified multi-step model to an uncertain system may, therefore, lead to unsafe behavior and insufficient performance. A promising approach to control uncertain systems is stochastic MPC (SMPC) [13], [14]. An SMPC controller can be formulated when the uncertainty in the model follows a known probability distribution function and relies on the formulation of chance constraints, that is, constraints with a specified probability of violation.

In this chapter, an approach to identify a probabilistic multi-step model from noise-affected data is presented and the resulting model is employed in an SMPC formulation. As a key characteristic of the proposed method, the probabilistic multi-step identification is derived for linear systems without knowledge of the process and measurement noise covariance and has favorable theoretical properties. In particular, it implicitly includes a Kalman filter, yields unbiased parameter estimates and considers parametric uncertainties from the identification task. Furthermore, it is shown that these advantages are not shared with an identified and recursively evaluated state-space model. The proposed method is investigated in two case studies, demonstrating its effectiveness even for a nonlinear system.

This chapter summarizes the main findings of the publication [42]. The full publication is provided in Appendix C.4.

4.1 Probabilistic multi-step model

In this section, an uncertain linear system is introduced and recursively evaluated to obtain a distributed multi-step prediction. Initially, it is assumed that the system matrices are known and the multi-step identification task is discussed in Section 4.2. The multi-step distribution with known or identified parameters is then used to formulate an SMPC controller, as described in Section 4.3.

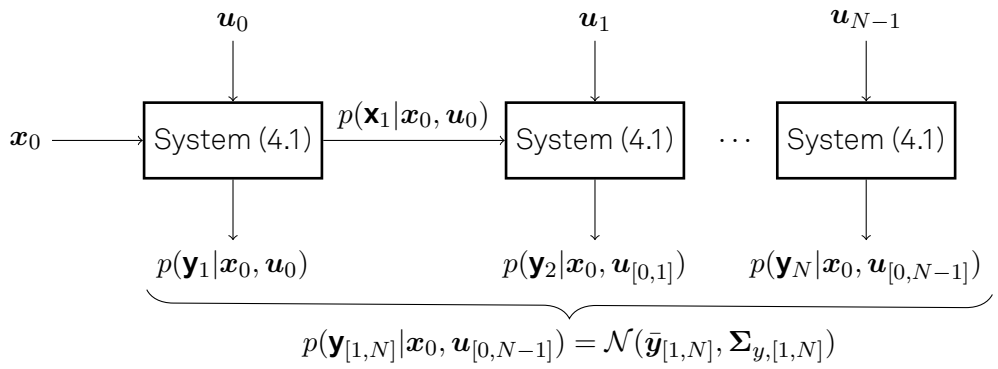

Figure 4.1: Computing the distributed multi-step prediction from $\boldsymbol{x}_0$ by recursively evaluating the dynamic system.

In the context of this chapter, the previously introduced linear dynamic system (3.1) is now subject to non-deterministic process noise $\mathbf{e}_x \in \mathbb{R}^{n_{e,x}}$ and measurement noise $\mathbf{e}_y \in \mathbb{R}^{n_{e,y}}$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{E}_x\mathbf{e}_{x,k} \tag{4.1a}$$
$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k + \boldsymbol{E}_y\mathbf{e}_{y,k}, \tag{4.1b}$$

with $\boldsymbol{E}_x \in \mathbb{R}^{n_x \times n_{e,x}}$ and $\boldsymbol{E}_y \in \mathbb{R}^{n_y \times n_{e,y}}$. To obtain a tractable distribution for the multi-step model and to facilitate the identification task, the following assumptions are introduced.

Assumption 4.1. *The additive measurement and process noise in* (4.1) *are normally distributed, that is,* $\mathbf{e}_{x,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_x)$ *and* $\mathbf{e}_{y,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_y)$, $\forall k$.

Assumption 4.2. *System* (4.1) *is subject to noisy state-feedback, that is,* $\boldsymbol{C} = \boldsymbol{I}$ *and* $\boldsymbol{D} = \mathbf{0}$.

Assumption 4.2 can be satisfied for an observable linear system by introducing a state from finite sequences of past outputs and measurements, as shown in Appendix A.2, and significantly simplifies the successive notation. There are, however, limitations due to Assumptions 4.2 which are discussed in Subsection 4.4.2.

4.1.1 Probabilistic model with known initial state

As the first step to derive the probabilistic multi-step model, the conditional distribution:

$$p(\mathbf{y}_{[1,N]}|\boldsymbol{x}_0, \boldsymbol{u}_{[0,N-1]}), \tag{4.2}$$

is formulated. This distribution, conditional on the initial state $\boldsymbol{x}_0$ and the sequence of inputs $\boldsymbol{u}_{[0,N-1]}$, can be obtained by recursively evaluating the system, as indicated in Figure 4.1. For the noise-affected dynamic system (4.1), this yields:

$$\begin{aligned}\boldsymbol{y}_{[0,N-1]} =& \mathcal{O}_N(\boldsymbol{A}, \boldsymbol{I})\boldsymbol{x}_0 + \mathcal{T}_N(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{I}, \mathbf{0})\boldsymbol{u}_{[0,N-1]} + \mathcal{T}_N(\boldsymbol{A}, \boldsymbol{E}_x, \boldsymbol{I}, \mathbf{0})\mathbf{e}_{x,[0,N-1]} \\ &+ (\boldsymbol{I}_N \otimes \boldsymbol{E}_y)\mathbf{e}_{y,[0,N-1]},\end{aligned} \tag{4.3}$$

where $\otimes$ denotes the Kronecker product. As main differences to the deterministic multi-step model (3.4), the process noise is propagated and the measurement noise is added. Furthermore, Assumption 4.2 is considered, that is, $\boldsymbol{C} = \boldsymbol{I}$ and $\boldsymbol{D} = \boldsymbol{0}$. As a consequence, it is possible to predict the shifted sequence $\boldsymbol{y}_{[1,N]}$ instead of $\boldsymbol{y}_{[0,N-1]}$, from the sequence of inputs $\boldsymbol{u}_{[0,N-1]}$. To this end, the matrices $\mathcal{O}_N^+(\cdot)$ and $\mathcal{T}_N^+(\cdot)$ are introduced, of which block i,j is obtained as follows:

$$[\mathcal{O}_N^+(\boldsymbol{A})]_{i,1} = \boldsymbol{A}^i \qquad \forall i \in \mathbb{I}_{[1,N]}, \tag{4.4a}$$

$$[\mathcal{T}_N^+(\boldsymbol{A},\boldsymbol{B})]_{i,j} = \begin{cases} \boldsymbol{A}^{i-j}\boldsymbol{B}, & \text{if } i \geq j \\ \boldsymbol{0}, & \text{otherwise} \end{cases} \qquad \forall i,j \in \mathbb{I}_{[1,N]}. \tag{4.4b}$$

These matrices are partitions of the matrices in (3.3). With (4.4), the reformulation of (4.3) yields:

$$\begin{aligned} \boldsymbol{y}_{[1,N]} =& \mathcal{O}_N^+(\boldsymbol{A})\boldsymbol{x}_0 + \mathcal{T}_N^+(\boldsymbol{A},\boldsymbol{B})\boldsymbol{u}_{[0,N-1]} + \mathcal{T}_N^+(\boldsymbol{A},\boldsymbol{E}_x)\mathbf{e}_{x,[0,N-1]} \\ &+ (\boldsymbol{I}_N \otimes \boldsymbol{E}_y)\mathbf{e}_{y,[1,N]}, \end{aligned} \tag{4.5}$$

where the sequences of the process and measurement noise are normally distributed, according to Assumption 4.1:

$$\mathbf{e}_{x,[0,N-1]} \sim \mathcal{N}(0, \boldsymbol{I}_N \otimes \boldsymbol{\Sigma}_x), \quad \mathbf{e}_{y,[1,N]} \sim \mathcal{N}(0, \boldsymbol{I}_N \otimes \boldsymbol{\Sigma}_y). \tag{4.6}$$

The desired distribution for the probabilistic multi-step model in (4.2) is obtained as the linear transformation in (4.5) of the normal distributions in (4.6) [74, 20.23 b]. By further considering the mixed product rule for the Kronecker product [74, 11.11 a], we obtain the normal distribution:

$$p(\mathbf{y}_{[1,N]}|\boldsymbol{x}_0, \boldsymbol{u}_{[0,N-1]}) = \mathcal{N}(\bar{\boldsymbol{y}}_{[1,N]}, \boldsymbol{\Sigma}_{y,[1,N]}), \tag{4.7a}$$

with mean and covariance:

$$\bar{\boldsymbol{y}}_{[1,N]} = \mathcal{O}_N^+(\boldsymbol{A})\boldsymbol{x}_0 + \mathcal{T}_N^+(\boldsymbol{A},\boldsymbol{B})\boldsymbol{u}_{[0,N-1]}, \tag{4.7b}$$

$$\boldsymbol{\Sigma}_{y,[1,N]} = \mathcal{T}_N^+(\boldsymbol{A},\boldsymbol{E}_x)(\boldsymbol{I}_N \otimes \boldsymbol{\Sigma}_x)\mathcal{T}_N^+(\boldsymbol{A},\boldsymbol{E}_x)^\top + \boldsymbol{I}_N \otimes (\boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top). \tag{4.7c}$$

4.1.2 Probabilistic model with state estimation

The multi-step distribution in (4.7) is conditional on the initial state $\boldsymbol{x}_0$. However, due to the presence of measurement noise, the initial state is unknown and must be estimated from the respective measurement $\boldsymbol{y}_0$. The optimal state estimator for system (4.1), and considering Assumption 4.1, is the *Kalman filter* [8]. The Kalman filter yields a distribution for the estimated initial state which also affects the distribution of the multi-step prediction. Combining the state estimation and the multi-step prediction, as shown in Figure 4.2, yields the desired distribution:

$$p(\hat{\mathbf{y}}_{[1,N]}|\boldsymbol{y}_0, \boldsymbol{u}_{[0,N-1]}). \tag{4.8}$$

As the main difference to (4.2), the distribution (4.8) is conditional on the noise-affected measurements $\boldsymbol{y}_0$. As the first step to derive the expression (4.8), the Kalman

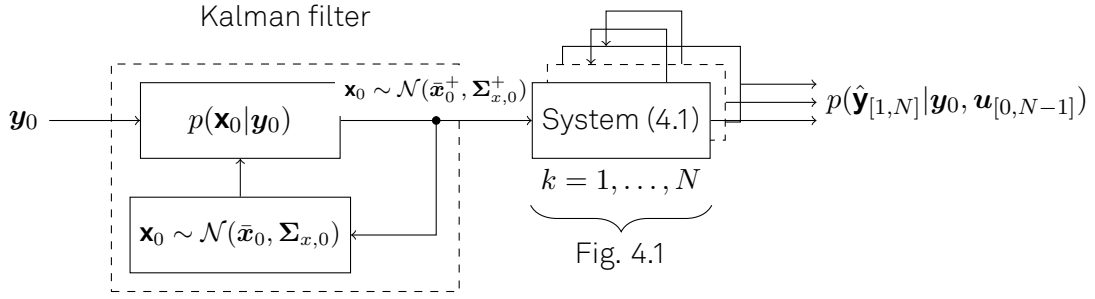

Figure 4.2: Computing the distributed multi-step prediction from $\boldsymbol{y}_0$ by estimating the initial state distribution and recursively evaluating the dynamic system. For conciseness, the recursive evaluation, shown in Figure 4.1, is compacted and omits the sequence of inputs.

filter correction step yields the conditional probability $p(\mathbf{x}_0|\boldsymbol{y}_0)$ by considering the obtained measurement $\boldsymbol{y}_0$ and a prior distribution $p(\mathbf{x}_0)$. Without loss of generality and for ease of notation, it is assumed that this prior has a zero mean.

Assumption 4.3. *The states of the dynamic system are distributed according to* $\mathbf{x}_0 \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{x,0})$.

With the prior from Assumption 4.3, the Kalman filter correction step yields [8]:

$$p(\mathbf{x}_0|\boldsymbol{y}_0) = \mathcal{N}(\bar{\boldsymbol{x}}_0^+, \boldsymbol{\Sigma}_{x,0}^+), \tag{4.9a}$$

where:

$$\bar{\boldsymbol{x}}_0^+ = \bar{\boldsymbol{x}}_0 + \boldsymbol{L}(\boldsymbol{y}_0 - \boldsymbol{C}\bar{\boldsymbol{x}}_0), \tag{4.9b}$$
$$\boldsymbol{\Sigma}_{x,0}^+ = (\boldsymbol{I} - \boldsymbol{L}\boldsymbol{C})\boldsymbol{\Sigma}_{x,0}, \tag{4.9c}$$

depends on the *Kalman gain*:

$$\boldsymbol{L} = \boldsymbol{\Sigma}_{x,0}\boldsymbol{C}^\top(\boldsymbol{C}\boldsymbol{\Sigma}_{x,0} + \boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top)^{-1}. \tag{4.9d}$$

With the estimated initial state distribution from (4.9), the dynamic system (4.1) can be recursively evaluated, as presented in the previous subsection. The sequential state estimation and multi-step prediction, displayed in Figure 4.2, is combined to directly obtain the desired distribution (4.8), as shown in the following lemma.

Lemma 4.1. *Assumption 4.1-4.3 hold. The distribution of the multi-step prediction, conditional on* $\boldsymbol{y}_0$ *and* $\boldsymbol{u}_{[0,N-1]}$*, is:*

$$p(\hat{\mathbf{y}}_{[1,N]}|\boldsymbol{y}_0, \boldsymbol{u}_{[0,N-1]}) = \mathcal{N}(\hat{\bar{\boldsymbol{y}}}_{[1,N]},\ \hat{\boldsymbol{\Sigma}}_{y,[1,N]}), \tag{4.10a}$$

with mean and covariance:

$$\hat{\bar{\boldsymbol{y}}}_{[1,N]} = \mathcal{O}_N^+(\boldsymbol{A})\boldsymbol{L}\boldsymbol{y}_0 + \mathcal{T}_N^+(\boldsymbol{A}, \boldsymbol{B})\boldsymbol{u}_{[0,N-1]}, \tag{4.10b}$$
$$\hat{\boldsymbol{\Sigma}}_{y,[1,N]} = \boldsymbol{\Sigma}_{y,[1,N]} + \mathcal{O}_N^+(\boldsymbol{A})(\boldsymbol{I} - \boldsymbol{L})\boldsymbol{\Sigma}_{x,0}\mathcal{O}_N^+(\boldsymbol{A})^\top, \tag{4.10c}$$

where $\boldsymbol{\Sigma}_{y,[1,N]}$ *is defined in* (4.7c)*, and the Kalman gain is defined in* (4.9d)*.*

Proof. The proof is shown in our work [42, Lemma 1]. □

The probabilistic multi-step model in (4.10) is further simplified by introducing:

$$\hat{\tilde{\boldsymbol{y}}}_{[1,N]} = \begin{bmatrix} \mathcal{O}_N^+(\boldsymbol{A})\boldsymbol{L} & \mathcal{T}_N^+(\boldsymbol{A},\boldsymbol{B}) \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{u}_{[0,N-1]} \end{bmatrix} = \hat{\boldsymbol{W}}^\top \boldsymbol{v}, \tag{4.11}$$

where $\boldsymbol{v}^\top = \begin{bmatrix} \boldsymbol{y}_0^\top & \boldsymbol{u}_{[0,N-1]}^\top \end{bmatrix}$. The distribution of the multi-step prediction (4.10) can thus be computed as:

$$p(\hat{\mathbf{y}}_{[1,N]} | \boldsymbol{y}_0, \boldsymbol{u}_{[0,N-1]}) = \mathcal{N}(\hat{\boldsymbol{W}}^\top \boldsymbol{v}, \hat{\boldsymbol{\Sigma}}_{y,[1,N]}), \tag{4.12}$$

with matrices $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{\Sigma}}_{y,[1,N]}$. Probabilistic multi-step identification boils down to obtaining these matrices and is discussed in the next section.

As a final remark, the prediction step of the Kalman filter is omitted in Lemma 4.1 for reasons that will be clarified in Subsection 4.3.1. For the discussion in this section, the prediction step is implicitly considered through Assumption 4.3, that is, it is assumed that the true prior distribution of the states is known.

4.2 Probabilistic multi-step identification

In the previous section, a probabilistic multi-step model in the form of (4.12) is derived from the known system matrices in (4.1) and the covariance matrices of the process and measurement noise in Assumption 4.1. However, for most real-world problems, these matrices are unknown. As a main contribution of our work [42], we propose to use maximum likelihood estimation to directly identify the parameters of the multi-step distribution in (4.12) from system data. As discussed, this distribution incorporates the effect of estimating the initial state from noisy measurements. This is a key aspect and it is shown in this section that it leads to favorable theoretical properties of the identified multi-step model. Furthermore, it is shown that these favorable properties are not shared with an identified and recursively evaluated state-space model.

For the identification task, data is obtained by collecting m sequences from the dynamic system (4.1), which are arranged in the matrices:

$$\boldsymbol{V}^\top = \begin{bmatrix} \boldsymbol{y}_0^{(1)} & \cdots & \boldsymbol{y}_0^{(m)} \\ \boldsymbol{u}_{[0,N-1]}^{(1)} & \cdots & \boldsymbol{u}_{[0,N-1]}^{(m)} \end{bmatrix}, \quad \boldsymbol{Y}_N^\top = \begin{bmatrix} \boldsymbol{y}_{[1,N]}^{(1)} & \cdots & \boldsymbol{y}_{[1,N]}^{(m)} \end{bmatrix}, \tag{4.13}$$

and form the set $\mathcal{D} = \{\boldsymbol{V},\ \boldsymbol{Y}_N\}$. As the main difference to the previous chapter, the system is now subject to process and measurement noise and Assumption 4.2 holds. Furthermore, the matrix in (4.13) is transposed in comparison to its counterpart in (3.10). This harmonizes the notation with the machine learning literature, where the first dimension of a data matrix typically refers to the samples.

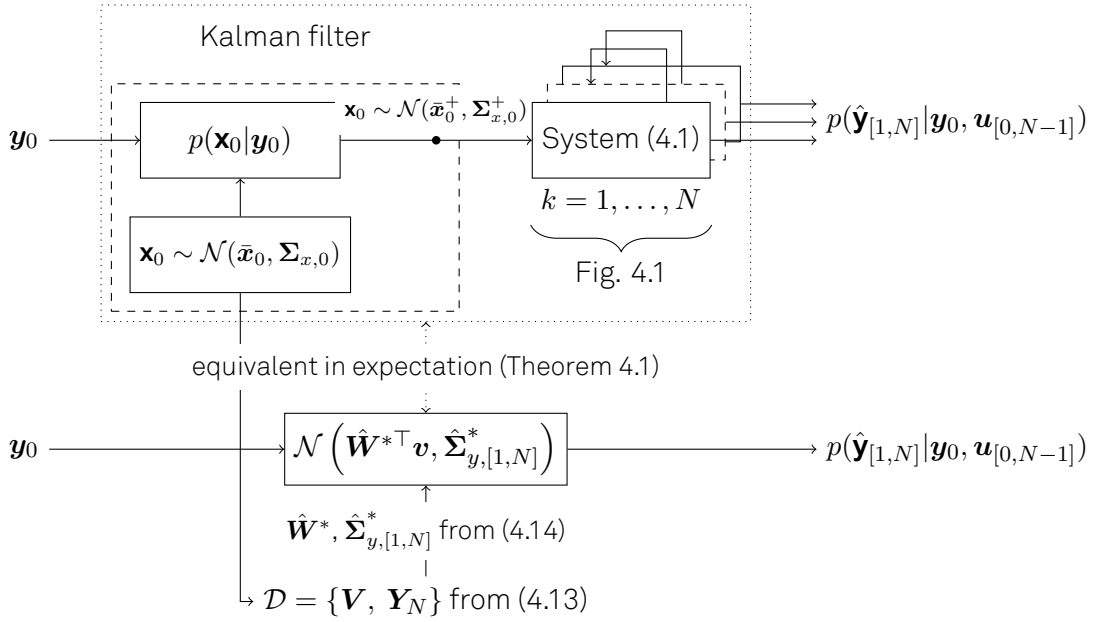

Figure 4.3: Relationship between true and identified probabilistic multi-step models. For conciseness, the recursive evaluation, shown in Figure 4.1, is compacted and omits the sequence of inputs.

4.2.1 Parameter and covariance estimation

The probabilistic multi-step identification task, with data in the form of (4.13), is formalized in the following theorem.

Theorem 4.1. *Assumption 4.2 and 4.1 hold. The collected data* (4.13) *from system* (4.1) *satisfies Assumption 4.3 and Assumption 3.2. The maximum likelihood approach yields the estimated parameters and bias corrected covariance matrix:*

$$\hat{\boldsymbol{W}}^* = \hat{\boldsymbol{\Sigma}}_p^* \boldsymbol{V}^\top \boldsymbol{Y}_N, \tag{4.14a}$$

$$\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]} = \frac{1}{m - (Nn_u + n_x)} (\boldsymbol{V}\hat{\boldsymbol{W}}^* - \boldsymbol{Y}_N)^\top (\boldsymbol{V}\hat{\boldsymbol{W}}^* - \boldsymbol{Y}_N), \tag{4.14b}$$

with:

$$\hat{\boldsymbol{\Sigma}}_p^* = (\boldsymbol{V}^\top \boldsymbol{V})^{-1}. \tag{4.14c}$$

The estimated distributed multi-step prediction is:

$$p(\hat{\mathbf{y}}^*_{[1,N]}|\mathcal{D}, \boldsymbol{v}) = \mathcal{N}(\hat{\boldsymbol{W}}^{*\top}\boldsymbol{v}, \hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}). \tag{4.15}$$

The parameters and covariance matrix have the property:

$$\mathbb{E}[\hat{\boldsymbol{W}}^*] = \hat{\boldsymbol{W}}, \tag{4.16a}$$

$$\mathbb{E}[\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}] = \hat{\boldsymbol{\Sigma}}_{y,[1,N]}. \tag{4.16b}$$

In expectation, the estimated parameters and covariance matrix are thus identical to the true parameters and covariance matrix in (4.12)*.*

Proof. The proof is shown in our work [42, Theorem 1]. □

The result of Theorem 4.1 is also summarized in Figure 4.3. With data from the uncertain system (4.1), the identified multi-step model yields, in expectation, the same distribution as the true multi-step model, including the Kalman filter state estimation. To understand the importance of Theorem 4.1 it must be considered that the multi-step identification task boils down to a linear regression problem with measurement error in the independent variables. Typically, this situation is a major challenge for linear regression, coined *error-in-variables* or *measurement-error-models* [76], [77], and leads to biased parameter estimates. Theorem 4.1 shows that the problem is circumvented by identifying the composition of state estimation and multi-step model, for which the obtained parameters are unbiased. It is desirable to include the state estimation because the identified model is successively evaluated with noise-affected measurements. However, it is necessary to consider minor limitations of Theorem 4.1, which are discussed in Subsection 4.3.1.

4.2.2 Parametric uncertainty

Theorem 4.1 allows to efficiently estimate the unknown parameters and the full covariance matrix of the probabilistic multi-step model in (4.12). While the estimated parameters converge to the true values in the limit of infinitely many samples, in practice, only a finite number of samples are available. Consequently, the parametric uncertainties of the estimated parameters must be considered. As the second main contribution of [42], we introduce the following theorem.

Theorem 4.2. *Assumption 4.2 and 4.1 hold. The collected data* (4.13) *from system* (4.1) *satisfies Assumption 4.3 and Assumption 3.2. We have a non-informative prior distribution for the parameters $\hat{\boldsymbol{W}}$ and maximum likelihood estimates of the parameter and covariance matrix from* (4.14)*. Considering the posterior distribution of the parameters, that is, the parametric uncertainty, we have the distributed multi-step prediction:*

$$p(\hat{\mathbf{y}}^*_{[1,N]}|\mathcal{D}, \boldsymbol{v}) = \mathcal{N}(\hat{\boldsymbol{W}}^{*\top}\boldsymbol{v},\ \alpha(\boldsymbol{v})\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}), \tag{4.17a}$$

with:

$$\alpha(\boldsymbol{v}) = (1 + \boldsymbol{v}^\top \hat{\boldsymbol{\Sigma}}^*_p \boldsymbol{v}), \tag{4.17b}$$

where $\boldsymbol{v}^\top = [\boldsymbol{y}_0^\top, \boldsymbol{u}_{[0,N-1]}]$, and with $\hat{\boldsymbol{\Sigma}}^*_p$, $\hat{\boldsymbol{W}}^*$ and $\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}$ from (4.14)*.*

Proof. The proof is shown in our work [42, Theorem 2]. □

With Theorem 4.2, considering the parametric uncertainty becomes a straightforward extension of the identified probabilistic multi-step model in (4.15). The scalar expression $\alpha(\boldsymbol{v}) \in [1, \infty)$ inflates the estimated covariance matrix obtained from Theorem 4.1. Theorems 4.1 and 4.2 yield consistent results, as for $m \to \infty$, it can be shown that $\alpha(\boldsymbol{v}) \to 1$ [73]. In other words, the parametric uncertainty vanishes for infinitely many samples and the expected values for the parameter and covariance matrix are obtained.

4.2.3 State-space identification

Theorem 4.1 establishes that the identified probabilistic multi-step model yields, in expectation, the same distribution as the true multi-step model, including the Kalman filter state estimation. In the following, it is shown that this favorable property is not obtained with an identified and recursively evaluated state-space model. In this way, the advantages of multi-step identification over state-space identification are shown.

State-space identification is a special case of multi-step identification with prediction horizon $N = 1$ and the parameters $\hat{\boldsymbol{W}}^*$ and $\hat{\boldsymbol{\Sigma}}^*_{y,1}$ can be estimated according to Theorem 4.1. To simplify the successive notation, we partition the estimated parameter matrix $\hat{\boldsymbol{W}}^*$ as:

$$\hat{\boldsymbol{W}}^* = \begin{bmatrix} \hat{\boldsymbol{A}}^* & \hat{\boldsymbol{B}}^* \end{bmatrix}. \tag{4.18}$$

This allows to state the following corollary.

Corollary 4.1. *As a consequence of Theorem 4.1, the identified parameters $\hat{\boldsymbol{W}}^*$ and covariance matrix $\hat{\boldsymbol{\Sigma}}^*_{y,1}$, of the multi-step model with prediction horizon $N = 1$, have the following expectation:*

$$\mathbb{E}[\hat{\boldsymbol{A}}^*] = \boldsymbol{A}\boldsymbol{L}, \tag{4.19a}$$

$$\mathbb{E}[\hat{\boldsymbol{B}}^*] = \boldsymbol{B}, \tag{4.19b}$$

$$\mathbb{E}[\hat{\boldsymbol{\Sigma}}^*_{y,1}] = \boldsymbol{E}_x \boldsymbol{\Sigma}_x \boldsymbol{E}_x^\top + \boldsymbol{E}_y \boldsymbol{\Sigma}_y \boldsymbol{E}_y^\top + \boldsymbol{A}(\boldsymbol{\Sigma}_{x,0} + \boldsymbol{L}\boldsymbol{\Sigma}_{x,0})\boldsymbol{A}^\top, \tag{4.19c}$$

where $\hat{\boldsymbol{A}}^*$ and $\hat{\boldsymbol{B}}^*$ are block elements of $\hat{\boldsymbol{W}}^*$ according to in (4.18)*. The Kalman gain $\boldsymbol{L}$ is defined in* (4.9d)*. With the identified parameters, the probabilistic state-space model:*

$$p(\hat{\mathbf{y}}^*_1 | \boldsymbol{y}_0, \boldsymbol{u}_0) = \mathcal{N}(\hat{\boldsymbol{A}}^* \boldsymbol{y}_0 + \hat{\boldsymbol{B}}^* \boldsymbol{u}_0, \hat{\boldsymbol{\Sigma}}^*_{y,1}), \tag{4.20}$$

describes, in expectation, the true distribution of the measurements at $\boldsymbol{y}_1$, given $\boldsymbol{y}_0$ and $\boldsymbol{u}_0$.

Proof. The proof is shown in Appendix A.3. □

As for the probabilistic multi-step model, the state-space identification yields unbiased parameters and the resulting distribution (4.20) includes the effect of estimating the initial state. Several problems arise, however, when (4.20) is recursively evaluated to obtain a distribution for the multi-step prediction $\hat{\mathbf{y}}_{[1,N]}$. The recursive evaluation of (4.20) is obtained without considering the parametric uncertainty, and yields:

$$p(\hat{\mathbf{y}}_{[1,N]} | \boldsymbol{y}_0, \boldsymbol{u}_{[0,N-1]}) = \mathcal{N}(\hat{\bar{\boldsymbol{y}}}_{[1,N]},\, \hat{\boldsymbol{\Sigma}}_{y,[1,N]}), \tag{4.21a}$$

$$\text{with:} \quad \hat{\bar{\boldsymbol{y}}}_{[1,N]} = \mathcal{O}_N^+(\hat{\boldsymbol{A}}^*)\boldsymbol{y}_0 + \mathcal{T}_N^+(\hat{\boldsymbol{A}}^*, \hat{\boldsymbol{B}}^*)\boldsymbol{u}_{[0,N-1]}, \tag{4.21b}$$

$$\hat{\boldsymbol{\Sigma}}_{y,[1,N]} = \mathcal{T}_N^+(\hat{\boldsymbol{A}}^*, \boldsymbol{I})(\boldsymbol{I}_N \otimes \hat{\boldsymbol{\Sigma}}^*_{y,1})\mathcal{T}_N^+(\hat{\boldsymbol{A}}^*, \boldsymbol{I})^\top. \tag{4.21c}$$

To discuss the behavior of the identified and recursively evaluated state-space model, the expected values of the estimated parameters (4.19) are considered. As the first problem, the expected value of the estimated covariance (4.21c) contains contributions from both the process noise and the measurement noise. The recursively

evaluated state-space model (4.21) thus propagates the measurement noise which incorrectly inflates the uncertainty of the multi-step prediction. For the second problem, we observe that the mean of the multi-step prediction (4.21b), with expected values (4.19), yields:

$$\hat{\bar{\boldsymbol{y}}}_{[1,N]} = \mathcal{O}_N^+(\boldsymbol{AL})\boldsymbol{y}_0 + \mathcal{T}_N^+(\boldsymbol{AL}, \boldsymbol{B})\boldsymbol{u}_{[0,N-1]}. \tag{4.22}$$

This is in contrast to the true mean of the multi-step prediction in (4.10b):

$$\hat{\bar{\boldsymbol{y}}}_{[1,N]} = \mathcal{O}_N^+(\boldsymbol{A})\boldsymbol{L}\boldsymbol{y}_0 + \mathcal{T}_N^+(\boldsymbol{A}, \boldsymbol{B})\boldsymbol{u}_{[0,N-1]}.$$

This discrepancy is a major disadvantage of state-space identification over multi-step identification for MPC. If both models are identified from data of a linear dynamic system with measurement noise, only the multi-step model yields unbiased predictions. Importantly, the second problem also applies to the identification of non-probabilistic models, as presented in Chapter 3. The final disadvantage of state-space identification is the parametric uncertainty discussed in Subsection 4.2.2. This uncertainty can be considered with (4.17) and $N = 1$. However, there is no closed-form solution for the recursive application of (4.20) if parametric uncertainties are considered.

4.3 Stochastic model predictive control

In this section, stochastic model predictive control (SMPC) is introduced. First, the stochastic optimal control problem is formulated for the probabilistic multi-step model in (4.10) with known system parameters. Successively, the identified probabilistic multi-step model (4.17) and the identified probabilistic state-space model (4.20) are employed for the formulation of a data-based SMPC problem.

Given the known system parameters, a distributed multi-step model (4.10), yielding the distribution $\hat{\mathbf{y}}_{[1,N]}$, can be used to formulate a stochastic optimal control problem in the following way [13], [14]:

$$\begin{aligned} \min_{\boldsymbol{u}_{[0,N-1]} \in \mathbb{U}} \quad & \mathbb{E}\left[\|\hat{\mathbf{y}}_{[1,N]}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}_{[0,N-1]}\|_{\boldsymbol{R}}^2\right] \\ \text{s.t.} \quad & P\left[\boldsymbol{a}_j^\top \hat{\mathbf{y}}_{[1,N]} \leq b_j\right] \geq (1-\epsilon) \quad \forall j \in \mathbb{I}_{[1,n_c]}, \end{aligned} \tag{4.23}$$

with expectation $\mathbb{E}(\cdot)$ and probability $P(\cdot)$. The cost function is formulated with positive definite weighting matrices, that is, $\boldsymbol{Q} \succ 0$, $\boldsymbol{R} \succ 0$, and the inputs are constrained to the convex set $\mathbb{U}$. The n_c *chance constraints* can be violated with probability $\epsilon \in (0, 1)$, and are introduced as individual halfspaces [78, p. 2.2.1] with $\boldsymbol{a}_j \in \mathbb{R}^{n_y}$ and $b_j \in \mathbb{R}$ for $j \in \mathbb{I}_{[1,n_c]}$.

In the described setting, the stochastic optimal control problem (4.23) can be reformulated as a deterministic problem that yields the same optimal solution [14]:

$$\begin{aligned} \min_{\boldsymbol{u}_{[0,N-1]} \in \mathbb{U}} \quad & \|\hat{\bar{\boldsymbol{y}}}_{[1,N]}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}_{[0,N-1]}\|_{\boldsymbol{R}}^2 + \operatorname{trace}\left(\boldsymbol{Q}\hat{\boldsymbol{\Sigma}}_{y,[1,N]}\right) \\ \text{s.t.} \quad & \boldsymbol{a}_j^\top \hat{\bar{\boldsymbol{y}}}_{[1,N]} \leq b_j - c_p(\epsilon)\|\boldsymbol{a}_j\|_{\hat{\boldsymbol{\Sigma}}_{y,[1,N]}} \quad \forall j \in \mathbb{I}_{[1,n_c]}, \end{aligned} \tag{4.24}$$

with $\hat{\bar{\boldsymbol{y}}}_{[1,N]}$ and $\hat{\boldsymbol{\Sigma}}_{y,[1,N]}$ from (4.10b) and (4.10c) . The factor $c_p(\epsilon) = \sqrt{2}\text{erf}^{-1}(1-2\epsilon)$ is the quantile of the standard normal distribution and erf^{-1} denotes the inverse error-function [79]. If the covariance of the multi-step distribution, that is, $\hat{\boldsymbol{\Sigma}}_{y,[1,N]}$, is constant, the cost function in (4.24) can be further simplified by omitting the respective term. However, as shown in Subsection 4.2.2, this is not the case for the identified multi-step model. Problem (4.24) is a quadratic optimization problem that depends on the measured initial state $\boldsymbol{y}_0$ and yields the feedback control law:

$$\boldsymbol{u}_k = \kappa_{\text{SMPC}}(\boldsymbol{y}_k). \tag{4.25}$$

4.3.1 Stochastic MPC with identified multi-step model

The multi-step distribution in (4.17) is the data-based equivalent to (4.10), including the parametric uncertainty from the identification task. With (4.17), the deterministic formulation of the stochastic optimal control problem (4.24) is given as follows:

$$\min_{\boldsymbol{u}_{[0,N-1]} \in \mathbb{U}} \|\hat{\bar{\boldsymbol{y}}}^*_{[1,N]}\|^2_{\boldsymbol{Q}} + \|\boldsymbol{u}_{[0,N-1]}\|^2_{\boldsymbol{R}} + \text{trace}\left(\alpha(\boldsymbol{v})\boldsymbol{Q}\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}\right) \tag{4.26a}$$

$$\text{s.t. } \boldsymbol{a}_j^\top \hat{\bar{\boldsymbol{y}}}^*_{[1,N]} \leq b_j - c_p(\epsilon)\|\boldsymbol{a}_j\|_{\alpha(\boldsymbol{v})\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}} \quad \forall j \in \mathbb{I}_{[1,n_c]}. \tag{4.26b}$$

The weight matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, constraint set $\mathbb{U}$ and the chance constraints, defined with $\boldsymbol{a}_j$, b_j $\forall j \in \mathbb{I}_{[1,n_c]}$ and ϵ, are analogous to (4.24). As the main difference between (4.24) and (4.26), the mean $\hat{\bar{\boldsymbol{y}}}^*_{[1,N]}$ and covariance $\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}$ now stem from distribution (4.17). The parametric uncertainty of the identified model results in the factor $\alpha(\boldsymbol{v})$, introduced in (4.17b), where $\boldsymbol{v}^\top = [\boldsymbol{y}_0^\top, \boldsymbol{u}^\top_{[0,N-1]}]$.

Due to the parametric uncertainty, and in contrast to (4.24), the optimization problem in (4.26) is no longer a quadratic problem. It is shown in [42, Appendix A] that problem (4.26) constitutes a convex second-order cone program [78]. Solving Problem (4.26) repeatedly for the measured initial state $\boldsymbol{y}_k$ yields the control law:

$$\boldsymbol{u}_k = \kappa^{\text{MSM}}_{\text{SMPC}}(\boldsymbol{y}_k). \tag{4.27}$$

The proposed data-based SMPC controller (4.27) allows to safely operate the unknown system (4.1). However, some limitations of the approach must be considered. As shown in Theorem 4.1, multi-step identification yields the parameters and covariance matrix for the distribution in (4.12). This distribution incorporates a Kalman filter correction step, considering the prior distribution of the initial state introduced in Assumption 4.3. This prior coincides with the distribution of the initial state from the sampled data $p_{\text{samp.}}(\boldsymbol{x}_0)$, as required for Theorem 4.1. This relationship is also indicated in Figure 4.3. As a consequence, the multi-step model always considers the same prior distribution, that is, $p_{\text{prior}}(\boldsymbol{x}_0) = p_{\text{samp.}}(\boldsymbol{x}_0)$.

This has two important limitations. First, it is imperative that the multi-step model is evaluated only for initial states that are reasonably probable in terms of the distribution of the initial state from the sampled data. This limitation is intuitive since performing a Kalman filter correction step with an incorrect prior state distribution is detrimental. Similarly, it may be detrimental to evaluate a data-based model for extrapolation points. It is, however, a limitation in the sense that data for

the identification task must be sampled in the same range that is expected for the closed-loop operation. The second limitation arises from the fact that the multi-step identification is performed prior to the control task. As a consequence, the prior state distribution is not updated during the control task. Updating the prior distribution is typically an important step of Kalman filtering and allows for convergence to the true state distribution over multiple iterations. We indicate that the prior distribution is unchanged in Figure 4.3. In practice, this limitation increases the uncertainty of the predicted future measurements and may lead to more conservative control actions of the SMPC controller. In Chapter 6, future steps to remedy both limitations are discussed.

Apart from the specific problems of the proposed data-based approach, SMPC also has general challenges, for example, proving recursive feasibility and stability, for which the reader is referred to the literature [13], [14].

4.3.2 Stochastic MPC with identified state-space model

To illustrate the advantages of the proposed SMPC formulation based on probabilistic multi-step identification, we also investigate an SMPC controller based on probabilistic state-space identification in [42]. In particular, the SMPC problem in (4.24) is formulated using the identified state-space model as:

$$\min_{\boldsymbol{u}_{[0,N-1]} \in \mathbb{U}} \ \|\hat{\bar{\boldsymbol{y}}}^*_{[1,N]}\|^2_{\boldsymbol{Q}} + \|\boldsymbol{u}_{[0,N-1]}\|^2_{\boldsymbol{R}} \tag{4.28a}$$

$$\text{s.t. } \boldsymbol{a}_j^\top \hat{\bar{\boldsymbol{y}}}^*_{[1,N]} \leq b_j - c_p(\epsilon)\|\boldsymbol{a}_j\|_{\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}} \ \forall j \in \mathbb{I}_{[1,n_c]}, \tag{4.28b}$$

where the mean $\hat{\bar{\boldsymbol{y}}}^*_{[1,N]}$ and covariance $\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}$ now stem from distribution (4.21). This multi-step distribution is obtained by recursively evaluating (4.20) without considering the parametric uncertainty. For this reason, the cost function in (4.28a) also omits the constant trace term of the covariance matrix. Solving Problem (4.28) repeatedly for the measured initial state $\boldsymbol{y}_k$ yields the control law:

$$\boldsymbol{u}_k = \kappa^{\text{SSM}}_{\text{SMPC}}(\boldsymbol{y}_k). \tag{4.29}$$

The data-based SMPC controller (4.29) with identified state-space model is expected to have significant shortcomings that are demonstrated in the following case studies. The main disadvantage stems from the recursive evaluation of the identified probabilistic state-space model. As discussed in Subsection 4.2.3, this recursive evaluation yields, in expectation, a biased mean prediction, an inflated covariance matrix, and does not consider the parametric uncertainty. Comparing the SMPC controller based on state-space identification (4.29) with the variant based on multi-step identification (4.27) is still relevant, however, as state-space identification is commonly employed in practice.

4.4 Case studies

The proposed SMPC controller based on probabilistic multi-step identification is investigated in two case studies. In the first study, the investigated system satisfies

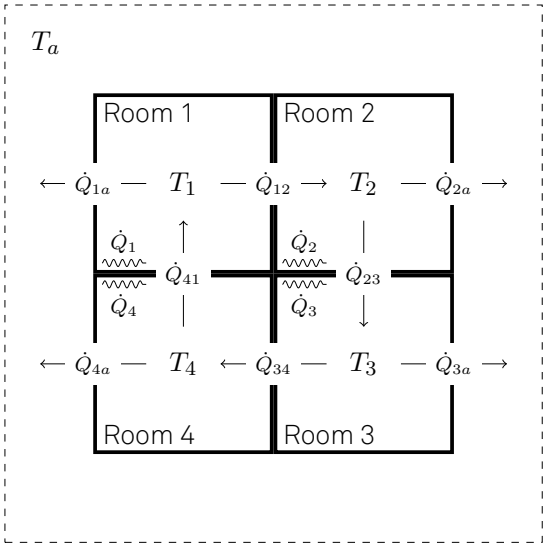

Figure 4.4: Resistance-capacitance network representation of the building system.

Assumptions 4.1 and 4.2. This is to highlight that the identified multi-step distribution (4.17) incorporates the state estimation, as shown in Theorem 4.1, and considers the parametric uncertainty derived in Theorem 4.2. In the second investigation, the applicability of the proposed method for a nonlinear system with output-feedback is investigated. In both investigations, the SMPC controller based on multi-step identification (4.27) is compared with a variant based on state-space identification (4.29).

4.4.1 Linear building system

In the first case study, a linear building system is investigated. The system is modeled as a resistance-capacitance network [80], with parameters from [81], and is depicted in Figure 4.4. The full model description is provided in Appendix B.3 and further details can be found in [42]. The main objective of the investigation is to illustrate the results of Theorems 4.1 and 4.2.

As a prerequisite for the investigation, a prior distribution of the initial state is defined according to Assumption 4.3 This prior distribution is used to sample data for the system identification, as shown in Figure 4.3. With the obtained data, two multi-step models, both with horizon $N = 12$, are identified and compared. The first multi-step model uses $m = 1000$ samples for the identification and is denoted MSM$_a$. For comparison, a multi-step model using only $m = 100$ samples is identified and denoted MSM$_b$. Both models are then evaluated to predict the multi-step distribution and compared with the ground truth. The ground truth distribution is obtained as shown in the upper half of Figure 4.3, that is, by estimating the initial state with a Kalman filter and recursively evaluating the dynamic system using the true system matrices and covariances.

Theorem 4.1 concludes that the identified multi-step model yields, in expectation, the true multi-step distribution. In Figure 4.5 it is shown that this conclusion also holds true in practice. The distributed multi-step prediction $p(\hat{\mathbf{y}}_{[1,N]})$ is displayed in terms of the mean and the standard deviation for three selected states. Furthermore,

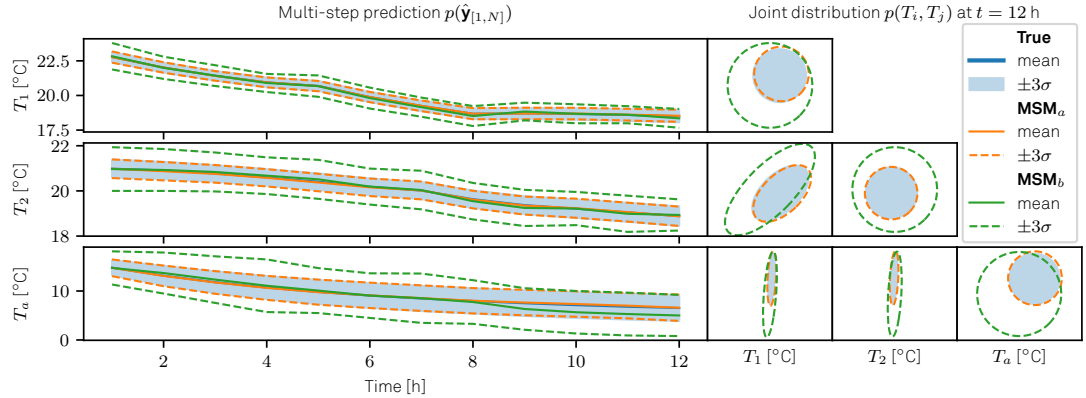

Figure 4.5: Distributed multi-step prediction of the building system for an initial measurement $\boldsymbol{y}_0$ and random control sequence $\boldsymbol{u}_{[0,N-1]}$ for $N = 12$. Comparison of true distribution (4.10) and two identified multi-step models (MSMs). The distribution of MSM$_a$ with $m = 1000$ samples and MSM$_b$ with $m = 100$ samples are obtained according to (4.17) [42, Figure 1].

the joint distributions $p(T_i, T_j)$ at $t = 12$ h are shown. For the multi-step model with $m = 1000$ samples (MSM$_a$), the identified parameters are close to their expected values. Consequently, the ground truth distribution and predicted distribution are almost identical. For the multi-step model with $m = 100$ samples (MSM$_b$), the effect of the parametric uncertainty, derived in Theorem 4.2, can also be seen in Figure 4.5. The predicted mean deviates slightly from the ground truth mean. However, this shortcoming is compensated by an increased predicted variance, resulting from the parametric uncertainty. The relatively high parametric uncertainty stems from the fact that $m = 100$ is close to the lower limit of $m_{\text{min}} = n_x + Nn_u = 65$ samples that are required to satisfy Assumption 3.2.

Using the identified multi-step model MSM$_a$, we formulate an SMPC controller in our work [42] and compare the controller to a variant based on state-space identification. As a main conclusion from this investigation, the SMPC formulation based on the identified multi-step model yields significantly better control performance. In terms of the overall energy consumption of the building, an average reduction of 8.3 % is observed [42]. It is found that this reduction is primarily due to the significantly larger predicted variances from the identified and recursively evaluated state-space model [42]. Following the discussion in Subsection 4.2.3, this increased variance is expected and leads to a larger back-off from the chance constraints.

4.4.2 Nonlinear CSTR

For the second case study, the CSTR system introduced in Subsection 2.3.2 is investigated. The system is depicted in Figure 2.1 and the full model description is provided in Appendix B.1. The nonlinear CSTR is investigated in two configurations, with state-feedback and with output-feedback. The main objective of this case study is to show that the proposed method can achieve excellent performance even if the underlying assumptions are violated. In this way, the applicability to real-world problems is demonstrated.

Table 4.1: Performance of SMPC for the control of the CSTR. Comparison between SMPC with an identified multi-step model (MSM) and an identified state-space model (SSM), either with state-feedback or output-feedback. All performance indicators are computed as mean and standard deviation over 50 experiments [42, Table 3]

Output-feedback	MSM	product [mol]	6.15±0.23
		max. cons. viol [°C]	0.00±0.00
	SSM	product [mol]	5.79±1.10
		max. cons. viol [°C]	0.01±0.06
State-feedback	MSM	product [mol]	7.17±0.21
		max. cons. viol [°C]	0.00±0.00
	SSM	product [mol]	6.16±0.21
		max. cons. viol [°C]	0.00±0.00

The underlying assumptions of the proposed method are violated because the investigated CSTR is nonlinear and may exhibit output-feedback. Both properties violate Assumption 4.1, that is, the process and measurement noise are not independent and normally distributed. Any nonlinear system can be reformulated as a linear system with additive process noise. In this reformulation, the process noise incorporates the nonlinearities of the system and, therefore, is not normally distributed. Furthermore, Assumption 4.1 is violated for systems with output-feedback. An observable linear system with output-feedback can be reformulated as a linear system with state-feedback, as shown in Appendix A.2. This requires the definition of the new state (3.6) from sequences of past inputs and measurements. Assumption 4.1 is violated in that case because successive measurements of the new state are correlated [16, Sec. 5].

Despite the discussed challenges, an SMPC controller with identified probabilistic multi-step model is employed. As in the previous case study, the obtained SMPC controller is compared with a variant based on state-space identification. For all investigated variants, the system is identified from $m = 500$ simulated sequences with a sequence length of $L = N + t$ and $N = 20$. For the variants with output-feedback, it is assumed that only the reactor temperature T_R and the product concentration c_B are measured and $t = 3$ is selected. The initial state of each sequence is obtained as a random uniform sample, which further violates Assumption 4.3. For all identified models, the SMPC problem is formulated with the economic control objective to maximize the concentration of the desired product c_b and considers a chance constraint for the safety-critical reactor temperature, that is, $P(T_R \leq 135\,°\text{C}) \geq 99.9\,\%$. Further details are given in our work [42].

To evaluate the performance of all investigated controller variants, 50 closed-loop trajectories are generated, starting from a random initial state. The performance is then quantified in terms of the obtained product and the maximum constraint violation. The results are presented in Table 4.1 as mean and standard deviation of the performance metrics. It can be seen that SMPC with multi-step model (MS-SMPC) with and without output-feedback yields excellent control performance for the investigated nonlinear CSTR system. The reactor is operated safely without constraint violations and the product yield is maximized. In contrast, SMPC with state-space model (SS-SMPC) results in minor constraint violations in the case of output-feedback

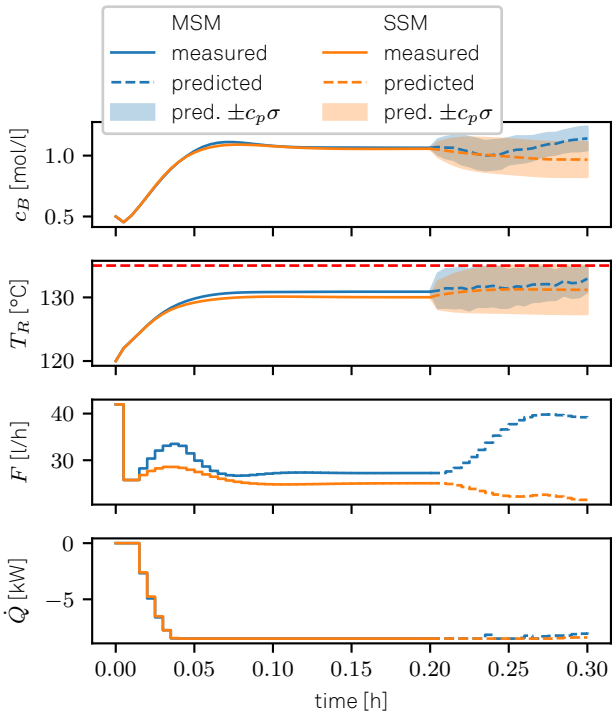

Figure 4.6: Closed-loop results and future prediction with uncertainty bounds for the CSTR system. Comparison of SMPC with identified multi-step model (MSM) and identified state-space model (SSM) in the case of output-feedback [42, Figure 4].

and yields less product in both cases. In the case of state-feedback, the MS-SMPC controllers results in 11.2 % more product than the SS-SMPC controller. To explain these results, Figure 4.6 shows an exemplary closed-loop trajectory for the MS-SMPC and the SS-SMPC controlled system, in the case of output-feedback. The figure also shows the predicted and optimized distributions at the final time-step. It is important to realize that the spread of the distributions is identified from data and approximates the nonlinearities of the CSTR system. The open-loop prediction at the last time-step shows that both controllers consider the uncertainty of the predicted future reactor temperature and back-off from the respective chance-constraint. However, due to the discussed shortcoming of state-space identification, this back-off is unnecessarily conservative for the SS-SMPC controller. As a result, the MS-SMPC controller yields more product.

4.5 Conclusions

In this chapter, probabilistic multi-step identification for stochastic MPC is introduced as an attractive solution for data-based control under uncertainty. It is shown that an identified multi-step model yields, in expectation, the true multi-step distribution of an unknown linear system with process and measurement noise. The identified multi-step distribution also incorporates the effect of estimating the initial state distribution and the parametric uncertainty from the identification task. In contrast, if a state-

space model is identified from the same data, the resulting multi-step prediction is biased. Furthermore, state-space identification cannot easily consider the parametric uncertainty for the distributed multi-step prediction. As a conclusion, the multi-step identification approach has inherent advantages over state-space identification and is well suited for the formulation of data-based SMPC controllers.

While the probabilistic multi-step identification approach is derived for linear systems with state-feedback, it is shown that the approach can also be applied to a nonlinear system with output-feedback. In this setting, the identified covariance of the multi-step model incorporates the effects of the nonlinearities of the system. In the investigated case study, the resulting SMPC controller safely controls the system with excellent performance.

For future work, a natural, but non-trivial, extension is to combine multi-step identification and nonlinear identification. For example, a multi-step model could be identified with neural networks, as introduced in Chapter 2. A prerequisite for this potential application is uncertainty quantification with neural networks, which is discussed in the next chapter.

Chapter 5

Uncertainty quantification with neural networks

A challenge for the application of traditional neural networks in system identification and control is their tendency to overfit and their inability to express uncertainty [26]. *Bayesian neural networks* (BNNs), in which the weights and predictions are probability distributions, are a theoretical concept to address this shortcoming. For practical applications, BNNs are typically approximated with sampling-based Markov chain Monte Carlo or variational inference methods [26]. However, due to the requirement to sample the predictive distribution, both categories of methods have only limited application for optimization-based control.

In this chapter, neural networks with *Bayesian last layer* [45], [46] are introduced. They constitute a simplified BNN where only the weights of the last layer and the prediction follow a Gaussian distribution and the remaining layers contain deterministic weights. In this way, neural networks with Bayesian last layer are a promising compromise between tractability and expressiveness and may serve as a versatile and important tool for system identification for predictive control. To leverage these promises, two important challenges must be addressed. Neural networks with Bayesian last layer should be trained by maximizing the marginal likelihood, which has previously been found unsuitable for gradient-based optimization. Additionally, the predictive distribution of a neural network with Bayesian last layer may be overconfident for extrapolation points. Solutions to overcome these challenges are proposed in this chapter. The improved uncertainty quantification of neural networks with Bayesian last layer is demonstrated in a simulation study and potential applications in the realm of system identification for control are discussed.

This chapter summarizes the main findings of the publication [43]. The full publication is provided in Appendix C.5.

5.1 Neural networks with Bayesian last layer

As in Chapter 2, a neural network (2.6) is used to approximate an unknown function $t = f(\boldsymbol{v})$, using the data $\mathcal{D} = \{\boldsymbol{v}^{(i)}, t^{(i)}\}_{i=1}^{m}$. For ease of notation, a scalar output is assumed in the following and the extension to the multivariate case can be found

in [43]. A prerequisite for neural networks with Bayesian last layer is the following assumption.

Assumption 5.1. *The NN* (2.6) *has a linear activation function in the output layer, that is,* $g_{L+1}(\boldsymbol{h}_{L+1}) = \boldsymbol{h}_{L+1}$.

While this Assumption was not explicitly required in Chapter 2, it is common practice for regression tasks to have a linear function in the last layer. To simplify the following notation, the output of the last internal layer of the neural network is denoted as:

$$\tilde{\boldsymbol{\phi}} = \boldsymbol{a}_L, \text{ and } \boldsymbol{\phi} = [\boldsymbol{a}_L^\top,\ 1]^\top, \tag{5.1}$$

with $\tilde{\boldsymbol{\phi}} \in \mathbb{R}^{\tilde{n}_\phi}$ and $\boldsymbol{\phi} \in \mathbb{R}^{n_\phi}$. In summary, a neural network that satisfies Assumption 5.1 is a linear function with respect to the features $\boldsymbol{\phi}$ in the last layer. Under similar premises as for the probabilistic multi-step identification in Chapter 4, it is, therefore, possible to obtain a distributed prediction. To formally introduce the neural network with Bayesian last layer, two additional assumptions are required.

Assumption 5.2. *The NN* (2.6) *provides a feature space* $\boldsymbol{\phi}(\boldsymbol{v}; \mathbb{W}_L) \in \mathbb{R}^{n_\phi}$ *from which the targets are obtained as:*

$$t^{(i)} = \boldsymbol{\phi}\left(\boldsymbol{v}^{(i)}; \mathbb{W}_L\right)^\top \boldsymbol{w}_{L+1} + \mathrm{e}^{(i)} \qquad \forall i \in \mathbb{I}_{[1,m]}, \tag{5.2}$$

where $\mathbb{W}_L = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L\}$ *denotes the weights until layer* L *and* $\boldsymbol{w}_{L+1}$ *are the weights in the last layer for a scalar output. The additive noise* $\mathrm{e} \in \mathbb{R}$ *is zero-mean normally distributed, that is,* $\mathrm{e}^{(i)} \sim \mathcal{N}(0, \sigma_e^2)$ *for all* $i \in \mathbb{I}_{[1,m]}$.

Assumption 5.3. *We have a zero-mean Gaussian prior belief for the weights of the output layer, that is,* $\mathbf{w}_{L+1} \sim \mathcal{N}(0, \sigma_w^2 \boldsymbol{I})$.

Considering Assumption 5.2 and 5.3, the parameters of the probabilistic model (5.2) are introduced as:

$$\Theta = \{\mathbb{W}_L, \sigma_e, \sigma_w\}. \tag{5.3}$$

Finally, the training data $\mathcal{D}$ is used to form the feature matrix and the target vector:

$$\boldsymbol{\Phi}^\top = \begin{bmatrix} \boldsymbol{\phi}\left(\boldsymbol{v}^{(1)}; \mathbb{W}_L\right) & \ldots & \boldsymbol{\phi}\left(\boldsymbol{v}^{(m)}; \mathbb{W}_L\right) \end{bmatrix}, \tag{5.4}$$

$$\boldsymbol{t}^\top = \begin{bmatrix} t^{(1)} & \ldots & t^{(m)} \end{bmatrix}, \tag{5.5}$$

With these prerequisites, the following Lemma can be stated.

Lemma 5.1. *Assumptions 5.1-5.3 hold. The neural network has normally distributed weights in the last layer, that is:*

$$p(\mathbf{w}_{L+1} | \mathcal{D}, \Theta) = \mathcal{N}(\bar{\boldsymbol{w}}_{L+1}, \boldsymbol{\Lambda}_p^{-1}) \tag{5.6a}$$

$$\bar{\boldsymbol{w}}_{L+1} = \sigma_e^{-2} \boldsymbol{\Lambda}_p^{-1} \boldsymbol{\Phi}^\top \boldsymbol{t} \tag{5.6b}$$

$$\boldsymbol{\Lambda}_p = \sigma_e^{-2} \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \sigma_w^{-2}, \tag{5.6c}$$

with feature matrix $\boldsymbol{\Phi}$ *from* (5.4) *and target vector* $\boldsymbol{t}$ *from* (5.5)*. For a test point* $\boldsymbol{v}$*, the predicted output of the neural network is normally distributed with:*

$$p(\mathbf{t}|\mathcal{D}, \Theta, \boldsymbol{v}) = \mathcal{N}(\bar{\boldsymbol{t}}, \boldsymbol{\Sigma}_t), \tag{5.7a}$$

$$\bar{\boldsymbol{t}} = \boldsymbol{\phi}^\top \bar{\boldsymbol{w}}_{L+1}, \tag{5.7b}$$

$$\boldsymbol{\Sigma}_t = \boldsymbol{\phi}^\top \boldsymbol{\Lambda}_p^{-1} \boldsymbol{\phi} + \sigma_e^2, \tag{5.7c}$$

Proof. The proof is shown in our work [43, Lemma 1 & Result 1]. □

In the proof of Lemma 5.1 *Bayes' law* [73] is applied to the weights of the last layer to obtain their posterior distribution. This is in contrast to all other weights of the neural network which are considered deterministic. In this sense, the neural network has a Bayesian last layer.

5.1.1 Marginal likelihood maximization

A neural network with Bayesian last layer is reminiscent of a probabilistic linear model, as introduced in Chapter 4. Consequently, it is possible to obtain maximum likelihood estimates of the weights in the last layer, as well as the covariance matrix, similarly to Theorem 4.1. However, this requires the features in the last layer, or equivalently the weights in all previous layers, to be fixed and does not fully utilize the capabilities of neural networks for function approximation.

To obtain suitable values for the deterministic weights, and, simultaneously, the unknown noise and prior variances σ_e and σ_w, it is desirable to maximize the *marginal likelihood* of the probabilistic model. Maximizing the marginal likelihood has the attractive interpretation of favoring the least complex model that suitably describes the available data [73].

The negative log-marginal likelihood of the neural network with Bayesian last layer is derived as [43, Result 1]:

$$\begin{aligned} J(\Theta; \mathcal{D}) = &\frac{m}{2} \log(2\pi) + n_\phi \log(\sigma_w) + m \log(\sigma_e) \\ &+ \frac{1}{2} \log\det(\boldsymbol{\Lambda}_p) + \frac{1}{2\sigma_e^2} \|\bar{\boldsymbol{t}} - \boldsymbol{t}\|_2^2 + \frac{1}{2\sigma_w^2} \|\bar{\boldsymbol{w}}_{L+1}\|_2^2, \end{aligned} \tag{5.8}$$

with parameters Θ from (5.3). The mean of the weights in the last layer, that is, $\bar{\boldsymbol{w}}_{L+1}$ stems from (5.6b), and is computed with the precision matrix $\boldsymbol{\Lambda}_p$ in (5.4) and the target vector $\boldsymbol{t}$ in (5.5). The predicted mean $\bar{\boldsymbol{t}}$ is obtained from (5.7a). The optimal parameters are then obtained as:

$$\begin{aligned} \Theta^*,\ \bar{\boldsymbol{w}}_{L+1}^* = \arg\min_{\Theta,\, \bar{\boldsymbol{w}}_{L+1}} \ & J(\Theta, \bar{\boldsymbol{w}}_{L+1}; \mathcal{D}) \\ \text{s.t.} \quad & (5.6b). \end{aligned} \tag{5.9}$$

Unfortunately, this optimization problem poses significant challenges in practice. The computation of $\bar{\boldsymbol{w}}_{L+1}$ in (5.6b) requires the inverse of the precision matrix (5.6c) which depends on the optimization variables Θ. The inverse operation is numerically challenging and computationally expensive and, therefore, prohibits sufficiently fast

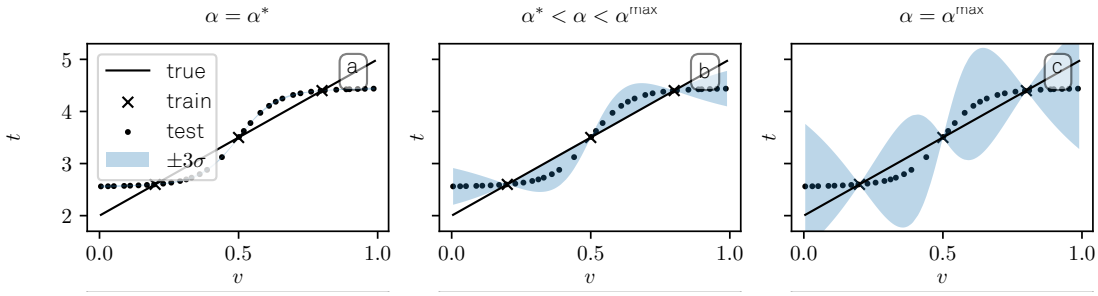

Figure 5.1: Neural network with Bayesian last layer: Predicted mean and standard deviation (5.7) after training on $m = 3$ samples. The effect of parameter α on the predicted uncertainty is shown by comparing the optimal α^* (Θ^* from (5.9) and α according to (5.13)) with suggested improved α^{max} [43, Figure 2].

optimization. Previous works on neural networks with Bayesian last layer have circumvented problem (5.9). In [18], [22], [47], [82], [83] the neural network is trained by minimizing the mean-squared-error (2.10) yielding $\mathbb{W}_{L+1}$. The noise and prior variances are either assumed to be known [22], [47], [82] or can be obtained by solving (5.9) with fixed parameters $\mathbb{W}_L$ [18], [83]. Subsequently, the distribution of the neural network with Bayesian last layer (5.7) can be evaluated.

As main contribution of our work [43], we present the following theorem which allows to directly solve (5.9).

Theorem 5.1. *The optimal solution* $(\Theta^*, \bar{\boldsymbol{w}}^*_{L+1})$ *of problem* (5.9) *is identical to the optimal solution obtained from the unconstrained problem:*

$$\Theta^*,\ \bar{\boldsymbol{w}}^*_{L+1} = \min_{\Theta, \bar{\boldsymbol{w}}} \quad J(\Theta, \bar{\boldsymbol{w}}_{L+1}; \mathcal{D}), \tag{5.10}$$

Where, in comparison to (5.9)*, the equality constraint has been dropped.*

Proof. The proof is shown in our work [43, Theorem 1]. □

Theorem 5.1 therefore enables the efficient maximization of the log marginal likelihood by avoiding the explicit computation of $\bar{\boldsymbol{w}}$ in (5.6b). Solving (5.10) yields the parameters $\mathbb{W}_{L+1}$ and the unknown noise and prior variances.

5.1.2 Improving the extrapolative uncertainty quantification

An important challenge with data-based probabilistic models is their predictive behavior in the extrapolation regime. The probabilistic model should indicate extrapolation by predicting an increased variance. However, it is difficult to achieve this behavior during training because the training data, by definition, does not contain extrapolation points.

To address this challenge, and as another main contribution of our work [43], a method to tune the extrapolative uncertainty quantification of a trained neural

network with Bayesian last layer is presented in the following. As a preliminary step, the precision matrix in (5.6c) is reformulated as:

$$\begin{aligned}\boldsymbol{\Lambda}_p &= \sigma_e^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma_w^{-2}\tilde{\boldsymbol{I}}_{n_\phi} \\ &= \sigma_e^{-2}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \alpha^{-1}\tilde{\boldsymbol{I}}_{n_\phi}\right) = \sigma_e^{-2}\bar{\boldsymbol{\Lambda}}_p,\end{aligned} \tag{5.11}$$

where [43, Result 2, Assumption 4]:

$$\bar{\boldsymbol{\Lambda}}_p = \boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \alpha^{-1}\tilde{\boldsymbol{I}}_{n_\phi}, \tag{5.12}$$

$$\alpha = \sigma_w^2/\sigma_e^2, \tag{5.13}$$

and $\tilde{\boldsymbol{I}}_{n_\phi} = \mathrm{diag}(\boldsymbol{I}_{n_\phi}, 0)$. It is shown in our work [43, Theorem 2] that the introduced parameter α directly influences the extrapolative uncertainty quantification of the neural network with Bayesian last layer. Leveraging this relationship, we propose in [43] to tune α after training the neural network with (5.9). In particular, the selected α should maximize the log-predictive density, that is:

$$\log\bar{p}(\boldsymbol{t}^{\text{test}}) = \frac{1}{m_{\text{test}}}\sum_{i=1}^{m_{\text{test}}}\log p(\mathrm{t} = t_i^{\text{test}}|\mathcal{D}, \Theta, \boldsymbol{x}^{\text{test}}), \tag{5.14}$$

on additional test data. The log-predictive density is an expressive measure for the quality of the identified probabilistic model. High values indicate high confidence in correct predictions, whereas low values indicate high confidence in incorrect predictions.

The effect of increasing α, relative to the optimal value obtained from (5.9), is shown in Figure 5.1. For this toy-example, a simple regression task with $m = 3$ data points is designed to provoke extrapolation. The training samples, the true function and the predicted distribution of the neural network with Bayesian last layer are depicted in the figure. Figure 5.1 a) shows the predicted distribution obtained with the optimal parameters from (5.9). Considering the sparse training data, the neural network predicts suitable mean values. However, the predicted standard deviations reveal that the model is overconfident. Increasing α, as shown in Figure 5.1 b) & c) remedies this shortcoming and achieves the desired performance. The final value of α^{max} is obtained by maximizing (5.14) [43, Algorithm 1].

5.1.3 Simulation study

The proposed algorithm to train and tune neural networks with Bayesian last layer is presented in a simulation study. The investigated problem constitutes a nonlinear regression task with $n_v = 1$ input, $n_t = 2$ outputs and additive noise with variances $\sigma_{e,1} = 0.05$ and $\sigma_{e,2} = 0.2$. Both noise variances are assumed to be unknown. A neural network with Bayesian last layer, configured with $L = 2$ layers, $n_{a,i} = 20$ neurons and activation function $g_i(\cdot) = \tanh(\cdot)$, $\forall i \in [1, 2, 3]$, is investigated to approximate the unknown function. Training is conducted as shown in [43, Algorithm 1], that is, the multivariate log-marginal likelihood (5.9) is maximized, yielding Θ^*. In the second step, α^*, obtained from (5.13), is tuned with additional data to improve the extrapolative uncertainty quantification. The log-predictive density (5.14) is maximized for α^{max}.

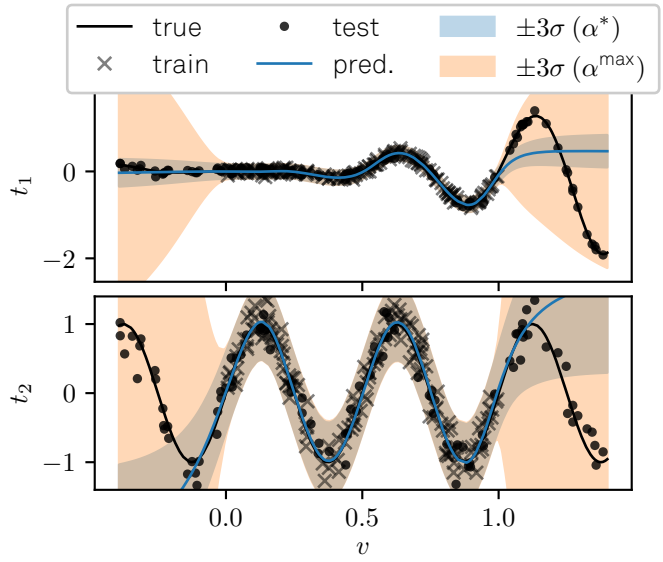

Figure 5.2: Multivariate neural network with Bayesian last layer: Predicted mean and standard deviation for two outputs with different and unknown noise level. The optimal parameters Θ^* are obtained with (5.10) and yield α^* with (5.13). This value is then adapted to improve the uncertainty quantification in the extrapolation regime by maximizing (5.14), yielding $\alpha^{\max}$ [43, Figure 4].

The results of the simulation study are shown in Figure 5.2. For the optimal value α^* and the tuned value $\alpha^{\max}$, the predicted distribution is depicted in terms of mean and standard deviation. Both distributions suitably describe the training data and capture the behavior of the unknown function in the interpolation regime. The estimated noise variances $\sigma_{e,1} = 0.051$ and $\sigma_{e,2} = 0.17$ are close to the true values. However, only the distribution with tuned extrapolative uncertainty quantification, that is, with $\alpha^{\max}$, is suitable to describe the extrapolation regime.

5.2 Applications & conclusions

Neural networks with Bayesian last layer are a tractable approach for uncertainty quantification with neural networks. With the proposed algorithm, which is based on maximizing the marginal likelihood and tuning the extrapolative uncertainty quantification, they are a promising solution for safe data-based control applications. In this section, two potential applications of the proposed method are discussed and may be investigated in future works.

5.2.1 State-space identification for stochastic MPC

Neural networks with Bayesian last layer can be used as a straightforward extension of the identification approach outlined in Chapter 2. Instead of the deterministic

state-space model in (2.11), a neural network with Bayesian last layer (5.7) yields the following Gaussian distribution:

$$p(\hat{\mathbf{x}}_{k+1}|\boldsymbol{x}_k, \boldsymbol{u}_k) = \mathcal{N}\left(\bar{\boldsymbol{x}}_{k+1}, \boldsymbol{\Sigma}_{x,k+1}\right). \tag{5.15}$$

To obtain a safe data-based predictive controller, this distribution may be incorporated in a stochastic optimal control problem (4.23). The main challenge in this regard is to obtain the required multi-step prediction. Unfortunately, recursively evaluating (5.15) does not warrant an analytical expression for the resulting distribution.

As an approximation, a tractable multi-step prediction can be obtained from (5.15) by propagating the uncertainty with a linearized model, in a manner similar to that of an *extended Kalman filter* [8, Sec. 4.5.2]. This approach has previously been applied in the literature for Gaussian processes [20], [21] and yields a Gaussian distribution for the multi-step prediction. The resulting distribution is thus suitable to formulate the stochastic optimal control problem in (4.24). However, propagating the uncertainty through linearization has two drawbacks. First, the obtained multi-step distribution is only an approximation and a derived stochastic MPC controller cannot guarantee safe control actions. Second, the linearization requires sequential backpropagation and, therefore, yields a nested computational graph that might impede sufficiently fast optimization.

In addition to the challenges introduced by linearization, an identified and recursively evaluated state-space model may be biased in the presence of measurement noise, as discussed in Section 4.2.3. This last challenge is not present, however, when approximating a non control-oriented model, as is the case for the water distribution network in Chapter 2. In this setting, probabilistic state-space identification with neural networks with Bayesian last layer may be a suitable solution.

5.2.2 Multi-step identification for stochastic MPC

Instead of identifying a nonlinear state-space model, neural networks with Bayesian last layer may also be used to directly identify a probabilistic multi-step model. In this setting, the neural network with Bayesian last layer (5.7) yields the following distribution:

$$p(\hat{\mathbf{x}}_{[k+1,k+N]}|\boldsymbol{x}_k, \boldsymbol{u}_{[k,k+N-1]}) = \mathcal{N}\left(\bar{\boldsymbol{x}}_{[k+1,k+N]}, \boldsymbol{\Sigma}_{x,[k+1,k+N]})\right). \tag{5.16}$$

The resulting distribution is directly suitable to formulate a stochastic optimal control problem (4.23) and can be reformulated to the deterministic problem (4.24).

Multi-step identification may have advantages over the previously discussed state-space identification because it avoids the challenge of propagating the uncertainty. Furthermore, some advantages of multi-step models over state-space models, discussed in Section 4.2.3, may also apply in the nonlinear setting. However, there are also specific challenges associated with nonlinear multi-step identification. Most importantly, a multi-step model has a larger input space than a state-space model for the same system. Consequently, a neural network with significantly more parameters may be required to identify the multi-step model. At the same time, fewer data samples are available for training. A sequence of length $N+1$ yields only one sample to train a multi-step model with a horizon of N. In contrast, the same sequence yields

N samples to identify a state-space model. With overlapping elements from a longer sequence this disadvantage is mitigated to some extent. However, samples from overlapping sequences are correlated and may result in biased estimates of the model parameters. Another solution to mitigate these challenges is to train N independent models for the N steps of the prediction horizon. The first model then corresponds to the state-space model in (5.15). With this approach, the described challenges become significant only for models that predict the final steps of the sequence.

In conclusion, both state-space and multi-step models using neural networks with Bayesian last layer are promising solutions to enable safe data-based predictive control. Both approaches have distinct advantages and disadvantages. For future works, it remains to investigate and compare their performance for concrete control tasks.

Chapter 6

Conclusions and outlook

This thesis explores the integration of model predictive control with advanced methods for system identification, specifically focusing on nonlinear identification and multi-step identification with deterministic and probabilistic models. This chapter summarizes the proposed methods, presents advantages and limitations and discusses their potential to facilitate the widespread adoption of model predictive control. Further steps towards achieving this goal are presented in the outlook.

6.1 Conclusions

Neural networks are an attractive model type for nonlinear state-space system identification. These models can learn complex system behavior from large datasets and are suitable for gradient-based optimization, as required for efficient model predictive control. Data for the identification task can either stem from measurements of an existing system or from simulation results of a non control-oriented model. Neural network system identification for systems with state-feedback and output-feedback is proposed and demonstrated in two simulation studies. It is shown that, with suitably sampled data, an MPC controller with neural network system model can yield near-identical results to a variant with exact system model for a wide range of operating conditions. Sampling data for the identification task, and formulating the MPC controller is facilitated by the introduced open-source software package do-mpc. In the second simulation study it is demonstrated that MPC with identified neural network system model can be employed for the energy-optimal control of a water distribution network. To reduce the complexity of the neural network and the resulting optimal control problem, neural network system identification is combined with a data-based hierarchical clustering approach. For the water distribution network, the proposed model predictive controller achieves significant improvements in energy efficiency compared to the default rule-based controller.

An alternative to state-space identification is multi-step identification. Multi-step models are derived for linear systems by condensing the recursive evaluation of a state-space model. It is shown that multi-step identification boils down to a tractable linear regression task, and the resulting model is directly suitable for the formulation of a model predictive control problem. The proposed controller with identified multi-step model is compared with the popular data-enabled predictive control method,

which combines the system identification and control tasks in a single optimization problem. As a main contribution, it is shown that with data from a deterministic system, both formulations yield equivalent results. Consequently, it is argued that data-enabled predictive control repeatedly estimates the multi-step parameters resulting in a significantly increased computational cost. The relationship between both methods becomes more intricate for non-deterministic systems as the data-enabled predictive control problem requires heuristic modifications to avoid infeasibility. For comparability, similar adaptations are proposed for the multi-step model predictive control problem and it is shown that both methods yield equivalent results under specific conditions. In the general case, both methods yield different optimal control actions and result in minor differences in control performance. These differences are compared in a simulation study where multi-step model predictive control slightly outperforms data-enabled predictive control in terms of the control objective, but significantly outperforms it in terms of computation time.

Multi-step identification is an attractive approach to derive a model predictive controller directly from data. However, in the presence of uncertainty, the proposed multi-step model predictive controller may result in unsafe behavior and constraint violations. As a solution, probabilistic multi-step identification for stochastic model predictive control is proposed. Stochastic model predictive control considers the distribution of the future behavior of the system, as obtained from a probabilistic multi-step model, and seeks to avoid constraints with a specified probability. Three major contributions are presented in this regard. First, a probabilistic multi-step model is derived by condensing a linear system with process and measurement noise. Importantly, the probabilistic multi-step model incorporates the estimated initial state distribution obtained from a Kalman filter. Second, a multi-step identification approach, based on maximum likelihood estimation, is proposed. It is shown that the identified model yields, in expectation, the true distributed multi-step prediction, including the state estimation task. Finally, a method to consider the parametric uncertainty of the identified multi-step model is presented. The probabilistic perspective also provides evidence for the inherent advantage of identified multi-step models over identified and recursively evaluated state-space models. These advantages also apply to data-enabled predictive control, which implicitly estimates a multi-step model.

Probabilistic multi-step identification enables the formulation of a data-based stochastic model predictive controller. The controller shows excellent performance and safe behavior in the presence of uncertainty in two simulation studies. In both studies, a comparison with a variant based on probabilistic state-space identification is presented, further demonstrating the advantages of multi-step identification. The proposed stochastic model predictive controller with identified multi-step model also achieves excellent performance for an investigated nonlinear system, despite not being originally derived for this context. The application to nonlinear systems is possible as the estimated covariance approximately incorporates the effect of the nonlinearities. In conclusion, probabilistic multi-step identification should be preferred over regular multi-step identification for nonlinear systems, even if the system is deterministic.

Combining probabilistic system identification with neural networks is an attractive extension of the proposed methods. As a prerequisite, neural networks with Bayesian

last layer are introduced in this thesis. They constitute a simplified Bayesian neural network and offer an attractive compromise between tractability and expressiveness of the predicted distribution. Two main contributions are presented to unlock the full potential of neural networks with Bayesian last layer. First, an efficient training algorithm based on a reformulation of the marginal likelihood is proposed. Maximizing the marginal likelihood enables joint training of weights and unknown noise variances, favoring the simplest model that adequately describes the data. Unfortunately, the marginal likelihood is ill-suited for gradient-based optimization. The proposed reformulation of the maximization problem overcomes this issue and satisfies the same conditions of optimality as the original formulation. Second, an algorithm to tune the extrapolative uncertainty quantification of the neural network with Bayesian last layer is proposed. This solves the challenge that the probabilistic model is trained, by definition, with data from the interpolation regime. Therefore, the predictive distribution may exhibit arbitrary and unsuitable behavior in the extrapolation regime. With the proposed algorithm, a single scalar parameter is tuned to mitigate this limitation. The effectiveness of the proposed method is demonstrated in a simulation study, where a multivariate nonlinear function with unknown noise variances is accurately recovered from data and where the predicted distribution correctly identifies high uncertainty in the extrapolation regime.

6.2 Outlook and future work

In this thesis, several contributions in the field of nonlinear state-space identification with neural networks, multi-step identification and probabilistic multi-step identification are presented. Using the identified models, nonlinear, economic and stochastic model predictive controllers can be synthesized directly from data and their performance is demonstrated in various case studies. Promising extensions for future work are presented in the following.

Neural networks for system identification hold the promise to identify complex nonlinear behavior from large quantities of data. However, with traditional neural networks, it is unclear when the model is extrapolating and how confident the predictions are in the extrapolation regime. To mitigate this limitation, neural networks with Bayesian last layer are introduced in this thesis as a tractable method for uncertainty quantification. A natural extension is the application of neural networks with Bayesian last layer for probabilistic nonlinear system identification and stochastic model predictive control. Two possible applications are outlined in this thesis and remain to be investigated in the future. Neural networks with Bayesian last layer can be used to identify nonlinear probabilistic state-space models. While the identification task is a straightforward application of the proposed method, it is challenging to successively formulate a stochastic model predictive controller. The recursive evaluation of the identified probabilistic state-space model does not yield an analytical distribution and suitable approximations may be investigated in future work. Alternatively, neural networks with Bayesian last layer may be employed to identify nonlinear probabilistic multi-step models. This application avoids the challenge of propagating the uncertainty but may require significantly larger neural networks, due to the higher dimensionality of the input space. A comparison between nonlinear multi-step identification

and nonlinear state-space identification using neural networks with Bayesian last layer is therefore an important task for future work.

Probabilistic linear multi-step identification for stochastic model predictive control presents further important research questions. The proposed method is derived for linear systems with noisy state-feedback. For systems with output-feedback, the identification task suffers from correlated measurement noise and introduces a bias. While this may be negligible in practice, as demonstrated in a case study, a rigorous method to treat this situation is desirable.

A further extension of probabilistic multi-step identification is to continuously update the identified model during the control application. This adaptive controller not only reacts to changing behavior of the system but may mitigate a further limitation of the proposed method. The implicit Kalman filter of the identified multi-step model does not include the prediction step, that is, an update of the prior state distribution. Instead, the prior state distribution corresponds to the sample distribution of the data used for the identification task. If this data is continuously updated, as proposed in the potential extension, the sample distribution is also updated accordingly.

In future works, it may also be investigated to combine the proposed extensions. For example, it is conceivable to update a nonlinear probabilistic multi-step model continuously during the control application. This does not necessarily imply that the entire neural network is retrained. Instead, the nonlinear feature function provided by the neural network may be kept unchanged and only the linear model of the Bayesian last layer is updated.

Apart from further methodological advancements, it is also desirable to investigate the performance of the proposed methods in larger-scale systems and for real-world applications.

Bibliography

[1] H. Lee, K. Calvin, D. Dasgupta, *et al.*, *IPCC, 2023: Climate Change 2023: Synthesis Report, Summary for Policymakers. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Geneva, Switzerland: IPCC, 2023.

[2] P. Shukla, J. Skea, R. Slade, *et al.*, *IPCC, 2022: Climate Change 2022: Mitigation of Climate Change. Contribution of Working Group III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. United Kingdom: Cambridge University Press, 2022.

[3] R. Madurai Elavarasan, R. Pugazhendhi, M. Irfan, L. Mihet-Popa, I. A. Khan, and P. E. Campana, "State-of-the-art sustainable approaches for deeper decarbonization in Europe – An endowment to climate neutral vision", *Renewable and Sustainable Energy Reviews*, vol. 159, p. 112 204, May 2022.

[4] D. Blum, Z. Wang, C. Weyandt, D. Kim, M. Wetter, T. Hong, and M. A. Piette, "Field demonstration and implementation analysis of model predictive control in an office HVAC system", *Applied Energy*, vol. 318, p. 119 104, 2022.

[5] M. G. Forbes, R. S. Patwardhan, H. Hamadah, and R. B. Gopaluni, "Model Predictive Control in Industry: Challenges and Opportunities", *IFAC-PapersOnLine*, In 9th IFAC Symposium on Advanced Control of Chemical Processes, vol. 48, no. 8, pp. 531–538, Jan. 1, 2015.

[6] S. Vazquez, J. I. Leon, L. G. Franquelo, J. Rodriguez, H. A. Young, A. Marquez, and P. Zanchetta, "Model Predictive Control: A Review of Its Applications in Power Electronics", *IEEE Industrial Electronics Magazine*, vol. 8, no. 1, pp. 16–31, Mar. 2014.

[7] S. Kouro, M. A. Perez, J. Rodriguez, A. M. Llor, and H. A. Young, "Model Predictive Control: MPC's Role in the Evolution of Power Electronics", *IEEE Industrial Electronics Magazine*, vol. 9, no. 4, pp. 8–21, Dec. 2015.

[8] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd edition. Madison, WI, USA: Nob Hill Publishing, 2017.

[9] J. B. Rawlings, D. Angeli, and C. N. Bates, "Fundamentals of economic model predictive control", in *IEEE Conference on Decision and Control*, 2012, pp. 3851–3861.

[10] M. Ellis, J. Liu, and P. D. Christofides, *Economic Model Predictive Control* (Advances in Industrial Control). Switzerland: Springer International Publishing, 2017.

[11] M. Morari and J. H. Lee, "Model predictive control: Past, present and future", *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, May 1999.

[12] F. Fiedler, B. Karg, L. Lüken, D. Brandner, M. Heinlein, F. Brabender, and S. Lucia, "Do-mpc: Towards FAIR nonlinear and robust model predictive control", *Control Engineering Practice*, vol. 140, p. 105 676, Nov. 1, 2023.

[13] A. Mesbah, "Stochastic Model Predictive Control: An Overview and Perspectives for Future Research", *IEEE Control Systems*, vol. 36, no. 6, pp. 30–44, Dec. 2016.

[14] M. Farina, L. Giulioni, and R. Scattolini, "Stochastic linear model predictive control with chance constraints–a review", *Journal of Process Control*, vol. 44, pp. 53–67, 2016.

[15] L. Ljung, "Perspectives on system identification", *Annual Reviews in Control*, vol. 34, no. 1, pp. 1–12, 2010.

[16] K. J. Åström and P. Eykhoff, "System identification—A survey", *Automatica*, vol. 7, no. 2, pp. 123–162, Mar. 1, 1971.

[17] L. Ljung, *System Identification: Theory for the User*. NJ, USA: Prentice Hall PTR, 1999.

[18] C. D. McKinnon and A. P. Schoellig, "Meta learning with paired forward and inverse models for efficient receding horizon control", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3240–3247, 2021.

[19] J. Dong, "Robust Data-Driven Iterative Learning Control for Linear-Time-Invariant and Hammerstein-Wiener Systems", *IEEE Transactions on Cybernetics*, pp. 1–14, 2021.

[20] C. D. McKinnon and A. P. Schoellig, "Learning Probabilistic Models for Safe Predictive Control in Unknown Environments", presented at the European Control Conference, Jun. 2019, pp. 2472–2479.

[21] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious Model Predictive Control Using Gaussian Process Regression", *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, Nov. 2020.

[22] K. P. Wabersich and M. N. Zeilinger, "Nonlinear learning-based model predictive control supporting state and input dependent model uncertainty estimates", *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8897–8915, 2021.

[23] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön, "Deep Learning and System Identification", *IFAC-PapersOnLine*, In 21st IFAC World Congress, vol. 53, no. 2, pp. 1175–1181, Jan. 1, 2020.

[24] Y. Wang, "A new concept using LSTM Neural Networks for dynamic system identification", in *American Control Conference*, May 2017, pp. 5324–5329.

[25] O. Nelles, *Nonlinear Dynamic System Identification*. Springer, 2020.

[26] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, "Hands-On Bayesian Neural Networks—A Tutorial for Deep Learning Users", *IEEE Computational Intelligence Magazine*, vol. 17, no. 2, pp. 29–48, 2022.

[27] W. Favoreel, B. De Moor, and M. Gevers, "SPC: Subspace predictive control", *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 4004–4009, 1999.

[28] E. Terzi, L. Fagiano, M. Farina, and R. Scattolini, "Learning-based predictive control for linear systems: A unitary approach", *Automatica*, vol. 108, p. 108 473, 2019.

[29] E. Terzi, M. Farina, L. Fagiano, and R. Scattolini, "Robust multi-rate predictive control using multi-step prediction models learned from data", *Automatica*, vol. 136, p. 109 852, 2022.

[30] J. Köhler, K. P. Wabersich, J. Berberich, and M. N. Zeilinger, "State space models vs. multi-step predictors in predictive control: Are state space models complicating safe data-driven designs?", in *IEEE Conference on Decision and Control*, Cancun, Mexico, Dec. 2022, pp. 491–498.

[31] J. Coulson, J. Lygeros, and F. Dörfler, "Data-enabled predictive control: In the shallows of the DeePC", in *European Control Conference*, 2019, pp. 307–312.

[32] I. Markovsky and F. Dörfler, "Behavioral systems theory in data-driven analysis, signal processing, and control", *Annual Reviews in Control*, vol. 52, pp. 42–64, 2021.

[33] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "On the design of terminal ingredients for data-driven MPC", *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 257–263, 2021.

[34] I. Markovsky, L. Huang, and F. Dörfler, "Data-Driven Control Based on the Behavioral Approach: From Theory to Applications in Power Systems", *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 28–68, Oct. 2023.

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

[36] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Linear Tracking MPC for Nonlinear Systems—Part II: The Data-Driven Case", *IEEE Transactions on Automatic Control*, vol. 67, no. 9, pp. 4406–4421, Sep. 2022.

[37] D. Reinhardt and S. Gros, "Data-Driven Predictive Control for Nonlinear Systems Using Feature Selection", in *IEEE Conference on Decision and Control*, Singapore, Singapore, Dec. 2023, pp. 2576–2583.

[38] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, "A note on persistency of excitation", *Systems & Control Letters*, vol. 54, no. 4, pp. 325–329, 2005.

[39] J. Coulson, J. Lygeros, and F. Dörfler, "Distributionally robust chance constrained data-enabled predictive control", *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3289–3304, 2021.

[40] F. Fiedler, A. Cominola, and S. Lucia, "Economic nonlinear predictive control of water distribution networks based on surrogate modeling and automatic clustering", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 16 636–16 643, 2020.

[41] F. Fiedler and S. Lucia, "On the relationship between data-enabled predictive control and subspace predictive control", in *European Control Conference*, 2021, pp. 222–229.

[42] F. Fiedler and S. Lucia, "Probabilistic Multi-Step Identification With Implicit State Estimation for Stochastic MPC", *IEEE Access*, vol. 11, pp. 117 018–117 029, 2023.

[43] F. Fiedler and S. Lucia, "Improved uncertainty quantification for neural networks with Bayesian last layer", *IEEE Access*, vol. 11, pp. 123 149–123 160, 2023.

[44] M. Abdar, F. Pourpanah, S. Hussain, *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges", *Information Fusion*, vol. 76, pp. 243–297, 2021.

[45] M. Lázaro-Gredilla and A. R. Figueiras-Vidal, "Marginalized neural network mixtures for large-scale regression", *IEEE Transactions on Neural Networks*, vol. 21, no. 8, pp. 1345–1351, 2010.

[46] J. Watson, J. A. Lin, P. Klink, J. Pajarinen, and J. Peters, "Latent Derivative Bayesian Last Layer Networks", presented at the International Conference on Artificial Intelligence and Statistics, 2021, pp. 1198–1206.

[47] F. Fiedler and S. Lucia, "Model predictive control with neural network system model and Bayesian last layer trust regions", in *IEEE International Conference on Control & Automation*, Jun. 2022, pp. 141–147.

[48] C. Döpmann, F. Fiedler, S. Lucia, and F. Tschorsch, "Optimization-Based Predictive Congestion Control for the Tor Network: Opportunities and Challenges", *ACM Transactions on Internet Technology*, vol. 22, no. 4, pp. 1–30, Nov. 30, 2022.

[49] P. Guillén, F. Fiedler, H. Sarnago, S. Lucía, and O. Lucía, "Deep Learning Implementation of Model Predictive Control for Multioutput Resonant Converters", *IEEE Access*, vol. 10, pp. 65 228–65 237, 2022.

[50] C. Döpmann, F. Fiedler, S. Lucia, and F. Tschorsch, "Towards Optimization-Based Predictive Congestion Control for the Tor Network", *Electronic Communications of the EASST*, vol. 80, 2021.

[51] F. Fiedler, C. Döpmann, F. Tschorsch, and S. Lucia, "PredicTor: Predictive Congestion Control for the Tor Network", in *IEEE Conference on Control Technology and Applications*, Aug. 2020, pp. 863–870.

[52] N. Krausch, S. Hans, F. Fiedler, S. Lucia, P. Neubauer, and M. N. C. Bournazou, "From screening to production: A holistic approach of high-throughput model-based screening for recombinant protein production", in *Computer Aided Chemical Engineering*, vol. 48, Elsevier, 2020, pp. 1723–1728.

[53] F. Fiedler, D. Baumbach, A. Börner, and S. Lucia, "A Probabilistic Moving Horizon Estimation Framework Applied to the Visual-Inertial Sensor Fusion Problem", in *European Control Conference*, May 2020, pp. 1009–1016.

[54] F. Bonassi, M. Farina, and R. Scattolini, "Stability of discrete-time feed-forward neural networks in NARX configuration", *IFAC-PapersOnLine*, 19th IFAC Symposium on System Identification, vol. 54, no. 7, pp. 547–552, Jan. 1, 2021.

[55] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control", *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[56] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming", *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[57] S. Lucia, T. Finkler, and S. Engell, "Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty", *Journal of Process Control*, vol. 23, no. 9, pp. 1306–1319, Oct. 1, 2013.

[58] "Rapid Control Prototyping", in *Rapid Control Prototyping: Methoden und Anwendungen*, D. Abel and A. Bollig, Eds., Berlin, Heidelberg: Springer, 2006, pp. 295–318.

[59] M. Schleipen, S.-S. Gilani, T. Bischoff, and J. Pfrommer, "OPC UA & Industrie 4.0 - Enabling Technology with High Diversity and Variability", *Procedia CIRP*, vol. 57, pp. 315–320, Jan. 1, 2016.

[60] M. Abadi, A. Agarwal, P. Barham, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems", 2016. arXiv: 1603.04467.

[61] K. Klatt and S. Engell, "Gain-scheduling trajectory control of a continuous stirred tank reactor", *Computers & Chemical Engineering*, vol. 22, no. 4, pp. 491–502, Jan. 1, 1998.

[62] C. M. Chini and A. S. Stillwell, "The State of U.S. Urban Water: Data and the Energy-Water Nexus", *Water Resources Research*, vol. 54, no. 3, pp. 1796–1811, Mar. 2018.

[63] E. S. Spang and F. J. Loge, "A High-Resolution Approach to Mapping Energy Flows through Water Infrastructure Systems", *Journal of Industrial Ecology*, vol. 19, no. 4, pp. 656–665, Aug. 2015.

[64] C. Biscos, M. Mulholland, M. V. Le Lann, C. A. Buckley, and C. J. Brouckaert, "Optimal operation of water distribution networks by predictive control using MINLP", *Water SA*, vol. 29, no. 4, pp. 393–404, 2003.

[65] G. Cembrano, J. Quevedo, V. Puig, *et al.*, "PLIO: A generic tool for real-time operational predictive optimal control of water networks", *Water Science and Technology*, vol. 64, no. 2, pp. 448–459, 2011.

[66] C. Ocampo-Martinez, D. Barcelli, V. Puig, and A. Bemporad, "Hierarchical and decentralised model predictive control of drinking water networks: Application to Barcelona case study", *IET Control Theory & Applications*, vol. 6, no. 1, p. 62, 2012.

[67] A. Ostfeld, E. Salomons, L. Ormsbee, *et al.*, "Battle of the Water Calibration Networks", *Journal of Water Resources Planning and Management*, vol. 138, no. 5, pp. 523–532, Sep. 2012.

[68] L. A. Rossman, "EPANET 2: Users manual", 2000.

[69] D. R. Broad, G. C. Dandy, and H. R. Maier, "Water Distribution System Optimization Using Metamodels", *Journal of Water Resources Planning and Management*, vol. 131, no. 3, pp. 172–180, May 2005.

[70] Y. Wang, V. Puig, and G. Cembrano, "Non-linear economic model predictive control of water distribution networks", *Journal of Process Control*, vol. 56, pp. 23–34, 2017.

[71] J. H. Ward, "Hierarchical Grouping to Optimize an Objective Function", *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244, Mar. 1963.

[72] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[73] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006, 738 pp.

[74] G. A. F. Seber, *A Matrix Handbook for Statisticians*. John Wiley & Sons, 2008.

[75] S. J. Qin, "An overview of subspace identification", *Computers & Chemical Engineering*, Papers Form Chemical Process Control VII, vol. 30, no. 10, pp. 1502–1513, Sep. 12, 2006.

[76] W. A. Fuller, *Measurement Error Models*. John Wiley & Sons, 2009.

[77] J. P. Buonaccorsi, *Measurement Error: Models, Methods, and Applications*. CRC press, 2010.

[78] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[79] L. Blackmore, M. Ono, and B. C. Williams, "Chance-Constrained Optimal Path Planning With Obstacles", *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, Dec. 2011.

[80] B. Bueno, L. Norford, G. Pigeon, and R. Britter, "A resistance-capacitance network model for the analysis of the interactions between the energy performance of buildings and the urban climate", *Building and Environment*, vol. 54, pp. 116–125, Aug. 2012.

[81] L. Hewing, K. P. Wabersich, and M. N. Zeilinger, "Recursively feasible stochastic model predictive control using indirect feedback", *Automatica*, vol. 119, p. 109 095, Sep. 1, 2020.

[82] J. Harrison, A. Sharma, R. Calandra, and M. Pavone, "Control adaptation via meta-learning dynamics", in *NeurIPS*, 2018.

[83] J. Snoek, O. Rippel, K. Swersky, *et al.*, "Scalable bayesian optimization using deep neural networks", in *International Conference on Machine Learning*, PMLR, 2015, pp. 2171–2180.

Appendix A

Proofs and derivations

A.1 Proof of Lemma 3.2: Multi-step model

For ease of notation, the symbols $\mathcal{O}_t$ and $\mathcal{T}_t$ are used to denote $\mathcal{O}_t(\boldsymbol{A}, \boldsymbol{C})$ and $\mathcal{T}_t(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ in the following. To recover the initial state of system (3.1) from finite sequences of past measurements and inputs, the multi-step model (3.4) for a sequence of length t, according to Assumption 3.1, is rearranged as follows:

$$\begin{aligned} \mathcal{O}_t \boldsymbol{x}_0 &= \begin{bmatrix} \boldsymbol{I} & -\mathcal{T}_t \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,t-1]} \end{bmatrix} \\ \Leftrightarrow \; \boldsymbol{x}_0 &= \mathcal{O}_t^\dagger \begin{bmatrix} \boldsymbol{I} & -\mathcal{T}_t \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,t-1]} \end{bmatrix}. \end{aligned} \tag{A.1}$$

The recovered initial state $\boldsymbol{x}_0$ is unique as $\mathcal{O}_t$ has full rank, due to Assumption 3.1. To predict the future sequences from the recovered initial state, (A.1) is substituted into (3.4) with $L = t + N$ and $N > 0$:

$$\boldsymbol{y}_{[0,L-1]} = \mathcal{O}_L \mathcal{O}_t^\dagger \begin{bmatrix} \boldsymbol{I} & -\mathcal{T}_t \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,t-1]} \end{bmatrix} + \mathcal{T}_L \boldsymbol{u}_{[0,L-1]}. \tag{A.2}$$

To obtain only the future outputs of the system, that is, $\boldsymbol{y}_{[t,L-1]}$, (A.2) is reformulated, by partitioning $\mathcal{O}_L$ and $\mathcal{T}_L$ (note that $L = t + N$):

$$\mathcal{O}_L = \begin{bmatrix} \mathcal{O}_t \\ \mathcal{O}_N \boldsymbol{A}^t \end{bmatrix}, \quad \mathcal{T}_L = \begin{bmatrix} \mathcal{T}_t & \boldsymbol{0} \\ \mathcal{T}_t^N & \mathcal{T}_N \end{bmatrix}. \tag{A.3}$$

Substituting (A.3) in (A.2) yields:

$$\begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{y}_{[t,L-1]} \end{bmatrix} = \begin{bmatrix} \mathcal{O}_t \mathcal{O}_t^\dagger \\ \mathcal{O}_N \boldsymbol{A}^t \mathcal{O}_t^\dagger \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & -\mathcal{T}_t \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,t-1]} \end{bmatrix} + \begin{bmatrix} \mathcal{T}_t & \boldsymbol{0} \\ \mathcal{T}_t^N & \mathcal{T}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_{[0,t-1]} \\ \boldsymbol{u}_{[t,L-1]} \end{bmatrix}. \tag{A.4}$$

With t larger than or equal to the system lag, it holds that $\mathcal{O}_t\mathcal{O}_t^\dagger = \boldsymbol{I}$. Considering this equality and multiplying the matrices in (A.4) yields:

$$\begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{y}_{[t,L-1]} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & -\mathcal{T}_t \\ \mathcal{O}_N \boldsymbol{A}^t \mathcal{O}_t^\dagger & -\mathcal{O}_N \boldsymbol{A}^t \mathcal{O}_t^\dagger \mathcal{T}_t \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,t-1]} \end{bmatrix} + \begin{bmatrix} \mathcal{T}_t & \boldsymbol{0} \\ \mathcal{T}_t^N & \mathcal{T}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_{[0,t-1]} \\ \boldsymbol{u}_{[t,L-1]} \end{bmatrix}. \quad \text{(A.5)}$$

The first block-row of the above equation yields the expected equality $\boldsymbol{y}_{[0,t-1]} = \boldsymbol{y}_{[0,t-1]}$. Importantly, the second block-row yields the desired relationship:

$$\boldsymbol{y}_{[t,L-1]} = \underbrace{\begin{bmatrix} \mathcal{O}_N \boldsymbol{A}^t \mathcal{O}_t^\dagger & -\mathcal{O}_N \boldsymbol{A}^t \mathcal{O}_t^\dagger \mathcal{T}_t + \mathcal{T}_t^N & \mathcal{T}_N \end{bmatrix}}_{\boldsymbol{W}} \begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,t-1]} \\ \boldsymbol{u}_{[t,L-1]} \end{bmatrix}, \quad \text{(A.6)}$$

and corresponds to (3.5), which concludes the proof.

A.2 Reformulation of a linear system with output-feedback

A linear dynamic system (3.1) with output-feedback can be reformulated to exhibit state-feedback, as shown in the following. If Assumption 3.1 holds, multi-step predictions of length N can be obtained with (3.5). As a special case, $N = 1$ yields:

$$\boldsymbol{y}_t = \boldsymbol{W} \begin{bmatrix} \boldsymbol{y}_{[0,t-1]} \\ \boldsymbol{u}_{[0,t-1]} \\ \boldsymbol{u}_t \end{bmatrix}. \tag{A.7}$$

The reformulated system representation considers the newly introduced state:

$$\tilde{\boldsymbol{x}}_t^\top = \begin{bmatrix} \boldsymbol{y}_{[0,t-1]}^\top & \boldsymbol{u}_{[0,t-1]}^\top \end{bmatrix}. \tag{A.8}$$

To obtain the next state $\tilde{\boldsymbol{x}}_{t+1}$, the matrix $\boldsymbol{W}$ is partitioned as:

$$\begin{aligned} \boldsymbol{W} &= \begin{bmatrix} \boldsymbol{W}_{y,0} & \dots & \boldsymbol{W}_{y,t-1} & \boldsymbol{W}_{u,0} & \dots & \boldsymbol{W}_{u,t-1} & \boldsymbol{W}_{u,t} \end{bmatrix}, \\ &= \begin{bmatrix} \boldsymbol{W}_y & \boldsymbol{W}_u \end{bmatrix}, \end{aligned} \tag{A.9}$$

and the system dynamics are formulated as:

$$\underbrace{\begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_{t-1} \\ \boldsymbol{y}_t \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{t-1} \\ \boldsymbol{u}_t \end{bmatrix}}_{\tilde{\boldsymbol{x}}_{t+1}} = \underbrace{\begin{bmatrix} 0 & \boldsymbol{I} & 0 & & & \dots & & 0 \\ \vdots & & \ddots & & & & & \vdots \\ 0 & \dots & 0 & \boldsymbol{I} & 0 & \dots & & 0 \\ \boldsymbol{W}_{y,0} & \dots & \dots & \boldsymbol{W}_{y,t-1} & \boldsymbol{W}_{u,0} & \dots & \dots & \boldsymbol{W}_{u,t-1} \\ 0 & & \dots & & 0 & \boldsymbol{I} & 0 & 0 \\ \vdots & & & & & & \ddots & 0 \\ & & & & & & & \boldsymbol{I} \\ 0 & & & \dots & & & & 0 \end{bmatrix}}_{\tilde{\boldsymbol{A}}} \underbrace{\begin{bmatrix} \boldsymbol{y}_0 \\ \vdots \\ \boldsymbol{y}_{t-2} \\ \boldsymbol{y}_{t-1} \\ \boldsymbol{u}_0 \\ \vdots \\ \boldsymbol{u}_{t-2} \\ \boldsymbol{u}_{t-1} \end{bmatrix}}_{\tilde{\boldsymbol{x}}_t} + \underbrace{\begin{bmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{W}_{u,t} \\ 0 \\ \vdots \\ 0 \\ \boldsymbol{I} \end{bmatrix}}_{\tilde{\boldsymbol{B}}} \boldsymbol{u}_t.$$

In summary, the equivalent representation of the dynamic system (3.1) is written as:

$$\tilde{\boldsymbol{x}}_{t+1} = \tilde{\boldsymbol{A}}(\boldsymbol{W}_y, \boldsymbol{W}_u)\tilde{\boldsymbol{x}}_t + \tilde{\boldsymbol{B}}(\boldsymbol{W}_u)\boldsymbol{u}_t. \tag{A.10a}$$

A.3 Proof of Corollary 4.1: Probabilistic state-space model

As a special case for $N = 1$, the multi-step model in the form of (4.10) describes the distribution:

$$p(\hat{\mathbf{y}}_1|\boldsymbol{y}_0, \boldsymbol{u}_0) = \mathcal{N}(\mathcal{O}_1^+(\boldsymbol{A})\boldsymbol{L}\boldsymbol{y}_0 + \mathcal{T}_1^+(\boldsymbol{A}, \boldsymbol{B})\boldsymbol{u}_0, \hat{\boldsymbol{\Sigma}}_{y,1}),$$

Considering (4.4), that is:

$$\mathcal{O}_1^+(\boldsymbol{A}) = \boldsymbol{A}, \quad \mathcal{T}_1^+(\boldsymbol{A}, \boldsymbol{B}) = \boldsymbol{B}, \tag{A.11}$$

we obtain:

$$p(\hat{\mathbf{y}}_1|\boldsymbol{y}_0, \boldsymbol{u}_0) = \mathcal{N}(\boldsymbol{A}\boldsymbol{L}\boldsymbol{y}_0 + \boldsymbol{B}\boldsymbol{u}_0, \hat{\boldsymbol{\Sigma}}_{y,1}). \tag{A.12}$$

Considering (A.11), the parameter matrix $\hat{\boldsymbol{W}}$, introduced in (4.11), is:

$$\hat{\boldsymbol{W}} = \begin{bmatrix} \boldsymbol{A}\boldsymbol{L} & \boldsymbol{B} \end{bmatrix}. \tag{A.13}$$

Furthermore, it follows from (A.11) that the covariance matrix $\hat{\boldsymbol{\Sigma}}_{y,1}$, defined in (4.10c), is:

$$\hat{\boldsymbol{\Sigma}}_{y,1} = \boldsymbol{\Sigma}_{y,1} + \boldsymbol{A}(\boldsymbol{\Sigma}_{x,0} + \boldsymbol{L}\boldsymbol{\Sigma}_{x,0})\boldsymbol{A}^\top, \tag{A.14}$$

Substituting $\boldsymbol{\Sigma}_{y,1}$ from (4.7c), that is:

$$\boldsymbol{\Sigma}_{y,1} = \boldsymbol{E}_x\boldsymbol{\Sigma}_x\boldsymbol{E}_x^\top + \boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top, \tag{A.15}$$

in (A.14) yields:

$$\hat{\boldsymbol{\Sigma}}_{y,1} = \boldsymbol{E}_x\boldsymbol{\Sigma}_x\boldsymbol{E}_x^\top + \boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top + \boldsymbol{A}(\boldsymbol{\Sigma}_{x,0} + \boldsymbol{L}\boldsymbol{\Sigma}_{x,0})\boldsymbol{A}^\top. \tag{A.16}$$

According to Theorem 4.1, we obtain the following expected values of the estimated parameters $\hat{\boldsymbol{W}}^*$ and covariance $\hat{\boldsymbol{\Sigma}}_{y,1}^*$:

$$\mathbb{E}[\hat{\boldsymbol{W}}^*] = \hat{\boldsymbol{W}}, \tag{A.17a}$$

$$\mathbb{E}[\hat{\boldsymbol{\Sigma}}_{y,1}^*] = \hat{\boldsymbol{\Sigma}}_{y,1}. \tag{A.17b}$$

Considering the true parameter matrix $\hat{\boldsymbol{W}}$ (A.13) and covariance matrix $\hat{\boldsymbol{\Sigma}}_{y,1}$ (A.16), yields the equality in (4.19). The identified probabilistic state-space model in (4.20) with the expected values of the estimated parameters (4.19) is equivalent to (A.12), which concludes the proof.

Appendix B

Investigated systems

B.1 Continuously stirred tank reactor

The continuously stirred tank reactor (CSTR) in Figure 2.1 is described by the ordinary differential equations:

$$\dot{c}_{\mathrm{A}} = F \cdot (c_{\mathrm{A},0} - c_{\mathrm{A}}) - k_1 \cdot c_{\mathrm{A}} - k_3 \cdot c_{\mathrm{A}}^2, \tag{B.1a}$$

$$\dot{c}_{\mathrm{B}} = -F \cdot c_{\mathrm{B}} + k_1 \cdot c_{\mathrm{A}} - k_2 \cdot c_{\mathrm{B}}, \tag{B.1b}$$

$$\dot{T}_{\mathrm{R}} = -\frac{k_1 \cdot c_{\mathrm{A}} \cdot \Delta H_{\mathrm{R},ab} + k_2 \cdot c_{\mathrm{B}} \cdot \Delta H_{\mathrm{R},bc} + k_3 \cdot c_{\mathrm{A}}^2 \cdot \Delta H_{\mathrm{R},ad}}{\rho \cdot c_p} + F \cdot (T_{\mathrm{in}} - T_{\mathrm{R}}) + \frac{k_w \cdot A_{\mathrm{R}} \cdot (T_{\mathrm{K}} - T_{\mathrm{R}})}{\rho \cdot c_p \cdot V_{\mathrm{R}}}, \tag{B.1c}$$

$$\dot{T}_{\mathrm{K}} = \frac{\dot{Q} + k_w \cdot A_{\mathrm{R}} \cdot (T_{\mathrm{R}} - T_{\mathrm{K}})}{m_k \cdot c_{p,k}}, \tag{B.1d}$$

where the rate coefficients k_1, k_2, and k_3 are obtained with Arrhenius' law:

$$k_1 = k_{0,\mathrm{ab}} \cdot \exp\left(\frac{-E_{\mathrm{A,ab}}}{T_{\mathrm{R}} + 273.15}\right), \tag{B.2a}$$

$$k_2 = k_{0,\mathrm{bc}} \cdot \exp\left(\frac{-E_{\mathrm{A,bc}}}{T_{\mathrm{R}} + 273.15}\right), \tag{B.2b}$$

$$k_3 = k_{0,\mathrm{ad}} \cdot \exp\left(\frac{-E_{\mathrm{A,ad}}}{T_{\mathrm{R}} + 273.15}\right). \tag{B.2c}$$

The parameters for (B.1) and (B.2) are listed in Table B.2. Furthermore, we display the bounds for the states and inputs in Table B.1.

Table B.1: System bounds of the CSTR system.

	$\boldsymbol{x}$				$\boldsymbol{u}$	
	c_{A} mol L^{-1}	c_{B} mol L^{-1}	T_{R} °C	T_{K} °C	F h^{-1}	$\dot{Q}$ kJ h^{-1}
lb	0.1	0.1	50.0	50.0	5.0	-8500.0
ub	2.0	2.0	135.0	140.0	100.0	0.0

Table B.2: Model parameters of the CSTR system.

Parameter	Value	Unit	Description
$k_{0,\mathrm{ab}}$	1.287×10^{12}	h^{-1}	Arrhenius factor
$k_{0,\mathrm{bc}}$	1.287×10^{12}	h^{-1}	Arrhenius factor
$k_{0,\mathrm{ad}}$	9.043×10^{9}	$\mathrm{L\,mol^{-1}\,h^{-1}}$	Arrhenius factor
R_{gas}	8.314×10^{-3}	$\mathrm{kJ\,mol^{-1}\,K^{-1}}$	ideal gas constant
$E_{\mathrm{A,ab}}$	9758.3	$\mathrm{kJ\,mol^{-1}}$	activation energy
$E_{\mathrm{A,bc}}$	9758.3	$\mathrm{kJ\,mol^{-1}}$	activation energy
$E_{\mathrm{A,ad}}$	8560.0	$\mathrm{kJ\,mol^{-1}}$	activation energy
$\Delta H_{\mathrm{R,ab}}$	4.2	$\mathrm{kJ\,mol^{-1}}$	reaction enthalpy
$\Delta H_{\mathrm{R,bc}}$	−11.0	$\mathrm{kJ\,mol^{-1}}$	reaction enthalpy
$\Delta H_{\mathrm{R,ad}}$	−41.85	$\mathrm{kJ\,mol^{-1}}$	reaction enthalpy
ρ	0.9342	$\mathrm{kg\,L^{-1}}$	reactor density
c_p	3.01	$\mathrm{kJ\,kg^{-1}\,K^{-1}}$	reactor heat capacity
$c_{p,k}$	2.0	$\mathrm{kJ\,kg^{-1}\,K^{-1}}$	coolant heat capacity
A_{R}	0.215	$\mathrm{m^2}$	are of reactor wall
V_{R}	10.01	L	liquid volume of reactor
m_k	5.0	kg	mass of coolant
T_{in}	130.0	°C	inlet temperature
k_w	4032.0	$\mathrm{kJ\,h^{-1}\,m^{-2}\,K^{-1}}$	heat transfer coefficient reactor wall

Table B.3: Model parameters of the triple mass spring system.

Parameter	Value					Unit
i =	1	2	3	4		
Θ_i	2.25	2.25	2.25		$\times 10^{-4}$	kg m^2
c_i	2.697	2.66	3.05	2.86	$\times 10^{-3}$	N m
d_i	6.78	8.01	8.82		$\times 10^{-5}$	N m s
τ					10^{-2}	s

B.2 Triple-mass-spring system

The ordinary differential equations of the triple-mass-spring system are

$$\Theta_1 \ddot{\phi}_1 = -c_1 (\phi_1 - \phi_{m,1}) - c_2 (\phi_1 - \phi_2) - d_1 \dot{\phi}_1, \tag{B.3a}$$
$$\Theta_2 \ddot{\phi}_2 = -c_2 (\phi_2 - \phi_1) - c_3 (\phi_2 - \phi_3) - d_2 \dot{\phi}_2, \tag{B.3b}$$
$$\Theta_3 \ddot{\phi}_3 = -c_3 (\phi_3 - \phi_2) - c_4 (\phi_3 - \phi_{m,2}) - d_3 \dot{\phi}_3, \tag{B.3c}$$
$$\dot{\phi}_{m,1} = \frac{1}{\tau}(\phi_{m,1}^{\text{set}} - \phi_{m,1}), \tag{B.3d}$$
$$\dot{\phi}_{m,2} = \frac{1}{\tau}(\phi_{m,2}^{\text{set}} - \phi_{m,2}), \tag{B.3e}$$

where ϕ_i [rad] denotes the angle of disc i. The angular velocity of disc i is $\dot{\phi}_i$ [rad s^{-1}]. Finally, $\phi_{m,1}$ [rad] denotes the current position of motor 1. The inputs are $\boldsymbol{u} = \begin{bmatrix} \phi_{m,1}^{\text{set}} & \phi_{m,2}^{\text{set}} \end{bmatrix}$, where $\phi_{m,i}^{\text{set}}$ [rad] is the desired position for motor i. The parameters of the system are listed in Table B.3. The discrete-time formulation can be obtained by reformulating (B.3) in the form of

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_c \boldsymbol{x} + \boldsymbol{B}_c \boldsymbol{u}, \tag{B.4}$$

with state vector

$$\boldsymbol{x}^\top = \begin{bmatrix} \phi_1 & \phi_2 & \phi_3 & \dot{\phi}_1 & \dot{\phi}_2 & \dot{\phi}_3 & \phi_{m,1} & \phi_{m,2}. \end{bmatrix}$$

Finally, the discrete-time system matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are obtained as

$$\boldsymbol{A} = e^{\boldsymbol{A}_c \Delta t}, \tag{B.5a}$$
$$\boldsymbol{B} = \boldsymbol{A}^{-1} \left(e^{\boldsymbol{A}_c \Delta t} - \boldsymbol{I} \right) \boldsymbol{B}_c, \tag{B.5b}$$

where $\Delta t = 0.1$s is the timestep.

Table B.4: Model parameters of the building system.

Parameter	Value				Unit
i =	1	2	3	4	
C_i	50	110	80	90	MJ K^{-1}
g_i	2.1	1.9	1.0	2.0	kW K^{-1}
g_{ia}	0.3	0.5	0.4	0.6	kW K^{-1}

B.3 Building system

The building system, shown in Figure 4.4, is governed by the ordinary differential equations:

$$C_1 \frac{dT_1}{dt} = -\dot{Q}_{12} + \dot{Q}_{14} - \dot{Q}_{1a} + \dot{Q}_1, \tag{B.6}$$

$$C_2 \frac{dT_2}{dt} = \dot{Q}_{12} - \dot{Q}_{23} - \dot{Q}_{2a} + \dot{Q}_2, \tag{B.7}$$

$$C_3 \frac{dT_3}{dt} = \dot{Q}_{23} - \dot{Q}_{34} - \dot{Q}_{3a} + \dot{Q}_3, \tag{B.8}$$

$$C_4 \frac{dT_4}{dt} = \dot{Q}_{34} - \dot{Q}_{14} - \dot{Q}_{4a} + \dot{Q}_4, \tag{B.9}$$

where the heat flows between the rooms are directed as shown in Figure 4.4 and are computed as:

$$\dot{Q}_{12} = g_1(T_1 - T_2), \tag{B.10}$$

$$\dot{Q}_{23} = g_2(T_2 - T_3), \tag{B.11}$$

$$\dot{Q}_{34} = g_3(T_3 - T_4), \tag{B.12}$$

$$\dot{Q}_{14} = g_4(T_4 - T_1). \tag{B.13}$$

The heat flow of each room with the environment is computed as:

$$\dot{Q}_{ia} = g_{ia}(T_i - T_a) \quad \forall i \in \mathbb{I}_{[1,4]}. \tag{B.14}$$

The temperatures in each room and the environment constitute the states of the system, that is, $\boldsymbol{x}^\top = [T_1,\, T_2,\, T_3,\, T_4,\, T_a]$ [°C], and the control inputs are the radiators in each room, that is, $\boldsymbol{u}^\top = [\dot{Q}_1,\, \dot{Q}_2,\, \dot{Q}_3,\, \dot{Q}_4]$ [kW]. The required parameters are listed in Table B.4. Finally, it is assumed that the ambient temperature follows the forecasted temperature T_f, that is:

$$\dot{T}_a = \frac{1}{\tau_a}(T_f - T_a), \tag{B.15}$$

with $\tau_a = 72.000$ s. With (B.6) and (B.15), we have a linear continuous-time system in the form of (B.4), which is discretized with (B.5) using the timestep $\Delta t = 3600$ s.

Appendix C

Pre-published content

This thesis is the result of research activities at the chair of Process Automation Systems at the TU Dortmund University. Parts of this thesis have been previously published in journal articles and conference proceedings, as declared in the following.

Chapter 1	1.1	partly modified from	[42]
	1.2	partly modified from	[12], [42]
	1.2.2	partly modified from	[42]
Chapter 2	2.1.2	partly modified from	[12]
	2.2	partly modified from	[12]
	2.2.1	partly modified from	[12], [43]
	2.3.2	partly modified from	[12]
	2.3.3	partly modified from	[40]
Chapter 3	3.1.2	partly modified from	[41]
	3.2	partly modified from	[41]
	3.4.1	partly modified from	[41]
	3.4.2	partly modified from	[41]
	3.5.1	partly modified from	[41]
	3.5.2	partly modified from	[41]
Chapter 4	Introduction	partly modified from	[42]
	4.1	partly modified from	[42]
	4.1.1	partly modified from	[42]
	4.1.2	partly modified from	[42]
	4.2	partly modified from	[42]
	4.2.1	partly modified from	[42]
	4.2.2	partly modified from	[42]
	4.2.3	partly modified from	[42]
	4.3	partly modified from	[42]

	4.3.1	partly modified from	[42]
	4.3.2	partly modified from	[42]
	4.4.1	partly modified from	[42]
	4.4.2	partly modified from	[42]
	4.5	partly modified from	[42]
Chapter 5	Introduction	partly modified from	[43]
	5.1	partly modified from	[43]
	5.1.1	partly modified from	[43]
	5.1.2	partly modified from	[42], [43]
	5.1.3	partly modified from	[43]

Publications

The referenced list of journal articles and conference proceedings is given in the following.

Journal articles

[12] F. Fiedler, B. Karg, L. Lüken, D. Brandner, M. Heinlein, F. Brabender, and S. Lucia, "Do-mpc: Towards FAIR nonlinear and robust model predictive control", *Control Engineering Practice*, vol. 140, p. 105 676, Nov. 1, 2023

[42] F. Fiedler and S. Lucia, "Probabilistic Multi-Step Identification With Implicit State Estimation for Stochastic MPC", *IEEE Access*, vol. 11, pp. 117 018–117 029, 2023

[43] F. Fiedler and S. Lucia, "Improved uncertainty quantification for neural networks with Bayesian last layer", *IEEE Access*, vol. 11, pp. 123 149–123 160, 2023

Peer-reviewed conference proceedings

[40] F. Fiedler, A. Cominola, and S. Lucia, "Economic nonlinear predictive control of water distribution networks based on surrogate modeling and automatic clustering", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 16 636–16 643, 2020

[41] F. Fiedler and S. Lucia, "On the relationship between data-enabled predictive control and subspace predictive control", in *European Control Conference*, 2021, pp. 222–229

Control Engineering Practice 140 (2023) 105676

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

do-mpc: Towards FAIR nonlinear and robust model predictive control

Felix Fiedler *, Benjamin Karg, Lukas Lüken, Dean Brandner, Moritz Heinlein, Felix Brabender, Sergio Lucia

Chair of Process Automation Systems, TU Dortmund University, Emil-Figge-Str. 70, 44227, Dortmund, Germany

ARTICLE INFO

Keywords:
Nonlinear model predictive control
Learning-based control
Robust control

ABSTRACT

Over the last decades, model predictive control (MPC) has shown outstanding performance for control tasks from various domains. This performance has further improved in recent years with advanced MPC schemes for nonlinear systems under uncertainty including economic control objectives. These recent improvements often fail to bridge the gap between MPC researchers and control practitioners in academia and industry, where classical control approaches and traditional linear MPC still dominate most applications. This is despite the fact that advanced MPC controllers can lead to significant energy savings, yield improvements, safer operation and other benefits.

In this work, we identify four main obstacles hindering the widespread adoption of advanced MPC methods. These are the unavailability of models, the challenges associated with deploying complex controllers on physical systems, the scarcity of rapid prototyping tools for advanced methods and the limited reproducibility and reusability of advanced MPC controllers and their results. We find that the FAIR principles (findable, accessible, interoperable, reusable) for scientific data-management and research software can play an important role in tackling these obstacles. Following these guidelines, we discuss FAIR solutions and present the open-source software do-mpc as a concrete implementation. The presented solutions include interoperability with neural network toolboxes to simplify nonlinear system identification, interoperability with the OPC UA communication protocol for deployment, and a reproducible data-sampling framework for transparent controller validation, system identification and approximate MPC. The potential of the proposed solutions is illustrated with several simulation studies.

1. Introduction

Model predictive control (MPC) is a method for controlling multi-variable systems with constraints and has important applications in the process industry (Forbes, Patwardhan, Hamadah, & Gopaluni, 2015), building control (Blum et al., 2022; Yao & Shekhar, 2021), power electronics (Kouro et al., 2015; Vazquez et al., 2014) and other fields. MPC is based on the formulation of a finite horizon optimal control problem, in which a control objective function is minimized while taking into account the system dynamics and additional performance or safety critical constraints. In recent years, there has been an increasing research focus on advanced MPC solutions, for example with nonlinear systems, economic cost functions (Rawlings, Angeli, & Bates, 2012), and consideration of uncertainties in robust or stochastic MPC (Lucia, Finkler, & Engell, 2013; Mayne, Kerrigan, van Wyk, & Falugi, 2011; Mesbah, 2016). Solving the complex and potentially nonlinear optimization problems has been enabled by improved hardware and advances in software and algorithms, such as IPOPT (Wächter & Biegler, 2006), FORCES (Zanelli, Domahidi, Jerez, & Morari, 2020), OpEn (Sopasakis, Fresk, & Patrinos, 2020), acados (Verschueren et al., 2022), GRAMPC (Englert, Völz, Mesmer, Rhein, & Graichen, 2019) and others (Chen, Bruschetta, Picotti, & Beghi, 2019; Findeisen, Graichen, & Mönnigmann, 2018; Risbeck & Rawlings, 2016). Due to these advances, MPC is nowadays applicable to large-scale systems as well as systems with high control frequency, for example, in process industries or power electronics.

Despite these developments, advanced MPC approaches are often not widely adopted outside the core MPC research community (Blum et al., 2022). For instance, if MPC is applied in process industries and building control it is mostly limited to linear and nominal MPC (Forbes et al., 2015; Yao & Shekhar, 2021). However, especially in these domains there is a significant potential for improved operations with nonlinear, robust and economic MPC (Rawlings et al., 2012). Other research communities and industries with secondary control tasks often resort to traditional control algorithms.

As a main contribution of this work, we identify four challenges that hinder the widespread adoption of advanced MPC approaches

* Corresponding author.
E-mail address: felix.fiedler@tu-dortmund.de (F. Fiedler).

https://doi.org/10.1016/j.conengprac.2023.105676
Received 23 May 2023; Received in revised form 11 August 2023; Accepted 29 August 2023
Available online 7 September 2023

Table 1
Challenges and FAIR solutions for the adoption of advanced MPC.

Challenge	Reason	FAIR solution
1 Control oriented model is unavailable	Modeling is expensive or available model is not suitable for optimization	Neural network system identification and **interoperability**, e.g. through the ONNX standard. Accessible and **reproducible** data for identification
2 Deployment of MPC algorithm to physical system is complicated	Research tools not tailored for deployment or require significant coding experience. Solving advanced MPC on resource limited hardware can be intractable	Interoperability with OPC UA standard to communicate with plant. Approximate MPC with **accessible** and **reproducible** data and validation
3 Rapid prototyping is difficult for advanced MPC	Commercial tools require significant upfront investment. Research tools require experience and may be insufficient for holistic rapid prototyping approach	Accessible open-source MPC software with comprehensive and modular features from initial draft to deployment and **reusable** templates
4 MPC results are not reproducible and not reusable	Inaccessible or incomprehensible controller implementation. Data for system identification, controller validation etc. is not accessible or reproducible	Comprehensible code with documented open-source software. Reproducible data sampling framework for validation, identification and approximate MPC

and outline solutions to overcome these challenges. A summary of our discussion is shown in Table 1. In particular, we illustrate how applying the FAIR principles (findable, accessible, interoperable and reusable) for scientific data-management and research software (Chue Hong et al., 2021; Wilkinson et al., 2016) can play an important role in tackling these challenges. Related discussions have been previously reported in the literature (Blum et al., 2022; Forbes et al., 2015). However, the authors in Forbes et al. (2015) mostly focus on traditional linear MPC and concrete industrial challenges, e.g. the interaction of operator and MPC application. In Blum et al. (2022), the authors focus on practical challenges for the deployment of MPC to a building system. The novelty of our discussion therefore lies in the focus on advanced MPC and the connection to the FAIR principles. We believe that as data and machine learning become increasingly important also in the context of MPC, adherence to the FAIR principles becomes crucial in designing MPC controllers that can be actually deployed in real-world applications.

As another contribution of this work, and in the spirit of FAIR research software, we present do-mpc,[1] a nonlinear and robust MPC software. Throughout this work, we highlight how do-mpc, by considering the FAIR solutions in Table 1, facilitates the adoption of advanced MPC outside of the core MPC research community. Additionally, we investigate how comparable toolboxes for advanced MPC deal with the outlined challenges. This comparison considers a non-exhaustive selection of tools, and shows that do-mpc can be an attractive solution for researchers and practitioners of advanced MPC.

This work is structured as follows. In Section 2, we discuss the challenges and FAIR solutions for the widespread adoption of advanced MPC. As a concrete implementation of these solutions, we present do-mpc in Section 3, with an introduction of the control theoretical background in Section 4. Important steps towards the FAIR solutions are introduced in the successive sections. In particular, we discuss neural network system identification, the sampling framework, and the ONNX conversion module in Section 5. In Section 6, approximate MPC as a versatile means of deployment is discussed with a focus on sensitivity-based neural network training. A summary of our work, as well as an outlook and future research directions, are presented in Section 7.

2. Challenges for the adoption of advanced MPC

In Table 1, we summarize four main challenges that are hindering the widespread adoption of advanced MPC in research and industry. We proceed by explaining these challenges and discuss solutions in the next subsection.

The first challenge 1 is the difficulty to obtain a control-oriented model, which has been recognized as a core problem of model-based control for decades (Morari & H. Lee, 1999). Physical modeling is time consuming, requires expert knowledge and is typically repeated for each investigated system. Additionally, complex system models are often created in dedicated dynamic modeling software which is typically not suitable for optimization and MPC. While obtaining a control oriented model is a challenge for model-based control in general, it is even more profound for advanced MPC. For example, economic MPC often steers the system close to its constraints where a plant model mismatch might have severe consequences.

The second challenge 2 lies in the deployment, that is, interfacing the physical system with the controller running on the designated hardware, for example a programmable logic controller (PLC) or a micro-controller. Especially with research tools, deploying the control algorithm may require significant effort and experience (Blum et al., 2022). On the other hand, industrial software is designed for deployment, but may not be flexible enough for all use-cases (Blum et al., 2022). Furthermore, software licenses and dedicated hardware for commercial solutions require a significant upfront investment.

Deployment is the final step of controller design which typically starts with a first prototype that is tested and improved gradually. This established process, coined *rapid control prototyping* (Abel & Bollig, 2006), has important advantages. For example, expected cost reductions can be used to justify further work. Unfortunately, the rapid prototyping approach faces serious obstacles for advanced MPC solutions, which forms the third identified challenge 3. For industrial MPC software, rapid protyping can be prohibited by the upfront investment in software and hardware. A related issue also applies to open-source research tools for advanced MPC. These tools often have a steep learning curve and require significant manual work, advanced coding skills and experience. These challenges contribute to a significant time investment before a first prototype is working. Research software can also be insufficient for a holistic rapid prototyping approach, lacking, for example, efficient means for software-in-the-loop (Abel & Bollig, 2006) testing. These tests are essential to consider potential effects of deployment, for example due to communication issues and time-delays stemming from solving the MPC problem.

Finally, as the fourth challenge 4, published work on advanced MPC methods is often not reusable and reproducible because the code, software, or system model may be inaccessible, problem-specific, poorly documented, or not maintained. Without access to the code or usable code, results cannot be validated.

Furthermore, the development of advanced MPC solutions may require sampled data, for example, for system identification or probabilistic validation. Only if this data is reproducible, the overall results are reproducible and the approach can be reused if the data is updated. Naturally, control solutions with industrial software are entirely inaccessible to third-party users and cannot be reused or reproduced.

[1] https://www.do-mpc.com.

2.1. FAIR solutions to enable advanced MPC

In 2016, the authors (Wilkinson et al., 2016) published the FAIR principles for scientific data management, which guide that research data should be findable, accessible, interoperable and reusable. Since this publication, the FAIR principles are having an important impact on various fields (Artrith et al., 2021; Wise et al., 2019) and further extensions and applications have been proposed, such as the FAIR principles for scientific research software (Chue Hong et al., 2021). We find that the FAIR principles also offer important guidance on how to tackle the outlined challenges for the widespread adoption of advanced MPC. In the following, we will discuss FAIR solutions to enable advanced MPC with a summary of our discussion presented in Table 1.

For the first issue (1), that is, the availability of a system model, we consider data-based system identification (Ljung, 2010) as a key solution. In many cases, system data is readily available from historic plant measurements or simulation results of a non-control oriented high-fidelity model. Depending on the investigated system and prior knowledge, either a pure black-box model or a hybrid model can be identified. The main advantage of data-based models is that they potentially require no additional engineering effort when rolling out MPC solutions to different systems. Furthermore, data-based linear system identification is already an established approach for traditional MPC schemes (Morari & H. Lee, 1999). Model selection is a challenge for nonlinear system identification (Morari & H. Lee, 1999) but especially neural networks have shown promising results in recent years (Ljung, Andersson, Tiels, & Schön, 2020; Wang, 2017). These models can capture complex nonlinear system behavior from large datasets and are suitable for fast gradient-based optimization. An important advantage of neural networks is also the availability of powerful open-source software, for example Tensorflow (Abadi et al., 2016), Pytorch (Paszke et al., 2019) but also the Open Neural Network Exchange[2] (ONNX) standard for the exchange of neural network models. While these toolboxes simplify training a neural network for system identification, most MPC software solutions cannot directly incorporate them. It is therefore imperative that state-of-the-art MPC software has an interface to neural network models. Interoperability, in the sense of the FAIR principles, can be achieved by supporting the ONNX standard. Another challenge for nonlinear system identification in general is the test input signal design (Morari & H. Lee, 1999). The FAIR principles play an important role in this regard as well, because the resulting data and the input signal algorithm should be findable, accessible and reusable. If the identification process is transparent, the trust in the obtained model will be improved and it will also showcase its potential limitations. Furthermore, similar identification tasks can be simplified, facilitating the development of further MPC applications.

We also consider neural networks as a key enabling technology for the deployment of advanced MPC solutions (2). It has recently been shown that deep neural networks can exactly represent the piecewise affine control law of a linear MPC controller (Karg & Lucia, 2020). Importantly, the evaluation of deep neural networks can be orders of magnitudes faster than solving an optimization problem. This makes approximate MPC a popular approach for control tasks with high frequency and it can also be used for advanced MPC controllers (Chen et al., 2018; Guillén, Fiedler, Sarnago, Lucía, & Lucía, 2022; Kumar, Rawlings, & Wright, 2021; Paulson & Mesbah, 2020). While the neural network represents only an approximation of the original MPC controller, probabilistic performance and safety guarantees can be obtained (Karg, Alamo, & Lucia, 2021). In recent work (Lüken, Brandner, & Lucia, 2023; Winqvist, Venkitaraman, & Wahlberg, 2021), it has also been shown that sample efficient approximate MPC controllers can be obtained by retrieving the sensitivity information of the optimal control problem and by incorporating this information in the neural network training. Especially for approximate MPC, it is important to follow the FAIR guidelines when sampling the original controller, when training the neural network controller, or for its probabilistic validation. Only if these steps are transparent and automated, the approximate MPC is reproducible and reusable (4). This also enables successive changes of the approximate MPC, for example, a modification of the control task.

Complementing approximate MPC with neural networks, we consider interoperability with the OPC UA (Unified Architecture by the OPC foundation) standard, an important element for the deployment of MPC (2). OPC UA is a machine-to-machine communication protocol, used especially in the process industries (Schleipen, Gilani, Bischoff, & Pfrommer, 2016). Through interoperability with OPC UA, advanced MPC can be deployed even without code generation, using Docker containers (Blum et al., 2022). OPC UA also enables software-in-the-loop (Abel & Bollig, 2006) testing that takes communication and calculation delays into consideration. In this way, interoperability with OPC UA benefits a holistic rapid control prototyping (3) approach.

As an important prerequisite for rapid prototyping, an open-source software for advanced MPC is required. This software must be accessible to research and industry with secondary control tasks while supporting full customization for advanced control research. To this end, it is paramount to have a versatile but robust interface, a modular design and a comprehensive documentation. Relevant examples come from the machine learning community, where open-source tools like Pytorch and Tensorflow require only minutes from installation to training a first model but can also be used for the most complex tasks. This is also enabled by the extensive documentation of both tools and various tutorials with example code. The availability of such tools, both in machine learning and for advanced control, also strongly facilitates the reusability and reproducibility (4) of methods and results that were developed using them.

3. Towards FAIR advanced MPC with do-mpc

As a contribution of this work, we introduce do-mpc,[3] an open-source software for robust, economic and nonlinear MPC which is built around CasADi (Andersson et al., 2019). As in our previous work (Lucia, Tătulea-Codrean, Schoppmeyer, & Engell, 2017), do-mpc supports nonlinear differential–algebraic system models, orthogonal collocation on finite elements for discretization, robust multi-stage MPC with economic cost and nonlinear soft-constraints. In comparison to our previous efforts (Lucia et al., 2017), we have fundamentally redeveloped do-mpc in a modular fashion with a focus on the FAIR principles, and with important new features including moving horizon estimation (MHE), a data sampling framework, interoperability with ONNX and OPC UA and sensitivity calculation. An overview of the structure of do-mpc, highlighting the most important modules and their interconnection, is displayed in Fig. 1. We will elaborate in the following how obstacles towards the adoption of advanced MPC in research and industry are tackled in do-mpc with FAIR solutions. The contribution of the different do-mpc modules towards the challenges in Table 1 are also highlighted in Fig. 1. Furthermore, we present a summary of the proposed FAIR solutions in do-mpc in comparison to a non-exhaustive selection of advanced MPC tools in Table 2. The comparison excludes most commercial MPC solutions as they are explicitly not positioned as FAIR research software.

We find that data-based system identification with neural networks can be an important solution to mitigate the challenge of obtaining a control oriented model (1). To this end, do-mpc facilitates the incorporation of neural network system models through an interface to the ONNX open standard for machine learning interoperability, as shown in Fig. 1. ONNX enables model sharing and is supported by common

[2] https://onnx.ai.

[3] https://www.do-mpc.com.

Table 2
Comparing FAIR solutions to the challenges in Table 1 in different advanced MPC toolboxes. Indication of advantages (+) and neutral aspects (○).

Tool & short summary	1 Control oriented model is unavailable	2 Deployment of MPC algorithm to physical system is complicated	3 Rapid prototyping is difficult for advanced MPC	4 MPC results are not reproducible and not reusable
do-mpc: Nonlinear and robust multi-stage MPC and MHE committed to the FAIR principles. Uses CasADi (Andersson, Gillis, Horn, Rawlings, & Diehl, 2019) as its backend.	+ ONNX conversion for neural network model + Data-sampling for system identification	+ OPC UA interface + Reproducible and sample efficient approximate MPC for export to target hardware ○ No code generation	+ Python interface + Modular design + OPC UA for software-in-the-loop + Quick start and deep customization	+ Extensive documentation and examples + Validation with reproducible sampling framework + Readable model and controller formulation
acados (Verschueren et al., 2022): Nonlinear MPC and MHE with code generation for fast and embedded applications. Uses CasADi (Andersson et al., 2019) for differentiation.	○ No solution outlined	+ Code generation and compilation for target hardware	+ Matlab and Python interface + Modular design + Compiled controller with Matlab/Simulink interface ○ Advanced coding and control experience	○ Documentation and examples
GRAMPC (Englert et al., 2019): Nonlinear MPC and MHE with rapid but suboptimal solutions. Controller design by adapting C templates.	○ No solution outlined	+ Code compilation for target hardware	+ Compiled controller with Matlab/Simulink interface ○ Advanced coding and control experience	○ Examples but sparse documentation
MATMPC (Chen et al., 2019): Nonlinear MPC in Matlab with fast compiled MEX functions and using CasADi (Andersson et al., 2019) for integration.	○ No solution outlined	○ Deployment not outlined or demonstrated	+ Matlab/Simulink interface	○ Sparse examples and documentation
MPCTools (Risbeck & Rawlings, 2016): Nonlinear MPC using CasADi (Andersson et al., 2019) in Matlab.	○ No solution outlined	○ Deployment not outlined or demonstrated	+ Matlab interface	○ Examples but sparse documentation
Matlab MPC Toolbox: Nonlinear MPC from Mathworks® for Matlab and Simulink.	+ Linear system identification toolbox	+ Code generation with additional toolbox + Explicit MPC for linear systems	+ Matlab ecosystem for rapid control prototyping + Modular design ○ Limited customization	+ Extensive documentation and examples ○ Expensive licensing
OpEn (Sopasakis et al., 2020): Nonlinear optimization in Rust for deployment on embedded hardware. Positioned for control applications. Uses CasADi (Andersson et al., 2019) for differentiation.	○ No solution outlined	+ Code generation and compilation for target hardware + TCP socket server	+ Matlab and Python interface ○ Advanced coding and control experience ○ Custom controller implementation	+ Extensive documentation and examples

deep-learning tools, e.g. Matlab, Pytorch and Tensorflow. By including an ONNX conversion tool, do-mpc allows to seamlessly incorporate previously trained neural networks in the optimal control problem. If neural network system identification is used to create surrogate models of high-fidelity simulation models, do-mpc can additionally support the data-generation process. To this end, do-mpc features the sampling module shown in Fig. 1, which supports the definition of sampling scenarios, for instance with different initial conditions of the system, and provides a multiprocessing-enabled sampler which can generate closed-loop trajectories. Importantly, the sampling module is designed to make the data generation process transparent and reproducible 4. As shown in Table 2, we find that data-based system identification is not a strong focus of most other open-source MPC solutions. In particular, the seamless integration of neural networks and the transparent and reproducible data-generation are important propositions of do-mpc.

Regarding the issue of deployment 2, do-mpc enables or facilitates both solutions listed in Table 1. As shown in Fig. 1, do-mpc provides an OPC UA interface and supports the automatic derivation of OPC UA clients from the core do-mpc modules. Clients can be connected to an existing OPC UA server or to a locally launched server for testing purposes. Deployment with OPC UA is also implicitly available to the tools in Table 2 which have a Matlab/Simulink interface. However, this requires an additional license for the industrial communication toolbox from Mathworks® and a custom configuration. Alternatively, various tools in Table 2 enable deployment through code generation (Sopasakis et al., 2020; Verschueren et al., 2022) or directly use a compiled language (Englert et al., 2019). This is especially useful for embedded applications but can be challenging without advanced software and hardware experience. For embedded applications, do-mpc follows a different strategy, by facilitating the development of approximate MPC. In particular, the sampling module can be used to generate closed-loop trajectories of the MPC controlled system which are used to train the approximate MPC controller. These results are reproducible by simply sharing the human readable sampling plan which also makes the entire process highly transparent. This ensures that the approximate MPC controller can be reproduced, iteratively improved after modifying the control task, or reused for a related use-case 4. With the sensitivity module, do-mpc further facilitates approximate MPC deployment by supporting the computation of sensitivities. This additional information can be incorporated when training the neural network-based approximate MPC controller (Cocola & Hand, 2020; Czarnecki, Osindero, Jaderberg, Swirszcz, & Pascanu, 2017; Lüken et al., 2023).

One of the most important strengths of do-mpc is that it enables rapid prototyping 3 of advanced MPC solutions, for example, robust multi-stage nonlinear MPC. Through the well documented, intuitive and robust Python interface, do-mpc enables users with basic control experience to design a first prototype within minutes. At the same time, do-mpc allows for the deep customization that is required in active research and has been used in several recent publications (Guillén et al., 2022; Marzullo, Dey, Long, Leiva Vilaplana, & Henze, 2022; Patria, Rossi, Fernandez, & Dominguez, 2021). While other tools listed in Table 2 are also well adopted in the MPC community, they often require a steeper learning curve for the initial setup and successive adaptations. Rapid control prototyping with do-mpc also benefits from its highly

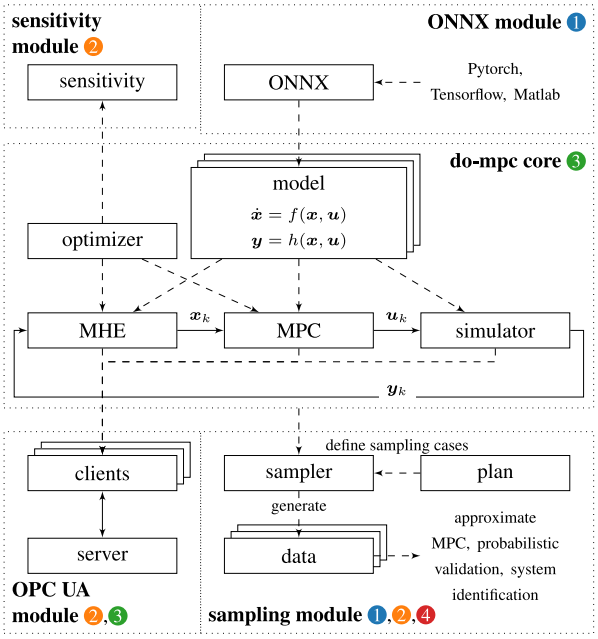

Fig. 1. The core elements of do-mpc and important modules with reference to the challenges summarized in Table 1.

modular structure. Development and testing of control schemes can start with just a model, a controller and the simulator from the do-mpc core, shown in Fig. 1. As development continues, additional modules and features can be included, such as a moving horizon estimator or the robust multi-stage approach. Finally, the OPC UA module is used for software-in-the-loop testing and deployment to the real plant. Software-in-the-loop testing is also possible with other toolboxes listed in Table 2, but is typically not an explicit focus. As an exception, the Matlab MPC toolbox has a comprehensive list of features in this regard but suffers from expensive licensing and limited customization due to proprietary code.

Control algorithms designed with do-mpc are also highly human readable and can be extended by other researchers. This also means that FAIR publications with do-mpc enable other researchers to reuse the code and reproduce the results 4. An important element in this regard is also the extensive documentation of do-mpc. This is a major advantage in comparison to several other tools presented in Table 2 which offer competitive features but often suffer from sparse documentation and examples.

In the following sections, we will introduce the control theoretical background and implemented methods in do-mpc. Furthermore, we will present the important features that are enabling the FAIR solutions summarized in Table 1. We introduce two systems to demonstrate in simulation studies how these features, and do-mpc in general, can tackle the challenges in practice. In the spirit of illustrating the FAIR principles, all the code for the examples, the resulting data or the required sampling plan are provided online[4] and can be executed with do-mpc version >4.5.0.

4. From LQR to robust nonlinear model predictive control

The core modules of do-mpc, shown in Fig. 1, are based on the formulation of a dynamic system model. We consider a general uncertain discrete-time system of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}_k), \tag{1a}$$

$$0 = \boldsymbol{g}_{\text{alg}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}_k), \tag{1b}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}_k), \tag{1c}$$

with states $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^{n_x}$, inputs $\boldsymbol{u} \in \mathbb{U} \subset \mathbb{R}^{n_u}$, algebraic states $\boldsymbol{z} \in \mathbb{Z} \subset \mathbb{R}^{n_z}$, measurements $\boldsymbol{y} \in \mathbb{Y} \subset \mathbb{R}^{n_y}$, and potentially uncertain parameters $\boldsymbol{p} \in \mathbb{P} \subset \mathbb{R}^{n_p}$. The subscript k denotes the discrete-time instance and we introduce the time-step Δt. We use bold symbols to denote vectors or vector valued functions, e.g. $\boldsymbol{x}$, and regular symbols for scalars, e.g. k.

Apart from discrete-time models in the form of (1), do-mpc also natively supports continuous-time differential–algebraic (DAE) systems. In this case, orthogonal collocation on finite elements is used as a full discretization approach (Biegler, 2010), For ease of notation, we proceed with the discrete-time formulation.

In the following we introduce the three main controller classes implemented in do-mpc. First, we present the nominal advanced model predictive controller, which can handle, for example, nonlinear systems and economic control goals. Secondly, we present the robust multi-stage MPC formulation, which can additionally consider multiple scenarios of the parameters $\boldsymbol{p}$ to mitigate the effect of uncertainty (Lucia et al., 2013). Finally, we discuss the linear quadratic regulator (LQR) for regulation and set-point control of unconstrained linear systems.

4.1. Nominal MPC

Model predictive control solves the constrained optimal control problem, for a finite horizon with N elements, repeatedly at each time step and applies the first input of the optimal trajectory to the controlled system. The input trajectory from 0 to N is denoted as $\boldsymbol{u}_{[0:N]}$. The set of feasible states $\mathbb{X}$, inputs $\mathbb{U}$ and other constraints are described by the constraint function $\boldsymbol{g}_{\text{con}}(\boldsymbol{x}_{[0:N+1]}, \boldsymbol{u}_{[0:N]}, \boldsymbol{z}_{[0:N]}, \boldsymbol{p}_{[0:N]}) \leq \boldsymbol{0}$. The optimal control problem for known values of the parameters $\hat{\boldsymbol{p}}_{[0:N]}$ is shown below.

$$\min_{\boldsymbol{x}_{[0:N+1]}, \boldsymbol{u}_{[0:N]}, \boldsymbol{z}_{[0:N]}} \boldsymbol{J}(\boldsymbol{x}_{[0:N+1]}, \boldsymbol{u}_{[0:N]}, \boldsymbol{z}_{[0:N]}, \boldsymbol{p}_{[0:N]}) \tag{2a}$$

$$\text{s.t.}: \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{initial}}, \tag{2b}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \hat{\boldsymbol{p}}_k), \ \forall k \in \mathbb{I}_{[0,N]}, \tag{2c}$$

$$0 = \boldsymbol{g}_{\text{alg}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \hat{\boldsymbol{p}}_k), \ \forall k \in \mathbb{I}_{[0,N]}, \tag{2d}$$

$$\boldsymbol{g}_{\text{con}}(\boldsymbol{x}_{[0:N+1]}, \boldsymbol{u}_{[0:N]}, \boldsymbol{z}_{[0:N]}, \hat{\boldsymbol{p}}_{[0:N]}) \leq \boldsymbol{0}, \tag{2e}$$

where $\mathbb{I}_{[0,N]}$ denotes the set of integers from 0 to N. The cost function (2a) is of the form:

$$\boldsymbol{J}(\boldsymbol{x}_{[0:N+1]}, \boldsymbol{u}_{[0:N]}, \boldsymbol{z}_{[0:N]}, \boldsymbol{p}_{[0:N]}) = \sum_{k=0}^{N} \left(l(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{z}_k, \boldsymbol{p}_k) + \boldsymbol{r}^\top \Delta \boldsymbol{u}_k^2 \right) + m(\boldsymbol{x}_{N+1}), \tag{3}$$

with stage cost $l(\cdot)$ and terminal cost $m(\cdot)$. To penalize rapid changes of inputs, $\Delta \boldsymbol{u}_k = \boldsymbol{u}_k - \boldsymbol{u}_{k-1}$ is element-wise squared, weighted with positive $\boldsymbol{r} \in \mathbb{R}_+^{n_u}$ and added to the cost. The stage cost and terminal cost can be arbitrary expressions, for example economic cost functions. The constraints (2c) and (2d) enforces the optimized trajectory to follow the nonlinear system dynamics in (1). The optimal control problem can also be formulated with soft-constraints, as discussed in Zeilinger, Morari, and Jones (2014). To this end, do-mpc automatically introduces slack-variables for the inequality constraints (2e) and adds their the weighted ℓ_1-norm to the cost (2a).

For MPC, the optimal control problem is recomputed at each time step k with the new initial state $\boldsymbol{x}_{\text{init}}$, yielding the implicit feedback control law:

$$\boldsymbol{u}_k = \kappa_{\text{MPC}}(\boldsymbol{x}_k). \tag{4}$$

Note that in (4), we omit further arguments of the control law, such as $\boldsymbol{p}_k$ and $\boldsymbol{u}_{k-1}$ for ease of notation.

[4] https://github.com/pas-tudo/2023_do_mpc_paper.

4.2. Multi-stage MPC

In the case when the future trajectory of the parameters $\boldsymbol{p}_{[0:N]}$ is uncertain, controlling the system with nominal MPC may result in constraint violations and potentially unsafe operation. Instead, robust MPC schemes need to be employed. Multi-stage MPC, as presented in Lucia et al. (2013), takes the uncertainties into account by optimizing over multiple scenarios of the possible realizations of the unknown parameters. The realizations of the parameters can be chosen as the extreme realizations, which can be branched in every step of the prediction horizon in a tree-like structure. Multi-stage MPC is a closed-loop robust MPC approach, which, in comparison to open-loop robust approaches, results in less conservative control actions (Lucia et al., 2013).

Branching the tree in each step would result in an exponential growth of scenarios with the prediction horizon. To avoid this intractable complexity, a heuristic approximation is to stop the branching after the so called *robust horizon*. This still leads to good results in practice, even for nonlinear systems (Lucia et al., 2013).

In contrast to many other robust MPC approaches (e.g. tube based MPC (Kouvaritakis & Cannon, 2016)), multi-stage MPC does not require to pre-compute linear feedback or back-up terms, while still being able to plan the future reaction to different parameter realizations. As a heuristic robust nonlinear controller it is conceptually applicable to all systems of the form (1). Thus, it is more accessible, as it also requires less in-depth knowledge and prerequisites to set up as other approaches. With do-mpc, multi-stage robust MPC is a straight-forward extension of the nominal MPC case. The control engineer only has to supply possible realizations of the uncertain parameters and the robust horizon.

4.3. Linear quadratic regulator

The linear quadratic regulator (LQR) is the optimal controller for linear systems with quadratic cost and without constraints. The LQR yields an explicit linear feedback control law:

$$\boldsymbol{u}_k = \kappa_{\text{LQR}}(\boldsymbol{x}_k) = \boldsymbol{K}\boldsymbol{x}_k, \tag{5}$$

where $\boldsymbol{K} \in \mathbb{R}^{n_u \times n_x}$ can be pre-computed offline. The explicit control law is a great advantage and can be used as a benchmark for more complex approaches, for the calculation of terminal sets in many MPC schemes, for data-generation as shown in Section 5.2, and is still of great interest in current research (Goel & Hassibi, 2022; Pfrommer & Sojoudi, 2022; Scampicchio, Aravkin, & Pillonetto, 2021).

For the design of the LQR, do-mpc considers a dedicated linear model class, which can be readily obtained by automatically linearizing the system (1) or directly by supplying the linear system dynamics. For linear continuous-time systems, do-mpc can also derive the respective discrete-time formulation with (Virtanen et al., 2020). Additionally, even differential–algebraic models of index one can be automatically transformed into ordinary differential equations required for the LQR.

4.4. Simulation study

To showcase the robust and nonlinear control capabilities of do-mpc, we investigate a continuously stirred tank reactor (CSTR). The system was previously introduced in Klatt and Engell (1998) and is depicted in Fig. 2. The CSTR has $n_x = 4$ states, that is, the concentration of the educt c_A, the concentration of the product c_B, the reactor temperature T_R and the temperature of the coolant T_K. Apart from the main reaction, two additional reactions form the byproducts C and D. The system is controlled with $\boldsymbol{u} = [F,\ \dot{Q}]^\top \in \mathbb{R}^2$, that is, the normalized inlet flow $F = \dot{V}/V_R$ and the heat removed by the coolant $\dot{Q}$. All system parameters and the system dynamics can be found in Klatt and Engell (1998). Additionally, the system model, our controller implementation

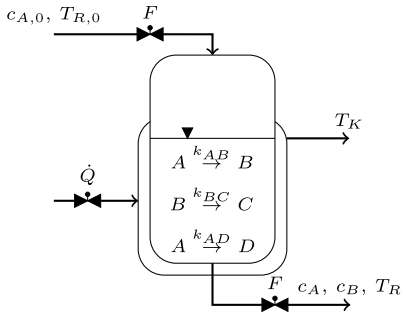

Fig. 2. Continuously stirred tank rector producing product B from educt A. The inflow $F = \dot{V}/V_R$ and heat flow $\dot{Q}$ removed from the coolant are used to control the system.

Table 3
Lower (lb) and upper (ub) bound for the CSTR system shown in Fig. 2.

	$\boldsymbol{x}$				$\boldsymbol{u}$	
	c_A mol L^{-1}	c_B mol L^{-1}	T_R °C	T_K °C	F h^{-1}	$\dot{Q}$ kJ h^{-1}
lb	0.1	0.1	50.0	50.0	5.0	−8500.0
ub	2.0	2.0	135.0	140.0	100.0	0.0

and all results can be found online.[5] To investigate the robust multi-stage approach, two important parameters, the activation energy $E_{A,AD}$ and the rate constant $k_{0,AB}$ are considered uncertain. As in previous work (Lucia, 2015), this is achieved by introducing multipliers $\alpha \in [0.95,\ 1.05]$ and $\beta \in [0.9,\ 1.1]$, such that $\hat{E}_{A,AD} = \alpha E_{A,AD}$ and $\hat{k}_{0,AB} = \beta k_{0,AB}$.

The control task in this investigation is to track a desired concentration of the product c_B, denoted as c_B^{set}, while complying with the constraints shown in Table 3. To this end, the MPC cost function in (3) is formulated with $l(c_B) = m(c_B) = (c_B - c_B^{\text{set}})^2$, and $\boldsymbol{r} = [1, 1 \cdot 10^{-4}]$ to weight the input-rate penalization.

To control the CSTR system, we propose to use a robust multi-stage MPC and investigate a nominal MPC controller for comparison. Both controllers are configured with a horizon of $N = 20$ and time-step of $\Delta t = 18$ s. Furthermore, the upper bound for T_R is implemented as a soft-constraint in both formulations.

As the only difference, the robust multi-stage MPC controller has a robust horizon of one and considers the Cartesian product $\mathbb{A} \times \mathbb{B}$ of the sets $\mathbb{A} = \{0.95,\ 1, 1.05\}$ and $\mathbb{B} = \{0.9,\ 1, 1.1\}$ as scenarios of the uncertain parameters α and β. We investigate the controller performance for a closed-loop simulation with fixed initial state x_0 and set-point $c_B^{\text{set}} = 1$ mol L^{-1}. The results are shown in Fig. 3 for two scenarios: In Fig. 3(a), the plant is simulated with the nominal values $\alpha = \beta = 1$, whereas in Fig. 3(b), we randomly chose $\alpha = 0.96$ and $\beta = 1.02$ for the simulated plant. For the first scenario in Fig. 3(a), it can be seen that the nominal MPC controller performs excellent, reaching the desired set-point faster than the robust multi-stage controller. The reason for the more conservative control action of the robust multi-stage MPC can be explained with the predicted scenarios which are also shown in Fig. 3. Some of these scenarios suggest a rapid increase of the reactor temperature T_R, suggesting that a significant backoff from the constraint is necessary.

In the second scenario, the simulation model is setup with values of α and β that differ from the nominal values. The results in Fig. 3(b) show that in this scenario the nominal MPC controller immediately violates the temperature constraints in Table 3. While the controller continuously yields feasible solutions due to the use of soft-constraints, the performance is insufficient. On the other hand, the robust multi-stage MPC safely operates the CSTR without violating the constraints.

[5] https://github.com/pas-tudo/2023_do_mpc_paper.

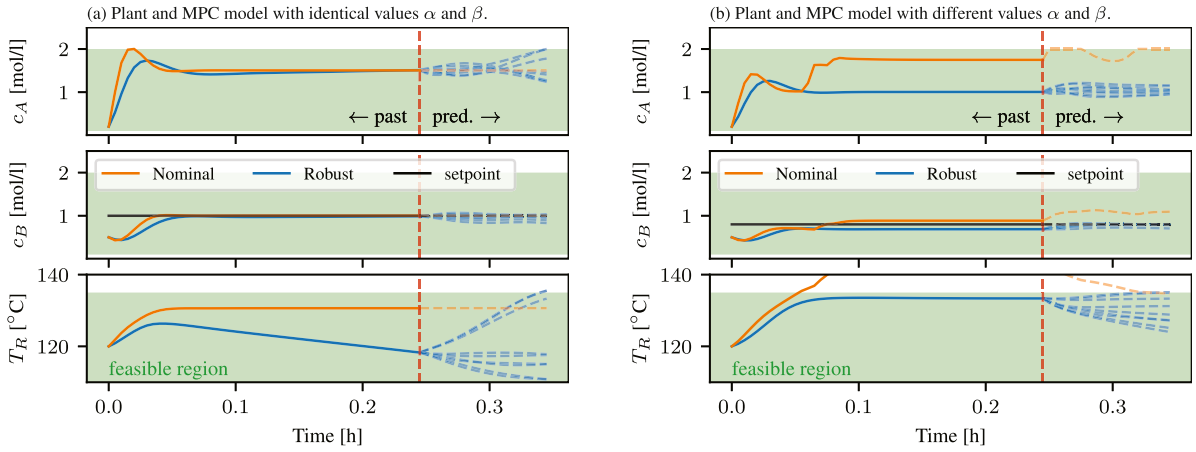

Fig. 3. Comparison of multi-stage robust and nominal MPC for the CSTR system in Fig. 2. In scenario (a) the nominal MPC controller uses a model with the same values of α and β as in the simulation model (plant). In scenario (b) the parameters of the plant are uncertain and not identical to those used in nominal MPC. The feasible region, defined by the bounds in Table 3, is highlighted in green. . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Note that rigorous guarantees for the constraint satisfaction are not the focus of this illustrative example but can be obtained as shown in Lucia, Paulen, and Engell (2014).

4.5. Deployment with OPC UA

To deploy the proposed MPC solution shown in the previous subsection, do-mpc features an interface to OPC UA by extending the OPC UA-Asyncio library.[6] The OPC UA machine-to-machine communication protocol is commonly used in process industries to connect sensors and actuators of chemical plants. In a nutshell, a central server is connected to multiple clients that can read states of the plant and write control actions. The core do-mpc modules shown in Fig. 1 can be used to automatically derive a client. Clients are then connected to a specified server and assigned read or write access to their respective fields. Additionally, do-mpc can automatically configure a local server for software-in-the-loop (Abel & Bollig, 2006) testing.

For the demonstration of the OPC UA interface of do-mpc, we present a software-in-the-loop investigation of the CSTR example shown in Fig. 2. We initialize a nominal MPC controller and a simulator with identical values for the uncertain parameters α and β, as in Fig. 3(a). Both objects are then used to derive an OPC client and to configure a local server with the desired fields. Finally, the simulator and the MPC controller are launched in individual processes and triggered at sampling times $\Delta t_{\text{MPC}} = 18$ s and $\Delta t_{\text{Sim}} = 2$ s. In this way, the obtained closed-loop trajectory realistically considers time-delays that arise from the computation time of the MPC controller, communication delays, and discrete sampling intervals. In our investigation, these effects contributed to an increase of 2.88% of the closed-loop cost and no constraint violations.

5. MPC with interoperable data-based models

As shown in the previous section, do-mpc enables the synthesis of robust, nonlinear and economic MPC and facilitates the deployment with OPC UA. However, a central prerequisite is the availability of a control oriented model, which is unavailable for many applications.

To tackle this challenge, we consider data-based system identification, in particular with neural networks, a promising solution. Apart from their ability to learn complex nonlinear behavior from large datasets, neural networks also benefit from their ease of applicability due to toolboxes such as Matlab, Pytorch and Tensorflow. The drawback of these tools, however, is that they cannot be directly incorporated in most MPC frameworks. To fill this gap, we contribute a tool that can parse neural network models from the ONNX standard in do-mpc. Through interoperability with ONNX, do-mpc can directly import deep learning models that were previously created and trained, for example in Matlab, Pytorch and Tensorflow. Furthermore, do-mpc facilitates the data-generation process for system identification, especially if this data is obtained from a high-fidelity non-control oriented model. To this end, do-mpc features the sampling module which can be used to efficiently generate reproducible closed-loop data from the high-fidelity model.

In this section, we discuss neural network system identification and present the ONNX conversion module as well as the sampling module shown in Fig. 1.

5.1. System identification with neural networks

We introduce system identification as a regression task with dataset $\mathcal{D} = \{\mathcal{V}, \mathcal{T}\}$ consisting of m data pairs of inputs $\boldsymbol{v} \in \mathbb{R}^{n_v}$ and targets $\boldsymbol{t} \in \mathbb{R}^{n_t}$ from which the sets $\mathcal{V} = \left\{\boldsymbol{v}^{(i)}\right\}_{i=1}^{m}$ and $\mathcal{T} = \left\{\boldsymbol{t}^{(i)}\right\}_{i=1}^{m}$ are formed. In the simplest case, measured sequences of all states $\boldsymbol{x}_{[k:k+m+1]}$ and inputs $\boldsymbol{u}_{[k:k+m]}$ are available and we have $\boldsymbol{v}^{(i)} = \left[\boldsymbol{x}_i^\top, \boldsymbol{u}_i^\top\right]^\top$ and $\boldsymbol{t}^{(i)} = \boldsymbol{x}_{i+1}$. The incorporation of further parameters and algebraic states, as introduced in (1), is straightforward and omitted here for clarity.

A feed-forward neural network with L hidden layers, yielding prediction $\hat{\boldsymbol{t}}$, is introduced as:

$$\hat{\boldsymbol{t}} = NN(\boldsymbol{v}; \mathbb{W}_{L+1}) = g_{L+1} \circ h_{L+1} \circ \ldots \circ g_1 \circ h_1(\boldsymbol{v}), \tag{6}$$

where $\circ$ denotes function composition, $g_l(\cdot), \forall l = 1, \ldots, L+1$ are activation functions and

$$\boldsymbol{h}_l = h_l(\boldsymbol{\phi}_{l-1}) = \left[\boldsymbol{\phi}_{l-1}^\top,\ 1\right] \boldsymbol{W}_l, \tag{7a}$$

$$\boldsymbol{\phi}_l = g_l(\boldsymbol{h}_l), \tag{7b}$$

are the operations with n_{ϕ_l} neurons in layer l. Additionally, we have $\boldsymbol{\phi}_0 = \boldsymbol{v}$ for the input layer. The set of weight matrices with cardinality $L+1$ is denoted $\mathbb{W}_{L+1} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_{L+1}\}$. For a regression task it is common to minimize the mean-squared-error loss function:

$$L_{\text{MSE}} = \frac{1}{m} \sum_{i=1}^{m} \left\| NN\left(\boldsymbol{v}^{(i)}; \mathbb{W}_{L+1}\right) - \boldsymbol{t}^{(i)} \right\|_2^2, \tag{8}$$

where $\|\cdot\|_2$ denotes the 2-norm. The optimal set of weights is then obtained with:

$$\mathbb{W}_{L+1}^* = \arg\min_{\mathbb{W}_{L+1}} L_{\text{MSE}}\left(\mathcal{D}; \mathbb{W}_{L+1}\right). \tag{9}$$

[6] https://github.com/FreeOpcUa/opcua-asyncio.

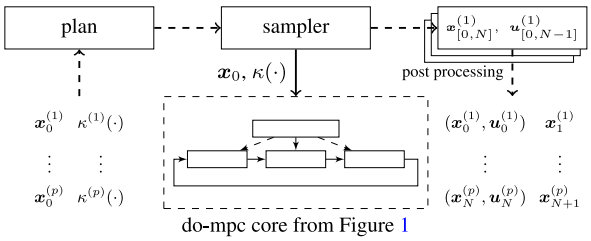

Fig. 4. Data-generation for system identification with the do-mpc sampling tools.

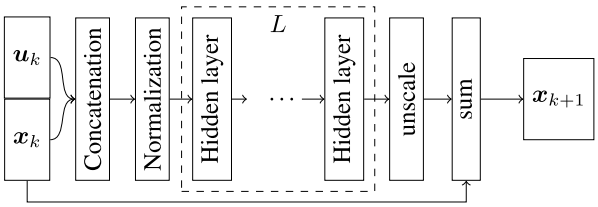

Fig. 5. Investigated neural network architecture for system identification of the CSTR system shown in Fig. 2. The neural network is exported as an ONNX model from Tensorflow converted automatically to a discrete-time do-mpc model.

With the dataset $\mathcal{D}$ as described above, the trained neural network thus approximates the system as:

$$\hat{\boldsymbol{x}}_{k+1} = NN(\boldsymbol{x}_k, \boldsymbol{u}_k; \mathbb{W}^*_{L+1}). \tag{10}$$

For observable systems with only output-feedback $\boldsymbol{y}_k$, we can instead identify a data-based model with nonlinear auto-regressive structure and exogeneous inputs (NARX) (Bonassi, Farina, & Scattolini, 2021):

$$\hat{\boldsymbol{y}}_{k+1} = NN^{\text{NARX}}(\boldsymbol{y}_{[k-l:k]}, \boldsymbol{u}_{[k-l:k]}; \mathbb{W}^{\text{NARX}}_{L+1}). \tag{11}$$

The parameter l in (11) denotes the model order and is typically unknown. It is often determined from system knowledge (Bonassi et al., 2021), trial and error, or more rigorously through model comparison using an information criterion (Baldacchino, Anderson, & Kadirkamanathan, 2012).

5.2. Data generation

The obvious prerequisite for data-based system identification is the availability of a training dataset $\mathcal{D}$. This dataset must contain sufficiently many relevant samples which explore the state-space of the system and contain persistently exciting inputs.

In order to create surrogates for non-control oriented or very large models, it can be required to create this dataset by sampling a pre-existing simulation model. To facilitate this process, do-mpc contains a sampling module which is depicted in Fig. 4. In a nutshell, the sampling module contains three elements: the planner, the sampler and features for post-processing. The planner is designed to facilitate the process of planning conditions for the desired samples. For the application of system identification, these conditions can be the initial state of the system $\boldsymbol{x}_0$ and an input strategy $\kappa(\boldsymbol{x})$. The sampler is designed to call an arbitrary sample generating function for each of the p planned cases. For the system identification task, this function can be a simulation of the system, depending on the initial state $\boldsymbol{x}_0^{(i)}$ and an input strategy $\kappa^{(i)}(\cdot)$ for $i \in \mathbb{I}_{[1,p]}$. The resulting data, as well as the plan, are then stored and can be loaded for post-processing. The strategy of planning, generating and post-processing samples is summarized in Fig. 4.

With the described data-generation approach in do-mpc it is therefore sufficient to share the plan and the sample generating function to retrieve the full dataset. Furthermore, both elements are human-readable and reveal the designed data-generation strategy. In this way, do-mpc facilitates transparent research and allows for reproducible and reusable code and results.

5.3. Simulation study

To illustrate neural network system identification with the described do-mpc modules, that is, the ONNX conversion tool and the sampling tool, we investigate the CSTR example introduced in Section 4.4. To simplify the demonstration, we assume that full state-feedback is possible and identify a neural network state-space model in the form of (10). Data is generated as depicted in Fig. 4 by evaluating the exact simulation model recursively for $N = 50$ steps. A sampling case $i \in \mathbb{I}_{[1,p]}$ is defined in terms of the initial state $\boldsymbol{x}_0^{(i)}$, the set-point $c_b^{\text{set}(i)}$ and an input strategy defined by $\alpha^{(i)} \in [0, 1]$, such that:

$$\kappa^{(i)}(\boldsymbol{x}) = \left(1 - \alpha^{(i)}\right) \kappa_{\text{LQR}}(\boldsymbol{x}) + \alpha^{(i)} \kappa_{\text{RND}}(\boldsymbol{x}), \tag{12}$$

where $\kappa_{\text{RND}}(\boldsymbol{x})$ returns uniform random inputs, within the defined bounds in Table 3, and with random hold time. The LQR controller $\kappa_{\text{LQR}}(\boldsymbol{x})$ is directly obtained with do-mpc, as described in Section 4.3. By combining the set-point tracking LQR with random inputs, the data contains plausible input sequences around the desired set-point as well as more exploratory sequences. This approach could also be applied to certain unstable systems by restricting $\alpha \ll 1$.

To illustrate the capabilites of the ONNX conversion tool, we design a more complex neural network architecture in comparison to the formulation in (6). The proposed architecture shown in Fig. 5 has multiple inputs, contains concatenation and normalization layers and has a residual network structure (He, Zhang, Ren, & Sun, 2016).

For the CSTR example, the architecture in Fig. 5 is configured with $L = 3$ and $n_\phi = 32$ neurons in each layer, and is successively trained with the generated dataset consisting of 50.000 samples. Ultimately, the resulting data-based model should be used in the MPC formulation (2) and is therefore evaluated in this context. In a similar fashion as for the data-generation task, we use the do-mpc sampling tools to generate $p = 50$ test cases in terms of the initial state $\boldsymbol{x}_0^{(i)}$, the set-point $c_b^{\text{set (i)}}$ and use the MPC with NN system model to generate closed-loop trajectories with $N = 50$ steps. The performance is then compared to an MPC controller with exact system model.

The results of this comparison are shown in Fig. 6 in terms of the closed-loop cost and maximum constraint violation, both depicted in relationship to the set-point c_b^{set}. It shows that the MPC controller with data-based model performs almost equivalent to the variant with exact system model. The closed-loop cost is nearly indistinguishable with minor exceptions in two out of the $p = 50$ investigated cases. The maximum constraint violation is larger for the MPC with neural network system model, especially for the state c_A, but still small, not exceeding 0.02 mol L^{-1}. If rigorous performance guarantees of the obtained data-based controller are required, the presented methods can be readily combined with the probabilistic validation approach presented in Karg et al. (2021). In particular, this approach can also be used to determine a suitable number of investigated samples p.

6. Simple deployment with approximate MPC

The deployment of MPC remains another core challenge for the adoption of advanced MPC. This problem is further intensified when robust MPC is needed, for systems with very fast sampling times or when the controller should be deployed on restricted hardware. Hardware limitation occur, for example, due to memory limitations or restrictions of computation power, such as in the case of power electronics (Guillén et al., 2022) or for fast mechanical systems. A promising solution in this setting can be approximate MPC (Chen et al., 2018; Karg & Lucia, 2020; Kumar et al., 2021; Paulson & Mesbah, 2020), which is presented in the following. Furthermore, we showcase how the FAIR solutions outlined

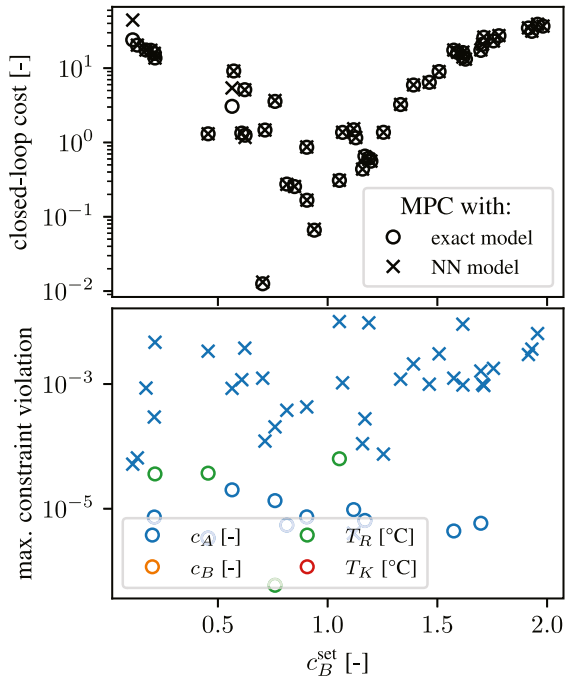

Fig. 6. MPC controller for the CSTR in Fig. 2 with exact system model and with approximate NN system model. Comparison of closed-loop cost and maximum constraint violation over $N = 50$ steps and $p = 50$ samples defined by $\boldsymbol{x}_0$ and c_B^{set}. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

in Section 2.1, and which are implemented in do-mpc, can facilitate the synthesis of reproducible and transparent approximate MPC.

The fundamental idea of approximate MPC is that the implicit MPC control law (4) should be replaced with an explicit function. This is reminiscent of explicit MPC (Bemporad, Morari, Dua, & Pistikopoulos, 2002), which is typically realized by partitioning the state-space and applying a different linear control law in each region (Karg & Lucia, 2020). While explicit MPC is exact in special cases, its memory footprint can become intractable for larger problems (Karg & Lucia, 2020).

Approximate MPC uses function approximation to obtain an efficient but inexact MPC law $\hat{\kappa}_{\text{MPC}}(\boldsymbol{x}_k)$. Neural networks, due to being universal function approximators (Hornik, Stinchcombe, & White, 1989), are particularly well suited for this task. Furthermore, it can be shown that neural networks can exactly represent certain MPC laws (Karg & Lucia, 2020). We therefore introduce:

$$\hat{\boldsymbol{u}}_k = \hat{\kappa}_{\text{MPC}} = NN(\boldsymbol{x}_k; \mathbb{W}^*_{L+1}), \tag{13}$$

as the approximate MPC controller using a neural network in the form of (6) and with optimized weights $\mathbb{W}^*_{L+1}$. For the sake of clarity, we again omit further potential inputs, e.g. parameters, in (13).

Training data must be obtained from the original MPC law (2) yielding a dataset $\mathcal{D} = \{\mathcal{V}, \mathcal{T}\}$, with $\mathcal{V} = \{\boldsymbol{x}_k^{(i)}\}_{i=1}^m$ and $\mathcal{T} = \{\kappa_{\text{MPC}}(\boldsymbol{x}_k^{(i)})\}_{i=1}^m$. The optimal parameters $\mathbb{W}^*_{L+1}$ can then be found by solving (9).

After successfully training the neural network, only the approximate MPC control law (13) is evaluated online which can be orders of magnitude faster than solving an optimization problem (Kumar et al., 2021).

6.1. Sample efficient approximate MPC with sensitivities

A challenge for the synthesis of an approximate MPC control law is the requirement to repeatedly solve the optimal control problem (2).

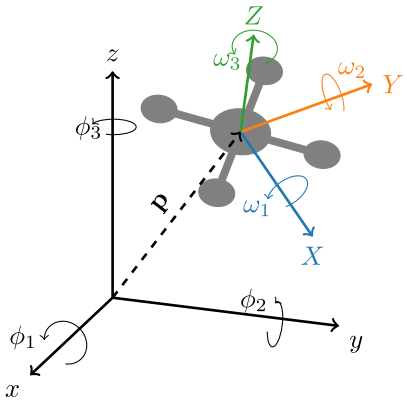

Fig. 7. Investigated quadcopter system.

Additionally, large systems and complex control task may require more data and also have longer solution times, potentially rendering the process prohibitive (Krishnamoorthy, 2022). As a remedy, additional information can be extracted from each sampled data-point with the sensitivity matrix $\boldsymbol{S} \in \mathbb{R}^{n_u \times n_x}$:

$$\boldsymbol{S} = \frac{\partial \kappa_{\text{MPC}}}{\partial \boldsymbol{x}_k} = \frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{x}_k}. \tag{14}$$

Intuitively, the sensitivity $\boldsymbol{S}$ represents the change of the optimal control law κ_{MPC} in (4) with respect to the current system state $\boldsymbol{x}_k$.

Two challenges remain. First, sensitivity information must be obtained from the optimal control problem. This can be achieved in a solver agnostic fashion via implicit differentiation of the Karush-Kuhn–Tucker conditions (Blondel et al., 2022; Fiacco & Ishizuka, 1990; Krishnamoorthy, 2022), and is readily available in do-mpc. The second challenge is to incorporate the sensitivity information when training the neural network. To this end, the Sobolev loss (Cocola & Hand, 2020; Czarnecki et al., 2017) can be used:

$$L_{\text{Sob}} = L_{\text{MSE}} + \frac{\gamma}{m} \sum_{i=1}^{m} \left\| \frac{\partial NN}{\partial \boldsymbol{x}_k}\bigg|_{\boldsymbol{x}_k^{(i)}} - \boldsymbol{S}^{(i)} \right\|_F^2, \tag{15}$$

with $\|\cdot\|_F$ denoting the Frobenius norm and $\gamma \geq 0$ being a weighting factor. Sobolev training requires that the training data is augmented with the sensitivity information, $\mathcal{S} = \{\boldsymbol{S}^{(i)}\}_{i=1}^m$, resulting in $\mathcal{D}_{\text{Sob}} = \mathcal{D} \cup \{\mathcal{S}\}$.

As previously shown in Lüken et al. (2023) and Winqvist et al. (2021), the approximate MPC control law $\hat{\kappa}_{\text{Sob}}(\boldsymbol{x}_k)$ obtained with Sobolev training can result in better controller performance with fewer training samples.

6.2. Simulation study

To showcase the application of approximate MPC with and without sensitivity information, we investigate a nonlinear, continuous-time quadcopter model previously introduced in Elokda, Coulson, Beuchat, Lygeros, and Dörfler (2021) and with parameters from Förster (2015). The full model formulation can also be found online. The model, shown in Fig. 7, has $n_x = 12$ states, that is, position $\boldsymbol{p}$, velocity $\dot{\boldsymbol{p}}$, orientation $\boldsymbol{\phi}$ and angular velocity $\boldsymbol{\omega}$ in three dimensions, and $n_u = 4$ control inputs, representing the thrust of the individual rotors. The control task is to follow a trajectory of position set-points $\boldsymbol{p}_{\text{set}}$, while maintaining the orientation and with penalties on velocity, angular velocity and for changes of the control inputs. For the stage cost and terminal cost in (3), we define:

$$l(\boldsymbol{x}) = m(\boldsymbol{x}) = \|\boldsymbol{p} - \boldsymbol{p}_{\text{set}}\|_2^2 + 10^{-3}\|\dot{\boldsymbol{p}}\|_2^2 + 10^{-2}\|\boldsymbol{\phi}\|_2^2 + 5 \cdot 10^{-3}\|\boldsymbol{\omega}\|_2^2,$$

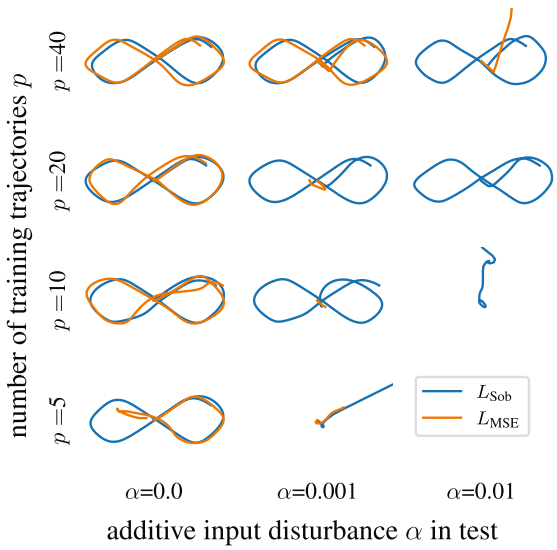

Fig. 8. Closed-loop trajectories with approximate MPC of quadcopter in Fig. 7 in (p_x, p_y)-plane. Comparison of approximate MPC controller trained with sensitivities (Sobolev loss L_{Sob}) vs. training with MSE (L_{MSE}) and for different numbers of trajectories used for training and different magnitude of additive input disturbance α. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and $\boldsymbol{r}^\top = [5,\ 5,\ 5,\ 5]$. The optimal control problem in (2) is formulated with a horizon of $N = 30$, time-step $\Delta t = 50$ ms, and with constrained control inputs $0 \leq \boldsymbol{u} \leq 0.3$.

To obtain training data for the approximate MPC controller, we sample closed-loop trajectories following variants of figure-eight curves in the (p_x, p_y)-plane with oscillating set-point for the height p_z. Furthermore, we provoke stabilizing feedback action by applying an additive disturbance to the inputs, that is, $\boldsymbol{u}_k = \kappa_{\text{MPC}}(\boldsymbol{x}_k) + \alpha\kappa_{\text{RND}}(\boldsymbol{x}_k)$, where $\kappa_{\text{RND}}(\boldsymbol{x}_k)$ returns random uniform values and is scaled with α. However, only the actual MPC control input $\kappa_{\text{MPC}}(\boldsymbol{x}_k)$, not $\boldsymbol{u}_k$, is stored for training the approximate MPC controller. Sampling is conducted, similarly as shown in Section 5, with the do-mpc sampling tools by varying parameters of the set-point trajectory and α. A total of $p = 50$ trajectories, each containing 200 steps (representing eight seconds of simulation time), are collected. Additionally, we use do-mpc to obtain sensitivity information in the form of (14).

We investigate a neural network with $L = 1$ hidden layers and $n_\phi = 80$ neurons which is trained either by minimizing the mean-squared-error loss L_{MSE} or the Sobolev loss L_{Sob} with $\gamma = 100$ to incorporate the sensitivity information. Additionally, we train multiple approximate MPC controller by varying the number of trajectories used for training from $p = 5$ to $p = 40$ and keep 10 trajectories for validation. Training is conducted in all cases for at most 5000 steps and with an early stopping criterion monitoring the validation loss.

The resulting approximate MPC variants are then examined in a closed-loop investigation, aiming to track a previously unseen trajectory and for different values of the additive input noise scaling factor α.

The resulting closed-loop trajectory in the (p_x, p_y)-plane is shown in Fig. 8. While multiple variants of the approximate MPC lead to satisfactory control performance, the advantages of approximate MPC trained with Sobolev loss (15) are evident. By considering the sensitivity information from (14), training with only $p = 5$ trajectories of the MPC controlled system can be sufficient for closed-loop stability, albeit with insufficient robustness to disturbances. Increasing the number of samples yields also suitable performance of the approximate MPC with regular training. However, only the approximate MPC with Sobolev training can still control the system with significant input disturbance ($\alpha = 10^{-2}$) after having seen at least $p = 20$ trajectories in the training data. The main advantage of the approximate MPC is that its evaluation takes on average 4.2 ± 2.7 ms, whereas solving the original optimal control problem requires 116 ± 23 ms.

7. Conclusions

Nonlinear, robust and economic model predictive control has shown outstanding results, leading to significant energy savings, yield improvements or solving previously unsolvable tasks. However, these results are often not widely adopted outside the core MPC research community, leaving a large untapped potential. In this work, we discuss four main challenges that hinder the widespread adoption of advanced MPC in research and industry. These challenges are the unavailability of models, the challenges associated with deploying advanced MPC, the scarcity of rapid prototyping tools and the difficulties in reproducing advanced MPC results. We find that the FAIR principles for scientific data management and research software offer important guidelines on how to tackle these challenges. First, we show that neural network system identification plays an important role in obtaining control-oriented models. Through interoperability with the ONNX standard, neural network models trained in popular toolboxes like Tensorflow and Pytorch can be directly employed in an MPC formulation. Deployment of advanced MPC is also enabled by interoperability with the OPC UA standard or can be achieved through approximate MPC. Especially for approximate MPC, system identification and controller validation, we highlight the importance of FAIR data generation and propose a versatile data sampling framework. Rapid prototyping is enabled by accessible open-source software and also benefits from reproducible data sampling, for example when updating an approximate MPC controller.

As a concrete implementation of the discussed FAIR solutions, we present the open-source software do-mpc. Apart from enabling the rapid development of multi-stage robust, nonlinear and economic MPC, we provide the important interfaces to ONNX and OPC UA and facilitate system identification, controller validation and approximate MPC through a sampling module. With the sensitivity module, we enable data-efficient approximate MPC. With do-mpc, obstacles towards the first steps of advanced MPC are minimized and the tool can support engineers from these first steps to the deployment of powerful control solutions.

In future work, we seek to further improve do-mpc contributing features such as the extended Kalman filter and tube-based robust MPC, and intend to conduct further field testing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research leading to these results has received funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant agreement number 423857295 and 466380688 – within the priority program “SPP 2331: Machine Learning in Chemical Engineering”.

The authors would also like to thank Alexandru Tatulea-Codrean and Sebastian Engell for discussions on OPC UA and earlier ideas for do-mpc. The authors also thank Joel A.E. Andersson for intensive discussions on CasADi.

Finally, the authors want to express their sincere gratitude to the reviewers for their extensive and constructive feedback. It has lead to interesting discussions, new thoughts and important improvements of this work.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467.

Abel, D., & Bollig, A. (Eds.), (2006). Rapid control prototyping. In *Rapid control prototyping: methoden und anwendungen* (pp. 295–318). Springer.

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi: A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36.

Artrith, N., Butler, K. T., Coudert, F.-X., Han, S., Isayev, O., Jain, A., et al. (2021). Best practices in machine learning for chemistry. *Nature Chemistry*, *13*(6), 505–508.

Baldacchino, T., Anderson, S. R., & Kadirkamanathan, V. (2012). Structure detection and parameter estimation for NARX models in a unified EM framework. *Automatica*, *48*(5), 857–865.

Bemporad, A., Morari, M., Dua, V., & Pistikopoulos, E. N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, *38*(1), 3–20.

Biegler, L. T. (2010). *Nonlinear programming: concepts, algorithms, and applications to chemical processes, vol. 10*. Siam.

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-Lopez, F., et al. (2022). Efficient and modular implicit differentiation. *Advances in Neural Information Processing Systems*, *35*, 5230–5242.

Blum, D., Wang, Z., Weyandt, C., Kim, D., Wetter, M., Hong, T., et al. (2022). Field demonstration and implementation analysis of model predictive control in an office HVAC system. *Applied Energy*, *318*, Article 119104.

Bonassi, F., Farina, M., & Scattolini, R. (2021). Stability of discrete-time feed-forward neural networks in NARX configuration. *IFAC-PapersOnLine*, *54*(7), 547–552.

Chen, Y., Bruschetta, M., Picotti, E., & Beghi, A. (2019). MATMPC - A MATLAB based toolbox for real-time nonlinear model predictive control. In *18th European control conference* (pp. 3365–3370).

Chen, S., Saulnier, K., Atanasov, N., Lee, D. D., Kumar, V., Pappas, G. J., et al. (2018). Approximating explicit model predictive control using constrained neural networks. In *American control conference* (pp. 1520–1527).

Chue Hong, N. P., Katz, D. S., Barker, M., Lamprecht, A. L., Martinez, C., Psomopoulos, F. E., et al. (2021). FAIR principles for research software (FAIR4RS principles), research data alliance. *Research Data Alliance*.

Cocola, J., & Hand, P. (2020). Global convergence of Sobolev training for overparameterized neural networks. In G. Nicosia, V. Ojha, E. La Malfa, G. Jansen, V. Sciacca, P. Pardalos, G. Giuffrida, & R. Umeton (Eds.), *Machine learning, optimization, and data science, vol. 12565* (pp. 574–586). Springer International Publishing.

Czarnecki, W. M., Osindero, S., Jaderberg, M., Swirszcz, G., & Pascanu, R. (2017). Sobolev training for neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems, vol. 30*. Curran Associates, Inc..

Elokda, E., Coulson, J., Beuchat, P. N., Lygeros, J., & Dörfler, F. (2021). Data-enabled predictive control for quadcopters. *International Journal of Robust and Nonlinear Control*, *31*(18), 8916–8936.

Englert, T., Völz, A., Mesmer, F., Rhein, S., & Graichen, K. (2019). A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC). *Optimization and Engineering*, *20*(3), 769–809.

Fiacco, A. V., & Ishizuka, Y. (1990). Sensitivity and stability analysis for nonlinear programming. *Annals of Operations Research*, *27*(1), 215–235.

Findeisen, R., Graichen, K., & Mönnigmann, M. (2018). Eingebettete Optimierung in der Regelungstechnik – Grundlagen und Herausforderungen. *at - Automatisierungstechnik*, *66*(11), 877–902.

Forbes, M. G., Patwardhan, R. S., Hamadah, H., & Gopaluni, R. B. (2015). Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine*, *48*(8), 531–538.

Förster, J. (2015). *System identification of the crazyflie 2.0 nano quadrocopter* (BSc. thesis), ETH Zurich.

Goel, G., & Hassibi, B. (2022). The power of linear controllers in LQR control. In *IEEE 61st conference on decision and control* (pp. 6652–6657).

Guillén, P., Fiedler, F., Sarnago, H., Lucía, S., & Lucía, O. (2022). Deep learning implementation of model predictive control for multioutput resonant converters. *IEEE Access*, *10*, 65228–65237.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Karg, B., Alamo, T., & Lucia, S. (2021). Probabilistic performance validation of deep learning-based robust NMPC controllers. *International Journal of Robust and Nonlinear Control*, *31*(18), 8855–8876.

Karg, B., & Lucia, S. (2020). Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, *50*(9), 3866–3878.

Klatt, K. U., & Engell, S. (1998). Gain-scheduling trajectory control of a continuous stirred tank reactor. *Computers & Chemical Engineering*, *22*(4), 491–502.

Kouro, S., Perez, M. A., Rodriguez, J., Llor, A. M., & Young, H. A. (2015). Model predictive control: MPC's role in the evolution of power electronics. *IEEE Industrial Electronics Magazine*, *9*(4), 8–21.

Kouvaritakis, B., & Cannon, M. (2016). *Advanced textbooks in control and signal processing, Model predictive control*. Springer International Publishing.

Krishnamoorthy, D. (2022). A sensitivity-based data augmentation framework for model predictive control policy approximation. *IEEE Transactions on Automatic Control*, *67*(11), 6090–6097.

Kumar, P., Rawlings, J. B., & Wright, S. J. (2021). Industrial, large-scale model predictive control with structured neural networks. *Computers & Chemical Engineering*, *150*, Article 107291.

Ljung, L. (2010). Perspectives on system identification. *Annual Reviews in Control*, *34*(1), 1–12.

Ljung, L., Andersson, C., Tiels, K., & Schön, T. B. (2020). Deep learning and system identification. *IFAC-PapersOnLine*, *53*(2), 1175–1181.

Lucia, S. (2015). *Robust multi-stage nonlinear model predictive control* (Ph.D. thesis), TU Dortmund.

Lucia, S., Finkler, T., & Engell, S. (2013). Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, *23*(9), 1306–1319.

Lucia, S., Paulen, R., & Engell, S. (2014). Multi-stage nonlinear model predictive control with verified robust constraint satisfaction. In *53rd IEEE conference on decision and control* (pp. 2816–2821).

Lucia, S., Tătulea-Codrean, A., Schoppmeyer, C., & Engell, S. (2017). Rapid development of modular and sustainable nonlinear model predictive control solutions. *Control Engineering Practice*, *60*, 51–62.

Lüken, L., Brandner, D., & Lucia, S. (2023). Sobolev training for data-efficient approximate nonlinear MPC. In *22nd IFAC world congress*.

Marzullo, T., Dey, S., Long, N., Leiva Vilaplana, J., & Henze, G. (2022). A high-fidelity building performance simulation test bed for the development and evaluation of advanced controls. *Journal of Building Performance Simulation*, *15*(3), 379–397.

Mayne, D. Q., Kerrigan, E. C., van Wyk, E. J., & Falugi, P. (2011). Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, *21*(11), 1341–1353.

Mesbah, A. (2016). Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems*, *36*(6), 30–44.

Morari, M., & H. Lee, J. (1999). Model predictive control: Past, present and future. *Computers & Chemical Engineering*, *23*(4), 667–682.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32*.

Patria, D., Rossi, C., Fernandez, R. A. S., & Dominguez, S. (2021). Nonlinear control strategies for an autonomous wing-in-ground-effect vehicle. *Sensors*, *21*(12), 4193.

Paulson, J. A., & Mesbah, A. (2020). Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction. *IEEE Control Systems Letters*, *4*(3), 719–724.

Pfrommer, S., & Sojoudi, S. (2022). LQR control with sparse adversarial disturbances. In *IEEE 61st conference on decision and control* (pp. 2346–2353). IEEE.

Rawlings, J. B., Angeli, D., & Bates, C. N. (2012). Fundamentals of economic model predictive control. In *51st IEEE conference on decision and control* (pp. 3851–3861).

Risbeck, M., & Rawlings, J. (2016). MPCTools: Nonlinear model predictive control tools for CasADi.

Scampicchio, A., Aravkin, A., & Pillonetto, G. (2021). Stable and robust LQR design via scenario approach. *Automatica*, *129*, Article 109571.

Schleipen, M., Gilani, S.-S., Bischoff, T., & Pfrommer, J. (2016). OPC UA & industrie 4.0 - enabling technology with high diversity and variability. *Procedia CIRP*, *57*, 315–320.

Sopasakis, P., Fresk, E., & Patrinos, P. (2020). OpEn: code generation for embedded nonconvex optimization. *IFAC-PapersOnLine*, *53*(2), 6548–6554.

Vazquez, S., Leon, J. I., Franquelo, L. G., Rodriguez, J., Young, H. A., Marquez, A., et al. (2014). Model predictive control: A review of its applications in power electronics. *IEEE Industrial Electronics Magazine*, *8*(1), 16–31.

Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., van Duijkeren, N., Zanelli, A., et al. (2022). Acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, *14*, 147–183.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature methods*, *17*(3), 261–272.

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25–57.

Wang, Y. (2017). A new concept using LSTM neural networks for dynamic system identification. In *American control conference* (pp. 5324–5329).

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., et al. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*(1), Article 160018.

Winqvist, R., Venkitaraman, A., & Wahlberg, B. (2021). Learning models of model predictive controllers using gradient data. *IFAC-PapersOnLine*, *54*(7), 7–12.

Wise, J., de Barron, A. G., Splendiani, A., Balali-Mood, B., Vasant, D., Little, E., et al. (2019). Implementation and relevance of FAIR data principles in biopharmaceutical R&D. *Drug Discovery Today*, *24*(4), 933–938.

Yao, Y., & Shekhar, D. K. (2021). State of the art review on model predictive control (MPC) in heating ventilation and air-conditioning (HVAC) field. *Building and Environment*, *200*, Article 107952.

Zanelli, A., Domahidi, A., Jerez, J., & Morari, M. (2020). FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, *93*(1), 13–29.

Zeilinger, M. N., Morari, M., & Jones, C. N. (2014). Soft constrained model predictive control with robust stability guarantees. *IEEE Transactions on Automatic Control*, *59*(5), 1190–1202.

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 16636–16643

Economic nonlinear predictive control of water distribution networks based on surrogate modeling and automatic clustering

Felix Fiedler [*,***] **Andrea Cominola** [**,***] **Sergio Lucia** [*,***]

[*] *Chair of Internet of Things for Smart Buildings, Technische Universität Berlin, Einsteinufer 17, 10587 Berlin, Germany*

[**] *Chair of Smart Water Networks, Technische Universität Berlin, Straße des 17. Juni 135, 10623, Berlin, Germany*

[***] *Einstein Center Digital Future, Technische Universität Berlin, Wilhelmstraße 67, 10117 Berlin, Germany*

Abstract: The operation of large-scale water distribution networks (WDNs) is a complex control task due to the size of the problem, the need to consider key operational, quality and safety-related constraints as well as because of the presence of uncertainties. An efficient operation of WDNs can lead to considerable reduction in the energy used to distribute the required amounts of water, leading to significant economic savings. Many model predictive control (MPC) schemes have been proposed in the literature to tackle this control problem. However, finding a control-oriented model that can be used in an optimization framework, which captures nonlinear behavior of the water network and is of a manageable size is a very important challenge faced in practice. We propose the use of a data-based automatic clustering method that clusters similar nodes of the network to reduce the model size and then learn a deep-learning based model of the clustered network. The learned model is used within an economic nonlinear MPC framework. The proposed method leads to a flexible scheme for economic robust nonlinear MPC of large WDNs that can be solved in real time, leads to significant energy savings and is robust to uncertain water demands. The potential of the proposed approach is illustrated by simulation results of a benchmark WDN model.

Keywords: model predictive control, water distribution networks, machine learning.

1. INTRODUCTION

Water distribution networks (WDNs) are large-scale interconnected systems that need to be operated reliably, while complying with operational constraints as well as water supply and quality standards, under varying climate and demand conditions. Water resources have the largest energy intensity in the urban metabolism of cities (Chini and Stillwell, 2018), with the water distribution phase often contributing the largest energy intensity share of water provision (Spang and Loge, 2015). Energy-related costs can constitute up to 65% of a utility's operating budget (Boulos et al., 2001).

The optimal management of WDNs can lead to considerable energy reduction and related economic savings. A case study in Valencia (Spain), for instance, demonstrated that potential savings of approximately 17% of the operational costs could be obtained via optimal pump scheduling and valve control, while a similar example in Wallan (Victoria, Australia) estimated a reduction in the average daily pumping costs of 20.6% with optimal operation of the WDNs (Broad et al., 2010). However, the optimal operation of such systems is a challenging task of increasing complexity and importance, due to the growing size of urban areas and the presence of multiple, potentially conflicting objectives.

Model predictive control (MPC) is an advanced control technique that can deal with nonlinear multivariable systems, including constraints and other control goals different than traditional set-point tracking (Mayne and Rawlings, 2009). For these reasons, it is an increasingly popular control method applied in many different fields and case studies and it has been already widely studied in the context of water distribution networks (Biscos et al., 2003; Cembrano et al., 2011; Ocampo-Martinez et al., 2012).

The application of MPC for WDNs faces two important challenges: (i) the presence of uncertainty and (ii) the size of the model for real-world networks. To deal with the uncertainty of the model parameters and water demands, different approaches based on chance constraints (Grosso et al., 2014), Gaussian processes (Wang et al., 2016) or scenario-tree methods (Sampathirao et al., 2018) have been recently proposed.

⋆ Felix Fiedler acknowledges the support of the Helmholtz Einstein International Berlin Research School in Data Science (HEIBRiDS), German Aerospace Center (DLR), and Technische Universität Berlin.

10.1016/j.ifacol.2020.12.793

Possible solutions to obtain computationally tractable MPC schemes for large-scale networks include the consideration of hierarchical (Duzinkiewicz et al., 2005; Grosso et al., 2013) and decentralized MPC schemes (Ocampo-Martinez et al., 2012) that avoid the centralized detailed optimization of all network elements at the same time. Alternatively, surrogate models that simplify the hydraulic equations used in the modeling of the WDNs can be used. Surrogate models based on quadratic functions were presented by Nowak et al. (2018) while linear models were used by Sampathirao et al. (2018). Some works have used artificial neural networks (ANNs) to obtain models to be coupled with genetic optimization algorithms (Broad et al., 2005), or deep learning techniques (Wu and Rahman, 2017) to derive predictive models. However, nonlinear surrogate models based on deep learning have not been exploited to be used in a gradient-based optimization framework typical of nonlinear MPC schemes, which is one of the main contributions of our work.

Obtaining a surrogate model for each relevant node in a large network can still be a difficult task, since often the main challenge is the amount of nodes considered in the network and not the complexity of the model. A reduction of the network nodes often requires the use of heuristics (Broad et al., 2005) or detailed system knowledge (Wang et al., 2017). As an additional contribution of this work, we propose the use of data-based clustering techniques to simplify the network model that needs to be learned. Clustering techniques in the context of WDNs have been previously studied (e.g. Di Nardo et al., 2015; Kirstein et al., 2015), but were not used for the design of MPC controllers. A clustering method is especially convenient in an MPC setting as it mitigates the effect of uncertain water demand predictions, as we illustrate in the simulation results.

Compared to similar works, such as Wang et al. (2017), we propose the combination of clustering, surrogate modeling and nonlinear MPC, showing the advantages of deep learning compared to traditional ANNs. Our method is available with an open source implementation [1], does not rely on the detailed knowledge of the underlying model equations and does not need control-oriented models, but just requires a simulator of the water distribution network. We show the potential for energy savings of an economic nonlinear MPC scheme compared to a simple rule-based controller for the benchmark case study of the C-Town WDN. Finally, our proposed clustering method mitigates the effects of uncertain demands and the proposed controller is able to robustly control the system even when an error up to 20% affects the prediction of water demands.

The remainder of the paper is organized as follows. Section 2 describes the considered WDN and its modeling assumptions. The proposed clustering method is described in Section 3, while the surrogate modeling based on deep learning is presented in Section 4. The economic NMPC formulation is explained in Section 5, the simulation results for a benchmark WDN are presented in Section 6 and the paper is concluded in Section 7.

[1] Source code available at: `https://github.com/4flixt/2019_WNTR_Surrogate_Model`

2. OPTIMAL OPERATION OF LARGE-SCALE WATER DISTRIBUTION NETWORKS

The problem of optimal operation of large-scale WDNs has been investigated in the literature in the last 50 years (Mala-Jetmarova et al., 2017), with primary focus on optimal pump operation to target savings in energy use and related cost, while ensuring that demands and water quality standards in the network are satisfied.

In general, a discrete-time model of a drinking water network includes difference and algebraic equations, based on mass continuity and energy, which can be written as:

$$x(k+1) = f(x(k), z(k), u(k), d(k)), \tag{1a}$$
$$z(k) = h(x(k), u(k), d(k)), \tag{1b}$$

where the dynamic states $x(k)$ describe the levels of the tanks that form part of the water network at time step k. The control inputs $u(k)$ include the operation conditions for the valves and pumps. The algebraic states $z(k)$ include the pressure in all nodes of the network, resulting from flow balances. The equations are strongly affected by the water demands in each node, denoted by $d(k)$.

Several detailed simulators for large-scale water networks exist in the literature. Arguably, the most widely used software application for modelling drinking water distribution systems is EPANET, developed by the United States Environmental Protection Agency. EPANET can perform extended period simulation of pressurized pipe networks, computing both hydraulic and water quality simulation (Rossman et al., 2000). More recently, EPANET extensions have also been developed to enhance EPANET modeling capabilities for water security, resilience modeling, and hydraulic response to cyber-physical attacks (Taormina et al., 2019).

Yet, EPANET models are not control-oriented models and often include several switches and discrete operation conditions that make them not suitable for the direct application of gradient-based optimization approaches, which are necessary to solve large-scale problems.

2.1 C-Town benchmark problem

We formulate the optimal operation problem for the WDN of C-Town. C-Town is a benchmark medium-sized network first introduced by Ostfeld (2012) and later used in other state-of-the-art studies (e.g. Sousa et al., 2016; Taormina, 2018). The original C-Town WDN features one reservoir providing water to a network composed of 429 pipes, 388 junctions, 7 storage tanks, 5 pump stations with a total of 11 pumps, and 4 valves. In this study, we simplified the original structure of C-Town to reduce the amount of control inputs. The simplification consisted in removing redundancy in pumping stations, i.e. each pumping station is assumed to include only one pump. The resulting modified C-Town WDN, which we use here as a demonstrative case study, is composed of 419 pipes, 378 junctions, 7 storage tanks, 5 pumps, and 4 valves (see Fig. 1).

The C-Town WDN is described by the following variables, summarized in (2a) to (2d): tank levels (b), node pressure (p), node demand (d), valve setting (v), i.e. minor loss coefficient set for Throttle Control Valves and pressure

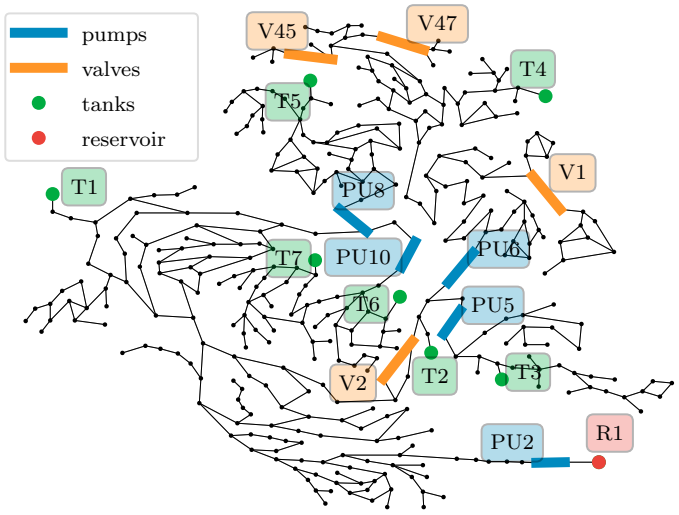

Fig. 1. Overview of the modified C-Town water distribution network, with highlighted pumps and valves (control inputs), tanks (system states) and reservoir.

setting of Pressure Reducing Valves, and relative pump speed (q):

$$x = [b_0, \dots, b_{n_{\text{tanks}}}] \in \mathbb{R}^7, \tag{2a}$$
$$z = [p_0, \dots, p_{n_{\text{junctions}}}] \in \mathbb{R}^{378}, \tag{2b}$$
$$u = [v_0, \dots, v_{n_{\text{valves}}}, q_0, \dots, q_{n_{\text{pumps}}}] \in \mathbb{R}^9, \tag{2c}$$
$$d = [d_0, \dots, d_{n_{\text{junctions}}}] \in \mathbb{R}^{378}. \tag{2d}$$

Tank levels in x are the observable state of the C-Town network, while pump and valve settings in u are the control inputs of the system.

In this study, we modeled C-Town in EPANET and simulate it as a demand-driven system. According to the EPANET engine, demand-driven simulation implies that node demands are always satisfied.

2.2 Summary of the proposed algorithm

To deal with the large-scale nature of WDNs, we propose to use, first, a clustering method that leads to a significant reduction of the number of node pressure (2b) and demands (2d) that are explicitly considered in the prediction model. Second, we generate a surrogate model via deep-learning, to obtain a control-oriented model which is used as a prediction model in an NMPC scheme. The data obtained from subsequent NMPC simulations can be used again to improve the quality of the surrogate model in an iterative scheme, as summarized in Algorithm 1.

3. CLUSTERING ALGORITHM

The investigated network in Fig. 1 has a total of 378 distinct junctions with individual demands and pressures. Obtaining a control-oriented surrogate model for such an extensive system is challenging and does not scale well. We therefore propose a clustering-based approach to simplify the problem significantly. Among the existing techniques for clustering, the hierarchical approach with connectivity constraints appears most suited for water networks. The approach is purely data-driven, but considers structural

Algorithm 1 Clustering and surrogate-based economic NMPC of large-scale WDNs

Input: Obtain initial training data (tank levels, junction pressure, demands, pump speed and valve setting) from simulations of available EPANET models or from historical real data.

1. Run hierarchical clustering algorithm (Section 3) to obtain current cluster.
2. Obtain deep learning-based surrogate model using current training data and current clustering (Section 4).
3. Run economic nonlinear MPC using the surrogate model from Step 2 as model for the predictions and a detailed EPANET simulator as reality based emulator (Section 5).
4. If performance not as desired, include the data obtained from Step 3 in your new training data and GOTO Step 1.

information of the network in the form of connectivity constraints. The connectivity matrix $A \in \mathbb{R}^{n_{\text{junc}} \times n_{\text{junc}}}$ can be derived from the EPANET model structure and represents the connections between nodes such that:

$$A_{ij} = \begin{cases} 1 \text{ if junction i and j are connected.} \\ 0 \text{ if junction i and j are not connected.} \end{cases} \tag{3}$$

Nodes are joined based on a variance-minimizing linkage criteria (Ward Jr, 1963) which minimizes the variance of the chosen features. The algorithm stops when the number of clusters matches the user defined parameter n_{clusters}. At each step of the recursive algorithm, clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ are merged, if for any $\mathcal{C}_k$, $k \neq i, j$ the variance of the features of the resulting cluster is such that $\text{var}(\mathcal{C}_i \bigcup \mathcal{C}_j) < \text{var}(\mathcal{C}_i \bigcup \mathcal{C}_k)$. We denote the final clusters as:

$$\mathcal{C}_j, \quad j = 1, \dots, n_{\text{clusters}}, \tag{4}$$

where each cluster contains a number of $|C_j|$ junctions. As features we use normalized pressures in the individual nodes at various time steps k. We normalize the pressure at each node with an individual parameter according to:

$$\tilde{p}_i(k) = \frac{p_i(k)}{f_{p,i}}, \quad \forall i = 1, \dots, n_{\text{junc}}. \tag{5}$$

To compute the scaling factors $f_{p,i}$ in (5) we use the mean value of the pressure for each junction from the samples used for training. Normalizing the pressure was found to be an important element to achieve a successful clustering. The scaling is motivated as connected nodes experience head loss on their linkage but will qualitatively follow the same pressure patterns. In Fig. 2, we showcase the pressure over time for two exemplary clusters, which can be identified in Fig. 3, as well as the computed normalized pressure according to (5). Fig. 2 shows that the average normalized pressure of the cluster is a good representation of the normalized pressure of each individual node. We compute the average normalized pressure for cluster j at each point in time k as:

$$\bar{p}_j(k) = \frac{1}{|\mathcal{C}_j|} \sum_{i \in \mathcal{C}_j} \tilde{p}_i(k). \tag{6}$$

The computed mean is used as a condensed representation of the original variables (2b). We evaluate the clustering performance by computing the normalized pressure variance $\sigma_j^2(k) = \text{Var}(\tilde{p}_i(k))$ with $i \in C_j$ for clusters j for all

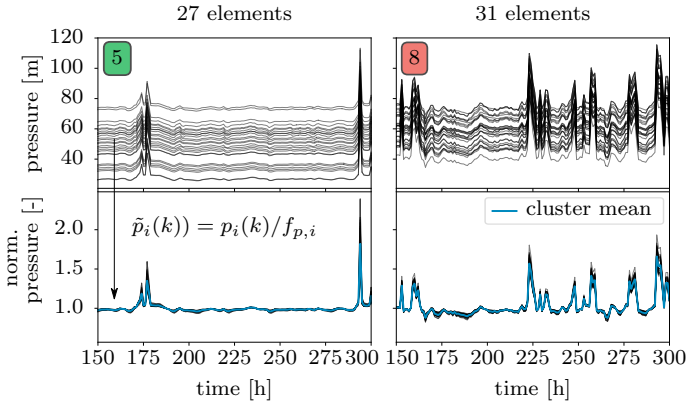

Fig. 2. Temporal evolution of pressure in clusters 5 and 8 (see Fig. 3) and the computed normalized pressure.

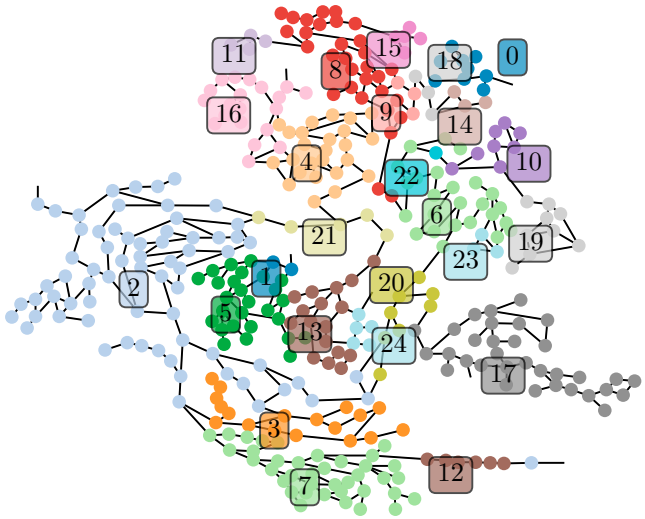

Fig. 3. Overview of the clustered junctions for the modified C-Town network.

times k. To obtain a global performance indicator we compute the mean of $\sigma_j^2(k)$ over all clusters j and times k. We choose $n_{\text{clusters}} = 25$ clusters. It shows that performance deteriorates with less individual clusters, while increasing the number has only a marginal effect. Clustering was performed in Python with scikit-learn (Pedregosa, 2011) with the feature agglomeration method. The resulting clustered nodes are presented in Fig. 3. We furthermore used the obtained clusters based on pressure similarities to group the demands in our network:

$$\bar{d}_j(k) = \sum_{i \in \mathcal{C}_j} d_i(k). \tag{7}$$

This is motivated because sections of the network which experience similar pressure patterns are likely to be affected by the same pump and valve actions, as well as tank levels. We are thus able to control these sections simultaneously and are interested in their aggregated demand. While junction pressure and demand are clustered we investigate tank levels (b), valve setting (v) and pump speed (q) individually. In summary the relevant variables and their respective dimensions for the dynamic system used in the remainder of the paper are:

$$x = [b_0, \dots, b_{n_{\text{tanks}}}] \in \mathbb{R}^7, \tag{8a}$$
$$\bar{z} = [\bar{p}_0, \dots, \bar{p}_{n_{\text{clusters}}}] \in \mathbb{R}^{25}, \tag{8b}$$
$$u = [v_0, \dots, v_{n_{\text{valves}}}, q_0, \dots, q_{n_{\text{pumps}}}] \in \mathbb{R}^9, \tag{8c}$$
$$\bar{d} = [\bar{d}_0, \dots, \bar{d}_{n_{\text{clusters}}}] \in \mathbb{R}^{25}. \tag{8d}$$

The clustering method significantly reduces the dimension of the algebraic states and demands as can be seen by comparing (2) and (8).

4. SURROGATE MODELING USING DEEP LEARNING

Artificial neural networks have been widely used as basis functions for surrogate modeling. Traditionally, shallow networks with only one intermediate (or hidden) layer are used, because under mild conditions they can approximate arbitrarily well any continuous function (Barron, 1993). Recently, the use of deep neural networks (DNNs) with several hidden layers has been very successful as an efficient approximation method for complex functions, mainly in the field of computer science and artificial intelligence (Silver et al., 2016). The choice of deep networks as opposed to shallow ones is motivated by recent theoretical results that support the increased representation power of deep networks (Safran and Shamir, 2017).

A standard DNN with fully connected layers is a simple sequence of function compositions of an affine transformation and a nonlinear function. A neural network $\mathcal{N} : \mathbb{R}^{n_{\text{in}}} \to \mathbb{R}^{n_{\text{out}}}$ can be written as:

$$\mathcal{N}(\zeta;\lambda) = \begin{cases} \alpha_{L+1} \circ \beta_L \circ \alpha_L \circ \cdots \circ \beta_1 \circ \alpha_1(x) & \text{for } L \geq 2, \\ \alpha_{L+1} \circ \beta_1 \circ \alpha_1(x), & \text{for } L = 1, \end{cases} \tag{9}$$

where the input of the network is $\zeta \in \mathbb{R}^{n_{\text{in}}}$ and the output is $\eta \in \mathbb{R}^{n_{\text{out}}}$, and λ denotes the network parameters. The number of hidden layers L and of neurons in each layer M determine the size of the neural network. If $L = 1$, the neural network is denoted as shallow, while deep neural networks have $L \geq 2$. Each hidden layer is composed of an affine function

$$\alpha_l(\xi_{l-1}) = W_l \xi_{l-1} + b_l, \tag{10}$$

where $\xi_{l-1} \in \mathbb{R}^M$ is the output of the previous layer and $\xi_0 = \zeta$, as well as a nonlinear nonlinear activation function β_l. The nonlinear activation function is a design parameter and usual choices include the rectifier linear units ($ReLU$), the sigmoid function and the hyperbolic tangent (tanh) which is used in this work:

$$\beta_l(\alpha_l) = \frac{e^{\alpha_l} - e^{-\alpha_l}}{e^{\alpha_l} + e^{-\alpha_l}}. \tag{11}$$

The parameters of all layers are summarized in $\lambda = \{\lambda_1, \dots, \lambda_{L+1}\}$ with

$$\lambda_l = \{W_l, b_l\} \quad \forall l = 1, \dots, L+1, \tag{12}$$

where W_l are the weights and b_l are the biases describing the corresponding affine functions. The neural network is trained using a set of training input-output data pairs $\{(\zeta^{(1)}, \eta^{(1)}), \dots, (\zeta^{(N_s)}, \eta^{(N_s)})\}$ where N_s is the number of training data points. The training consists of finding the parameters W_l, b_l that minimize the mean squared error, by solving the following optimization problem for a fixed value of L and M:

$$\lambda^* = \arg\min_{\lambda} \frac{1}{N_s} \sum_{i=1}^{N_s} (\eta^{(i)} - \mathcal{N}(\zeta^{(i)};\lambda))^2. \tag{13}$$

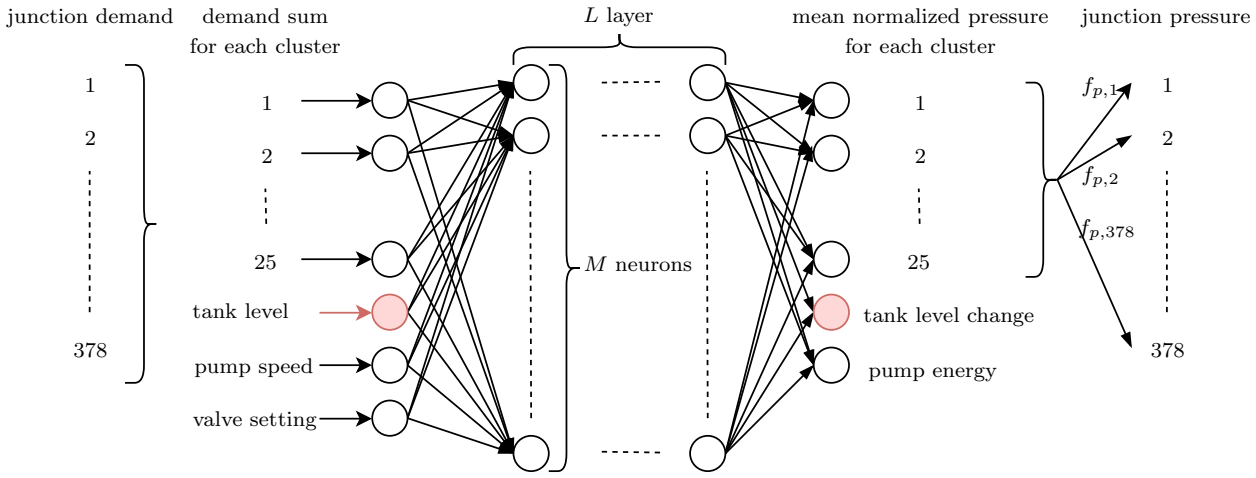

Fig. 4. Overview of the neural network inputs and outputs. The system states are marked in red.

4.1 Input and output structure

In our proposed Algorithm 1, we use DNNs to jointly learn the discrete-time and algebraic equations in (1) for the clustered variables (8). Fig. 4 illustrates the structure of the proposed model. We use as inputs the current clustered junction demands $\bar{d}(k)$, the tank level $b(k)$, the current pump speed $q(k)$ and valve position $v(k)$. As output we have the current mean value of the normalized pressures at each junction $\bar{p}(k)$, the current pump energy consumption $e(k)$ and the change of tank level $\Delta b(k)$. This allows to compute the tank level at the next time step as:

$$b(k+1) = b(k) + \Delta b(k). \tag{14}$$

Choosing the change of tank level versus the next tank level was found to be beneficial for the model accuracy and can be seen as an adaptive normalization. In practice it leads to smoother, more realistic predictions.
In summary, we have:

$$\zeta = \left[\bar{d}(k), b(k), q(k), v(k)\right] \in \mathbb{R}^{41} \tag{15a}$$

$$\eta = \left[\bar{p}(k), \Delta b(k), e(k)\right] \in \mathbb{R}^{37} \tag{15b}$$

Note that we can compute approximate values of the pressure in each junction i by rescaling the normalized pressure in cluster j value with its respective scaling factor:

$$p_i \approx f_{p,i}\bar{p}_j, \quad i \in \mathcal{C}_j \tag{16}$$

In many cases we are only interested in the lowest pressure within one cluster, for which we compute:

$$f_{\min,j} = \min(\{f_i \mid i \in \mathcal{C}_j\}), \tag{17a}$$

$$p^{\text{clust}}_{\min,j} \approx f_{\min,j}\bar{p}_j. \tag{17b}$$

The learned model can be evaluated recursively to achieve the desired prediction horizon in an MPC framework.

4.2 Training and model comparison

All data points used for training are obtained by running EPANET simulations with randomized demand scenarios and different controllers. The training is done using Tensorflow (Abadi, 2015) via Keras (Chollet et al., 2015). We are choosing Adam (Kingma and Ba, 2014), a variant of stochastic gradient descent, as optimizer to solve the problem (13). Training data was normalized, such that $|\bar{\zeta}^{(i)}| \leq 1$ and $|\bar{\eta}^{(i)}| \leq 1\ \forall i = 1, \ldots, n_{\text{train}}$. We investigated

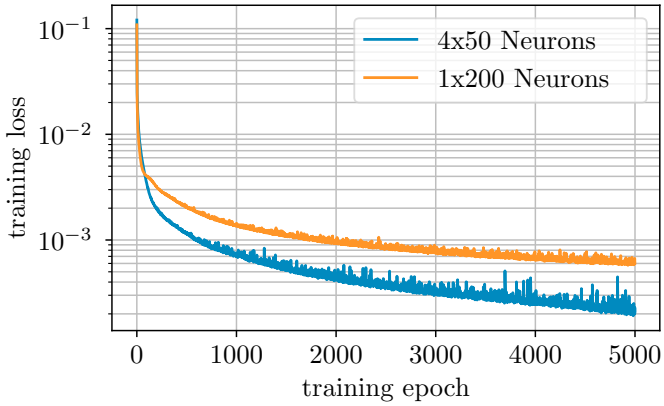

Fig. 5. Comparison of training loss for two investigated neural network architectures with the same number of neurons (nonlinear activations). Model 1 ($L = 4$, $M = 50$) with 11,637 weights and Model 2 ($L = 1$, $M = 200$) with 15,837 weights.

a number of models with different hyper parameters but limited our search space to architectures that are still feasible to optimize in real time. The training progress (mean squared error in (13)) for two investigated models that fulfil this criteria are displayed in Fig. 5. We showcase a shallow vs. a deep architecture with the same number of neurons. The deep architecture with $L = 4$, $M = 50$ has a total of 11,637 weights, whereas the shallow architecture with $L = 1$, $M = 200$ results in 15,837 weights. The training progress in Fig. 5 shows that the deep architecture achieves a significantly better performance. Similar differences between deep and shallow architectures are observed with a smaller number of neurons, leading to worse overall performance. On the other hand, larger architectures only show marginal improvements.
We also investigated a simple linear model for the same input structure and data as for the neural networks. The resulting loss was 3.67×10^{-3}, which is one order of magnitude larger than the the best obtained neural network. Because of the higher training and validation accuracy, we choose the deep learning model as prediction model embedded in the economic NMPC framework explained in the next section.

5. ECONOMIC NONLINEAR MODEL PREDICTIVE CONTROL FORMULATION

For the energy optimal control of the investigated WDN we propose the following economic nonlinear MPC formulation:

$$\begin{aligned}
\underset{\substack{u(0),\dots,u(N-1)\\ x(0),\dots,x(N)\\ \epsilon(0),\dots,\epsilon(N-1)}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \alpha_e e(x(k),u(k),\bar{d}(k)) \\
& + \alpha_\epsilon \epsilon(k) + \alpha_u \Delta u(k) && \text{(18a)}\\
\text{subject to:} \quad & x(k+1) = f(x(k),z(k),u(k),\bar{d}(k)), && \text{(18b)}\\
& z(k) = h(x(k),u(k),\bar{d}(k)), && \text{(18c)}\\
& g(x(k),z(k),u(k),\bar{d}(k),\epsilon(k)) \leq 0, && \text{(18d)}\\
& x_0 = x_{\text{init}}, && \text{(18e)}\\
& \text{for } k = 0,\dots,N-1. && \text{(18f)}
\end{aligned}$$

The first term in the mixed objective cost function in (18a) denotes the energy consumption of the pumps. If available, time-varying electricity prices can be incorporated in the formulation. In this work, we consider constant prices so that the economic operation cost is determined by the pump energy consumption which is part of the DNN output (15). Furthermore, we include slack variables ϵ to allow for soft constraints in (18d). The complete set of constraints is defined in (20). The last term of the cost function in (18a) penalizes rapid changes of the control input $\Delta u(k)$. We introduce the tuning factors $\alpha_e,\ \alpha_\epsilon,\ \alpha_u$ to balance the different terms of the objective function.
The difference equation (18b) and the algebraic equations (18c) that describe the dynamic and algebraic states are described by the surrogate model of the clustered network described in the previous subsection. A DNN as depicted in Fig. 4 computes the next state $x(k+1)$ and the algebraic states $z(k)$ as

$$[x(k+1)^T, z(k)^T]^T = \mathcal{N}(\bar{d}(k), x(k), u(k)). \tag{19}$$

The constraints in (18d) describe the maximum and minimum tank levels, as well as the input constraints for pumps and valves for all steps in the prediction horizon $k = 0,\dots,N-1$:

$$\begin{aligned}
& b_{\min,i} \leq b_i(k) + \epsilon_{\text{tank}}(k) \leq b_{\max,i}, && \forall i = 1,\dots,n_{\text{tanks}}, && \text{(20a)}\\
& p_{\min,i} \leq p^{\text{clust}}_{\min,i}(k) + \epsilon^{\text{clust}}_{\min,i}(k) && \forall i = 1,\dots,n_{\text{clusters}}, && \text{(20b)}\\
& v_{\min,i} \leq v_i(k) \leq v_{\max,i}, && \forall i = 1,\dots,n_{\text{valves}}, && \text{(20c)}\\
& q_{\min,i} \leq q_i(k) \leq q_{\max,i}, && \forall i = 1,\dots,n_{\text{pumps}}, && \text{(20d)}\\
& 0 \leq e(k), && && \text{(20e)}\\
& 0 \leq \epsilon(k). && && \text{(20f)}
\end{aligned}$$

As described in Section 3 and 4, we can compute the minimal pressure in each cluster from (17), which implicitly considers the constraint for all 378 junctions while only incorporating $n_{\text{clusters}} = 25$ constraints. The above mentioned slack variables for soft constraints are used for the tank level as well as the minimal pressure for each cluster. Since we are following an economic MPC formulation, softening these constraints is necessary as we are usually in their proximity. Having soft constraints on tank levels and pressures is unintuitive, but results from the demand-driven simulation that is implemented in EPANET. This modelling approach ensures that demands are satisfied, even when water cannot be supplied, thus resulting in negative pressures. Especially, when tank levels are close to zero, this scenario is bound to happen as tanks are the only storage terms in the network. For all other nodes, EPANET assumes stationary conditions under all circumstances. Therefore, we have to interpret negative pressures as not fully satisfied demand, which is undesirable but acceptable on rare occasions. To deal with unavoidable model uncertainty, and following ideas from tube-based MPC Mayne and Rawlings (2009), we tighten the constraints on the tank levels by increasing the lower bounds ($b_{\min}$).

Table 1. Comparison of different MPC solutions with the reference controller for different lower bounds of the tank levels.

	MPC for different $b_{\min}$			**Ref.**
	1.5	0.5	0.0	
energy [MWh]	106.49	105.56	99.97	111.57
energy saved [%]	4.55	5.39	10.39	-
cons. viol. [%]	0.05	0.13	0.51	0.0
avg. cons. viol. [m]	2.13	5.12	5.17	0.00

6. RESULTS

Due to the economic formulation of the MPC objective function in (18a), the main focus of investigating the proposed controller is energy consumption, while ensuring that the demand is satisfied under varying conditions. This can be realized by introducing a constraint tightening, i.e. increasing the lower bound on the tank levels in the form of a soft constraint. We investigate several options for the lower bound $b_{\min}$, which is applied to all tanks. The effect of this lower bound for the tank levels is demonstrated in Fig. 6, where we compare two different $b_{\min}$ values for the proposed MPC approach with the original rule-based controller. In this rule-based controller originally included in C-Town, pumps and valves are opened (closed) only when tank levels reach minimum (maximum) threshold values to guarantee the system can satisfy the expected demand. As expected, economic operation of the system leads to lower tank levels in general. At least one tank level (T1) remains close to zero. The effect of the increased lower bound $b_{\min}$ is visible in the mean values of the tank levels over time (horizontal lines in Fig. 6).

In Table 1 we showcase the energy consumption of the pumps over the simulation period of one month for the three investigated MPC controllers, as well as the rule-based reference controller. The computed energies are also expressed as relative savings in comparison to the reference controller. Furthermore, we display the constraint violation, expressed as percentage of instances where $p_i < 0$ over all nodes and time steps, as well as the mean constraint violation.

As we can see in Table 1, our proposed MPC controller leads to energy savings as high as 10.39% if no tightening of constraints is used ($b_{\min} = 0$). This comes at the cost of increased constraint violations, meaning that we cannot satisfy the demand in isolated instances. Nevertheless, constraint violations occur only during 0.51% of the operating time for each node. Constraint violations can be further reduced by tightening the lower bound of the tank level as shown in Table 1, achieving a trade-off between constraint violations and energy savings.

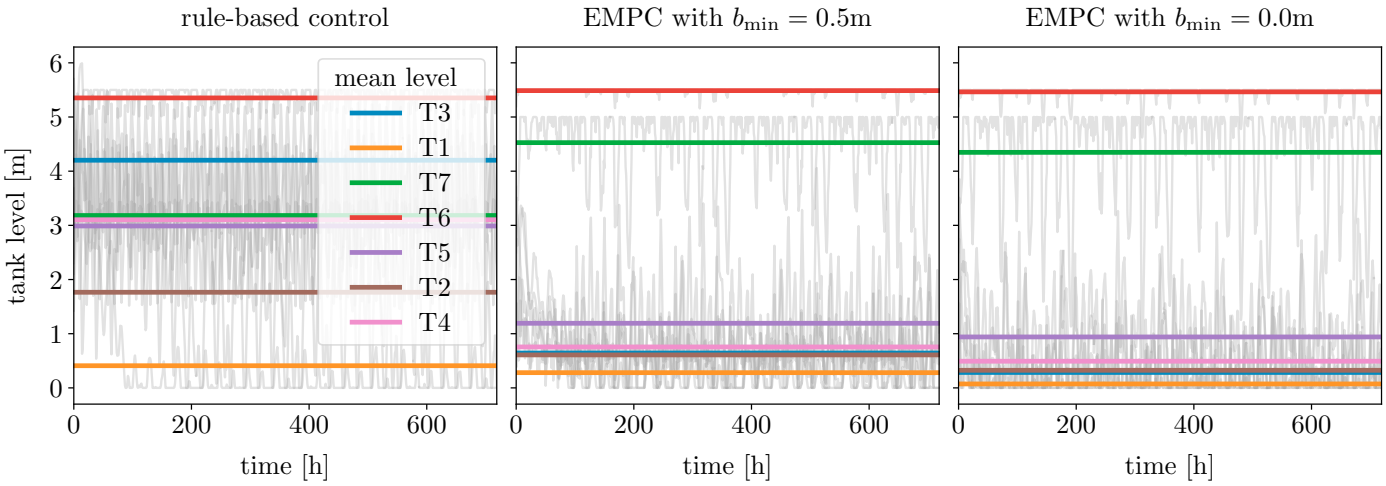

Fig. 6. Temporal evolution of the tank levels (system states) over the course of the simulation time (one month). Comparison of different values for the lower bound (soft constraint) for two EMPC solutions with the original rule-based controller. The mean value of the tank levels over time is marked with a horizontal line.

Table 2. MPC performance with $b_{\min} = 0.5$ m with error in the water demand predictions.

	Error demand predictions	
	0%	±20%
energy [MWh]	105.56	105.11
energy saved [%]	5.39	5.80
cons. viol. [%]	0.13	0.32
avg. cons. viol. [m]	5.12	9.59

Solving our proposed nonlinear MPC problem at each time step requires on average 2.5 s on a standard laptop, which enables the real-time implementation of the method.

6.1 Robust economic NMPC

Another feature of the proposed algorithm is its implicit robustness to demand uncertainty, which is a common problem for WDN. This features arises from the applied clustering method, where the demand for each node in a given cluster is summed to form an aggregated cluster demand. Fluctuations of individual nodes within a cluster are likely to compensate each other, especially for larger clusters. We investigate the robustness of our approach by disturbing the demand prediction with uniform noise in the range of ±20 %. The comparison of energy consumption and constraint satisfaction for the case of $b_{\min} = 0.5$ m are given in Table 2. The results show that the proposed NMPC scheme is able to robustly control the system even under the presence of significant uncertainty in the water demand forecasts.

7. CONCLUSIONS

Optimal operation of distribution water networks is a challenging problem due the large-scale and nonlinear nature of the system as well as the presence of important uncertainty and constraints. To deal with this problem, this paper proposes first to perform a clustering based on normalized pressures to reduce significantly the number of nodes that need to be explicitly considered in the model. We then obtain a control-oriented surrogate model based on the EPANET simulator, any other available simulator or real data and show that the use of deep learning networks leads to a better surrogate accuracy compared to standard shallow networks or linear models.

The deep learning-based surrogate model is embedded in an efficient nonlinear model predictive control framework that directly optimizes the economic performance of the system. By means of a simulation study of the C-town benchmark, we show that the proposed strategy leads to energy savings over 10% and, because of the proposed clustering, it is very robust to uncertain water demand predictions and it can be used in real time. The framework and the data used to obtain the results are openly available.

Future work includes the analysis of more advanced robust MPC strategies Lucia et al. (2013); Sampathirao et al. (2018) and stochastic approaches Lorenzen et al. (2017) that can consider other sources of uncertainty explicitly.

REFERENCES

Abadi, M. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Barron, A.R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3), 930–945.

Biscos, C., Mulholland, M., Le Lann, M., Buckley, C., and Brouckaert, C. (2003). Optimal operation of water distribution networks by predictive control using MINLP. *Water Sa*, 29(4), 393–404.

Boulos, P.F., Wu, Z., Orr, C.H., Moore, M., Hsiung, P., and Thomas, D. (2001). Optimal pump operation of water distribution systems using genetic algorithms. In *Distribution system symposium*.

Broad, D.R., Dandy, G.C., and Maier, H.R. (2005). Water distribution system optimization using metamodels. *Journal of Water Resources Planning and Management*, 131(3), 172–180.

Broad, D.R., Maier, H.R., and Dandy, G.C. (2010). Optimal operation of complex water distribution systems

using metamodels. *Journal of Water Resources Planning and Management*, 136(4), 433–443.

Cembrano, G., Quevedo, J., Puig, V., Pérez, R., Figueras, J., Verdejo, J., Escaler, I., Ramón, G., Barnet, G., Rodríguez, P., et al. (2011). Plio: A generic tool for real-time operational predictive optimal control of water networks. *Water Science and Technology*, 64(2), 448–459.

Chini, C.M. and Stillwell, A.S. (2018). The state of US urban water: data and the energy-water nexus. *Water Resources Research*, 54(3), 1796–1811.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Di Nardo, A., Di Natale, M., Giudicianni, C., Musmarra, D., Santonastaso, G.F., and Simone, A. (2015). Water distribution system clustering and partitioning based on social network algorithms. *Procedia Engineering*, 119(1), 196–205.

Duzinkiewicz, K., Brdys, M., and Chang, T. (2005). Hierarchical model predictive control of integrated quality and quantity in drinking water distribution systems. *Urban Water Journal*, 2(2), 125–137.

Grosso, J., Ocampo-Martínez, C., Puig, V., and Joseph, B. (2014). Chance-constrained model predictive control for drinking water networks. *Journal of Process Control*, 24(5), 504–516.

Grosso, J., Ocampo-Martínez, C., and Puig, V. (2013). Learning-based tuning of supervisory model predictive control for drinking water networks. *Engineering Applications of Artificial Intelligence*, 26(7), 1741 – 1750.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Kirstein, J.K., Albrechtsen, H.J., and Rygaard, M. (2015). Topological clustering as a tool for planning water quality monitoring in water distribution networks. *Water Supply*, 15(5), 1011–1018.

Lorenzen, M., Dabbene, F., Tempo, R., and Allgöwer, F. (2017). Constraint-tightening and stability in stochastic model predictive control. *IEEE Transactions on Automatic Control*, 62(7), 3165–3177.

Lucia, S., Finkler, T., and Engell, S. (2013). Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23, 1306–1319.

Mala-Jetmarova, H., Sultanova, N., and Savic, D. (2017). Lost in optimisation of water distribution systems? a literature review of system operation. *Environmental Modelling & Software*, 93, 209–254.

Mayne, D. and Rawlings, J. (2009). *Model Predictive Control: Theory and Design*. Nob Hill Publishing.

Nowak, D., Krieg, H., and Bortz, M. (2018). Surrogate models for the simulation of complex water supply networks. In *Proceedings of the 1st International WDSA/CCWI Joint Conference*.

Ocampo-Martinez, C., Barcelli, D., Puig, V., and Bemporad, A. (2012). Hierarchical and decentralised model predictive control of drinking water networks: Application to barcelona case study. *IET control theory & applications*, 6(1), 62–71.

Ostfeld, A. (2012). Battle of the water calibration networks. *Journal of Water Resources Planning and Management*, 138(5), 523–532.

Pedregosa, F. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Rossman, L.A. et al. (2000). Epanet 2: Users manual. *US Environmental Protection Agency. Office of Research and Development*.

Safran, I. and Shamir, O. (2017). Depth-width tradeoffs in approximating natural functions with neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2979–2987. JMLR. org.

Sampathirao, A.K., Sopasakis, P., Bemporad, A., and Panos Patrinos, P. (2018). GPU-Accelerated Stochastic Predictive Control of Drinking Water Networks. *IEEE Transactions on Control Systems Technology*, 26(2), 551–562.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529, 484.

Sousa, J., Muranho, J., Marques, A.S., and Gomes, R. (2016). Optimal management of water distribution networks with simulated annealing: The c-town problem. *Journal of Water Resources Planning and Management*, 142(5), C4015010.

Spang, E.S. and Loge, F.J. (2015). A high-resolution approach to mapping energy flows through water infrastructure systems. *Journal of Industrial Ecology*, 19(4), 656–665.

Taormina, R. (2018). Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks. *Journal of Water Resources Planning and Management*, 144(8), 04018048.

Taormina, R., Galelli, S., Douglas, H., Tippenhauer, N.O., Salomons, E., and Ostfeld, A. (2019). A toolbox for assessing the impacts of cyber-physical attacks on water distribution systems. *Environmental modelling & software*, 112, 46–51.

Wang, Y., Ocampo-Martinez, C., and Puig, V. (2016). Stochastic model predictive control based on Gaussian processes applied to drinking water networks. *IET Control Theory and Applications*, 10(8), 947–955.

Wang, Y., Puig, V., and Cembrano, G. (2017). Non-linear economic model predictive control of water distribution networks. *Journal of Process Control*, 56, 23–34.

Ward Jr, J.H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301), 236–244.

Wu, Z.Y. and Rahman, A. (2017). Optimized deep learning framework for water distribution data-driven modeling. *Procedia Engineering*, 186, 261–268.

2021 European Control Conference (ECC)
June 29 - July 2, 2021. Rotterdam, Netherlands

On the relationship between data-enabled predictive control and subspace predictive control

Felix Fiedler[1] and Sergio Lucia[1]

Abstract— **Data-enabled predictive control (DeePC) is a recently proposed approach that combines system identification, estimation and control in a single optimization problem, for which only recorded input/output data of the examined system is required. The same premise holds for the subspace predictive control (SPC) method in which a multi-step prediction model is identified from the same data as required for DeePC. This model is then used to formulate a similar optimal control problem. In this work we investigate the relationship between DeePC and SPC. Our primary contribution is to show that SPC is equivalent to DeePC in the deterministic case. We also show the equivalence of both methods in a special case for the non-deterministic formulation. We investigate the advantages and shortcomings of DeePC as opposed to SPC with and without measurement noise and illustrate them with a simulation example.**

I. Introduction

Model predictive control (MPC) is a popular strategy to control multivariable systems with constraints [1]. Traditionally, the main shortcoming of MPC is its computational complexity as it requires the online solution of an optimization problem. Due to advances in computing power, algorithms and efficient implementation strategies, solution time is often not an obstacle for its deployment anymore.

However, MPC still requires a dynamic model of reasonable accuracy to obtain a satisfactory performance. Such model can be obtained from physical modelling, using data-based system identification or a combination of both approaches. Learning from data has recently regained attention although it has been studied for a long time in the field of system identification [2], [3], [4], [5]. Important categories in this vast field are, among others, linear vs. nonlinear [2] and input/output vs. state-space models [3]. A concrete example of interest is the autoregressive model with exogeneous inputs (ARX) [5], a popular linear input/output approach. Typically, ARX models predict the next output of the system and can be evaluated repeatedly to compute finite future trajectories as required for an MPC formulation. Subspace predictive control (SPC) [6] follows a similar approach, where a multi-step ahead prediction model is used to compute the finite future trajectory in a single step. An extensive overview on the method is given in [7] and SPC is still an active field of research with the work in [8] investigating stability and horizon tuning.

*Felix Fiedler acknowledges the support of the Helmholtz Einstein International Berlin Research School in Data Science (HEIBRiDS)

[1] F. Fiedler and S. Lucia are with the Laboratory of Process Automation Systems at Technische Universität Dortmund, Emil-Figge-Str. 70, 44227 Dortmund

In recent years, a new trend in data-based control seeks to integrate the traditional sequential system identification and control in a unified approach. From the machine learning community, reinforcement-learning is a popular approach following this paradigm [9]. However, reinforcement-learning comes with its own challenges such as the demand for large quantities of data and produces highly variable outcomes [10]. Another approach to unify identification and control is the newly proposed data-enabled predictive control (DeePC) [11] algorithm. The approach describes a simple configuration of a predictive control problem which operates directly on matrices of collected input/output data. Furthermore, the required data has to satisfy only moderate requirements and can often be taken directly from a running process.

The initial work on DeePC [11] has thus sparked considerable interest in the control community with several extensions and modifications. The authors in [12] investigate DeePC with respect to stability and robustness and propose an adaptation to guarantee these properties. Distributionally robust DeePC is presented in [13], where it is assumed that data is sampled from a distribution of possible systems. In [14], DeePC is combined with an extended Kalman Filter to improve the performance in the case of noisy measurements.

However, DeePC remains in its nature a linear MPC scheme and is based on data which could also be used for sequential identification and control approaches such as SPC. The authors in [11] state that an equivalent classical MPC formulation exists for their proposed DeePC scheme. This equivalence is based on theorizing about the existence of an unknown system parameterization. But if such a parameterization could be known, what would be the advantages and potential shortcomings of the DeePC approach, especially in the linear setting, for which DeePC is derived?

The main contribution of this work is to help answer this question. In particular we investigate the relationship between DeePC and SPC. The performance of SPC and DeePC has recently been empirically compared in [15]. From the obtained similar performance the authors conclude that there are some similarities between the two formulations. In this work, we prove that in the linear deterministic case, DeePC is equivalent to SPC. The SPC multi-step prediction model can be determined from the exact same data that DeePC requires to operate. This exact equivalence has, to the best of our knowledge, not yet been established in the literature. For the non-deterministic case, DeePC requires some adaptations to deal with the noise corrupted data. We present a minor modification of DeePC and as a second

978-94-6384-236-5 ©2021 EUCA

contribution prove that in a special case this formulation is still equivalent to SPC. Finally, we compare both controllers numerically on a linear model with additive Gaussian noise. We include an analysis of the effect that the amount of available data and the regularization term of DeePC have on the closed-loop performance.

The remainder of this paper is structured as follows. In Section II we present the problem setup and review the data-enabled predictive control algorithm. We then present in Section III the subspace predictive control (SPC) algorithm and prove equivalence to the DeePC method in Theorem 1. We proceed to analyse the case of a non-deterministic linear system in Section IV. We present a minor modification of the original DeePC problem and prove equivalence to SPC in a special case in Theorem 2. Finally, we present a numerical comparison of both controllers in the general case in Section V. We finish this work with conluding remarks in Section VI.

II. Data-enabled predictive control

We investigate an unknown discrete-time LTI dynamic system represented in state-space form:

$$x_{k+1} = Ax_k + Bu_k, \tag{1a}$$
$$y_k = Cx_k + Du_k, \tag{1b}$$

with $x \in \mathbb{R}^n$ (states), $u \in \mathbb{R}^m$ (inputs), $y \in \mathbb{R}^p$ (measurements) and system matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$. We assume that the system with n states is given in its minimal representation.

From this unknown system, we collect T sequences of input/output data of length $L = T_{\text{ini}} + N$, where T_{ini} denotes the length of the initialization sequence and N the length of the prediction horizon. The collected data is arranged in matrices, such that:

$$U_L = \left[u_L^1, u_L^2, \dots, u_L^T\right],\ Y_L = \left[y_L^1, y_L^2, \dots, y_L^T\right], \tag{2}$$

where $u_L^1 = \left[\left(u_1^1\right)^\intercal, \dots, \left(u_L^1\right)^\intercal\right]^\intercal \in \mathbb{R}^{Lm}$ and $y_L^1 = \left[\left(y_1^1\right)^\intercal, \dots, \left(y_L^1\right)^\intercal\right]^\intercal \in \mathbb{R}^{Lp}$. These data matrices may or may not coincide with a Hankel or Page matrix (see [13]). We also assume that the sequences were collected starting from T unknown initial states x_1:

$$X_1 = \left[x_1^1, x_1^2, \dots, x_1^T\right], \tag{3}$$

where $X_1 \in \mathbb{R}^{n\times T}$. We divide the input/output data in (2) in two parts, such that:

$$U_L = \begin{bmatrix} U_{T_{\text{ini}}} \\ U_N \end{bmatrix},\ Y_L = \begin{bmatrix} Y_{T_{\text{ini}}} \\ Y_N \end{bmatrix}. \tag{4}$$

To clarify this notation, we now have $U_{T_{\text{ini}}} \in \mathbb{R}^{T_{\text{ini}}m\times T}$, $U_N \in \mathbb{R}^{Nm\times T}$, $Y_{T_{\text{ini}}} \in \mathbb{R}^{T_{\text{ini}}p\times T}$ and $Y_N \in \mathbb{R}^{Np\times T}$.

Definition 1 ([11]): With L, T such that $T \geq Lm$, the signals comprising matrix U_L are persistently exciting of order L if matrix U_L has full rank.

Definition 2 ([11]): We denote with $l(A,B,C,D)$ the lag of system (1). It is defined as the smallest integer l for which the observability matrix:

$$O_l(A,C) := \left(C,\ CA,\ \dots, CA^{l-1}\right), \tag{5}$$

has full rank.

Assumption 1: The number of recorded sequences is $T \geq Lm+n$. Matrix U_L is persistently exciting of order L according to Definition 1.

Assumption 2: The initialization sequence is $T_{\text{ini}} > l(A,B,C,D)$, with l according to Definition 2.

Assumption 3: The underlying matrix of initial states (3) has full rank, i.e. $\text{rank}(X_1) = n$ and the stacked matrix $[X_1^\intercal, U_L^\intercal]^\intercal$ has full rank.

Subsequently, we state a variant (see Remark 1) of the fundamental lemma of behavioral systems theory [16].

Lemma 1: Let (4) be input/output data of a deterministic system in the form of (1) which satisfies Assumption 1, Assumption 2 and Assumption 3. Any sequence $y_{T_{\text{ini}}} = \left[y_1^\intercal, \dots, y_{T_{\text{ini}}}^\intercal\right]^\intercal$, $y_N = \left[y_{T_{\text{ini}}+1}^\intercal, \dots, y_{T_{\text{ini}}+N}^\intercal\right]^\intercal$, $u_{T_{\text{ini}}} = \left[u_1^\intercal, \dots, u_{T_{\text{ini}}}^\intercal\right]^\intercal$ and $u_N = \left[u_{T_{\text{ini}}+1}^\intercal, \dots, u_{T_{\text{ini}}+N}^\intercal\right]^\intercal$ is a trajectory of system (1) if and only if, there exists a $g \in \mathbb{R}^T$, such that:

$$\begin{bmatrix} Y_{T_{\text{ini}}} \\ U_{T_{\text{ini}}} \\ U_N \\ Y_N \end{bmatrix} g = \begin{bmatrix} y_{T_{\text{ini}}} \\ u_{T_{\text{ini}}} \\ u_N \\ y_N \end{bmatrix}. \tag{6}$$

Remark 1: Note that the original statement of the fundamental lemma in [16] requires the matrices in (2) to be Hankel matrices. In [13] it is shown that (6) also holds for Page matrices which can be extended to show that arbitrary sequences are sufficient as long as the space spanned by the initial states of these sequences has full rank (see Assumption 3). The full proof of Lemma 1 is omitted here for brevity.

Based on the fundamental Lemma 1, the authors in [11] proposed the elegant data-enabled predictive control scheme (DeePC), where the optimal control policy is obtained as the solution of:

$$\begin{aligned} \min_{g,u_N,y_N} \quad & f(g,u_N,y_N) \\ \text{s.t.} \quad & (6) \\ & u_k \in \mathbb{U}\ \forall k \in \{1,\dots,N\}, \\ & y_k \in \mathbb{Y}\ \forall k \in \{1,\dots,N\}, \end{aligned} \tag{7}$$

with arbitrary objective function $f(g,u_N,y_N)$. Within the scope of this work, we only investigate the classical regulation task:

$$f(g,u_N,y_N) = \sum_{k=1}^{N} \left(\|y_k\|_Q^2 + \|u_k\|_R^2\right), \tag{8}$$

where $\|y_k\|_Q^2 = y_k^\intercal Q y_K$. We require that Q and R are positive definite.

III. Subspace predictive control

We start this section by stating a well known data-based approach for linear system identification: The auto-regressive model with external inputs (ARX) [5], where

$$y_{T_{\text{ini}}+1} = \bar{a}_1 y_1 + \cdots + \bar{a}_{T_{\text{ini}}} y_{T_{\text{ini}}} + \bar{b}_1 u_1 + \cdots + \bar{b}_{T_{\text{ini}}} u_{T_{\text{ini}}}. \quad (9)$$

Similarly as in Lemma 1, the equation can only hold if $T_{\text{ini}} > l(A,B,C,D)$, according to Definition 2 as shown in [17].

To identify the parameters

$$\bar{P} = \begin{bmatrix} \bar{a}_1, & \ldots, & \bar{a}_{T_{\text{ini}}}, & \bar{b}_1, & \ldots, & \bar{b}_{T_{\text{ini}}} \end{bmatrix} \quad (10)$$

based on the collected data in (2), one would typically solve the least-squares problem:

$$\min_{\bar{P}} \quad \|\bar{P} \begin{bmatrix} Y_{T_{\text{ini}}} \\ U_{T_{\text{ini}}} \end{bmatrix} - Y_+\|_F^2, \quad (11)$$

where $\|\cdot\|_F$ denotes the Frobenius-norm. The matrix Y_+ represents the first p rows of Y_N corresponding to the temporal elements of the output sequences at $T_{\text{ini}}+1$. Reminiscent of the ARX model in (9), the SPC approach [6] uses the collected data (2) to directly identify a linear multi-step ahead predictor:

$$\begin{aligned} y_N = a_1 y_1 + \cdots + a_{T_{\text{ini}}} y_{T_{\text{ini}}} \\ b_1 u_1 + \cdots + b_{T_{\text{ini}}+N} u_{T_{\text{ini}}+N}, \end{aligned} \quad (12)$$

where the parameters are again summarized as

$$P = \begin{bmatrix} a_1, & \ldots, & a_{T_{\text{ini}}}, & b_1, & \ldots, & b_{T_{\text{ini}}+N} \end{bmatrix}. \quad (13)$$

Lemma 2: Let (4) be input/output data of a deterministic system in the form of (1) which satisfies Assumption 1, Assumption 2 and Assumption 3. Let matrix $P^* \in \mathbb{R}^{Np \times Nm + T_{\text{ini}}(m+p)}$ be the solution of the least squares problem

$$P^* = \arg \min_P \ \|P \underbrace{\begin{bmatrix} Y_{T_{\text{ini}}} \\ U_{T_{\text{ini}}} \\ U_N \end{bmatrix}}_{M} - Y_N\|_F^2, \quad (14)$$

which is expressed explicitly in terms of the Moore-Penrose inverse (denoted with the superscript †) as:

$$P^* = Y_N M^\dagger. \quad (15)$$

Any sequence $u_{T_{\text{ini}}}$, u_N, $y_{T_{\text{ini}}}$, y_N is a trajectory of system (1) if

$$y_N = P^* \begin{bmatrix} y_{T_{\text{ini}}} \\ u_{T_{\text{ini}}} \\ u_N \end{bmatrix}. \quad (16)$$

Proof: The proof is shown in the appendix. ■

Lemma 2 allows to state the subspace predictive control problem (SPC) as

$$\begin{aligned} \min_{u_N, y_N} \quad & \sum_{k=1}^{N} \left(\|y_k\|_Q^2 + \|u_k\|_R^2 \right) \\ \text{s.t.} \quad & (16) \\ & u_k \in \mathbb{U}, \ \forall k \in \{1,\ldots,N\}, \\ & y_k \in \mathbb{Y}, \ \forall k \in \{1,\ldots,N\}. \end{aligned} \quad (17)$$

Remark 2: In the original work on SPC [6] the authors choose $T_{\text{ini}} = N$. For the comparison with DeePC we follow the SPC formulation from [8] with dedicated parameter T_{ini}.

A. Equivalence of DeePC and SPC

In the following we present our main contribution, a theorem stating that subspace predictive control (17) is an equivalent representation of the DeePC (7) method.

Theorem 1: Let U_L, Y_L in the form of (4), and $u_{T_{\text{ini}}}$, $y_{T_{\text{ini}}}$ be recorded from the linear system (1). The recorded data satisfies Assumption 1, Assumption 2 and Assumption 3. The optimal control policy u_N^* and optimal output trajectory y_N^* as obtained from the solution of the DeePC problem (7) is equal to the optimal control policy u_N^* and optimal output trajectory y_N^* as obtained from the solution of the SPC problem (17).

Proof: We modify problem (7) and eliminate the variable g from the formulation. First, we split (6) into:

$$\begin{bmatrix} Y_{T_{\text{ini}}} \\ U_{T_{\text{ini}}} \\ U_N \end{bmatrix} g = \begin{bmatrix} y_{T_{\text{ini}}} \\ u_{T_{\text{ini}}} \\ u_N \end{bmatrix}, \quad (18)$$

and

$$y_N = Y_N g. \quad (19)$$

From (18) we can explicitly compute g^* as:

$$g^* = M^\dagger b + \hat{g} \quad \forall \hat{g} \in \ker(M), \quad (20)$$

where we denote with $\ker(M)$ the null-space of matrix M. Inserting (20) in (19) yields:

$$y_N = Y_N g^*, \quad (21)$$

$$\overset{(20)}{\Longleftrightarrow} y_N = Y_N (M^\dagger b + \hat{g}). \quad (22)$$

We then need to show that $\ker(M) \subseteq \ker(Y_N)$, such that

$$Y_N \hat{g} = 0 \quad \forall \hat{g} \in \ker(M). \quad (23)$$

The relationship $\ker(M) \subseteq \ker(Y_N)$ holds because P^* as obtained from (15), according to Lemma 2, satisfies (16) and thus:

$$Y_N = P^* M. \quad (24)$$

For any $x \in \ker(M)$ we have:

$$Y_N x = P^* M x = 0, \quad (25)$$

which means $x \in \ker(Y_N)$ and consequently $\ker(M) \subseteq \ker(Y_N)$. Finally, we obtain from (22):

$$y_N = Y_N (M^\dagger b) \quad (26)$$

$$= P^* b. \quad (27)$$

We have thus eliminated g from the formulation of (7), where the constraint (6) now reads:

$$y_N = P^* \begin{bmatrix} y_{T_{\text{ini}}} \\ u_{T_{\text{ini}}} \\ u_N \end{bmatrix}. \quad (28)$$

This yields the presented SPC formulation (17) and therefore proves that (7) and (17) have the same solution. ■

As a consequence of the equivalence between problem (7) and (17), we argue that the DeePC optimization problem implicitly estimates the regressor (15) at each control iteration. Since the underlying data-matrices (2) are collected prior to the control application and are unchanged afterwards, there appears to be no reason for this repeated estimation in the deterministic case. The DeePC optimization problem (7) requires $T+(m+p)N$ optimization variables with $T \geq (T_{\text{ini}}+N)m+n$ (see Assumption 1) and $(m+p)(T_{\text{ini}}+N)$ equality constraints. The SPC formulation (17) has only $(m+p)N$ optimization variables and pN equality constraints. The increased computational cost of DeePC can be also seen in the numerical example presented in Section V.

IV. Non-deterministic case

We introduce the following linear system subject to zero-mean Gaussian noise $w_k \sim \mathcal{N}(0, \sigma_w^2 I)$:

$$x_{k+1} = Ax_k + Bu_k, \tag{29a}$$
$$y_k = Cx_k + Du_k + w_k. \tag{29b}$$

As for the deterministic case, data is collected, now resulting in the matrices:

$$U_L = \begin{bmatrix} U_{T_{\text{ini}}} \\ U_N \end{bmatrix},\ \tilde{Y}_L = \begin{bmatrix} \tilde{Y}_{T_{\text{ini}}} \\ \tilde{Y}_N \end{bmatrix},\ \tilde{M} = \begin{bmatrix} \tilde{Y}_{T_{\text{ini}}} \\ U_{T_{\text{ini}}} \\ U_N \end{bmatrix}.$$

A. Non-deterministic DeePC

In the non-deterministic setting, the DeePC formulation (7) must be adapted because constraint (6) can generally not be satisfied anymore. These modifications were already proposed in the original work on DeePC [11]. Following the authors in [12], we use ℓ_2-norms instead of ℓ_1-norms to penalize the slack variables in the cost function. We furthermore propose to add an additional slack variable σ_u which, similarly to σ_y, is motivated by additive input noise. This yields the following adapted DeePC formulation:

$$\begin{aligned} \min_{g,u_N,y_N,\sigma_y,\sigma_u} \quad & \sum_{k=1}^{N} \left(\|y_k\|_Q^2 + \|u_k\|_R^2 \right) \\ & + \lambda_g \|g\|_2^2 + \lambda_{\sigma_y} \|\sigma_y\|_2^2 + \lambda_{\sigma_u} \|\sigma_u\|_2^2 \\ \text{s.t.} \quad & \begin{bmatrix} \tilde{Y}_{T_{\text{ini}}} \\ U_{T_{\text{ini}}} \\ U_N \\ \tilde{Y}_N \end{bmatrix} g = \begin{bmatrix} y_{T_{\text{ini}}} \\ u_{T_{\text{ini}}} \\ u_N \\ y_N \end{bmatrix} + \begin{bmatrix} \sigma_y \\ \sigma_u \\ 0 \\ 0 \end{bmatrix}, \\ & u_k \in \mathbb{U}\ \forall k \in \{1,\dots,N\}, \\ & y_k \in \mathbb{Y}\ \forall k \in \{1,\dots,N\}. \end{aligned} \tag{30}$$

B. Non-deterministic SPC

Subspace predictive control in the non-deterministic setting requires no modifications for the identification step:

$$\tilde{P}^* = \tilde{Y}_N \tilde{M}^\dagger. \tag{31}$$

The SPC optimization problem is adapted to account for noisy initial conditions with slack variables σ_y and σ_u.

$$\begin{aligned} \min_{u_N,y_N,\sigma_y,\sigma_u} \quad & \sum_{k=1}^{N} \left(\|y_k\|_Q^2 + \|u_k\|_R^2 \right) + \lambda_{\sigma_y} \|\sigma_y\|_2^2 + \lambda_{\sigma_u} \|\sigma_u\|_2^2 \\ \text{s.t.} \quad & y_N = \tilde{P}^* \left(\begin{bmatrix} y_{T_{\text{ini}}} \\ u_{T_{\text{ini}}} \\ u_N, \end{bmatrix} + \begin{bmatrix} \sigma_y \\ \sigma_u \\ 0, \end{bmatrix} \right) \\ & u_k \in \mathbb{U},\ \forall k \in \{1,\dots,N\}, \\ & y_k \in \mathbb{Y},\ \forall k \in \{1,\dots,N\}. \end{aligned} \tag{32}$$

The main difference in the non-deterministic setting is that $\tilde{P}^*$ now cannot exactly map $\tilde{M}$ onto $\tilde{Y}_N$. However, due to the underlying least-squares formulation (14), the obtained regressor has favorable statistical properties in this setting, such as being unbiased [18].

C. Equivalence in the non-deterministic case

Equivalence of the adapted DeePC problem (30) and SPC problem (32) in the non-deterministic case cannot generally be established. There are, however, special cases in which both methods yield the exact same optimal control policy u_N^* and optimal output trajectory y_N^*. Subsequently, we present a theorem to show equivalence in such a special case. For this purpose, we reformulate the adapted DeePC problem (30) and SPC problem (32). We stack the optimization variables $v = [\sigma_y,\ \sigma_u,\ u_N]$ and introduce the block-diagonal weighting matrices:

$$\tilde{Q} = \begin{bmatrix} Q & & \\ & \ddots & \\ & & Q \end{bmatrix},\ \tilde{R} = \begin{bmatrix} R & & \\ & \ddots & \\ & & R \end{bmatrix},\ V = \begin{bmatrix} \lambda_{\sigma_y} & & \\ & \lambda_{\sigma_u} & \\ & & \tilde{R} \end{bmatrix}.$$

We then drop the inequality constraints in (30) and (34). This allows to state the unconstrained version of the adapted DeePC problem (30) as:

$$\begin{aligned} \min_{g,v,y_N} \quad & \frac{1}{2} g^\intercal (\lambda_g I) g + \frac{1}{2} y_N^\intercal \tilde{Q} y_N + \frac{1}{2} v^\intercal V v \\ \text{s.t.} \quad & y_N = \tilde{Y}_N g \\ & \underbrace{\begin{bmatrix} \tilde{Y}_{T_{\text{ini}}} \\ U_{T_{\text{ini}}} \\ U_N \end{bmatrix}}_{\tilde{M}} g = \underbrace{\begin{bmatrix} y_{T_{\text{ini}}} \\ u_{T_{\text{ini}}} \\ 0 \end{bmatrix}}_{b} + \underbrace{\begin{bmatrix} \sigma_y \\ \sigma_u \\ u_N \end{bmatrix}}_{v}, \end{aligned} \tag{33}$$

and the unconstrained version of the SPC problem (32) as:

$$\begin{aligned} \min_{v,y_N} \quad & \frac{1}{2} y_N^\intercal \tilde{Q} y_N + \frac{1}{2} v^\intercal V^\intercal v \\ \text{s.t.} \quad & y_N = \tilde{P} \left(\underbrace{\begin{bmatrix} y_{T_{\text{ini}}} \\ u_{T_{\text{ini}}} \\ 0 \end{bmatrix}}_{b} + \underbrace{\begin{bmatrix} \sigma_y \\ \sigma_u \\ u_N \end{bmatrix}}_{v} \right). \end{aligned} \tag{34}$$

Dropping the inequality constraints in (33) and (34) has been considered previously for DeePC in [15] and SPC in [8] as it allows for explicit solutions of (33) and (34) which are preferable for very fast control applications. Below, we state these explicit solutions in our notation.

Result 1: The optimal value v^* for (33) is computed explicitly as:

$$v^* = \tilde{M}(\lambda_g I + \tilde{M}^\intercal V \tilde{M} + Y_N^\intercal \tilde{Q} Y_N)^{-1} \tilde{M}^\intercal V b - b. \tag{35}$$

Proof: The proof is shown in the appendix. ∎

Result 2: The optimal value v^* for (34) is computed explicitly as:

$$v^* = -(V + \tilde{P}^\intercal Q \tilde{P})^{-1} \tilde{P}^\intercal \tilde{Q} \tilde{P} b. \tag{36}$$

Proof: The proof is shown in the appendix. ∎

We use Result 1 and Result 2 to establish equivalence of the DeePC and SPC solution v^* under certain conditions.

Theorem 2: Let $T = Lm + T_{\text{ini}}p$ and Assumption 1, Assumption 2 and Assumption 3 hold. Let $M \in \mathbb{R}^{Lm+T_{\text{ini}}p \times T}$ and $Y_N \in \mathbb{R}^{Np \times T}$ have full rank. The unconstrained DeePC problem (33) and unconstrained SPC problem (34) yield the same optimal solution v^*, if $\lambda_g = 0$.

Proof: We reformulate (36) by applying the following identity [19]:

$$(\Lambda + \Gamma)^{-1} = \Lambda^{-1} - (\Lambda + \Gamma)^{-1} \Gamma \Lambda^{-1} \tag{37}$$

with

$$\Lambda = \tilde{P}^\intercal \tilde{Q} \tilde{P}, \quad \Gamma = V. \tag{38}$$

Note that (37) only holds if Λ and $\Lambda + \Gamma$ are non-singular. According to the assumptions, we have that $\tilde{M}$ and $\tilde{Y}_N$ have full rank and $\tilde{M}$ is quadratic, such that:

$$\tilde{P}^* = Y_N \tilde{M}^\dagger = \tilde{Y}_N \tilde{M}^{-1} \tag{39}$$

also has full rank. By definition Q, R and therefore $\tilde{Q}$ and V are positive definite. Therefore we have that $\tilde{P}^\intercal \tilde{Q} \tilde{P}$ and $\tilde{P}^\intercal \tilde{Q} \tilde{P} + V$ are positive definite. Identity (37) can thus be applied to the explicit solution of the SPC problem (36) and yields:

$$\begin{aligned} v^* &= \left[-(\tilde{P}^\intercal \tilde{Q} \tilde{P})^{-1} + (V + \tilde{P}^\intercal \tilde{Q} \tilde{P})^{-1} V (\tilde{P}^\intercal \tilde{Q} \tilde{P})^{-1} \right] \tilde{P}^\intercal \tilde{Q} \tilde{P} b \\ &= (V + \tilde{P}^\intercal \tilde{Q} \tilde{P})^{-1} V b - b. \end{aligned} \tag{40}$$

We substitute (39) in (40):

$$\begin{aligned} v^* &= \left(V + (\tilde{M}^{-1})^\intercal \tilde{Y}_N^\intercal \tilde{Q} \tilde{Y}_N \tilde{M}^{-1} \right)^{-1} V b - b \\ &= \left(\left(\tilde{M}^{-1} \right)^\intercal \left(\tilde{M}^\intercal V \tilde{M} + \tilde{Y}_N^\intercal \tilde{Q} \tilde{Y}_N \right) \left(\tilde{M}^{-1} \right) \right)^{-1} V b - b, \end{aligned} \tag{41}$$

and computing the outer inverse, we obtain

$$v^* = \tilde{M} \left(\tilde{M}^\intercal V \tilde{M} + \tilde{Y}_N^\intercal \tilde{Q} \tilde{Y}_N \right)^{-1} \tilde{M}^\intercal V b - b. \tag{42}$$

This means that if $\lambda_g = 0$ as required in the theorem, the explicit solution (35) of the unconstrained adapted DeePC problem (30) is equal to (42) and therefore equal to the explicit solution (36) of the unconstrained SPC problem (32). ∎

Clearly the equivalence established in Theorem 2 holds only under strong assumptions (i.e. $T = Lm + T_{\text{ini}}p$ and $\lambda_g = 0$) and is shown here only for the unconstrained formulation of DeePC (33) and SPC (34). Furthermore, the presented DeePC formulation in (30) and (33) deviates from formulations in previous works through the addition of the slack variable σ_u. In future work, we therefore aim to investigate the constrained case which would also allow to enforce $\sigma_u = 0$.

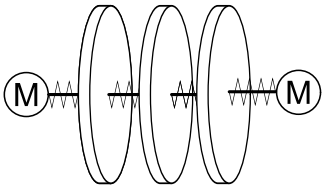

Fig. 1. Triple-mass-spring system (rotating) with two stepper motors as inputs (motor angle). Disc angles can be measured.

Bearing in mind the limitations of Theorem 2, we still argue that it shows that DeePC and SPC are very closely related, even in the non-deterministic formulation. This close relationship also holds in practice as shown in [15] and will be further illustrated in Section V. The main reason for deviations between the DeePC and SPC solution in the general case is due to the regularization of the decision variable g in DeePC with $\lambda_g > 0$. It becomes clear, by inspecting (35), that this regularization is required in the general case, as $\tilde{M}^\intercal V \tilde{M} + \tilde{Y}_N^\intercal \tilde{Q} \tilde{Y}_N$ is singular for $T > \max(Lm + T_{\text{ini}}p,\ Np)$. In Section V we numerically investigate the effect of λ_g on the obtained solution.

V. Simulation study

In this section, we present numerical evidence of our findings. We investigate a simple LTI system in the form of (1) in two scenarios: with and without additive Gaussian measurement noise. The investigated system, displayed in Figure 1, is a triple-mass-spring system (rotating discs) with two stepper motors ($m = 2$) attached to the outermost discs via additional springs. These disc angles are the measured output of the system ($p = 3$). The system has a total of $n = 8$ states. All materials to reproduce the results in this section (as well as the investigated system) are available in our accompanying repository[1]. According to Definition 2 we determine the system lag $l = 2$. To satisfy Assumption 2, we choose $T_{\text{ini}} = 4$. The prediction horizon is chosen as $N = 40$, which represents 4s. This means we must collect a minimum of $T > Lm + n = 96$ sequences for our data matrices in (2) to satisfy Assumption 1. Note that data is collected from independent experiments with random initial states fulfilling Assumption 3. The regulation objective is parameterized with $Q = I$ and $R = 0.1I$ and we constrain the inputs to $\mathbb{U} = \{-0.7, 0.7\}$. All controller variants are implemented using CasADi [20] with IPOPT [21].

A. Deterministic system

We first consider the described system without measurement noise and compare the deterministic DeePC algorithm (7) with SPC (17). The results are obtained with data from $T = 150$ captured sequences. In the deterministic setting, we have shown in Theorem 1 that both methods yield identical solutions. In Figure 2 we can see that the identical behavior also appears in practice. We present here an exemplary

[1] https://github.com/4flixt/DeePC_Perspective

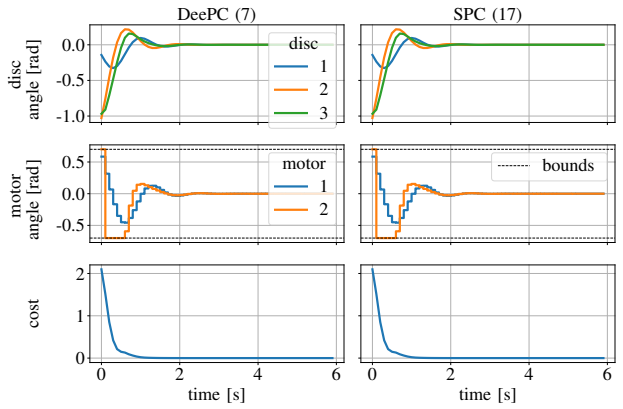

Fig. 2. Closed-loop trajectories of the mass-spring-system from Figure 1. Regulation after some initial excitation. Comparison of DeePC (7) and SPC (17) with deterministic data.

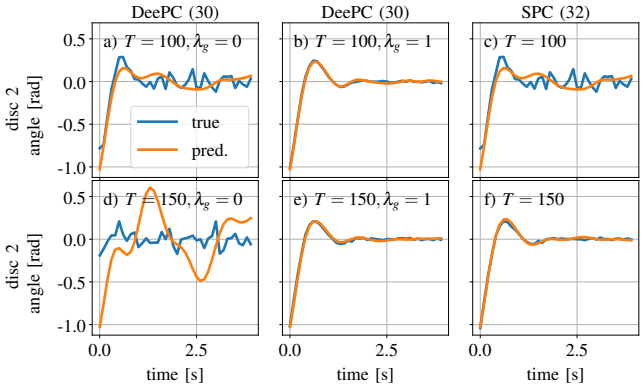

Fig. 3. Open-loop trajectories for the non-deterministic case with DeePC (30) vs. SPC (32). Comparison of predicted vs. true system response (when subjected to the optimal control trajectory). Only disc 2 (see Figure 2) is displayed for clarity.

closed-loop trajectory for both methods with random (but identical) excitation phase to obtain $y_{T_{\text{ini}}}$ and $u_{T_{\text{ini}}}$. To further quantify the similarity, we compare the cumulated cost:

$$c = \sum_{k=1}^{N_{\text{sim}}} \left(y_k^\intercal Q y_k + u_k^\intercal R u_k \right). \tag{43}$$

For the results in Figure 2 the cumulative cost differs between both controller variants on the order of 10^{-11} and equals to 5.617, which will serve as a benchmark for the non-deterministic case.

B. Non-deterministic system

In this subsection we investigate the previously described LTI system in the form of (29), where we choose $\sigma_w = 10^{-2}$ in all experiments. We choose $\lambda_y = 10^4$ and $\lambda_u = 10^4$ for non-deterministic formulations of DeePC (30) and SPC (32).

We want to compare the performance of both approaches and study the effect of the tuning parameter λ_g and of the number of recorded sequences T on the cost, according to (43), and computation time.

In Figure 3 we present predicted optimal trajectories obtained with (30) and (32) for the same initial input/output data $\tilde{y}_{T_{\text{ini}}}$ and $u_{T_{\text{ini}}}$ (noise disturbed). These trajectories (denoted as pred. in Figure 2) are compared with the true system response resulting from the sequence of optimal inputs. We vary the number of recorded sequences $T = \{100, 150\}$ and change the value of $\lambda_g = \{0, 1\}$. For a more concise representation we only showcase the output of the second disc angle. Notice that $T = 100$ barely exceeds $T = 96$, the lower bound to satisfy Assumption 1. With $T = 100$ and $\lambda_g = 0$ we also satisfy the assumptions for Theorem 2, meaning that we expect identical solutions from the unconstrained DeePC problem (33) and the SPC problem (34). This equivalence can also be observed for the constrained DeePC and SPC formulations in Figure 3 when comparing case a) and c). The open loop-cost of these cases differs on the order of 10^{-13}. Another observation from Figure 3 case d) is the unsatisfactory performance of DeePC, which stems from the fact that with $\lambda_g = 0$ and $T > \max(Lm + T_{\text{ini}}p,\ Np) = 120$ a singular matrix arises in the computation of v^*. Good performance with SPC can be obtained with $T = 150$ as seen in case f) or with DeePC for $T = 150$ and $\lambda_g = 1$ as shown in case e).

TABLE I

CLOSED-LOOP TRAJECTORIES FOR THE NON-DETERMINISTIC CASE WITH DEEPC (30) VS. SPC (32).

		Number of sequences T			**Ref.**
		100	150	200	
DeePC ($\lambda_g = 1$)	cost (43)	5.924	5.674	5.662	5.617
	comp. time [ms]	25.272	39.419	59.291	-
SPC	cost (43)	5.810	5.64	5.632	5.617
	comp. time [ms]	17.30	16.25	16.13	-

A further comparison is presented in Table I, where we analyze closed-loop trajectories obtained with DeePC (30), with $\lambda_g = 1$, and SPC (32). All results in this table are averages over ten simulation experiments, with independently sampled data matrices and measurement noise.

We take two main conclusions from the presented data in Table I. First, we see that SPC outperforms DeePC both in terms of overall cost and computation time in all scenarios. While the advantage of SPC with respect to the cost is minor, the difference in computation time is significant. Especially for the case $T = 150$, which seems to be a good compromise between performance and cost for DeePC and SPC, we notice a significant increase in computation time. Note that we compare only the online CPU time. The computation of the pseudo inverse is excluded, as it is performed offline and only once. As a second conclusion, we notice that even with $T = 100$, DeePC is outperformed by SPC. This is counter intuitive in comparison to Figure 3 case b) and c), where open-loop trajectories seem better with DeePC. However, open-loop predictions are not the same as close-loop control and we find that SPC shows less oscillatory behavior around the origin (not shown here), thus leading to a slightly lower closed-loop cost.

Note that we also investigated DeePC (30), with $\lambda_g \in \{0.1, 10\}$ and $T = 150$. Compared to the displayed case $\lambda_g = 1$ and $T = 150$ in Table I, the different parameters for λ_g lead to increased closed-loop costs, according to (43), of 5.707 and 5.984. As a conclusion, we found that $\lambda_g = 1$ is an appropriate choice for the presented comparison.

VI. Conclusions

In this work we investigated the relationship between data-enabled predictive control (DeePC) and subspace predictive control (SPC). Both methods require for their formulation the exact same data matrices, satisfying identical requirements. With DeePC, these matrices are directly incorporated in the optimal control problem, whereas SPC requires the data to identify a multi-step prediction model. We show that in the case of deterministic data, both formulations are equivalent. From the equivalence, we reason that DeePC implicitly estimates the same multi-step ahead prediction model at each iteration, thus adding significant additional computational cost to its online application. In particular, the online computational cost of DeePC grows with the number of collected data sequences.

For the non-deterministic case, the exact equivalence between DeePC and SPC does not hold in the general case. We propose a minor modification of DeePC and prove that this formulation is equivalent to SPC in the unconstrained case under special conditions. Even though the presented equivalence only holds under strong assumptions, it illustrates the close relationship between both methods.

In simulation studies with an exemplary LTI system with and without additive Gaussian noise, we showcase that the derived equivalences also hold in practice. Furthermore, we investigate the non-deterministic DeePC and SPC formulation with respect to the number of recorded data sequences and the regularization term for DeePC. We find that SPC outperforms DeePC in the investigated cases with minor improvements of the closed-loop cost and significant improvements in the online computation time.

In future work, we seek to further investigate the relationship between both methods and aim to extend the equivalence of the stochastic formulations to the constrained case. We also seek to compare both methods for nonlinear or time-varying systems where the implicit re-estimation of DeePC could be beneficial.

Appendix

A. Proof of Lemma 2

Proof: With the observability matrix $O_i(A,C)$ and the Toeplitz matrix $H_i(A,B,C,D)$

$$O_i = \begin{bmatrix} C, \\ CA, \\ \vdots \\ CA^{i-1} \end{bmatrix}, \; H_i = \begin{bmatrix} D & & & \\ CB & D & & \\ \vdots & & \ddots & \\ CA^{i-2}B & CA^{i-3}B & \dots & D \end{bmatrix}, \tag{44}$$

if the initial state x_1 is known, we can obtain $y_{T_{\text{ini}}}$ as:

$$y_{T_{\text{ini}}} = O_{T_{\text{ini}}} x_1 + H_{T_{\text{ini}}} u_{T_{\text{ini}}}, \tag{45}$$

$$\Leftrightarrow y_{T_{\text{ini}}} = \begin{bmatrix} O_{T_{\text{ini}}} & H_{T_{\text{ini}}} \end{bmatrix} \begin{bmatrix} x_1 \\ u_{T_{\text{ini}}} \end{bmatrix}, \tag{46}$$

Since the recorded sequences (4), (3) are trajectories of system (1) they must satisfy:

$$Y_{T_{\text{ini}}} = \begin{bmatrix} O_{T_{\text{ini}}} & H_{T_{\text{ini}}} \end{bmatrix} \begin{bmatrix} X_1 \\ U_{T_{\text{ini}}} \end{bmatrix}. \tag{47}$$

From (45) we can also uniquely obtain x_1 with $y_{T_{\text{ini}}}$ and $u_{T_{\text{ini}}}$ because Assumption 2 holds. This means there exists a known state-space representation of (1) with $\tilde{x} = \left[y_{T_{\text{ini}}}^\intercal, u_{T_{\text{ini}}}^\intercal \right]^\intercal$ and some system matrices $\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}$. Similarly as above, this allows to compute the future trajectory as:

$$y_N = \tilde{O}_N \tilde{x} + \tilde{H}_N u_N, \tag{48}$$

$$\Leftrightarrow \; y_N = \begin{bmatrix} \tilde{O}_N & \tilde{H}_N \end{bmatrix} \begin{bmatrix} y_{T_{\text{ini}}} \\ u_{T_{\text{ini}}} \\ u_N \end{bmatrix} \tag{49}$$

where $\tilde{O}_N$ is the observability matrix and $\tilde{H}_N$ is the Toeplitz matrix of the transformed system $(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D})$. The recorded sequences (4) must satisfy

$$Y_N = \begin{bmatrix} \tilde{O}_N & \tilde{H}_N \end{bmatrix} \begin{bmatrix} Y_{T_{\text{ini}}} \\ U_{T_{\text{ini}}} \\ U_N \end{bmatrix} = \begin{bmatrix} \tilde{O}_N & \tilde{H}_N \end{bmatrix} M. \tag{50}$$

We multiply both sides of the equation with $M^\dagger M$:

$$Y_N M^\dagger M = \begin{bmatrix} \tilde{O}_N & \tilde{H}_N \end{bmatrix} M M^\dagger M. \tag{51}$$

Considering the properties of the Moore-Penrose inverse ($MM^\dagger M = M$) we can write:

$$Y_N M^\dagger M = \begin{bmatrix} \tilde{O}_N & \tilde{H}_N \end{bmatrix} M, \tag{52}$$

which yields, considering (15),:

$$P^* M = \begin{bmatrix} \tilde{O}_N & \tilde{H}_N \end{bmatrix} M \tag{53}$$

We thus have that the projection of M with P^* is equivalent to the projection of M with the system matrices $\tilde{O}_N$ and $\tilde{H}_N$. From (50), we see that:

$$Y_N = P^* M, \tag{54}$$

which also holds for all linear combinations of the columns of M and Y_N with $\alpha \in \mathbb{R}^T$:

$$Y_N \alpha = P^* M \alpha \qquad \forall \alpha \in \mathbb{R}^T \tag{55}$$

$$\Leftrightarrow y_N = P^* \begin{bmatrix} u_{T_{\text{ini}}} \\ u_N \\ y_{T_{\text{ini}}} \end{bmatrix} \quad \forall \begin{bmatrix} y_{T_{\text{ini}}} \\ u_{T_{\text{ini}}} \\ u_N \end{bmatrix} \in \text{colspan}(M). \tag{56}$$

We thus have that any $\left[y_{T_{\text{ini}}}^\intercal, u_{T_{\text{ini}}}^\intercal, u_N^\intercal \right]^\intercal \in \text{colspan}(M)$ together with the resulting y_N computed via (16) is a sequence of system (1). To show that no other sequences $\left[y_{T_{\text{ini}}}^\intercal, u_{T_{\text{ini}}}^\intercal, u_N^\intercal \right]^\intercal \notin \text{colspan}(M)$ are trajectories of system (1), we introduce:

$$P^* = \begin{bmatrix} P^*_{y_{\text{ini}}}, P^*_{u_{\text{ini}}}, P^*_{u_N} \end{bmatrix} \tag{57}$$

With (47), (53) and (57), we can write:

$$Y_N = \begin{bmatrix} P^*_{y_{\text{ini}}} O_{T_{\text{ini}}} & P^*_{y_{\text{ini}}} H_{T_{\text{ini}}} + P^*_{u_{\text{ini}}} & P^*_{u_N} \end{bmatrix} \begin{bmatrix} X_1 \\ U_{T_{\text{ini}}} \\ U_N \end{bmatrix}. \quad (58)$$

By Assumption 1 and Assumption 3 the matrix $\begin{bmatrix} X_1^\intercal & U_{T_{\text{ini}}}^\intercal & U_N \end{bmatrix}^\intercal$ has full rank and thus spans the space of all initial states $x_1 \in \mathbb{R}^n$ and sequences $u_{T_{\text{ini}}} \in \mathbb{R}^{T_{\text{ini}}m}$, $u_N \in \mathbb{R}^{Nm}$ for which all possible sequences $y_{T_{\text{ini}}} \in \mathbb{R}^{T_{\text{ini}}p}$ according to (46) and $y_N \in \mathbb{R}^{Np}$ according to (16) are obtained. This shows that $\begin{bmatrix} y_{T_{\text{ini}}}^\intercal, u_{T_{\text{ini}}}^\intercal, u_N^\intercal, \end{bmatrix}^\intercal \in \text{colspan}(M)$ spans the subspace of all possible sequences of (1) for which (56) and thus (16) holds. ■

B. Proof of Result 1

We state the Lagrangian for problem (33):

$$\begin{aligned} \mathscr{L} = &\frac{1}{2} g^\intercal (\lambda_g I) g + \frac{1}{2} y_N^\intercal \tilde{Q} y_N + \frac{1}{2} v^\intercal V v \\ &- \mu_1^\intercal (y_N - \tilde{Y}_N g) - \mu_2^\intercal (\tilde{M} g - b - v) \end{aligned} \quad (59)$$

The first order conditions of optimality for problem (33) can be written in terms of the Lagrangian as:

$$\nabla_{y_N} \mathscr{L} = y_N^\intercal \tilde{Q} - \mu_1^\intercal = 0 \quad (60)$$
$$\nabla_v \mathscr{L} = v^\intercal V + \mu_2^\intercal = 0 \quad (61)$$
$$\nabla_g \mathscr{L} = g^\intercal (\lambda_g I) + \mu_1^\intercal \tilde{Y}_N - \mu_2^\intercal \tilde{M} = 0 \quad (62)$$
$$\nabla_{\mu_1} \mathscr{L} = -y_N^\intercal + g^\intercal \tilde{Y}_N^\intercal = 0 \quad (63)$$
$$\nabla_{\mu_2} \mathscr{L} = -g^\intercal \tilde{M}^\intercal + b^\intercal + v^\intercal = 0 \quad (64)$$

Inserting (60) and (61) into (62) yields:

$$g^\intercal (\lambda_g I) + y_N^\intercal \tilde{Q} \tilde{Y}_N + v^\intercal V \tilde{M} = 0. \quad (65)$$

Next we insert (63) and (64) in (65) and rearrange, which yields:

$$g = (\lambda_g I + \tilde{M}^\intercal V \tilde{M} + \tilde{Y}_N^\intercal \tilde{Q} \tilde{Y}_N)^{-1} \tilde{M}^\intercal V b. \quad (66)$$

The inverse in (66) always exists because $\tilde{M}^\intercal V \tilde{M}$ and $\tilde{Y}_N^\intercal \tilde{Q} \tilde{Y}_N$ are positive semi-definite and $\lambda_g I$ is positive definite. With (64) we obtain:

$$v^* = \tilde{M} (\lambda_g I + \tilde{M}^\intercal V \tilde{M} + \tilde{Y}_N^\intercal \tilde{Q} \tilde{Y}_N)^{-1} \tilde{M}^\intercal V b - b \quad (67)$$

C. Proof of Result 2

Proof: We state the Lagrangian for problem (34):

$$\mathscr{L} = \frac{1}{2} y_N^\intercal Q y_N + \frac{1}{2} v^\intercal V v - \mu^\intercal (y_N - \tilde{P} b - \tilde{P} b) \quad (68)$$

The first order conditions of optimality for problem (34) can be written in terms of the Lagrangian as:

$$\nabla_{y_N} \mathscr{L} = y_N^\intercal \tilde{Q} - \mu^\intercal = 0 \quad (69)$$
$$\nabla_v \mathscr{L} = v^\intercal V + \mu^\intercal \tilde{P} = 0 \quad (70)$$
$$\nabla_\mu \mathscr{L} = -y_N^\intercal + v^\intercal \tilde{P}^\intercal + b^\intercal \tilde{P}^\intercal = 0 \quad (71)$$

Inserting (71) in (69) and the resulting expression in (70) yields:

$$v^* = -(V + \tilde{P}^\intercal Q \tilde{P})^{-1} \tilde{P}^\intercal \tilde{Q} \tilde{P} b \quad (72)$$

after rearranging. The inverse in (72) always exists, because $\tilde{P}^\intercal \tilde{Q} \tilde{P}$ is positive semi-definite and by assumption V is positive definite. ■

ACKNOWLEDGMENT

The authors want to express their sincere gratitude to the reviewers for their extensive and constructive feedback. It has lead to many interesting discussions, new thoughts and important improvements of this work.

REFERENCES

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

[2] L. Ljung, *System Identification: Theory for the User*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1987.

[3] D. F. Delchamps, *State Space and Input-Output Linear Systems*. New York, NY, USA: Springer, 1988.

[4] A. Juditsky, H. Hjalmarsson, A. Benveniste, B. Delyon, L. Ljung, J. Sjöberg, and Q. Zhang, "Nonlinear black-box models in system identification: Mathematical foundations," *Automatica*, vol. 31, no. 12, pp. 1725–1750, Dec. 1995.

[5] L. Ljung, "System Identification," in *Wiley Encyclopedia of Electrical and Electronics Engineering*. Wiley, 2017, pp. 1–19.

[6] W. Favoreel, B. De Moor, and M. Gevers, "SPC: Subspace predictive control," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 4004–4009, 1999.

[7] B. Huang and R. Kadali, *Dynamic Modeling, Predictive Control and Performance Monitoring: A Data-Driven Subspace Approach*. Springer, 2008.

[8] S. Sedghizadeh and S. Beheshti, "Data-driven subspace predictive control: Stability and horizon tuning," *Journal of the Franklin Institute*, vol. 355, no. 15, pp. 7509–7547, 2018.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning, Second Edition: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[10] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 253–279, 2019.

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

[12] J. Berberich, J. Koehler, M. A. Muller, and F. Allgower, "Data-Driven Model Predictive Control with Stability and Robustness Guarantees," *IEEE Transactions on Automatic Control*, pp. 1–1, 2020.

[13] J. Coulson, J. Lygeros, and F. Dörfler, "Distributionally robust chance constrained data-enabled predictive control," *arXiv preprint arXiv:2006.01702*, 2020.

[14] D. Alpago, F. Dörfler, and J. Lygeros, "An Extended Kalman Filter for Data-Enabled Predictive Control," *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 994–999, 2020.

[15] P. G. Carlet, A. Favato, S. Bolognani, and F. Dörfler, "Data-driven predictive current control for synchronous motor drives," in *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE, 2020, pp. 5148–5154.

[16] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, "A note on persistency of excitation," *Systems & Control Letters*, vol. 54, no. 4, pp. 325–329, 2005.

[17] I. Markovsky, *Exact and Approximate Modeling of Linear Systems: A Behavioral Approach*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2006.

[18] M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least Squares Approach*, 1st ed. New York, NY, USA: Cambridge University Press, 2007.

[19] H. V. Henderson and S. R. Searle, "On deriving the inverse of a sum of matrices," *Siam Review*, vol. 23, no. 1, pp. 53–60, 1981.

[20] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.

[21] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 September 2023, accepted 16 October 2023, date of publication 20 October 2023, date of current version 26 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3326344

RESEARCH ARTICLE

Probabilistic Multi-Step Identification With Implicit State Estimation for Stochastic MPC

FELIX FIEDLER AND SERGIO LUCIA, (Member, IEEE)
Chair of Process Automation Systems, TU Dortmund University, 44227 Dortmund, Germany
Corresponding author: Felix Fiedler (felix.fiedler@tu-dortmund.de)

ABSTRACT Stochastic Model Predictive Control (SMPC) is a promising solution for controlling multivariable systems in the presence of uncertainty. However, a core challenge lies in obtaining a probabilistic system model. Recently, multi-step system identification has been proposed as a solution. Multi-step models simultaneously predict a finite sequence of future states, which traditionally involves recursive evaluation of a state-space model. Particularly in the stochastic context, the recursive evaluation of identified state-space models has several drawbacks, making multi-step models an appealing choice. As a main novelty of this work, we propose a probabilistic multi-step identification method for a linear system with noisy state measurements and unknown process and measurement noise covariances. We show that, in expectation, evaluating the identified multi-step model is equivalent to estimating the initial state distribution and subsequently propagating this distribution using the known system dynamics. Therefore, using only recorded data of an unknown linear system, our proposed method yields a probabilistic multi-step model, including the state estimation task, that can be directly used for SMPC. As an additional novelty, our proposed SMPC formulation considers parametric uncertainties of the identified multi-step model. We demonstrate our method in two simulation studies, showcasing its effectiveness even for a nonlinear system with output feedback.

INDEX TERMS Stochastic model predictive control, system identification, multi-step identification, data-based control.

I. INTRODUCTION

Model predictive control (MPC) is a popular strategy to control multivariable systems with constraints [1]. At its core, MPC uses a system model to predict and optimize the future behavior of the system. Obtaining a suitable system model is therefore a central requirement for MPC and has been recognized as a major challenge for decades [2]. Traditionally, state-space representations of the prediction model have been used for MPC and other control schemes. A state-space representation is the obvious choice for models that are obtained from first principles and typically employed for data-based system identification [3], [4]. Under ideal conditions, state-space models can exactly describe the system dynamics with the fewest number of parameters. To obtain predictions over multiple timesteps, as required for MPC, the state-space model can be evaluated recursively. However, a recent trend in system identification is to directly identify multi-step prediction models [5], [6], [7], [8].

A multi-step model can simultaneously predict finite sequences of future states with an individual function for each step of the sequence. At first glance, a multi-step model appears to add unnecessary complexity to the system identification and control task. However, recent work has shown that multi-step identification can have significant advantages over state-space identification [7], [8]. Most importantly, multi-step models can have a better accuracy than state-space models and complexity is added only to the offline identification task, not to the online control task.

A closely related trend to multi-step model identification is data-enabled predictive control (DeePC) [9], [10], [11], [12]. DeePC draws from behavioral systems theory and combines an implicit multi-step model identification and control task in a single optimization problem. The close relationship

The associate editor coordinating the review of this manuscript and approving it for publication was Ton Duc Do.

between DeePC and MPC with multi-step prediction models has recently been shown in [12], [13].

Another major challenge for the application of MPC is the presence of uncertainty. While nominal MPC provides a certain robustness, it is often not sufficient, especially if the system operates close to safety-critical constraints. A promising approach to handle this situation is stochastic MPC (SMPC) [14], [15]. SMPC can be applied when the uncertainty in the model follows a known probability distribution function and it relies on the formulation of chance constraints, that is, constraints with specified probability of violation.

Tackling the challenges of data-based identification and control under uncertainty with state-space and multi-step models is a prominent field of research [6], [7], [8], [17]. There are, however, important limitations in previous works using multi-step identification and SMPC. For example, it was previously assumed that the process and measurement noise covariances are known for the identification of the probabilistic multi-step model [8]. Unfortunately, this assumption significantly limits the applicability in practice, as the process noise typically represents unknown disturbances of the system. Additionally, most stochastic MPC formulations require state-feedback to solve the optimal control problem [14], [15], which is often not available in practice.

As a main contribution of this work, we propose a stochastic MPC formulation based on multi-step identification without knowledge of the process and measurement noise covariances. Our proposed method is derived for systems with noisy state measurements and we formulate the objective function and chance constraints of the SMPC problem in terms of these measurements. We show that with maximum likelihood estimation, we identify a multi-step model that, in expectation, describes the true distribution of the future measurements of the system. Crucially, this true distribution also considers the uncertainty for the initial state which is estimated from the initial measurement using a Kalman filter [1]. Therefore, evaluating our identified multi-step model with noisy state measurements is equivalent to estimating the initial state distribution and subsequently propagating this distribution using the known system dynamics. Finally, we show that this property is a unique advantage of multi-step models and does not apply to recursively evaluated state-space models.

The identified multi-step model is thus readily applicable for the formulation of an SMPC problem with noisy state-feedback. As another contribution of this work, our proposed SMPC formulation directly considers the parametric uncertainties of the identified multi-step model. This is in contrast to previous work, where the parametric uncertainties were included as an ellipsoidal uncertainty set [8]. As a final contribution, we investigate the proposed SMPC controller based on multi-step models, comparing it to a variant relying on identified state-space models. We conduct this evaluation in two simulation studies, showcasing the effectiveness of SMPC with identified multi-step model, even for a nonlinear system with output feedback.

This paper is structured as follows. In Section II, we introduce the stochastic optimal control problem for a linear dynamic system with known system matrices and noise covariances. In Section III, we assume that this information is not available and discuss probabilistic multi-step identification from data. In Section IV, the proposed SMPC problem based on the identified multi-step model is presented. We investigate our method with a linear building system in Section VI, and a nonlinear CSTR system in Section VII. The work is concluded in Section VIII.

II. STOCHASTIC MODEL PREDICTIVE CONTROL

We consider a linear and uncertain dynamic system given in the discrete-time formulation as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{E}_x\boldsymbol{e}_{x,k}, \tag{1a}$$

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{E}_y\boldsymbol{e}_{y,k}, \tag{1b}$$

with states $\boldsymbol{x} \in \mathbb{R}^{n_x}$, inputs $\boldsymbol{u} \in \mathbb{R}^{n_u}$, measurements $\boldsymbol{y} \in \mathbb{R}^{n_y}$, process noise $\boldsymbol{e}_{x,k} \in \mathbb{R}^{n_{e,x}}$ and measurement noise $\boldsymbol{e}_{y,k} \in \mathbb{R}^{n_{e,y}}$. The system is described with the matrices $\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{B} \in \mathbb{R}^{n_x \times n_u}$, $\boldsymbol{C} \in \mathbb{R}^{n_y \times n_x}$, $\boldsymbol{E}_x \in \mathbb{R}^{n_x \times n_{e,x}}$ and $\boldsymbol{E}_y \in \mathbb{R}^{n_y \times n_{e,y}}$. As in previous related work [8], we introduce the following assumption.

Assumption 1: The system (1) is subject to noisy state-feedback, that is $\boldsymbol{C} = \boldsymbol{I}$.

For practical applications, this assumption can be relaxed to output-feedback, as we demonstrate in Section VII. Furthermore, we assume the following properties of the system noise.

Assumption 2: The system (1) is subject to additive Gaussian process noise and measurement noise, that is: $\boldsymbol{e}_{x,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_x)$ *and* $\boldsymbol{e}_{y,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_y)$, $\forall k$.

As a main premise of this work, we consider the system matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{E}_x$ and $\boldsymbol{E}_y$ and the covariance matrices $\boldsymbol{\Sigma}_x$ and $\boldsymbol{\Sigma}_y$ as unknown. In the following subsection, we derive multi-step models from the state-space representation in (1), and propose an identification method of the parameters of the multi-step model in Section III.

A. MULTI-STEP PREDICTIONS

To formulate the stochastic model predictive control problem in Subsection II-C, we require the probability distribution of the future sequence of system measurements. If the system matrices are known, a multi-step prediction can be obtained by recursively evaluating the state state-space model in (1). Considering Assumption 1, we have:

$$\begin{aligned}\boldsymbol{y}_{[1,N]} = {} & \mathcal{O}(\boldsymbol{A})\boldsymbol{x}_0 + \mathcal{T}(\boldsymbol{A}, \boldsymbol{B})\boldsymbol{u}_{[0,N-1]} \\ & + \mathcal{T}(\boldsymbol{A}, \boldsymbol{E}_x)\boldsymbol{e}_{x,[0,N-1]} + (\boldsymbol{I}_N \otimes \boldsymbol{E}_y)\boldsymbol{e}_{y,[1,N]},\end{aligned} \tag{2}$$

where $\boldsymbol{y}_{[1,N]}^\top = \left[\boldsymbol{y}_1^\top, \ldots, \boldsymbol{y}_N^\top\right]$ denotes a finite sequence of outputs, $\otimes$ is the Kronecker product and $\boldsymbol{I}_N$ is the identity matrix of dimension N. The element in block i, j of the

matrices $\mathcal{T}(\cdot,\cdot)$ and $\mathcal{O}(\cdot)$ can be obtained as:

$$[\mathcal{T}(\boldsymbol{A},\boldsymbol{B})]_{i,j} = \begin{cases} \boldsymbol{A}^{i-j}\boldsymbol{B}, & \text{if } i \geq j \\ \boldsymbol{0}, & \text{otherwise} \end{cases} \quad \forall i,j \in \mathbb{I}_{[1,N]}, \tag{3a}$$

$$[\mathcal{O}(\boldsymbol{A})]_{i,1} = \boldsymbol{A}^i \quad \forall i \in \mathbb{I}_{[1,N]}. \tag{3b}$$

We denote with $\mathbb{I}_{[1,N]}$ the set of integers from 1 to N. Considering (2) and Assumption 2, it follows for the distributed multi-step predictions:

$$p(\boldsymbol{y}_{[1,N]}|\boldsymbol{x}_0, \boldsymbol{u}_{[0,N-1]}) = \mathcal{N}(\bar{\boldsymbol{y}}_{[1,N]}, \boldsymbol{\Sigma}_{y,[1,N]}), \tag{4a}$$

with:

$$\bar{\boldsymbol{y}}_{[1,N]} = \mathcal{O}(\boldsymbol{A})\boldsymbol{x}_0 + \mathcal{T}(\boldsymbol{A},\boldsymbol{B})\boldsymbol{u}_{[0,N-1]}, \tag{4b}$$

$$\boldsymbol{\Sigma}_{y,[1,N]} = \mathcal{T}(\boldsymbol{A},\boldsymbol{E}_x)(\boldsymbol{I}_N \otimes \boldsymbol{\Sigma}_x)\mathcal{T}(\boldsymbol{A},\boldsymbol{E}_x)^\top + \boldsymbol{I}_N \otimes (\boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top). \tag{4c}$$

The covariance in (4c) is obtained by considering the distributions $\boldsymbol{e}_{x,[0,N-1]} \sim \mathcal{N}(0, \boldsymbol{I}_N \otimes \boldsymbol{\Sigma}_x)$ and $\boldsymbol{e}_{y,[1,N]} \sim \mathcal{N}(0, \boldsymbol{I}_N \otimes \boldsymbol{\Sigma}_y)$, the properties of a linear transformation of a normally distributed variable [18, 20.23 b], and the mixed product rule for the Kronecker product [18, 11.11 a].

B. INCORPORATING THE STATE ESTIMATION

The predictive distribution in (4) is conditional on the initial state $\boldsymbol{x}_0$. However, due to the presence of measurement noise, the initial state $\boldsymbol{x}_0$ is unknown and must be estimated. The optimal state estimator for system (1) and considering Assumption 2 is the *Kalman filter* [1]. The Kalman filter yields a distribution for the estimated initial state which should also be considered for the distributed multi-step prediction.

In the following lemma, we incorporate the estimated distribution of the initial state to obtain the distribution $p(\hat{\boldsymbol{y}}_{[1,N]}|\boldsymbol{y}_0, \boldsymbol{u}_{[0,N-1]})$. In contrast to (4), this distribution is conditional on the noise affected measurement $\boldsymbol{y}_0$ instead of the unknown $\boldsymbol{x}_0$. Obtaining this distribution allows to directly state the stochastic optimal control problem for the measured initial state in the following subsection. Furthermore, we will revisit this distribution for the multi-step identification in Section III.

For the state estimation, we require a prior distribution of $\boldsymbol{x}_0$. Without loss of generality and for ease of notation, we assume that this prior has a zero mean.

Assumption 3: The states of the dynamic system are distributed according to $\boldsymbol{x}_0 \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{x,0})$.

This allows to state the following lemma.

Lemma 1: Assumption 1-3 hold. The distribution of the multi-step prediction, conditional on $\boldsymbol{y}_0$ *and* $\boldsymbol{u}_{[0,N-1]}$*, can be described with*:

$$p(\hat{\boldsymbol{y}}_{[1,N]}|\boldsymbol{y}_0, \boldsymbol{u}_{[0,N-1]}) = \mathcal{N}(\hat{\bar{\boldsymbol{y}}}_{[1,N]}, \hat{\boldsymbol{\Sigma}}_{y,[1,N]}). \tag{5}$$

The mean and covariance in (5) are:

$$\hat{\bar{\boldsymbol{y}}}_{[1,N]} = \mathcal{O}(\boldsymbol{A})\boldsymbol{L}\boldsymbol{y}_0 + \mathcal{T}(\boldsymbol{A},\boldsymbol{B})\boldsymbol{u}_{[0,N-1]}, \tag{6a}$$

$$\hat{\boldsymbol{\Sigma}}_{y,[1,N]} = \boldsymbol{\Sigma}_{y,[1,N]} + \mathcal{O}(\boldsymbol{A})(\boldsymbol{I}-\boldsymbol{L})\boldsymbol{\Sigma}_{x,0}\mathcal{O}(\boldsymbol{A})^\top, \tag{6b}$$

where $\boldsymbol{\Sigma}_{y,[1,N]}$ *is defined in (4c), and the Kalman gain can be computed as:*

$$\boldsymbol{L} = \boldsymbol{\Sigma}_{x,0}(\boldsymbol{\Sigma}_{x,0} + \boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top)^{-1}. \tag{6c}$$

Proof: With observed measurements $\boldsymbol{y}_0$*, and prior distribution of the states* $\boldsymbol{x}_0$ *from Assumption 3, we can perform a Kalman filter correction step [1]:*

$$\begin{aligned} \boldsymbol{L} &= \boldsymbol{\Sigma}_{x,0}\boldsymbol{C}^\top(\boldsymbol{C}\boldsymbol{\Sigma}_{x,0} + \boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top)^{-1}, \\ \bar{\boldsymbol{x}}_0^+ &= \bar{\boldsymbol{x}}_0 + \boldsymbol{L}(\boldsymbol{y}_0 - \boldsymbol{C}\bar{\boldsymbol{x}}_0), \\ \boldsymbol{\Sigma}_{x,0}^+ &= (\boldsymbol{I}-\boldsymbol{L}\boldsymbol{C})\boldsymbol{\Sigma}_{x,0}, \end{aligned}$$

with prior mean $\bar{\boldsymbol{x}}_0 = 0$*, posterior mean* $\bar{\boldsymbol{x}}_0^+$ *and posterior covariance* $\boldsymbol{\Sigma}_{x,0}^+$*. With* $\boldsymbol{C} = \boldsymbol{I}$*, due to Assumption 1, and* $\bar{\boldsymbol{x}}_0 = 0$*, due to Assumption 3, we obtain:*

$$\begin{aligned} \boldsymbol{L} &= \boldsymbol{\Sigma}_{x,0}(\boldsymbol{\Sigma}_{x,0} + \boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top)^{-1}, \\ \bar{\boldsymbol{x}}_0^+ &= \boldsymbol{L}\boldsymbol{y}_0, \\ \boldsymbol{\Sigma}_{x,0}^+ &= (\boldsymbol{I}-\boldsymbol{L})\boldsymbol{\Sigma}_{x,0}, \end{aligned}$$

With mean $\bar{\boldsymbol{x}}_0^+$ *and covariance* $\boldsymbol{\Sigma}_{x,0}^+$*, we have the posterior distribution:*

$$\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{L}\boldsymbol{y}_0,\ (\boldsymbol{I}-\boldsymbol{L})\boldsymbol{\Sigma}_{x,0}). \tag{7}$$

Substituting the distribution (7) in (2) yields (6). □

We omit the prediction step of the Kalman filter in Lemma 1 for reasons that will be clarified in Subsection IV-A. For the discussion in this section, the prediction step is implicitly considered through Assumption 3, that is, we assume to have the true prior distribution of the states.

C. STOCHASTIC OPTIMAL CONTROL PROBLEM

With the distributed multi-step system response (5), we can now state the stochastic optimal control problem. As is commonly the case for data-based MPC [9], we formulate cost and constraints for the measured outputs of the system. Furthermore, we directly incorporate the state estimation in the formulation and consider the uncertainty of the distributed initial state as shown in the previous subsection.

With expectation $\mathbb{E}(\cdot)$, and probability $P(\cdot)$, we have the stochastic optimal control problem:

$$\begin{aligned} \min_{\boldsymbol{u}_{[0,N-1]} \in \mathbb{A}} \quad & \mathbb{E}\left[\|\hat{\boldsymbol{y}}_{[1,N]}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}_{[0,N-1]}\|_{\boldsymbol{R}}^2\right] \\ \text{s.t.} \quad & P\left[\boldsymbol{a}_j^\top \hat{\boldsymbol{y}}_{[1,N]} \leq b_j\right] \geq (1-\epsilon) \quad \forall j \in \mathbb{I}_{[1,n_c]}, \end{aligned} \tag{8}$$

with $\hat{\boldsymbol{y}}_{[1,N]} \sim \mathcal{N}(\hat{\bar{\boldsymbol{y}}}_{[1,N]}, \hat{\boldsymbol{\Sigma}}_{y,[1,N]})$ obtained from (5), positive definite weighting matrices $\boldsymbol{Q} \succ 0$, $\boldsymbol{R} \succ 0$ and convex constraint set $\mathbb{A}$. We denote the norm $\|\boldsymbol{y}\|_{\boldsymbol{Q}}^2 = \boldsymbol{y}^\top\boldsymbol{Q}\boldsymbol{y}$. The n_c *chance constraints*, can be violated with probability $\epsilon \in]0,1[$, and are introduced as individual halfspaces [19, 2.2.1] with $\boldsymbol{a}_j \in \mathbb{R}^{n_y}$ and $b_j \in \mathbb{R}$ for $j \in \mathbb{I}_{[1,n_c]}$.

In the described setting, we can reformulate (8) as a deterministic problem, which yields the same optimal solution [15]:

$$\min_{\boldsymbol{u}_{[0,N-1]}\in\mathbb{A}} \|\hat{\bar{\boldsymbol{y}}}_{[1,N]}\|^2_{\boldsymbol{Q}} + \|\boldsymbol{u}_{[0,N-1]}\|^2_{\boldsymbol{R}} + \operatorname{trace}\left(\boldsymbol{Q}\hat{\boldsymbol{\Sigma}}_{y,[1,N]}\right)$$
$$\text{s.t. } \boldsymbol{a}_j^\top \hat{\bar{\boldsymbol{y}}}_{[1,N]} \leq b_j - c_p(\epsilon)\|\boldsymbol{a}_j\|_{\hat{\boldsymbol{\Sigma}}_{y,[1,N]}} \quad \forall j \in \mathbb{I}_{[1,n_c]}, \quad (9)$$

with $\hat{\bar{\boldsymbol{y}}}_{[1,N]}$ and $\hat{\boldsymbol{\Sigma}}_{y,[1,N]}$ from (6). The factor $c_p(\epsilon) = \sqrt{2}\text{erf}^{-1}(1-2\epsilon)$ is the quantile of the standard normal distribution and erf^{-1} denotes the inverse error-function [20].

Problem (9) is a quadratic optimization problem depending on the measured initial state $\boldsymbol{y}_0$ and implicitly considers a Kalman filter for the state estimation.

III. PROBABILISTIC MULTI-STEP IDENTIFICATION

For the formulation of the stochastic optimal control problem (9), we require the multi-step distribution in (5). Commonly, the parameters of this distribution are obtained from the system matrices in (1) and the covariances of process and measurement noise from Assumption 2. However, In many practical applications this information is unavailable.

In this section, we propose to use maximum likelihood estimation to identify directly the parameters of the multi-step distribution in (5) from system data. In contrast to previous work [8], we do not assume knowledge of the process and measurement noise covariances. As a main contribution, we present Theorem 1. The theorem establishes that, with the expected values for the estimated parameters and covariance matrix, we obtain the distributed multi-step prediction in (5). The identified multi-step model thus implicitly estimates the initial state distribution from noisy measurements.

A. PRELIMINARIES

To simplify the successive notation, we formulate the multi-step prediction model (2) as:

$$\boldsymbol{t} = \boldsymbol{W}^\top \boldsymbol{z} + \boldsymbol{e}_t, \quad (10)$$

with independent variable $\boldsymbol{z} = [\boldsymbol{x}_0^\top, \boldsymbol{u}_{[0,N-1]}^\top]^\top \in \mathbb{R}^{n_z}$, and response variable $\boldsymbol{t} = \boldsymbol{y}_{[1,N]} \in \mathbb{R}^{n_t}$. We have $n_z = n_x + Nn_u$ independent variables and $n_t = Nn_y$ response variables. The transposed parameters are

$$\boldsymbol{W}^T = [\mathcal{O}(\boldsymbol{A}),\ \mathcal{T}(\boldsymbol{A},\boldsymbol{B})], \quad (11)$$

and we have additive noise $\boldsymbol{e}_t \sim \mathcal{N}(0, \boldsymbol{\Sigma}_t)$, with $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_{y,[1,N]}$. It follows directly for the distribution of the response variables:

$$p(\boldsymbol{t}|\boldsymbol{z}, \boldsymbol{W}, \boldsymbol{\Sigma}_t) = \mathcal{N}(\boldsymbol{W}^\top \boldsymbol{z},\ \boldsymbol{\Sigma}_t). \quad (12)$$

Due to measurement noise, the true value of $\boldsymbol{z}$ is unknown and we introduce

$$\boldsymbol{v} = \boldsymbol{z} + \boldsymbol{e}_v, \quad (13)$$

with $\boldsymbol{e}_v \sim \mathcal{N}(0, \boldsymbol{\Sigma}_v)$ and $\boldsymbol{\Sigma}_v = \text{diag}(\boldsymbol{E}_y \boldsymbol{\Sigma}_y \boldsymbol{E}_y^\top, \boldsymbol{0})$. We denote with $\text{diag}(\boldsymbol{A}, \boldsymbol{B})$ the block-diagonal matrix with $\boldsymbol{A}$ and $\boldsymbol{B}$ on the diagonal.

B. PARAMETER AND COVARIANCE ESTIMATION

We seek to identify the multi-step system response from data. To this end, we gather m sequences of the dynamic system and introduce:

$$\boldsymbol{v}_i = \left[\boldsymbol{y}_0^{(i)\top},\ \boldsymbol{u}_{[0,N-1]}^{(i)\top}\right]^\top,\ \boldsymbol{t}_i = \boldsymbol{y}_{[1,N]}^{(i)} \quad \forall i \in \mathbb{I}_{[1,m]}, \quad (14)$$

We have the set $\mathcal{D} = \{\boldsymbol{V}, \boldsymbol{T}\}$ with *design matrix* [21] $\boldsymbol{V} = [\boldsymbol{v}_i, \ldots, \boldsymbol{v}_m]^\top$ and $\boldsymbol{T} = [\boldsymbol{t}_i, \ldots, \boldsymbol{t}_m]^\top$. The linear model (10) is compactly evaluated for all samples with:

$$\boldsymbol{T} = \boldsymbol{V}\boldsymbol{W} + \boldsymbol{E}_t, \quad (15)$$

where the vectorized residual matrix $\boldsymbol{E}_t$ has the distribution $\text{vec}(\boldsymbol{E}_t) \sim \mathcal{N}(0, \boldsymbol{\Sigma}_t \otimes \boldsymbol{I}_m)$. We require the following assumption for the design matrix.

Assumption 4: We have $m > n_z$ samples and the design matrix $\boldsymbol{V} \in \mathbb{R}^{m\times n_z}$ has full rank, that is, rank$(\boldsymbol{V}) = n_z$.

In practice, Assumption 4 also implies that the gathered input sequences must be persistently exciting [10].

This allows to state the following theorem for the identification of the probabilistic multi-step model.

Theorem 1: Assumption 1 and 2 hold. We have data samples from (14) that satisfy Assumption 3 and 4. The bias corrected maximum likelihood approach yields the estimated parameters and covariance matrix:

$$\hat{\boldsymbol{W}}^* = \boldsymbol{\Sigma}_p^* \boldsymbol{V}^\top \boldsymbol{T}, \quad (16a)$$
$$\hat{\boldsymbol{\Sigma}}_t^* = \frac{1}{m-n_z}(\boldsymbol{V}\hat{\boldsymbol{W}}^* - \boldsymbol{T})^\top(\boldsymbol{V}\hat{\boldsymbol{W}}^* - \boldsymbol{T}), \quad (16b)$$

with:

$$\hat{\boldsymbol{\Sigma}}_p^* = (\boldsymbol{V}^\top \boldsymbol{V})^{-1}. \quad (16c)$$

We partition the estimated parameters as

$$\hat{\boldsymbol{W}}^{*\top} = [\hat{\mathcal{O}}_{\boldsymbol{A}}^*,\ \hat{\mathcal{T}}_{\boldsymbol{A},\boldsymbol{B}}^*],$$

and have $\hat{\boldsymbol{\Sigma}}_t^* = \hat{\boldsymbol{\Sigma}}_{y,[1,N]}^*$. The estimated parameters and covariance matrix have the property:

$$\mathbb{E}[\hat{\mathcal{O}}_{\boldsymbol{A}}^*] = \mathcal{O}(\boldsymbol{A})\boldsymbol{L}, \quad (17a)$$
$$\mathbb{E}[\hat{\mathcal{T}}_{\boldsymbol{A},\boldsymbol{B}}^*] = \mathcal{T}(\boldsymbol{A},\boldsymbol{B}), \quad (17b)$$
$$\mathbb{E}[\hat{\boldsymbol{\Sigma}}_{y,[1,N]}^*] = \boldsymbol{\Sigma}_{y,[1,N]} + \mathcal{O}(\boldsymbol{A})(\boldsymbol{I}-\boldsymbol{L})\boldsymbol{\Sigma}_{x,0}\mathcal{O}(\boldsymbol{A})^\top. \quad (17c)$$

In expectation, the estimated parameters and covariance matrix are thus identical to the true parameters and covariance matrix in (5).

Proof: We consider the multi-step model in the form of (12) and have noise disturbed independent variables from (13). Only as an intermediate step of the derivation, we introduce the assumption that $\boldsymbol{u}_{[0,N-1]} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{u,0})$. This allows to state the distribution $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_z)$ with

$\boldsymbol{\Sigma}_z = diag(\boldsymbol{\Sigma}_{x,0}, \boldsymbol{\Sigma}_{u,0})$. *We state the joint distribution of* $\boldsymbol{t}$ *and* $\boldsymbol{v}$ *as shown in [22]:*

$$p\left([\mathbf{t}, \mathbf{v}]^\top | \boldsymbol{W}, \boldsymbol{\Sigma}_t\right) = \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\boldsymbol{W}^\top\boldsymbol{\Sigma}_z\boldsymbol{W} + \boldsymbol{\Sigma}_t & \boldsymbol{W}^\top\boldsymbol{\Sigma}_z \\ \boldsymbol{\Sigma}_z\boldsymbol{W} & \boldsymbol{\Sigma}_z + \boldsymbol{\Sigma}_v\end{bmatrix}\right), \tag{18}$$

and the resulting conditional probability follows directly from the properties of a Gaussian normal distribution:

$$p(\mathbf{t}|\mathbf{v}, \boldsymbol{W}, \boldsymbol{\Sigma}_t) = \mathcal{N}(\boldsymbol{W}^\top\boldsymbol{K}\boldsymbol{v},\ \boldsymbol{\Sigma}_t + \boldsymbol{W}^\top(\boldsymbol{\Sigma}_z - \boldsymbol{K}\boldsymbol{\Sigma}_z)\boldsymbol{W}), \tag{19}$$

with reliability matrix [23]:

$$\boldsymbol{K} = \boldsymbol{\Sigma}_z(\boldsymbol{\Sigma}_z + \boldsymbol{\Sigma}_v)^{-1}. \tag{20}$$

Furthermore, we introduce $\hat{\boldsymbol{W}}^\top = \boldsymbol{W}^\top\boldsymbol{K}$ *and* $\hat{\boldsymbol{\Sigma}}_t = \boldsymbol{\Sigma}_t + \boldsymbol{W}^\top(\boldsymbol{\Sigma}_z - \boldsymbol{K}\boldsymbol{\Sigma}_z)\boldsymbol{W}$ *and state (19) as*:

$$p(\mathbf{t}|\mathbf{v}, \hat{\boldsymbol{W}}, \hat{\boldsymbol{\Sigma}}_t) = \mathcal{N}(\hat{\boldsymbol{W}}^\top\mathbf{v},\ \hat{\boldsymbol{\Sigma}}_t). \tag{21}$$

Of the independent variables, only $\mathbf{y}_0$ *is affected by measurement error. We therefore introduce* $\boldsymbol{\Sigma}_v = diag(\boldsymbol{E}\boldsymbol{\Sigma}_y\boldsymbol{E}^\top, \mathbf{0})$, *and can further simplify (20):*

$$\begin{aligned}\boldsymbol{K} &= \begin{bmatrix}\boldsymbol{\Sigma}_{x,0} & 0\\ 0 & \boldsymbol{\Sigma}_{u,0}\end{bmatrix}\left(\begin{bmatrix}\boldsymbol{\Sigma}_{x,0} + \boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top & 0\\ 0 & \boldsymbol{\Sigma}_{u,0}\end{bmatrix}\right)^{-1}\\ &= \begin{bmatrix}\boldsymbol{\Sigma}_{x,0}(\boldsymbol{\Sigma}_{x,0} + \boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top)^{-1} & 0\\ 0 & \boldsymbol{I}\end{bmatrix} \overset{(6c)}{=} \begin{bmatrix}\boldsymbol{L} & 0\\ 0 & \boldsymbol{I}\end{bmatrix}.\end{aligned}$$

The previously introduced $\boldsymbol{\Sigma}_{u,0}$ *thus vanishes from the expression. We substitute* $\boldsymbol{K} = diag(\boldsymbol{L}, \boldsymbol{I})$ *in (19) and consider the definition of the parameters* $\boldsymbol{W}$ *from (11). We obtain*:

$$\hat{\boldsymbol{W}}^\top = [\mathcal{O}(\boldsymbol{A})\boldsymbol{L},\ \mathcal{T}(\boldsymbol{A}, \boldsymbol{B})], \tag{22a}$$

$$\hat{\boldsymbol{\Sigma}}_t = \boldsymbol{\Sigma}_{y,[1,N]} + \mathcal{O}(\boldsymbol{A})(\boldsymbol{I} - \boldsymbol{L})\boldsymbol{\Sigma}_{x,0}\mathcal{O}(\boldsymbol{A})^\top. \tag{22b}$$

With the distribution of the response variables in (21), we have the likelihood:

$$p(\boldsymbol{t} = \boldsymbol{T}|\boldsymbol{V}, \hat{\boldsymbol{W}}, \hat{\boldsymbol{\Sigma}}_t) = p(\mathcal{D}|\hat{\boldsymbol{W}}, \hat{\boldsymbol{\Sigma}}_t). \tag{23}$$

To compute the likelihood, we vectorize (15) and consider the properties of the Kronecker product [18, 11.16]:

$$p(\mathcal{D}|\hat{\boldsymbol{W}}, \hat{\boldsymbol{\Sigma}}_t) = \mathcal{N}((\boldsymbol{I} \otimes \boldsymbol{V})vec(\hat{\boldsymbol{W}}),\ \hat{\boldsymbol{\Sigma}}_t \otimes \boldsymbol{I}_m). \tag{24}$$

We then maximize the likelihood:

$$\hat{\boldsymbol{W}}^*,\ \hat{\boldsymbol{\Sigma}}_t^* = \arg\max_{\hat{\boldsymbol{W}}, \hat{\boldsymbol{\Sigma}}_t} p(\mathcal{D}|\hat{\boldsymbol{W}}, \hat{\boldsymbol{\Sigma}}_t). \tag{25}$$

Problem (25) has the explicit solution shown in (16) with bias corrected covariance matrix [21], [23]. The inverse in (16c) always exists and we have $m > n_z$ *in (16b) due to Assumption 4. The estimated parameters and covariance have the property that [21], [23]:*

$$\mathbb{E}(\hat{\boldsymbol{W}}^*) = \hat{\boldsymbol{W}},$$
$$\mathbb{E}(\hat{\boldsymbol{\Sigma}}_t^*) = \hat{\boldsymbol{\Sigma}}_t.$$

Considering the definition of $\hat{\boldsymbol{W}}$ *and* $\hat{\boldsymbol{\Sigma}}_t$ *in (22), we obtain (17).* □

As a main consequence of Theorem 1, we have that the estimated parameters and covariance in (16) are, in expectation, identical to the parameters and covariance of the distribution (5). Therefore, we directly obtain the distributed multi-step predictions that incorporates the state estimation.

Finally, we want to remark that the computational complexity of the estimation boils down to the operations in (16) which are tractable even for larger systems.

C. PARAMETRIC UNCERTAINTY

Theorem 1 allows to efficiently estimate the unknown parameters and the full covariance matrix of the probabilistic multi-step model in (5). While we have the favorable property that the estimated parameters are unbiased, we can, in practice, only consider a finite number of samples for the identification task. Consequently, we must consider the parametric uncertainty of the estimated parameters for which we introduce the following theorem.

Theorem 2: Assumptions 1-4 hold, and we have a non-informative prior distribution for the parameters $\hat{\boldsymbol{W}}$. *We have m samples of data from (14) and maximum likelihood estimates of the parameter and covariance matrix from (16). Considering the posterior distribution of the parameters, i.e. the parametric uncertainty, we have the distributed multi-step prediction*:

$$p(\hat{\mathbf{y}}^*_{[1,N]}|\mathcal{D}, \mathbf{y}_0, \boldsymbol{u}_{[0,N-1]}) = \mathcal{N}(\hat{\bar{\mathbf{y}}}^*_{[1,N]},\ \alpha(\boldsymbol{v})\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}), \tag{26a}$$

with:

$$\hat{\bar{\mathbf{y}}}^*_{[1,N]} = \hat{\mathcal{O}}^*_{\boldsymbol{A}}\mathbf{y}_0 + \hat{\mathcal{T}}^*_{\boldsymbol{A},\boldsymbol{B}}\boldsymbol{u}_{[0,N-1]}, \tag{26b}$$

$$\alpha(\boldsymbol{v}) = (1 + \boldsymbol{v}^\top\hat{\boldsymbol{\Sigma}}^*_p\boldsymbol{v}), \tag{26c}$$

where $\boldsymbol{v}^\top = [\mathbf{y}_0^\top, \boldsymbol{u}^\top_{[0,N-1]}]$.

Proof: We consider the multi-step model in the form of (12) and have noise disturbed independent variables from (13). We have estimated parameters $\hat{\boldsymbol{W}}^*$ and $\hat{\boldsymbol{\Sigma}}^*_t$ from Theorem 1. The posterior of the identified parameters is

$$p(\hat{\boldsymbol{W}}^*|\mathcal{D}, \hat{\boldsymbol{\Sigma}}^*_t) = \frac{p(\mathcal{D}|\hat{\boldsymbol{W}}^*, \hat{\boldsymbol{\Sigma}}^*_t)p(\hat{\boldsymbol{W}}^*)}{p(\mathcal{D}|\hat{\boldsymbol{\Sigma}}^*_t)}. \tag{27}$$

For the likelihood, we again consider the vectorized formulation in (24). With the non-informative prior, i.e. $p(\hat{\boldsymbol{W}}) \sim 1$, we obtain the posterior distribution for the vectorized parameters $\text{vec}(\hat{\boldsymbol{W}}^*)$:

$$p(\text{vec}(\hat{\boldsymbol{W}})|\mathcal{D}, \hat{\boldsymbol{\Sigma}}^*_t) = \mathcal{N}(\text{vec}(\hat{\boldsymbol{W}}), \hat{\boldsymbol{\Sigma}}^*_t \otimes \hat{\boldsymbol{\Sigma}}^*_p). \tag{28}$$

We then vectorize the linear model (10) for a single test point, yielding:

$$\begin{aligned}\boldsymbol{t}^\top &= \boldsymbol{v}^\top\hat{\boldsymbol{W}} + \boldsymbol{e}_t^\top\\ \overset{\text{vec}(\cdot)}{\Leftrightarrow}\ \boldsymbol{t} &= \text{vec}\left(\boldsymbol{v}^\top\hat{\boldsymbol{W}}\right) + \boldsymbol{e}_t\\ &= \left(\boldsymbol{I} \otimes \boldsymbol{v}^\top\right)\text{vec}\left(\hat{\boldsymbol{W}}\right) + \boldsymbol{e}_t.\end{aligned}$$

Considering the distribution of the vectorized parameters in (28), we obtain a multivariate normal distribution for the response variables with mean $\mathbb{E}[\boldsymbol{t}] = \hat{\boldsymbol{W}}^\top \boldsymbol{v}$ and covariance

$$\begin{aligned}\mathrm{Cov}[\boldsymbol{t}] &= (\boldsymbol{I} \otimes \boldsymbol{v}^\top)(\hat{\boldsymbol{\Sigma}}_t^* \otimes \hat{\boldsymbol{\Sigma}}_p^*)(\boldsymbol{I} \otimes \boldsymbol{v}^\top)^\top + \hat{\boldsymbol{\Sigma}}_t^* \\ &= \hat{\boldsymbol{\Sigma}}_t^* \otimes \boldsymbol{v}^\top \hat{\boldsymbol{\Sigma}}_p^* \boldsymbol{v} + \hat{\boldsymbol{\Sigma}}_t^*.\end{aligned}$$

The first equality stems from the properties of a linear transformation of a normally distributed random variable [18, 20.23 b], and the second equality follows from the mixed product property of the Kronecker product [18, 11.11]. For a single test point $\boldsymbol{v}$, the term $\boldsymbol{v}^\top \hat{\boldsymbol{\Sigma}}_p^* \boldsymbol{v}$ is a scalar and the Kronecker product simplifies to a scalar multiplication:

$$\mathrm{Cov}[\boldsymbol{t}] = (\boldsymbol{v}^\top \hat{\boldsymbol{\Sigma}}_p^* \boldsymbol{v})\boldsymbol{\Sigma}_t^* + \hat{\boldsymbol{\Sigma}}_t^* = (\boldsymbol{v}^\top \hat{\boldsymbol{\Sigma}}_p^* \boldsymbol{v} + 1)\hat{\boldsymbol{\Sigma}}_t^* = \alpha(\boldsymbol{v})\hat{\boldsymbol{\Sigma}}_t^*.$$

With mean and covariance, we thus have the distribution:

$$p(\boldsymbol{t}|\mathcal{D}, \boldsymbol{v}) = \mathcal{N}(\hat{\boldsymbol{W}}^{*\top} \boldsymbol{v}, \alpha(\boldsymbol{v})\hat{\boldsymbol{\Sigma}}_t^*). \tag{29}$$

From the definition of $\boldsymbol{v}, \boldsymbol{t}$ and considering (16), we obtain the distributed multi-step prediction in (26). □

The distributed multi-step prediction including the parametric uncertainty is thus given in (26). We observe the following behavior as the number of samples m used for the identification increases. As shown in [21, 3.3.2], we have for $m \to \infty$ that $\alpha(\boldsymbol{v}) \to 1$. In the same limit, the identified parameters and covariance matrix converge to their expectation in (17) and we recover exactly the distribution (5). We demonstrate this behavior in the numerical example in Section VI.

IV. STOCHASTIC MPC WITH IDENTIFIED MULTI-STEP MODEL

With the distribution in (26) we have the data-based equivalent to (5) including the parametric uncertainty from the identification task. The deterministic formulation of the stochastic optimal control problem (9) with identified multi-step model is given as:

$$\begin{aligned}\min_{\boldsymbol{u}_{[0,N-1]} \in \mathbb{A}} \quad & \|\hat{\bar{\boldsymbol{y}}}_{[1,N]}^*\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u}_{[0,N-1]}\|_{\boldsymbol{R}}^2 \\ & + \mathrm{trace}\left(\alpha(\boldsymbol{v})\boldsymbol{Q}\hat{\boldsymbol{\Sigma}}_{y,[1,N]}^*\right) && \text{(30a)} \\ \text{s.t.} \quad & \boldsymbol{a}_j^\top \hat{\bar{\boldsymbol{y}}}_{[1,N]}^* \leq b_j - c_p(\epsilon)\|\boldsymbol{a}_j\|_{\alpha(\boldsymbol{v})\hat{\boldsymbol{\Sigma}}_{y,[1,N]}^*} \quad \forall j \in \mathbb{I}_{[1,n_c]}. && \text{(30b)}\end{aligned}$$

The weight matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, constraint set $\mathbb{A}$ and the chance constraints, defined with $\boldsymbol{a}_j$, b_j $\forall j \in \mathbb{I}_{[1,n_c]}$ and ϵ, are analogous to (9). As a main difference between (9) and (30), the mean $\hat{\bar{\boldsymbol{y}}}_{[1,N]}^*$ and covariance $\hat{\boldsymbol{\Sigma}}_{y,[1,N]}^*$ now stem from distribution (26). The parametric uncertainty of the identified model results in the factor $\alpha(\boldsymbol{v})$, introduced in (26c), where $\boldsymbol{v}^\top = [\boldsymbol{y}_0^\top, \boldsymbol{u}_{[0,N-1]}^\top]$.

Due to the parametric uncertainty, and in contrast to (9), the optimization problem in (30) is not a quadratic problem anymore. We show in Appendix A that problem (30) constitutes a convex second-order cone program [19]. In contrast to previous work [8], where the parametric uncertainty is included as an ellipsoidal uncertainty set, we propose to solve problem (30) directly.

A. RECURSIVE APPLICATION

The stochastic optimal control problem in (30) is solved repeatedly at each sampling time to yield closed-loop control actions. This introduces the well known challenges to guarantee recursive feasibility and stability [14]. Additionally, we have to consider a limitation of the identified multi-step model. As shown in Theorem 1, the multi-step model identifies the parameters and covariance matrix for the distribution in (5). This distribution incorporates a Kalman filter correction step, considering the prior distribution of the initial state $p_{\text{prior}}(\boldsymbol{x}_0)$ and the current measurement. However, this prior for the initial state distribution is not explicitly formulated. Instead, it coincides with the distribution of the initial state from the sampled data $p_{\text{samp.}}(\boldsymbol{x}_0)$, as required for Theorem 1. In other words, the multi-step model always considers the same prior distribution, that is, $p_{\text{prior}}(\boldsymbol{x}_0) = p_{\text{samp.}}(\boldsymbol{x}_0)$.

This has two important limitations. First, it is imperative that the multi-step model is evaluated only for initial states $\boldsymbol{x}_{0,\text{eval}}$ that are reasonably probable in terms of the distribution of the initial state from the sampled data. This is an intuitive limitation, as it is detrimental to perform a Kalman filter correction step with poorly selected prior state distribution. It is, however, a limitation in the sense that data for the identification task must be sampled in the same range that is expected for the closed-loop operation.

The second limitation arises from the fact that with identified multi-step model the prior state distribution is not updated. Updating the prior distribution is typically an important step of Kalman filtering and allows for convergence to the true state distribution over multiple iterations. In practice, this limitation increases the uncertainty of the predicted future measurements and may lead to more conservative control actions of the SMPC controller.

V. STATE-SPACE IDENTIFICATION

To have a comparative baseline for our proposed method, we also employ a probabilistic state-space identification approach. We consider the same data for the identification task which stems from system (1) with unknown parameters and process and measurement noise covariances. In particular, we have the data with individual samples:

$$\boldsymbol{v}_i = \left[\boldsymbol{y}_0^{(i)\top}, \boldsymbol{u}_0^{(i)\top}\right]^\top, \quad \boldsymbol{t}_i = \boldsymbol{y}_1^{(i)} \quad \forall i \in \mathbb{I}_{[1,m]}. \tag{31}$$

A. CHALLENGES WITH STATE-SPACE IDENTIFICATION

We can identify a state-space model as a special case of the described multi-step identification approach with $N = 1$. In that case, it follows from (3) that $\mathcal{O}(\boldsymbol{A}) = \boldsymbol{A}$ and $\mathcal{T}(\boldsymbol{A}, \boldsymbol{B}) = \boldsymbol{B}$. Unfortunately, state-space identification poses two major challenges which both stem from the requirement that the

identified model must be recursively evaluated to obtain the multi-step prediction.

The first challenge is related to the measurement noise of the system. From Theorem 1, we obtain parameters $\hat{\boldsymbol{A}}^*$, $\hat{\boldsymbol{B}}^*$ and $\hat{\boldsymbol{\Sigma}}^*_{y,1}$ with the properties:

$$\begin{aligned}\mathbb{E}[\hat{\boldsymbol{A}}^*] &= \boldsymbol{A}\boldsymbol{L},\\ \mathbb{E}[\hat{\boldsymbol{B}}^*] &= \boldsymbol{B},\\ \mathbb{E}[\hat{\boldsymbol{\Sigma}}^*_{y,1}] &= \boldsymbol{\Sigma}_{y,1} + \boldsymbol{A}(\boldsymbol{\Sigma}_{x,0} + \boldsymbol{L}\boldsymbol{\Sigma}_{x,0})\boldsymbol{A}^\top,\end{aligned}$$

where

$$\boldsymbol{\Sigma}_{y,1} = \boldsymbol{A}\boldsymbol{E}_x\boldsymbol{\Sigma}_x\boldsymbol{E}_x^\top\boldsymbol{A}^\top + \boldsymbol{E}_y\boldsymbol{\Sigma}_y\boldsymbol{E}_y^\top. \tag{32}$$

In expectation, the identified parameters are thus identical to the parameters in (6). With:

$$\hat{\hat{\boldsymbol{y}}}^*_1 = \hat{\boldsymbol{A}}^*\boldsymbol{y}_0 + \hat{\boldsymbol{B}}^*\boldsymbol{u}_0, \tag{33}$$

we thus have the conditional distribution (5) for $N = 1$:

$$p(\hat{\boldsymbol{y}}^*_1|\boldsymbol{y}_0, \boldsymbol{u}_0) = \mathcal{N}(\hat{\hat{\boldsymbol{y}}}^*_1, \hat{\boldsymbol{\Sigma}}^*_{y,1}). \tag{34}$$

We can recursively evaluate (34) to obtain a distributed multi-step prediction $\hat{\boldsymbol{y}}_{[1,N]}$. Unfortunately, this will not yield the correct multi-step prediction in (5) for $N > 1$. This is due to the fact that (32) contains contributions from both the process noise and the measurement noise. The recursive evaluation of (34) thus propagates the measurement noise which can significantly inflate the uncertainty of the multi-step prediction.

The second challenge with state-space identification is the parametric uncertainty discussed in Subsection III-C. This uncertainty can be considered for a single-step prediction with (26) and $N = 1$. However, there is no closed-form solution for the recursive application of (34) if parametric uncertainties are considered.

B. IDENTIFIED STATE-SPACE MODEL FOR SMPC

The distributed multi-step system response obtained from the recursively evaluated state-space model is expected to have significant shortcomings due to the challenges discussed above. Regardless, we suggest an approach to formulate the stochastic MPC problem with the identified state-space model. The approach serves as an important comparative baseline, as it is obtained under the same premises as for the multi-step identification.

We formulate the stochastic MPC problem as in (9), using the identified state-space model with parameters $\hat{\boldsymbol{A}}^*$, $\hat{\boldsymbol{B}}^*$ and covariance $\hat{\boldsymbol{\Sigma}}^*_{y,1}$. The distributed multi-step prediction is obtained by recursively evaluating (34) without considering the parametric uncertainty.

VI. LINEAR CASE STUDY

In this and the following section, we investigate the proposed method in two simulation studies. We first consider a linear system with noisy state-feedback and investigate a nonlinear system with noisy output-feedback in Section VII. For both simulation studies, the complete code and our results are available online.[1]

A. SYSTEM DESCRIPTION

We consider the linear building model previously presented in [24]. The system has five states $\boldsymbol{x} = [T_1, \ldots, T_4, T_a]^\top$, where T_i [°C] is the temperature in room i and T_a [°C] the ambient temperature. The room temperatures are controlled with combined heating and cooling units $\boldsymbol{u} = [\dot{Q}_1, \ldots, \dot{Q}_4]^\top$ [kW]. The dynamics are modeled as a resistance network [25]. We assume that an uncertain forecast T_f [°C] of the ambient temperature is available, and model this situation with $\dot{T}_a = \tau_a(T_f - T_a) + e_{T,a}$, and significant process noise. The model is parameterized as in [24], including $\tau_a = 1/72.000$ and discretized with timestep $\Delta t = 3600\,\text{s}$. Furthermore, we define the variances $\sigma_x^\top = [0, 0, 0, 0, 0.5]$ and $\sigma_y^\top = 10^{-1} \cdot [1, 1, 1, 1, 1]$ for the process and measurement noise. The covariance matrices are then obtained as $\boldsymbol{\Sigma}_x = \text{diag}(\sigma_x^2)$ and $\boldsymbol{\Sigma}_y = \text{diag}(\sigma_y^2)$.

B. SYSTEM IDENTIFICATION

In our first investigation, we seek to show that with the proposed multi-step identification, we obtain a data-based model that represents the distributed system response in (5). We want to highlight, in particular, that this distribution incorporates the effect of the state estimation as discussed in Subsection II-B.

For the investigation, we first determine an initial state distribution according to Assumption 3. With standard deviation $\sigma_{x,0}^\top = [2, 2, 2, 2, 5]$ and mean $\bar{\boldsymbol{x}}_0^\top = [20, 20, 20, 20, 15]$, we have:

$$\boldsymbol{x}_0 \sim \mathcal{N}(\bar{\boldsymbol{x}}_0, \text{diag}(\sigma_{x,0}^2)). \tag{35}$$

Two multi-step models, both with horizon $N = 12$, are identified. We denote MSM_a for the identified multi-step model using $m = 1000$ sequences, and MSM_b for the identified multi-step model with $m = 100$ sequences. All sequences start from a random initial state according to distribution (35) and we normalize the data to achieve a zero-mean as required for Assumption 3.

Additionally, we obtain the ground truth for the predictive distribution. To this end, we consider the true system and covariance matrices and evaluate distribution (5), which includes the Kalman filter correction step for the prior distribution in (35).

In Figure 1, we display the distributed multi-step prediction $p(\hat{\boldsymbol{y}}_{[1,N]})$ over the prediction horizon. We show the mean and the standard deviation for T_1, T_2 and T_a. The other states are omitted from the plot to improve readability. Furthermore, we show the joint distribution $p(T_i, T_j)$ at $t = 12\,\text{h}$. In comparison to the ground truth, MSM_a yields almost the identical predictive distribution. This is consistent with our theoretical results presented in Theorem 1. Due to the large number of samples the identified parameters are almost

[1] https://github.com/4flixt/2023_Stochastic_MSM

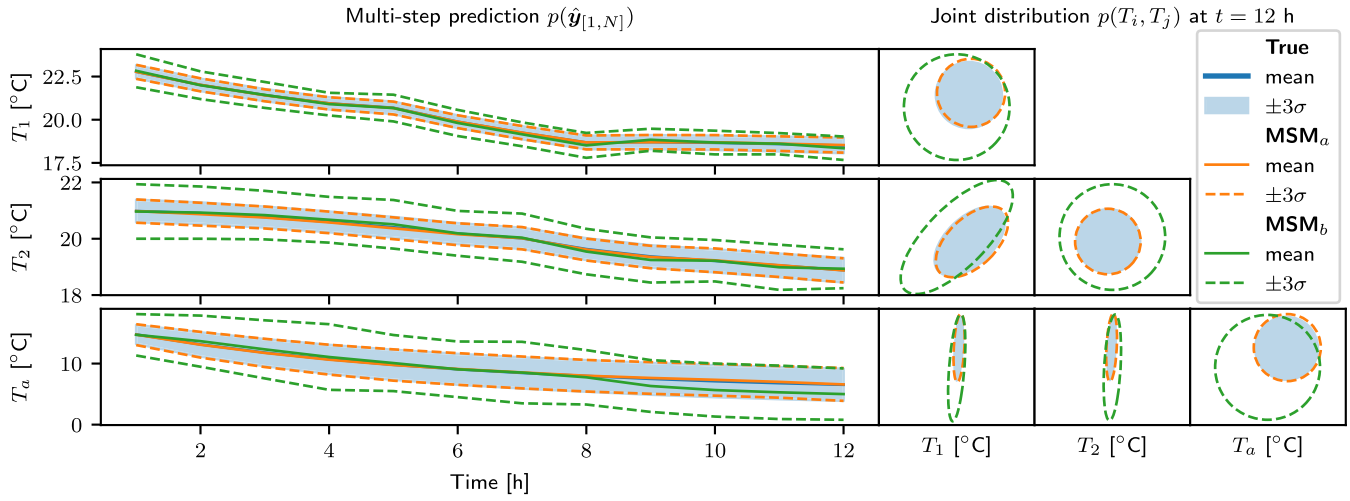

FIGURE 1. Distributed multi-step prediction of the building system for an initial measurement y_0 and random control sequence $u_{[0,N-1]}$ for $N = 12$. Comparison of true distribution (5) and two identified multi-step models (MSM). The distribution of MSM_a with $m = 1000$ samples and MSM_b with $m = 100$ samples are obtained according to (26).

identical to their expectation in (17), and we recover the true distribution in (5) from data. On the other hand, MSM_b is identified with only $m = 100$ samples and therefore experiences significant parametric uncertainty, according to Theorem 2. However, while the obtained distribution is broader and the mean is shifted, it still approximates the true distribution in (5).

C. STOCHASTIC MPC

We formulate a stochastic MPC controller based on the identified multi-step model (MSM-SMPC). As a comparative baseline, we also formulate a stochastic MPC controller based on an identified state-space model (SSM-SMPC). Both models are obtained from the same data as described in the previous subsection.

For the control objective, we choose the cost function:

$$J(\boldsymbol{u}_{[0,N-1]}) = \sum_{k=0}^{N-2} (5\|\boldsymbol{u}_k\|_2^2 + 10\|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|_2^2) + \|\boldsymbol{u}_{N-1}\|_2^2.$$

The actuators (heating and cooling power) are limited to $-6\,\text{kW} \leq \dot{Q}_i \leq 6\,\text{kW}$ for all rooms. We have the individual chance constraints $P(T_i \geq 18) \geq (1 - \epsilon)$, which can be violated with probability $\epsilon = 10^{-3}$. The stochastic optimal control problem is implemented and solved using CasADi [26] with IPOPT [27].

In a first investigation, we compare the open-loop predictions of both controllers. We define a scenario with initial state $\boldsymbol{x}_0 = [23,\ 20,\ 20,\ 20,\ 10]^\top\,°\text{C}$ and corresponding noise disturbed measurement $\boldsymbol{y}_0$. We then solve the optimal control problem in (30) for the multi-step model and (9) for the state-space model. As discussed in Section V, the SMPC controller based on the state-space model does not consider the parametric uncertainty.

TABLE 1. Building system: Closed-loop performance over a period of 50h with mean and standard deviation computed over 20 samples.

	MSM-SMPC	SSM-SMPC
$\sum_i Q_i$ [kWh]	640.9±32.4	698.8±29.9
cons. viol. [%]	0.0±0.0	0.0±0.0

The resulting open-loop predictions are shown in Figure 2. We display the predicted mean $\hat{\boldsymbol{y}}_{[1,N]}$ and standard deviation, considering $c_p(\epsilon)$ for the chosen violation probability. Furthermore, we show 50 samples of the true system response for the optimal open-loop input sequence, considering the same initial state and randomly drawn process and measurement noise. It can be seen in Figure 2 that the MSM-SMPC controller yields a suitable sequence of inputs to minimize the cost function while satisfying the chance constraints. While the chance constraints are also satisfied for the SSM-SMPC controller, the predicted uncertainty bounds are much larger and overly conservative. In the investigated scenario, with forecasted ambient temperature of $T_f = 10\,°\text{C}$, this leads to suboptimal controls action with unnecessarily high energy consumption. This effect is expected, as discussed in Section V, as the identified state-space model incorrectly propagates the measurement noise.

In a second investigation, we compare the closed-loop performance of the two controllers. To this end, we consider the same initial state as in the previous section and recursively control the system for a period of 50 h with either the MSM-SMPC or SSM-SMPC controller. The samples differ in the randomly drawn sequences of process and measurement noise. Finally, we obtain the percentage of constraint violations and the total energy consumption for each run and compute mean and standard deviation of these values. The performance metrics are shown in Table 1. The MSM-SMPC controller yields on average 8.3 % less energy consumption

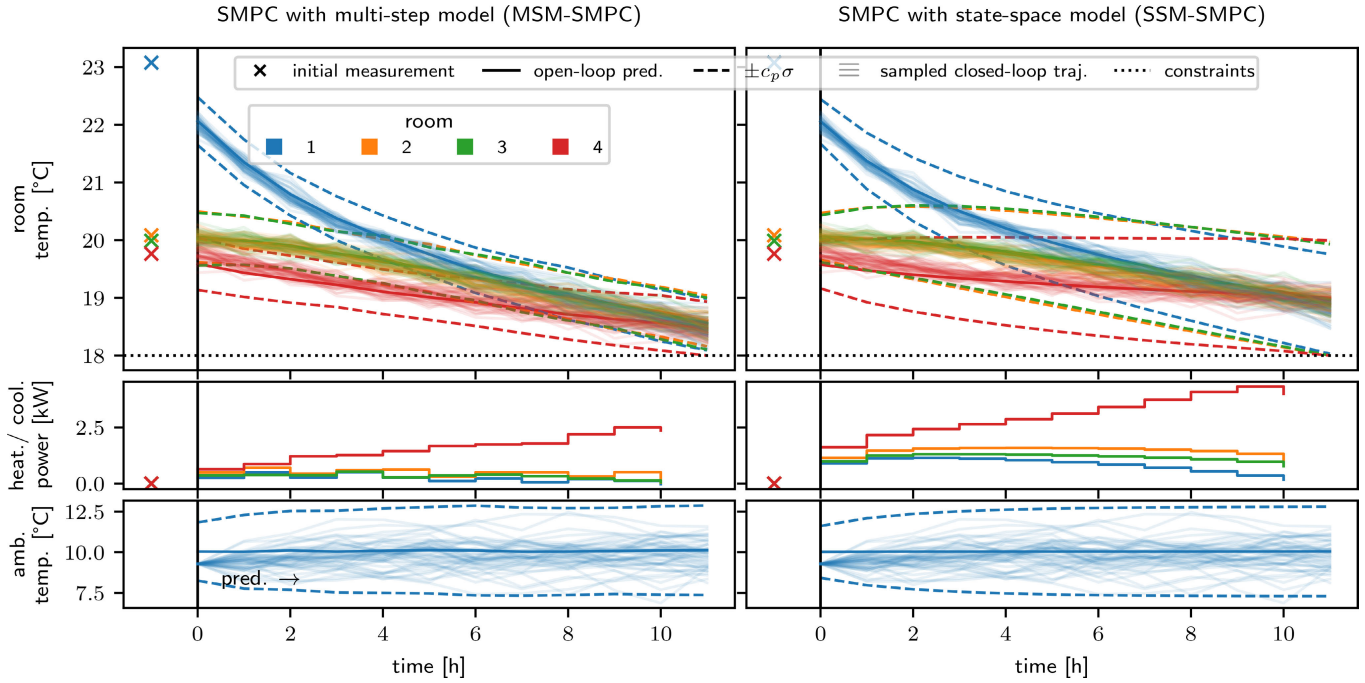

FIGURE 2. Building system: Comparison of open-loop prediction (mean and predicted standard deviation) of SMPC with multi-step model and SMPC with state-space model for the same initial measurement. The standard deviation is scaled with c_p introduced in (9). Using the open-loop control inputs, 50 samples of the true system response are drawn.

than the SSM-SMPC controller. Both controllers satisfy the chance constraints with no violations.

VII. NONLINEAR CASE STUDY WITH OUTPUT-FEEDBACK

The proposed multi-step identification approach is derived for linear systems with noisy state-feedback. Additionally, we explore its applicability to a nonlinear system with output-feedback in this section. The application to nonlinear systems is motivated by interpreting the process noise in (1a) as an additive nonlinear term. While this interpretation violates Assumption 2, that is, the process noise is not normally distributed, we expect to identify a multi-step model where the identified uncertainty approximately encompasses the nonlinearities.

As in the previous section, we compare the SMPC based on an identified multi-step model with a variant based on state-space identification.

A. SYSTEM DESCRIPTION

For the nonlinear case study, we consider the continuously stirred tank reactor (CSTR) previously introduced in [28]. The reactor is modeled with four states $\boldsymbol{x}^\top = [c_A, c_B, T_R, T_K]$, with the concentrations c_A [mol L^{-1}] and c_B [mol L^{-1}] and the temperatures T_R [°C] of the reactor and T_K [°C] of the cooling jacket. As control inputs, we have the flow rate $\dot{V}$ [m^3 s^{-1}] which is normalized with the reactor volume V_R [m^3], yielding $F = \dot{V}/V_R$, and the heat removed from the jacket $\dot{Q}$ [kJ h^{-1}]. For the safe operation of the CSTR, the states and inputs must lie within the bounds shown in Table 2. In our investigated scenario, the nonlinear

TABLE 2. Bounds for the CSTR system. [1] Chance constraint for SMPC.

	states				inputs	
	c_A mol L^{-1}	c_B mol L^{-1}	T_R °C	T_K °C	F h^{-1}	$\dot{Q}$ kJ h^{-1}
lower	0.1	0.1	50.0	50.0	5.0	-8500.0
upper	2.0	2.0	135.0[1]	140.0	60.0	0.0

CSTR system experiences measurement noise with standard deviation $\sigma_y^\top = [0.01, 0.01, 0.5, 0.5]$ but no additional process noise.

The nonlinear model is created and simulated in do-mpc [29] with a timestep of 18 s. The model equations and parameters can be found in [28] or online.[2]

B. SYSTEM IDENTIFICATION

As in the previous section, we compare the identified state-space model (SSM) and multi-step model (MSM). Furthermore, we also investigate the effect of having full state-feedback vs. output-feedback for system identification and the successive control application. As in related data-based approaches [5], [6], [9], [11], output-feedback can be incorporated in the proposed methods by introducing the state:

$$\boldsymbol{x}_k^\top = [\boldsymbol{y}_{[k-l,k]}^\top, \boldsymbol{u}_{[k-l,k-1]}^\top]. \tag{36}$$

If the system is observable, and with l chosen larger than the system lag, the introduction of state (36) allows to

[2] https://github.com/4flixt/2023_Stochastic_MSM

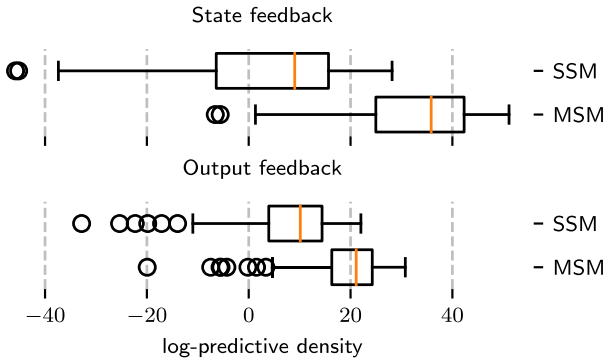

FIGURE 3. Nonlinear CSTR system: Comparison of log-predictive density for the identified state-space model (SSM) vs. the multi-step model (MSM). The predictive density is computed for 100 test cases as shown in (37). Representation as box-plot with filtered outliers, inter-quartile range, minimum-maximum range and median.

reformulate the system with output-feedback in the form of (1) [11] and satisfies Assumption 1. Unfortunately, considering output-feedback, by introducing the state in (36), violates Assumption 2 as the additive measurement noise on the newly introduced state (36) is now correlated. While this correlation is known to yield biased estimates [3, Sec. 5], we show in the following example that the proposed method can still lead to good results.

For the system identification, we gather $m = 500$ simulated sequences of length $L = N + l$ with $N = 20$ and $l = 1$ for the models with state-feedback and $l = 3$ for model with output-feedback. For both cases we also create $m = 100$ sequences of test data. All sampled sequences are created with uniformly random initial state, within the bounds shown in Table 2, and with persistently exciting random inputs.

In contrast to the linear case study in Section VI, there is no ground-truth distribution to evaluate the performance of the identified models. We therefore investigate the quality of the obtained probabilistic models by computing the logarithm of the predicted distribution, that is:

$$\log p\left(\boldsymbol{y}_{[1,N]} = \boldsymbol{y}^{(i)}_{[1,N]} | \mathcal{D}, \boldsymbol{y}^{(i)}_{0}, \boldsymbol{u}^{(i)}_{[0,N-1]}\right), \qquad (37)$$

for the test samples $\left(\boldsymbol{y}^{(i)}_{[0,N]}, \boldsymbol{u}^{(i)}_{[0,N-1]}\right)$ $\forall i \in \mathbb{I}_{[1,m]}$. The log-predictive density is an expressive measure for the quality of the identified probabilistic model. High values indicate high confidence in correct predictions, while low values indicate high confidence in incorrect predictions.

We display the log-predictive density in Figure 3 in the form of a box-plot to visualize the median, the quartile range, minimum and maximum as well as outliers. The identified multi-step model shows clear advantages over the recursively evaluated state-space model. Both, for state-feedback and output-feedback, the median and quartile ranges of the log-predictive density are significantly higher for the multi-step model. We also see that the log-predictive density obtained with the MSM is increased in the case of state-feedback. The state-space model, on the other hand, experiences more severe outliers for state-feedback, reducing the overall log-predictive density. We reason that with state-feedback the uncertainty in the predictive distribution is reduced as more information is available. However, this only benefits the log-predictive density of the MSM which yields accurate predictions with high confidence in that case.

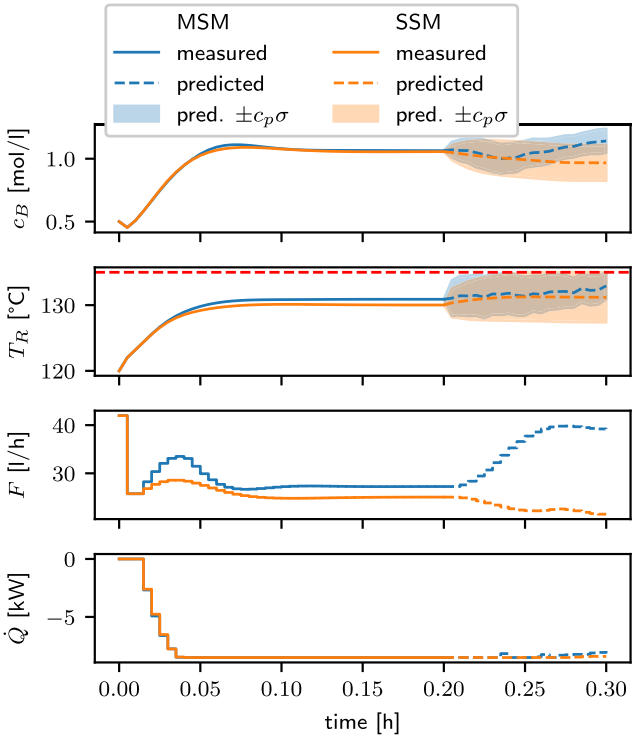

FIGURE 4. Nonlinear CSTR: Output-feedback SMPC closed-loop results and future prediction with uncertainty bounds. Comparison of SMPC and prediction with identified multi-step model (MSM) and identified state-space model (SSM) for an exemplary initial condition.

TABLE 3. Nonlinear CSTR system: Closed-loop SMPC performance over 50 experiments. Comparison of MSM and SSM models with state or output-feedback.

Output-feedback	MSM	product [mol]	6.15±0.23
		max. cons. viol [°C]	0.00±0.00
	SSM	product [mol]	5.79±1.10
		max. cons. viol [°C]	0.01±0.06
State-feedback	MSM	product [mol]	7.17±0.21
		max. cons. viol [°C]	0.00±0.00
	SSM	product [mol]	6.16±0.21
		max. cons. viol [°C]	0.00±0.00

C. STOCHASTIC MPC

We proceed our investigation with an analysis of the stochastic MPC controllers obtained with the multi-step model (MSM-SMPC) and the state-space model (SSM-SMPC). Furthermore, we continue to investigate the differences between

output-feedback and state-feedback for the identified systems and the control application.

All investigated controller variants are implemented with the control objective to maximize the production of component B. Furthermore, we seek to penalize rapid changes of the control inputs. To this end, we propose the cost function:

$$J(\boldsymbol{y}_{[1,N]}, \boldsymbol{u}_{[0,N-1]}) = -\sum_{k=1}^{N} c_{B,k} + \sum_{k=1}^{N-1} \left(10^{-2}\Delta F_K + 10^{-5}\Delta \dot{Q}_k\right),$$

with $\Delta F_k = F_k - F_{k-1}$ and $\Delta \dot{Q}_k = \dot{Q}_k - \dot{Q}_{k-1}$. The problem is formulated with horizon $N = 20$ and must consider the constraints in Table 2. Of those constraints, we formulate the safety critical bound of the reactor temperature as a chance constraint $P(T_R \leq 135) \geq (1-\epsilon)$, with probability of violation $\epsilon = 10^{-3}$.

To illustrate the behavior of the MSM-SMPC and SSM-SMPC controllers, we showcase an exemplary closed-loop trajectory in Figure 4, for the case of output-feedback. Both controllers run for 40 timesteps, corresponding to 12 min of simulation time. Additionally, we display mean and standard deviation, scaled with $c_p(\epsilon)$, of the open-loop prediction for the final timestep. Both controllers achieve satisfactory control performance in the example, yielding a high concentration of c_B while safely avoiding the constraint $T_R \leq 135$. However, we see in Figure 4 that the SMPC controller with multi-step model can operate significantly closer to the chance constraint for the reactor temperature. This allows to realize a higher product concentration c_B and higher normalized flow rate F, which yields overall more product.

To further quantify the performance of SSM-MPC and MSM-SMPC, also for the case of state-feedback, we present the results in Table 3. We display the amount of produced component B and the maximum constraint violation of the chance constraint $T_R \leq 135$ for all investigated controller variants. These performance indicators are computed as mean and standard deviation for 50 independent experiments with different initial state sampled uniformly within the bounds in Table 2. The table supports the qualitative finding from Figure 4. MSM-SMPC outperforms SSM-SMPC in the scenario with state-feedback and for the scenario with output-feedback. For the output-feedback scenario, we also see that SSM-SMPC leads to minor constraint violations. Finally, we observe that both controllers have a better performance, in terms of the obtained product, when state-feedback is available. In the case of state-feedback the MSM-SMPC controllers results, on average, in 16.4 % more product than the SSM-SMPC controller.

VIII. CONCLUSION

In this work, we propose a novel approach that combines probabilistic multi-step system identification with stochastic Model Predictive Control (SMPC). Our identification procedure is derived for linear systems with noisy state-feedback and with Gaussian process and measurement noise. In contrast to previous work, our proposed method does not require knowledge of the noise covariance matrices.

As a main contribution, we derive that the identified multi-step model yields, in expectation, the true distribution of the future measurements. We show that evaluating our identified model with noisy state-measurements is equivalent to estimating the initial state distribution and propagating this distribution with the known system dynamics. In this way, the identified multi-step model performs an implicit state estimation and can directly be used to formulate an SMPC problem for noisy state-measurements.

We demonstrate the theoretical findings and the performance of our proposed data-based SMPC controller in two simulation studies. In comparison to a SMPC controller based on an identified state-space model, we achieve significantly better performance and safer operation. Furthermore, we showcase in the second simulation study that our proposed method can also be applied to a nonlinear system with output-feedback, despite not being originally derived for this context.

In future work, we seek to rigorously extend our method to linear systems with output-feedback, by considering the correlation of the measurement noise. Furthermore, we seek to extend the method by updating the identified multi-step model with new data in a recursive fashion.

APPENDIX

A. CONVEXITY OF SMPC PROBLEM WITH PARAMETRIC UNCERTAINTY

We show that problem (30) is a convex optimization problem. The last term in the objective function in (30a) can be reformulated as:

$$\operatorname{trace}\left(\alpha(\boldsymbol{v})\boldsymbol{Q}\boldsymbol{\Sigma}^{*}_{y,[1,N]}\right) = \left(1+\boldsymbol{v}^{\top}\hat{\boldsymbol{\Sigma}}^{*}_{p}\boldsymbol{v}\right)\operatorname{trace}\left(\boldsymbol{Q}\boldsymbol{\Sigma}^{*}_{y,[1,N]}\right),$$

with positive definite matrix $\hat{\boldsymbol{\Sigma}}^{*}_{p}$ due to Assumption 4. Adding the convex term to the overall objective function does not change the convexity of the problem. The expression $\hat{\boldsymbol{y}}^{*}_{[1,N]}$ in (26b) is linear in the optimization variables $\boldsymbol{u}_{[0,N-1]}$. It remains to show that the inequality constraint (30b) is convex. To this end, we show in the following lemma that (30b) represents a second order cone and is thus convex [19].

Lemma 2: Let Assumption 4 hold. The constraint (30b) has the form $\forall j \in \mathbb{I}_{[1,n_c]}$*:*

$$\boldsymbol{g}_j(\boldsymbol{v}) \leq 0,$$

with:

$$\boldsymbol{g}_j(\boldsymbol{v}) = \tilde{\boldsymbol{a}}_j^{\top}\boldsymbol{v} + \sqrt{\boldsymbol{v}^{\top}\hat{\boldsymbol{\Sigma}}^{*}_{p}\boldsymbol{v}+1} - \tilde{b}_j, \tag{38}$$

where:

$$\tilde{\boldsymbol{a}}_j^{\top} = \boldsymbol{a}_j^{\top}[\hat{\mathcal{T}}^{*}_{\boldsymbol{A},\boldsymbol{B}},\ \hat{\mathcal{O}}^{*}_{\boldsymbol{A}}]c_p(\epsilon)^{-1}d_j^{-1/2}, \tag{39a}$$

$$\tilde{b}_j = b_j c_p(\epsilon)^{-1} d_j^{-1/2}, \tag{39b}$$

and:

$$d_j = \boldsymbol{a}_j^\top \hat{\boldsymbol{\Sigma}}^*_{y,[1,N]} \boldsymbol{a}_j. \tag{40}$$

The constraint represents a second order cone.

Proof: To obtain the form in (38), we first insert (26b) in (30b) and consider the definition of $\boldsymbol{v}$:

$$\boldsymbol{a}_j^\top [\hat{T}^*_{\boldsymbol{A},\boldsymbol{B}},\ \hat{\mathcal{O}}^*_{\boldsymbol{A}}]\boldsymbol{v} \leq b_j - c_p(\epsilon)\|\boldsymbol{a}_j\|_{\alpha\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}}. \tag{41}$$

We expand the term $\|\boldsymbol{a}_j\|_{\alpha\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}}$ and obtain:

$$\begin{aligned}\|\boldsymbol{a}_j\|_{\alpha\hat{\boldsymbol{\Sigma}}^*_{y,[1,N]}} &= \sqrt{\alpha \boldsymbol{a}_j^\top \hat{\boldsymbol{\Sigma}}^*_{y,[1,N]} \boldsymbol{a}_j} = \sqrt{d_j}\sqrt{\alpha}, \\ &= \sqrt{d_j}\sqrt{1 + \boldsymbol{v}^\top \hat{\boldsymbol{\Sigma}}^*_p \boldsymbol{v}},\end{aligned} \tag{42}$$

with constant d_j introduced in (40). We insert (42) in (41) and divide by $\sqrt{d_j}c_p(\epsilon)$. By introducing and substituting the expressions (39), we obtain the desired expression (38). □

REFERENCES

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

[2] M. Morari and J. H. Lee, "Model predictive control: Past, present and future," *Comput. Chem. Eng.*, vol. 23, nos. 4–5, pp. 667–682, May 1999.

[3] K. J. Åström and P. Eykhoff, "System identification—A survey," *Automatica*, vol. 7, no. 2, pp. 123–162, 1971.

[4] L. Ljung, "System identification," in *Signal Analysis and Prediction* (Applied and Numerical Harmonic Analysis). Basel, Switzerland: Birkhauser, 1998, pp. 163–173.

[5] W. Favoreel, B. D. Moor, and M. Gevers, "SPC: Subspace predictive control," *IFAC Proc. Volumes*, vol. 32, no. 2, pp. 4004–4009, Jul. 1999.

[6] E. Terzi, L. Fagiano, M. Farina, and R. Scattolini, "Learning-based predictive control for linear systems: A unitary approach," *Automatica*, vol. 108, Oct. 2019, Art. no. 108473.

[7] E. Terzi, M. Farina, L. Fagiano, and R. Scattolini, "Robust multi-rate predictive control using multi-step prediction models learned from data," *Automatica*, vol. 136, Feb. 2022, Art. no. 109852.

[8] J. Köhler, K. P. Wabersich, J. Berberich, and M. N. Zeilinger, "State space models vs. multi-step predictors in predictive control: Are state space models complicating safe data-driven designs?" in *Proc. IEEE 61st Conf. Decis. Control (CDC)*, Dec. 2022, pp. 491–498.

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

[10] I. Markovsky and F. Dörfler, "Behavioral systems theory in data-driven analysis, signal processing, and control," *Annu. Rev. Control*, vol. 52, pp. 42–64, Jan. 2021.

[11] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "On the design of terminal ingredients for data-driven MPC," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 257–263, 2021.

[12] 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, Feb. 2023.

[13] F. Fiedler and S. Lucia, "On the relationship between data-enabled predictive control and subspace predictive control," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2021, pp. 222–229.

[14] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Syst. Mag.*, vol. 36, no. 6, pp. 30–44, Dec. 2016.

[15] M. Farina, L. Giulioni, and R. Scattolini, "Stochastic linear model predictive control with chance constraints – a review," *J. Process Control*, vol. 44, pp. 53–67, Aug. 2016.

[16] M. Fochesato and J. Lygeros, "Data-driven distributionally robust bounds for stochastic model predictive control," in *Proc. IEEE 61st Conf. Decis. Control (CDC)*, Dec. 2022, pp. 3611–3616.

[17] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 6, pp. 2736–2743, Nov. 2020.

[18] G. A. Seber, *A Matrix Handbook for Statisticians*. Hoboken, NJ, USA: Wiley, 2008.

[19] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[20] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Trans. Robot.*, vol. 27, no. 6, pp. 1080–1094, Dec. 2011.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.

[22] W. A. Fuller, *Measurement Error Models*. Hoboken, NJ, USA: Wiley, 2009.

[23] J. P. Buonaccorsi, *Measurement Error: Models, Methods, and Applications*. Boca Raton, FL, USA: CRC Press, 2010.

[24] L. Hewing, K. P. Wabersich, and M. N. Zeilinger, "Recursively feasible stochastic model predictive control using indirect feedback," *Automatica*, vol. 119, Sep. 2020, Art. no. 109095.

[25] B. Bueno, L. Norford, G. Pigeon, and R. Britter, "A resistance-capacitance network model for the analysis of the interactions between the energy performance of buildings and the urban climate," *Building Environ.*, vol. 54, pp. 116–125, Aug. 2012.

[26] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019.

[27] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[28] K.-U. Klatt and S. Engell, "Gain-scheduling trajectory control of a continuous stirred tank reactor," *Comput. Chem. Eng.*, vol. 22, nos. 4–5, pp. 491–502, Jan. 1998.

[29] F. Fiedler, B. Karg, L. Lüken, D. Brandner, M. Heinlein, F. Brabender, and S. Lucia, "Do-MPC: Towards FAIR nonlinear and robust model predictive control," *Control Eng. Pract.*, vol. 140, Nov. 2023, Art. no. 105676.

FELIX FIEDLER received the M.Sc. degree in chemical engineering from TU Berlin University, Germany, in 2018. He was Ph.D. candidate at TU Berlin University with the HEIBRiDS Graduate School program, until 2020. In the same year, he joined TU Dortmund University, Germany, where he is currently pursuing his Ph.D. with the Chair of Process Automation Systems. His research interest include model predictive control under uncertainty, system identification and machine learning. He is also the current lead developer of the open-source do-mpc toolbox which enables rapid non-linear MPC developments in Python (www.do-mpc.com).

SERGIO LUCIA (Member, IEEE) received the M.Sc. degree in electrical engineering from the University of Zaragoza, Spain, in 2010, and the Dr. (Ing.) degree in optimization and automatic control from TU Dortmund University, Germany, in 2014. He joined Otto-von-Guericke Universität Magdeburg and visited the Massachusetts Institute of Technology as a Postdoctoral Fellow. He was an Assistant Professor with TU Berlin, between 2017 and 2020. Since 2020, he has been a Professor with TU Dortmund University and the Head of the Chair of Process Automation Systems. His research interests include decision-making under uncertainty, distributed control, and the interplay between machine learning techniques and control theory. He is currently an Associate Editor of the *Journal of Process Control*.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 October 2023, accepted 30 October 2023, date of publication 2 November 2023, date of current version 8 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3329685

RESEARCH ARTICLE

Improved Uncertainty Quantification for Neural Networks With Bayesian Last Layer

FELIX FIEDLER AND SERGIO LUCIA, (Member, IEEE)
Chair of Process Automation Systems, TU Dortmund University, 44227 Dortmund, Germany
Corresponding author: Felix Fiedler (felix.fiedler@tu-dortmund.de)

ABSTRACT Uncertainty quantification is an important task in machine learning - a task in which standard neural networks (NNs) have traditionally not excelled. This can be a limitation for safety-critical applications, where uncertainty-aware methods like Gaussian processes or Bayesian linear regression are often preferred. Bayesian neural networks are an approach to address this limitation. They assume probability distributions for all parameters and yield distributed predictions. However, training and inference are typically intractable and approximations must be employed. A promising approximation is NNs with Bayesian last layer (BLL). They assume distributed weights only in the linear output layer and yield a normally distributed prediction. To approximate the intractable Bayesian neural network, point estimates of the distributed weights in all but the last layer should be obtained by maximizing the marginal likelihood. This has previously been challenging, as the marginal likelihood is expensive to evaluate in this setting. We present a reformulation of the log-marginal likelihood of a NN with BLL which allows for efficient training using backpropagation. Furthermore, we address the challenge of uncertainty quantification for extrapolation points. We provide a metric to quantify the degree of extrapolation and derive a method to improve the uncertainty quantification for these points. Our methods are derived for the multivariate case and demonstrated in a simulation study. In comparison to Bayesian linear regression with fixed features, and a Bayesian neural network trained with variational inference, our proposed method achieves the highest log-predictive density on test data.

INDEX TERMS Bayesian last layer, Bayesian neural network, uncertainty quantification.

I. INTRODUCTION

Machine learning tries to capture patterns and trends through data. Both, the data and the identified patterns are subject to uncertainty [1], [2]. For many applications, especially those where machine learning is applied to safety critical tasks, it is imperative to quantify the uncertainty of the predictions. An important example is learning-based control, where probabilistic system models are identified from data and used for safe control decisions [3], [4], [5], [6], [7], [8].

Bayesian linear regression (BLR) [3], [4], [5], [6], [7] and Gaussian processes (GPs) [7], [8] are prominent methods for probabilistic system identification. Both assume that a nonlinear feature space can be mapped linearly to the outputs. As a main difference, BLR requires the features to be explicitly defined, while for GPs the features are implicitly defined through a kernel function [9]. This is an advantage of GPs, as the most suitable features for BLR are often challenging to determine. On the other hand, GPs scale poorly with the number of data samples [10]. For big data problems they are typically approximated with sparse GPs [11], and can be further improved with deep kernel learning [12], [13].

Especially in recent years, neural networks (NNs) and deep learning have gained significant popularity for a vast variety of machine learning tasks [14]. NNs have also been successfully applied for control applications [15], [16], often to infer the system model from data [17], [18], [19]. A challenge with NNs is their tendency to overfit and their inability to express uncertainty [1]. Bayesian neural networks (BNNs), in which the weights and predictions are probability distributions, are a concept to tackle this shortcoming. In practice, BNNs can be intractable to train and query and are often approximated [1], [2]. Most approximate BNN approaches fall into one of two categories: Markov chain

The associate editor coordinating the review of this manuscript and approving it for publication was Alba Amato.

Monte Carlo (MCMC) and variational inference (VI) [1]. MCMC methods do not require a classical training phase and instead sample directly the posterior weight distribution of the Bayesian neural network [20]. Unfortunately, MCMC scales poorly to large models which limits their applicability [1]. Variational inference is a popular alternative to MCMC and based on the idea of learning the parameters of a surrogate posterior distribution of the weights [1], [2]. For neural networks, variational inference is often implemented with the *Bayes by Backprop* algorithm [21].

Unfortunately, even a BNN trained with VI requires sampling to approximate the predictive distribution [1]. This can be a significant disadvantage in comparison to GPs and BLR, which yield analytical results. A promising compromise between tractability and expressiveness are NNs with Bayesian last layer (BLL) [10], [22]. NNs with BLL can be seen as a simplified BNN, where only the weights of the output layer follow a Gaussian distribution, and the remaining layers contain deterministic weights. At the same time, they can be interpreted as a deep kernel learning approach with linear kernel. NNs with BLL are also strongly related to BLR in that they consider a nonlinear feature space which is mapped linearly onto the outputs. Similarly to GPs, BLR and other probabilistic models [9], [23], NNs with BLL are trained by maximizing the marginal likelihood with respect to the parameters of the probabilistic model. These parameters include prior and noise variance and, importantly, the weights of the deterministic layers. While the log-marginal likelihood (LML) can be expressed analytically for NNs with BLL, it contains expressions such as the inverse of the precision matrix, making it unsuitable for direct gradient-based optimization. Previous works, especially for control applications, have therefore either assumed knowledge of all parameters [3], [5], [19], including prior and noise variance, or have maximized the LML after training a NN with fixed features [4], [24]. In previous works that did include the deterministic weights as parameters, maximizing the LML required sampling the surrogate posterior during training [22], or using an approximate precision matrix [12] to enable gradient-based optimization.

As a main contribution of this work, we propose an approach to maximize the exact LML of a NN with BLL that does not require sampling and is suitable for gradient-based optimization. Most importantly, we avoid the matrix inverse in the LML by reintroducing the weights of the last layer, which were marginalized, as optimization variables. We show that our reformulation of the LML satisfies the conditions of optimality for the same solution as the original formulation. In this way, we provide a simpler training procedure, in comparison to training with variational inference, and can outperform BLR with fixed features obtained from a NN. Our second main contribution is an algorithm to improve the uncertainty quantification for extrapolation points. To this end, we relate the computation of the BLL covariance to an intuitive metric for extrapolation, which is inspired by the definition of extrapolation in [25]. Based on this relationship, the proposed algorithm adjusts a scalar parameter to improve the log-predictive density on additional validation data. Our proposed methods are derived for the multivariate-case, estimating individual noise-variances for each output. This is in contrast to GP and BLR applications, where the multivariate case is typically tackled by fitting an individual model to each output [4], [7], [8] or by assuming i.i.d. noise for all outputs [5]. The advantage of a single model for the multivariate case is apparent for the application of system identification for optimization-based control, as multiple models may significantly increase computation times.

This work is structured as follows. In Section II, we introduce NNs with BLL. The marginal likelihood, which is maximized during training, is discussed in Section III. In Section IV, we discuss interpolation and extrapolation for NNs with BLL and present an algorithm to improve the predictive distribution in the extrapolation regime. We discuss the special considerations required for the multivariate case in Section V. The Bayes by Backprop method is introduced in Section VI. A simulation example to compare the proposed method with BLR and Bayes by Backprop is presented in Section VII. The paper is concluded in Section VIII.

II. BAYESIAN LAST LAYER

We investigate a dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{t}\}$ consisting of m data pairs of inputs $\boldsymbol{x} \in \mathbb{R}^{n_x}$ and targets $t \in \mathbb{R}$ from which the matrices $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_m]^\top \in \mathbb{R}^{m \times n_x}$ and $\boldsymbol{t} = [t_1, \dots, t_m]^\top \in \mathbb{R}^{m \times 1}$ are formed. We assume a scalar output for ease of notation and address the multivariate case in Section V. Regarding the notation, lower case bold symbols denote vectors, upper case bold symbols denote matrices, and regular lower case symbols are scalars. For the regression task, we introduce a feed-forward NN model with L hidden layers as

$$y = NN(\boldsymbol{x}; \mathbb{W}_{L+1}) = g_{L+1} \circ h_{L+1} \circ \cdots \circ g_1 \circ h_1(\boldsymbol{x}), \quad (1)$$

where $\circ$ denotes function composition. At each layer, we have a linear mapping $h_l(\cdot)$ followed by a nonlinear activation function $g_l(\cdot)$, that is:

$$\boldsymbol{h}_l^\top = h_l(\boldsymbol{a}_{l-1}) = \begin{bmatrix} \boldsymbol{a}_{l-1}^\top, & 1 \end{bmatrix} \boldsymbol{W}_l \quad l \in \mathbb{I}_{[1,L+1]}, \quad (2)$$

$$\boldsymbol{a}_l = g_l(\boldsymbol{h}_l) \quad l \in \mathbb{I}_{[1,L+1]}. \quad (3)$$

The set $\mathbb{I}_{[1,L+1]}$ denotes the set of integers from 1 to $L+1$. We have $\boldsymbol{a}_0 = \boldsymbol{x}$, $\boldsymbol{a}_{L+1} = \boldsymbol{y}$, and $\boldsymbol{a}_l \in \mathbb{R}^{n_{a,l}}$ for all $l \in \mathbb{I}_{[1,L]}$. The number of neurons in layer l is denoted as $n_{a,l}$, and we have the weights $\boldsymbol{W} \in \mathbb{R}^{n_{a,l-1}+1 \times n_{a,l}}$, which include the bias term. The set of weight matrices with cardinality $L+1$ is denoted $\mathbb{W}_{L+1} = \{\boldsymbol{W}_1, \dots, \boldsymbol{W}_{L+1}\}$. As a requirement for NNs with BLL, we state the following assumption.

Assumption 1: The NN (1) has a linear activation function in the output layer, i.e. $g_{L+1}(h_{L+1}) = h_{L+1}$.

With the linear mapping, the output of the last internal layer is of particular importance and we introduce the notation:

$$\tilde{\boldsymbol{\phi}} = \boldsymbol{a}_L, \text{ and } \boldsymbol{\phi} = [\boldsymbol{a}_L^\top, \ 1]^\top, \quad (4)$$

where $\tilde{\boldsymbol{\phi}} \in \mathbb{R}^{n_{\tilde{\phi}}}$ are referred to as *linear features* and $\boldsymbol{\phi} \in \mathbb{R}^{n_\phi}$ are the corresponding *affine features* with $n_\phi = n_{\tilde{\phi}} + 1$. With slight abuse of notation, we use $\tilde{\boldsymbol{\phi}} = \tilde{\boldsymbol{\phi}}(\boldsymbol{x}; \mathbb{W}_L)$ and $\boldsymbol{\phi} = \boldsymbol{\phi}(\boldsymbol{x}; \mathbb{W}_L)$ as both, the values of the features, and the function, parameterized with $\mathbb{W}_L$. We require two additional assumptions to state Lemma 1 which formally describes a NN with BLL.

Assumption 2: The NN (1) provides a feature space $\boldsymbol{\phi}(\boldsymbol{x}; \mathbb{W}_L) \in \mathbb{R}^{n_\phi}$ from which the targets are obtained through a linear mapping, according to Assumption 1:

$$\boldsymbol{t} = \boldsymbol{\phi}(\boldsymbol{X}; \mathbb{W}_L)^\top \boldsymbol{w} + \boldsymbol{\epsilon} = \boldsymbol{y} + \boldsymbol{\epsilon}, \tag{5}$$

where we introduce the set $\mathbb{W}_L = \{\boldsymbol{W}_1, \dots, \boldsymbol{W}_L\}$ for all weights until layer L and have $\boldsymbol{w} = \boldsymbol{W}_{L+1}$ the weights of the output layer. The additive noise $\boldsymbol{\epsilon} \in \mathbb{R}^m$ is zero-mean normally distributed, that is, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_E)$.

Assumption 3: We have a prior belief for the weights of the output layer $\mathbf{w} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_W)$.

We introduce the parameters of the probabilistic model (5) as:

$$\boldsymbol{\Theta} = \{\mathbb{W}_L, \boldsymbol{\Sigma}_E, \boldsymbol{\Sigma}_W\}, \tag{6}$$

and state the posterior distribution of the weights $\mathbf{w}$ of the last layer using Bayes' law:

$$p(\mathbf{w}|\mathcal{D}, \boldsymbol{\Theta}) = \frac{p(\mathcal{D}|\mathbf{w}, \boldsymbol{\Theta}) p(\mathbf{w}|\boldsymbol{\Theta})}{p(\mathcal{D}|\boldsymbol{\Theta})}. \tag{7}$$

A neural network with Bayesian last layer has deterministic weights in all hidden layers and distributed weights in the output layer. We obtain an analytical expression for the distribution of the the predicted outputs as shown in the following lemma [22].

Lemma 1: Assumptions 1-3 hold. The predicted outputs are normally distributed with:

$$p(\mathbf{y}|\mathcal{D}, \boldsymbol{\Theta}, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_y^{NN}(\boldsymbol{x}), \boldsymbol{\Sigma}_y^{NN}(\boldsymbol{x})), \tag{8a}$$

$$\boldsymbol{\mu}_y^{NN}(\boldsymbol{x}) = NN(\boldsymbol{x}; \mathbb{W}_{L+1}), \tag{8b}$$

$$\boldsymbol{\Sigma}_y^{NN}(\boldsymbol{x}) = \boldsymbol{\phi}^\top \boldsymbol{\Lambda}_p^{-1} \boldsymbol{\phi}, \tag{8c}$$

where $\boldsymbol{\Phi} = \boldsymbol{\phi}(\boldsymbol{X}; \mathbb{W}_L)$ is the feature matrix for the training data, $\boldsymbol{\phi} = \boldsymbol{\phi}(\boldsymbol{x}; \mathbb{W}_L)$ is the feature matrix for the test data, and with the precision matrix

$$\boldsymbol{\Lambda}_p = \boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_E^{-1} \boldsymbol{\Phi} + \boldsymbol{\Sigma}_W^{-1}. \tag{9}$$

Proof: The posterior $p(\mathbf{w}|\mathcal{D}, \boldsymbol{\Theta})$ in (7) yields a normal distribution in the weights:

$$p(\mathbf{w}|\mathcal{D}, \boldsymbol{\Theta}) = \mathcal{N}(\bar{\boldsymbol{w}}, \boldsymbol{\Lambda}_p^{-1}), \tag{10a}$$

$$\text{with:} \quad \bar{\boldsymbol{w}} = \boldsymbol{\Lambda}_p^{-1} \boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_E^{-1} \boldsymbol{t}, \tag{10b}$$

as shown for the case of arbitrary features in [23]. Finally, the predicted output $\boldsymbol{y}$ is a linear transformation of the random variable $\mathbf{w}$, yielding the distribution in (8). □

We can also obtain a posterior distribution for the targets, for which we consider (5) and obtain

$$p(\mathbf{t}|\mathcal{D}, \boldsymbol{\Theta}, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_y^{\text{NN}}(\boldsymbol{x}), \boldsymbol{\Sigma}_t^{\text{NN}}(\boldsymbol{x})), \tag{11a}$$

$$\boldsymbol{\Sigma}_t^{\text{NN}}(\boldsymbol{x}) = \boldsymbol{\Sigma}_y^{\text{NN}}(\boldsymbol{x}) + \boldsymbol{\Sigma}_E, \tag{11b}$$

where $\boldsymbol{\mu}_y^{\text{NN}}(\boldsymbol{x})$ and $\boldsymbol{\Sigma}_y^{\text{NN}}(\boldsymbol{x})$ stem from (8). We will revisit (11) again for the definition of performance metrics.

III. MARGINAL LIKELIHOOD MAXIMIZATION

The posterior distribution of the weights of the last layer can be obtained with (10) for given values of the parameters $\boldsymbol{\Theta}$. Determining suitable values of the parameters from prior knowledge is often challenging. Instead, the parameters can be inferred by maximizing the marginal likelihood in (7), which is also known as empirical Bayes or type-2 maximum likelihood [26]. In BLR, that is, for a fixed feature space, the log-marginal likelihood (LML) can be maximized as shown in [23]. Following this idea, the authors in [24] propose an approach where a fixed feature space is obtained through classical NN regression. After NN training, the LML is then maximized with these fixed features. However, the LML is also influenced by the weights of the deterministic layers and the resulting feature space. It is therefore reasonable to include the set of weights $\mathbb{W}_L$ until the last layer as parameters and train them by maximizing the LML. Unfortunately, this poses significant challenges.

We proceed by introducing the LML and discuss the challenge of maximizing this expression in the next subsection.

Lemma 2 (Log-Marginal Likelihood): If Assumptions 1-3 hold, the negative LML of (7), denoted as $J(\boldsymbol{\Theta}; \mathcal{D}) = -\log(\mathcal{D}|\boldsymbol{\Theta})$, results in:

$$J(\boldsymbol{\Theta}; \mathcal{D}) = \frac{m}{2}\log(2\pi) + \frac{1}{2}\log\det(\boldsymbol{\Sigma}_W) + \frac{1}{2}\log\det(\boldsymbol{\Sigma}_E) + \frac{1}{2}\log\det(\boldsymbol{\Lambda}_p) + \frac{1}{2}\|\boldsymbol{t} - \boldsymbol{y}\|^2_{\boldsymbol{\Sigma}_E^{-1}} + \frac{1}{2}\|\bar{\boldsymbol{w}}\|^2_{\boldsymbol{\Sigma}_W^{-1}}. \tag{12}$$

In this formulation, we have $\boldsymbol{\Lambda}_p$ from (9), $\bar{\boldsymbol{w}}$ from (10b), $\boldsymbol{y} = \boldsymbol{\Phi}\bar{\boldsymbol{w}}$ and $\boldsymbol{\Theta}$ from (6).

Proof: The proof for fixed features is shown in [23]. □

For practical applications, the formulation in (12) is further simplified by considering the following assumption, where $\text{diag}(\boldsymbol{A}, \boldsymbol{B})$ denotes a block-diagonal matrix with $\boldsymbol{A}$ and $\boldsymbol{B}$ on the diagonal.

Assumption 4: The additive noise introduced in Assumption 2 is i.i.d. for all samples m, yielding $\boldsymbol{\Sigma}_E = \sigma_e^2 \boldsymbol{I}_m$. The weight prior introduced in Assumption 3 is i.i.d. with a flat prior for the bias, that is, $\boldsymbol{\Sigma}_w^{-1} = \sigma_w^{-2}\text{diag}(\boldsymbol{I}_{n_{\tilde{\phi}}}, 0)$.

The assumption of a flat prior for the bias term will have a negligible effect in practice but is important for the theoretical results presented in Section IV. For ease of notation, we will reuse $J(\boldsymbol{\Theta}; \mathcal{D})$ and $\boldsymbol{\Theta}$ in following results.

Result 1: Applying Assumption 4 to Lemma 2, we can write the negative LML in (12) as:

$$J(\boldsymbol{\Theta}; \mathcal{D}) = \frac{m}{2}\log(2\pi) + n_\phi \log(\sigma_w) + m\log(\sigma_e) + \frac{1}{2}\log\det(\boldsymbol{\Lambda}_p) + \frac{1}{2\sigma_e^2}\|\boldsymbol{t} - \boldsymbol{y}\|_2^2 + \frac{1}{2\sigma_w^2}\|\bar{\boldsymbol{w}}\|_2^2. \tag{13}$$

We introduce $\tilde{\boldsymbol{I}}_{n_\phi} = \operatorname{diag}(\boldsymbol{I}_{n_{\bar{\phi}}}, 0)$ *and obtain simplified expressions for (9) and (10b):*

$$\boldsymbol{\Lambda}_p = \sigma_e^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{1}{\sigma_w^2}\tilde{\boldsymbol{I}}_{n_\phi}, \tag{14a}$$

$$\bar{\boldsymbol{w}} = \sigma_e^{-2}\boldsymbol{\Lambda}_p^{-1}\boldsymbol{\Phi}^\top\boldsymbol{t}. \tag{14b}$$

In this setting, we denote $\boldsymbol{\Theta} = \{\mathbb{W}_L, \sigma_e, \sigma_w\}$.

A. AUGMENTED LOG-MARGINAL LIKELIHOOD MAXIMIZATION

To train a NN with BLL we seek to minimize the negative LML:

$$\min_{\boldsymbol{\Theta}} \quad J(\boldsymbol{\Theta}; \mathcal{D}), \tag{15}$$

with $J(\boldsymbol{\Theta}; \mathcal{D})$ and $\boldsymbol{\Theta}$ according to Result 1. This problem excludes the weights in the last layer of the NN as they have been marginalized. A result of this marginalization is the expression (14b) which computes these weights explicitly. Unfortunately, this creates a major challenge when iteratively solving problem (15) for the optimal values $\boldsymbol{\Theta}^*$, as the computational graph contains unfavorable expressions such as the inverse of $\boldsymbol{\Lambda}_p$, which are both numerically challenging and computationally expensive. Furthermore, (14b) requires leveraging the entire training dataset for the computation of the gradient $\nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta}; \mathcal{D})$.

The authors in [22] circumvent these issues by variational inference, replacing the LML objective by the evidence lower bound objective (ELBO). As the variational posterior they choose a Gaussian distribution, parameterized with mean and covariance. This is the obvious choice in the BLL setting where the true posterior is Gaussian as shown in (8) and, consequently, the ELBO objective is equivalent to the LML [22]. Variational inference comes at the cost, however, of introducing as optimization variables the parameters to describe the variational posterior. Furthermore, the variational inference training loop requires sampling this variational posterior.

In this work, we present an alternative approach, which simultaneously avoids parameterizing the variational posterior and yields a computational graph with lower complexity than (13). The resulting formulation is suitable for fast gradient-based optimization. To obtain this result we reformulate (15) as

$$\begin{aligned} \min_{\boldsymbol{\Theta}, \bar{\boldsymbol{w}}} \quad & J(\boldsymbol{\Theta}, \bar{\boldsymbol{w}}; \mathcal{D}) \\ \text{s.t.} \quad & \bar{\boldsymbol{w}} = \frac{1}{\sigma_e^2}\boldsymbol{\Lambda}_p^{-1}\boldsymbol{\Phi}^\top\boldsymbol{t}, \end{aligned} \tag{16}$$

with $\boldsymbol{\Theta} = \{\mathbb{W}_L, \sigma_e, \sigma_w\}$. Importantly, we introduce $\bar{\boldsymbol{w}}$ as an optimization variable and add an equality constraint corresponding to (14b). The optimal solution $\boldsymbol{\Theta}^*$ of (16) is thus identical to the optimal solution of (15). As a main contribution of this work, we state the following theorem.

Theorem 1 (Augmented Log-Marginal Likelihood Maximization): The optimal solution $(\boldsymbol{\Theta}^*, \bar{\boldsymbol{w}}^*)$ of problem (16) is identical to the optimal solution obtained from the unconstrained problem:

$$\min_{\boldsymbol{\Theta}, \bar{\boldsymbol{w}}} \quad J(\boldsymbol{\Theta}, \bar{\boldsymbol{w}}; \mathcal{D}), \tag{17}$$

where, as the only difference to (15), $\bar{\boldsymbol{w}}$ is now an optimization variable and, in comparison to (16), the equality constraint has been dropped.

Proof: The Lagrangian of problem (16) can be written as:

$$\mathcal{L}(\boldsymbol{\Theta}, \bar{\boldsymbol{w}}, \lambda; \mathcal{D}) = J(\boldsymbol{\Theta}, \bar{\boldsymbol{w}}; \mathcal{D}) + \lambda^\top\left(\boldsymbol{\Lambda}_p\bar{\boldsymbol{w}} - \frac{1}{\sigma_e^2}\boldsymbol{\Phi}^\top\boldsymbol{t}\right).$$

We then state the first-order condition of optimality for the optimization variable $\bar{\boldsymbol{w}}$:

$$\nabla_{\bar{\boldsymbol{w}}}\mathcal{L}(\boldsymbol{\Theta}, \bar{\boldsymbol{w}}, \lambda; \mathcal{D}) = \nabla_{\bar{\boldsymbol{w}}} J(\boldsymbol{\Theta}, \bar{\boldsymbol{w}}; \mathcal{D}) + \lambda^\top\boldsymbol{\Lambda}_p \overset{!}{=} 0. \tag{18}$$

Considering (13) and (5), that is, $\boldsymbol{y} = \boldsymbol{\Phi}\bar{\boldsymbol{w}}$, we obtain:

$$\begin{aligned} &\nabla_{\bar{\boldsymbol{w}}} J(\boldsymbol{\Theta}, \bar{\boldsymbol{w}}; \mathcal{D}) \\ &\quad= \nabla_{\bar{\boldsymbol{w}}}\left(\frac{1}{2\sigma_e^2}\|\boldsymbol{t} - \boldsymbol{\Phi}\bar{\boldsymbol{w}}\|_2^2 + \frac{1}{2\sigma_w^2}\|\bar{\boldsymbol{w}}\|_2^2\right) && (19)\\ &\quad= \frac{2}{2\sigma_e^2}\left(\bar{\boldsymbol{w}}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi} - \boldsymbol{t}^\top\boldsymbol{\Phi}\right) + \frac{2}{2\sigma_w^2}\bar{\boldsymbol{w}}^\top && (20)\\ &\quad= \bar{\boldsymbol{w}}^\top\left(\frac{1}{\sigma_e^2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{1}{\sigma_w^2}\tilde{\boldsymbol{I}}_{n_\phi}\right) - \frac{1}{\sigma_e^2}\boldsymbol{t}^\top\boldsymbol{\Phi}. && (21) \end{aligned}$$

Using (14a), we obtain:

$$\nabla_{\bar{\boldsymbol{w}}} J(\boldsymbol{\Theta}, \bar{\boldsymbol{w}}; \mathcal{D}) = \bar{\boldsymbol{w}}^\top\boldsymbol{\Lambda}_p - \frac{1}{\sigma_e^2}\boldsymbol{t}^\top\boldsymbol{\Phi}. \tag{22}$$

We substitute (22) into (18) and have:

$$\begin{aligned} \lambda^\top\boldsymbol{\Lambda}_p &= -\bar{\boldsymbol{w}}^\top\boldsymbol{\Lambda}_p + \frac{1}{\sigma_e^2}\boldsymbol{t}^\top\boldsymbol{\Phi}, && (23)\\ \Leftrightarrow\ \lambda &= -\boldsymbol{\Lambda}_p^{-1}\boldsymbol{\Lambda}_p\bar{\boldsymbol{w}} + \frac{1}{\sigma_e^2}\boldsymbol{\Lambda}_p^{-1}\boldsymbol{\Phi}^\top\boldsymbol{t}, && (24)\\ \Leftrightarrow\ \lambda &= -\bar{\boldsymbol{w}} + \bar{\boldsymbol{w}} = 0, && (25) \end{aligned}$$

where in (24) we have substituted (14b). From (25), we obtain that $\lambda = 0$ and the Lagrangian of problem (16) thus simplifies to:

$$\mathcal{L}(\boldsymbol{\Theta}, \bar{\boldsymbol{w}}, \lambda; \mathcal{D}) = J(\boldsymbol{\Theta}, \bar{\boldsymbol{w}}; \mathcal{D}). \tag{26}$$

This is exactly the Lagrangian of problem (17) and therefore problem (16) and (17) yield the same optimal solution. □

Theorem 1 enables us to maximize the LML (13) without having to explicitly compute $\bar{\boldsymbol{w}}$ according to Equation (14b). This means, in particular, that we are not required to compute the inverse of $\boldsymbol{\Lambda}_p$ to express the LML. Consequentially, we can use back-propagation and gradient-based optimization to maximize the LML, which significantly simplifies training NNs with BLL.

B. CHANGE OF VARIABLES AND SCALING

With Theorem 1 we can state the LML as a function of $\bar{\boldsymbol{w}}$ which is now included in the set of parameters $\boldsymbol{\Theta} = \{\mathbb{W}_L, \bar{\boldsymbol{w}}, a, b\} = \{\mathbb{W}_{L+1}, a, b\}$. Furthermore, we propose a change of variables and scale the objective function to improve numerical stability. In particular, we introduce:

$$\alpha = \frac{\sigma_w^2}{\sigma_e^2}, \tag{27}$$

which can be interpreted as a signal-to-noise ratio and optimize over $\log(\alpha)$ and $\log(\sigma_e)$ to ensure that $\sigma_e > 0$ and $\sigma_w > 0$ without constraining the problem. These changes are formalized in the following result.

Result 2: Considering the definition of α in (27), we reformulate (14a):

$$\begin{aligned}\boldsymbol{\Lambda}_p &= \sigma_e^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma_w^{-2}\tilde{\boldsymbol{I}}_{n_\phi} \\ &= \sigma_e^{-2}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \alpha^{-1}\tilde{\boldsymbol{I}}_{n_\phi}\right) = \sigma_e^{-2}\bar{\boldsymbol{\Lambda}}_p,\end{aligned} \tag{28}$$

where

$$\bar{\boldsymbol{\Lambda}}_p = \boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \alpha^{-1}\tilde{\boldsymbol{I}}_{n_\phi}. \tag{29}$$

The scaled negative LML from (13) can then be written as:

$$\begin{aligned}&J(\boldsymbol{\Theta}; \mathcal{D}) \\ &= \frac{1}{2}\log(2\pi) + \frac{n_\phi}{2m}\log\alpha + \log\sigma_e \\ &\quad + \frac{1}{2m}\left(\log\det(\bar{\boldsymbol{\Lambda}}_p) + \sigma_e^{-2}\|\boldsymbol{t}-\boldsymbol{y}\|_2^2 + \alpha^{-1}\sigma_e^{-2}\|\bar{\boldsymbol{w}}\|_2^2\right),\end{aligned} \tag{30}$$

with $\boldsymbol{\Theta} = \{\mathbb{W}_{L+1}, \alpha, \sigma_e\}$.

To train a NN with BLL and univariate output, we consider in the following the LML and parameters in the form of Result 2. Additionally, the newly introduced parameter α in (27) will play an important role in the following discussion on interpolation and extrapolation and ultimately helps to improve the extrapolative uncertainty.

IV. IMPROVING THE EXTRAPOLATIVE UNCERTAINTY

One of the main challenges of the BLL predictive distribution (8) is that for arbitrary extrapolation points the required assumptions for Lemma 1 will not hold. In this section, we discuss the behavior of the predictive distribution in the interpolation and extrapolation regime and propose a method to improve the performance of NNs with BLL for extrapolation.

To formalize the notion of interpolation and extrapolation, we follow the definition of interpolation described in [25], for which we also need to define the convex hull.

Definition 1 (Convex Hull): The convex hull of a set of samples $\boldsymbol{X} \in \mathbb{R}^{m\times n_x}$ is defined as the set:

$$\mathbb{C}_{\boldsymbol{X}} = \left\{\boldsymbol{X}^\top\boldsymbol{v} | \boldsymbol{v} \in \mathbb{R}^m, \ \sum \boldsymbol{v} = 1, \boldsymbol{v} \geq 0\right\}.$$

Definition 2 (Interpolation): A sample $\boldsymbol{x}$ is considered to be an interpolation point of a set of samples $\boldsymbol{X}$, given a feature space $\tilde{\boldsymbol{\Phi}}(\boldsymbol{X}; \mathbb{W}_L)$ which satisfies Assumption 1 and 2, if: $\tilde{\boldsymbol{\phi}}(\boldsymbol{x}) \in \mathbb{C}_{\tilde{\boldsymbol{\Phi}}(\boldsymbol{X};\mathbb{W}_L)}$.

Interpolation is an attribute of the input space but its definition considers the learned feature space of the neural network. Considering the feature space in Definition 2 may seem counter-intuitive but applies well to reality, where nonlinear features for regression can show arbitrary behavior between data points, even for a univariate input. In this case, interpolation points in the input domain are rightly classified as extrapolation points by considering the feature domain.

Classifying a point as interpolation or extrapolation is a binary decision. In reality this is a shortcoming, as different degrees of extrapolation are possible. That is, a point "close" to the convex hull might still lead to a trustworthy prediction. The distance to a set, e.g. the convex hull, is defined below.

Definition 3 (Distance): For a set $\mathbb{X} \subset \mathbb{R}^{n_x}$ and a point $\boldsymbol{x} \in \mathbb{R}^{n_x}$ we define the distance $d(\boldsymbol{x}, \mathbb{X})$ as:

$$d(\boldsymbol{x}, \mathbb{X}) = \inf_{a\in\mathbb{X}} \|x - a\|_2^2. \tag{31}$$

A. QUANTIFICATION OF INTERPOLATION AND EXTRAPOLATION

In the following, we seek to define a metric to quantify the degree of extrapolation for a nonlinear regression model with feature space. Intuitively, such a metric could be based on the distance, according to Definition 3, to the convex hull of the features. Unfortunately, the distance to the convex hull results in an optimization problem that scales with the number of samples and the feature dimension, which can be a limitation for practical applications. Instead, we propose an approximate metric for which we introduce the *affine cost*. As related concepts of the affine cost, we define the well known *span* and *affine hull*.

Definition 4 (Span): The span of a set of samples $\boldsymbol{X} \in \mathbb{R}^{m\times n_x}$ is defined as the set:

$$\mathbb{S}_{\boldsymbol{X}} = \left\{\boldsymbol{X}^\top\boldsymbol{v} | \boldsymbol{v} \in \mathbb{R}^m\right\}.$$

Definition 5 (Affine Hull): The affine hull of a set of samples $\boldsymbol{X} \in \mathbb{R}^{m\times n_x}$ is defined as the set:

$$\mathbb{A}_{\boldsymbol{X}} = \left\{\boldsymbol{X}^\top\boldsymbol{v} | \boldsymbol{v} \in \mathbb{R}^m, \ \sum \boldsymbol{v} = 1\right\}.$$

Definition 6 (Affine Cost): The affine cost of a test point $\boldsymbol{x} \in \mathbb{R}^{n_x}$ and given the data $\boldsymbol{X} \in \mathbb{R}^{m\times n_x}$ is defined as the optimal cost:

$$c_{\mathbb{A}_{\boldsymbol{X}}}(\boldsymbol{x}) = \underset{\boldsymbol{v},\boldsymbol{e}}{\text{minimize}} \quad \|\boldsymbol{v}\|_2^2 + \gamma\|\boldsymbol{e}\|_2^2 \tag{32a}$$

$$\text{subject to :} \quad \boldsymbol{X}^\top\boldsymbol{v} + \boldsymbol{e} = \boldsymbol{x}, \tag{32b}$$

$$\sum \boldsymbol{v} = 1, \tag{32c}$$

where the spanning coefficient $\boldsymbol{v} \in \mathbb{R}^m$ and the residual variable $\boldsymbol{e} \in \mathbb{R}^{n_x}$ are optimization variables and $\gamma \in \mathbb{R}$ is a weighting factor for the residuals.

The affine cost naturally complements the definition of the affine hull by computing the norm of the respective spanning coefficients $\boldsymbol{v}$ from Definition 5. Importantly, test values

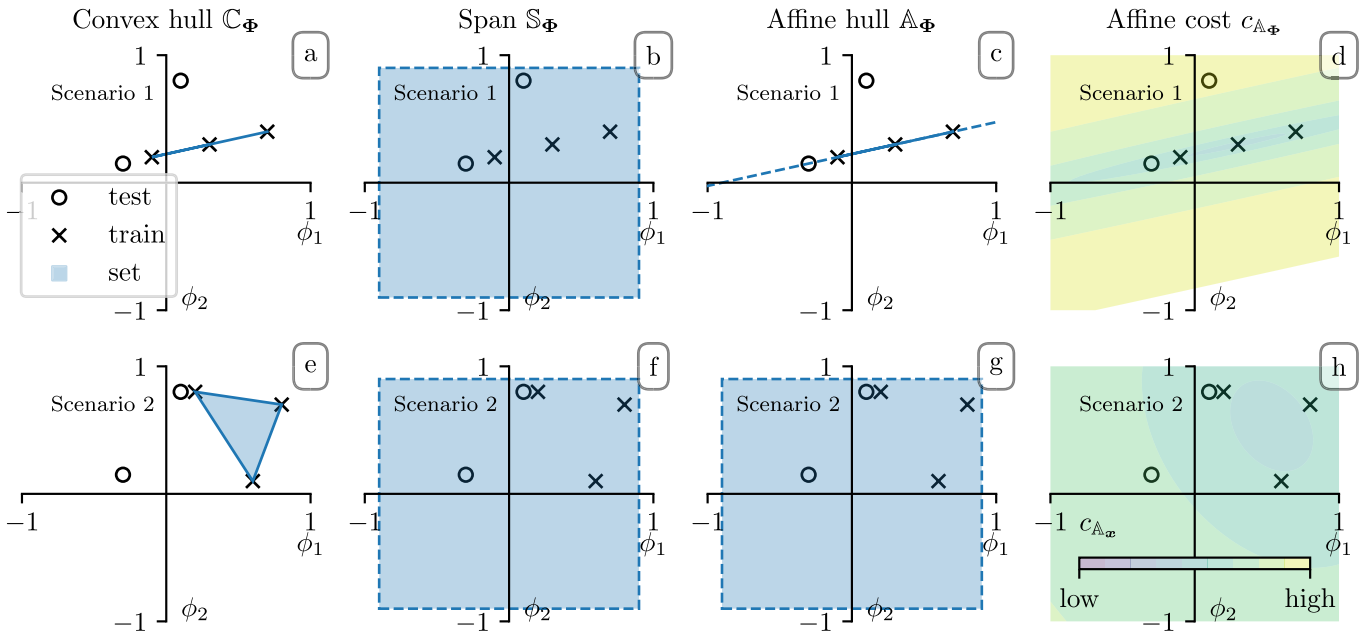

FIGURE 1. Comparison of convex hull (Definition 1), span (Definition 4), affine hull (Definition 5) and affine cost (Definition 6) for two exemplary sets of features $\tilde{\Phi} \in \mathbb{R}^{m \times n_{\tilde{\phi}}}$, both with $m = 3$ and $n_{\tilde{\phi}} = 2$.

$\boldsymbol{x} \notin \mathbb{A}_X$ also have a value assigned, for which the γ-weighted norm of the residuals $\boldsymbol{e}$ is considered.

The relationship of convex hull, span and affine hull as well as the affine cost can be inspected in Figure 1. In this figure, two exemplary sets of features $\tilde{\boldsymbol{\Phi}} \in \mathbb{R}^{m \times n_{\tilde{\phi}}}$, both with $m = 3$ and $n_{\tilde{\phi}} = 2$ are presented and we compare the relationship of test to training samples.

By comparing, for example Figure 1a) and d), we can see that the affine cost has a strong relationship with the convex hull on which we have based the notion of interpolation in Definition 2. In particular, we consider the level set

$$\mathbb{L}_{\mathbb{A}_{\tilde{\Phi}}} = \{\tilde{\boldsymbol{\phi}} \in \mathbb{R}^{n_{\tilde{\phi}}} | c_{\mathbb{A}_{\tilde{\Phi}}}(\tilde{\boldsymbol{\phi}}) \leq l\},$$

obtained with the affine cost and suitable level l as a soft approximation of the convex hull. Such level sets can be seen in Figure 1 d) and h) as the contour lines of $c_{\mathbb{A}_{\tilde{\Phi}}}$. It holds that for a test point $\tilde{\boldsymbol{\phi}}$, the affine cost grows with the distance to the level set $\mathbb{L}_{\mathbb{A}_{\tilde{\Phi}}}$. In this sense, we consider the distance $d(\tilde{\boldsymbol{\phi}}, \mathbb{L}_{\mathbb{A}_{\tilde{\Phi}}})$ and, more directly, the affine cost $c_{\mathbb{A}_{\tilde{\Phi}}}(\tilde{\boldsymbol{\phi}})$ itself as the desired metric for the degree of extrapolation.

For the behavior of this metric we distinguish two important cases. In the first case, small values of the affine cost are achieved for test points that are within the affine hull, i.e. $\tilde{\boldsymbol{\phi}} \in \mathbb{A}_{\tilde{\Phi}}$, and for small distances to the convex hull. According to Definition 2, these include all interpolation points and what we consider mild extrapolation. Both test points in Figure 1 h) are examples for this case.

In the second case, a test point is not within the affine hull $\tilde{\boldsymbol{\phi}} \notin \mathbb{A}_{\tilde{\Phi}}$. The affine cost is now influenced primarily through the parameter γ. This can be seen by inspection of (32), where the residuals $\boldsymbol{e}$ must be used if $\tilde{\boldsymbol{\phi}} \notin \mathbb{A}_{\tilde{\Phi}}$. The cost may then be dominated by the term $\|\boldsymbol{e}\|_2^2$ which is weighted with γ. We argue that in the second case the test point can be considered an extrapolation point and γ can be interpreted as a penalty for extrapolation. This case can be observed in Figure 1 d) for the test point with higher affine cost.

B. RELATIONSHIP OF AFFINE COST AND COVARIANCE

As another main contribution of this work, we introduce Theorem 2 to establish the relationship of the BLL covariance and the previously presented affine cost.

Theorem 2 (BLL Affine Cost): If Assumption 4 holds, and with $\gamma = \alpha$, the affine cost $c_{\mathbb{A}_{\tilde{\Phi}}}(\tilde{\boldsymbol{\phi}}(\boldsymbol{x}))$, according to Definition 6, is equivalent to the scaled BLL covariance (8c):

$$c_{\mathbb{A}_{\tilde{\Phi}}}(\tilde{\boldsymbol{\phi}}(\boldsymbol{x})) = \sigma_e^{-2} \boldsymbol{\Sigma}_{\hat{y}}^{NN}(\boldsymbol{x}), \tag{33}$$

where $\boldsymbol{\phi}$ and $\tilde{\boldsymbol{\phi}}$ describe the features obtained at the last internal layer of the neural network as defined in (4).

Proof: We reformulate the equality constraints of $c_{\mathbb{A}_{\tilde{\Phi}}}(\tilde{\boldsymbol{\phi}})$ shown in (32):

$$\begin{bmatrix} \tilde{\boldsymbol{\Phi}}^\top & \boldsymbol{I}_{n_{\tilde{\phi}}} \\ \mathbf{1}_{1,n_m} & \mathbf{0}_{1,n_{\tilde{\phi}}} \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{e} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{\phi}} \\ 1 \end{bmatrix}, \tag{34}$$

where $\mathbf{0}$ and $\mathbf{1}$ denote matrices filled with zeros or ones and their respective dimensions are given in the subscript. Considering also (4), we then introduce:

$$\boldsymbol{\Phi}^\top = \begin{bmatrix} \tilde{\boldsymbol{\Phi}}^\top \\ \mathbf{1}_{1,n_m} \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} \boldsymbol{I}_{n_{\tilde{\phi}}} \\ \mathbf{0}_{1,n_{\tilde{\phi}}} \end{bmatrix}, \quad \boldsymbol{p} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{e} \end{bmatrix} \text{ and } \boldsymbol{\phi} = \begin{bmatrix} \tilde{\boldsymbol{\phi}} \\ 1 \end{bmatrix},$$

which allows to reformulate Equation (34) as:

$$\begin{bmatrix} \boldsymbol{\Phi}^\top & \boldsymbol{M} \end{bmatrix} \boldsymbol{p} = \boldsymbol{\phi}. \tag{35}$$

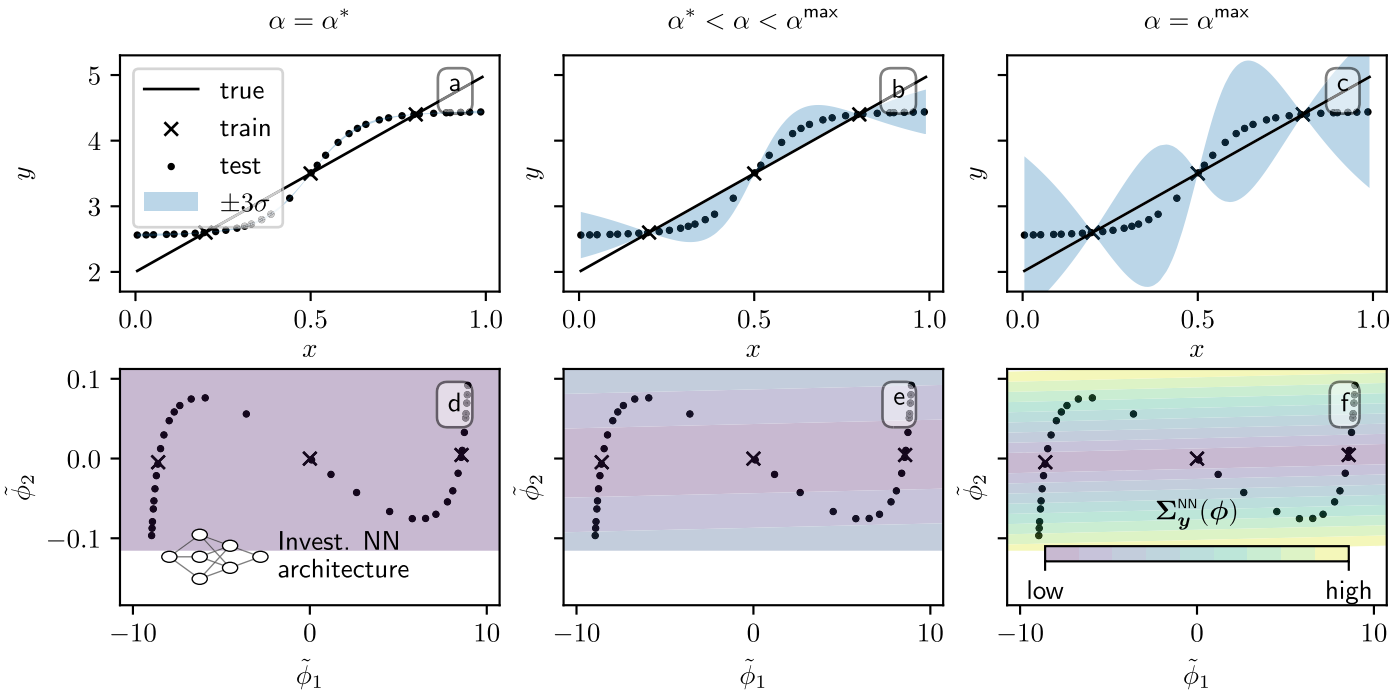

FIGURE 2. NN with BLL: Predicted mean and standard deviation (8) and feature space with $n_{\tilde{\phi}} = 2$ for $m = 3$ training samples. The effect of parameter α on the extrapolation uncertainty is shown by comparing the optimal α^* (maximization of LML (30)) with suggested improved $\alpha^{\max}$.

By further introducing $\boldsymbol{D} = \left[\boldsymbol{\Phi}^\top \; \boldsymbol{M}\right]$ and $\boldsymbol{W} = \operatorname{diag}(\boldsymbol{I}_m, \; \gamma \boldsymbol{I}_{n_{\tilde{\phi}}})$, Problem (32) can be stated as:

$$c_{\mathbb{A}_X}(\tilde{\boldsymbol{\phi}}) = \underset{\boldsymbol{p}}{\text{minimize}} \quad \|\boldsymbol{p}\|_{\boldsymbol{W}}^2 \qquad \text{subject to :} \quad \boldsymbol{D}\boldsymbol{p} = \boldsymbol{\phi}. \tag{36}$$

We have a weighted least-squares problem in (36) for which the solution can be obtained as:

$$\boldsymbol{p}^* = \boldsymbol{\phi}^\top (\boldsymbol{D}\boldsymbol{W}^{-1}\boldsymbol{D}^\top)^{-1} \boldsymbol{D}\boldsymbol{W}^{-1}. \tag{37}$$

Substituting $\boldsymbol{p}^*$ from (37) into the cost function of (36) yields the affine cost:

$$c_{\mathbb{A}_X}(\tilde{\boldsymbol{\phi}}) = \boldsymbol{\phi}^\top (\boldsymbol{D}\boldsymbol{W}^{-1}\boldsymbol{D}^\top)^{-1} \boldsymbol{\phi}, \tag{38}$$

where

$$\begin{aligned} \boldsymbol{D}\boldsymbol{W}^{-1}\boldsymbol{D}^\top &= \begin{bmatrix} \boldsymbol{\Phi}^\top & \boldsymbol{M} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_m & \\ & \gamma^{-1}\boldsymbol{I}_{n_{\tilde{\phi}}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{M}^\top \end{bmatrix} \\ &= \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \gamma^{-1} \tilde{\boldsymbol{I}}_{n_\phi}. \end{aligned} \tag{39}$$

Considering that $\gamma = \alpha$, as stated in the theorem, and (28)-(29), we have:

$$\boldsymbol{D}\boldsymbol{W}^{-1}\boldsymbol{D}^\top = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \alpha^{-1} \tilde{\boldsymbol{I}}_{n_\phi} = \bar{\boldsymbol{\Lambda}}_p = \sigma_e^2 \boldsymbol{\Lambda}_p, \tag{40}$$

where $\tilde{\boldsymbol{I}}_{n_\phi} = \operatorname{diag}(\boldsymbol{I}_{n_{\tilde{\phi}}}, \; 0)$. Substituting (40) in (38) yields:

$$c_{\mathbb{A}_X}(\tilde{\boldsymbol{\phi}}(\boldsymbol{x})) = \sigma_e^{-2} \boldsymbol{\phi}^\top \boldsymbol{\Lambda}_p^{-1} \boldsymbol{\phi} = \sigma_e^{-2} \boldsymbol{\Sigma}_{\hat{y}}^{\text{NN}}(\boldsymbol{x}), \tag{41}$$

which concludes the proof. □

Theorem 2 establishes the theoretical relationship between affine cost and BLL covariance. In the next subsection, we discuss how this relationship can be used to improve the extrapolative uncertainty of NNs with BLL.

C. IMPROVING THE UNCERTAINTY QUANTIFICATION WITH THE AFFINE COST INTERPRETATION

The first practical implication of Theorem 2 is that a NN with BLL should indeed be trained by maximizing the marginal likelihood as discussed in Section III. Using the features of a trained NN and then applying BLR, as previously shown in [4] and [24] might lead to suboptimal performance. The reason for this is the log det-regularization of the precision matrix $\boldsymbol{\Lambda}_p$ which arises only in the marginal likelihood cost formulation. This regularization encourages a low rank of the feature matrix $\tilde{\boldsymbol{\Phi}}$, resulting in a proper subspace for the affine hull, that is, $\mathbb{A}_{\tilde{\boldsymbol{\Phi}}} \subset \mathbb{R}^{n_{\tilde{\phi}}}$. Only in this setting can we potentially obtain test points with $\tilde{\boldsymbol{\phi}}(\boldsymbol{x}) \notin \mathbb{A}_{\tilde{\boldsymbol{\Phi}}}$, clearly indicating extrapolation through high values of the affine cost. The effect can be seen in Figure 1 by comparing subplot d) and h). The application of log det-regularization to obtain matrices with low rank is well known [27] and also applied in other fields such as compressed sensing [28].

The second important implication of Theorem 2 is the interpretation of the parameter α (or γ respectively) in the context of the affine cost from Definition 6. The parameter directly controls the affine cost and thus the variance for test points $\tilde{\boldsymbol{\phi}}(\boldsymbol{x}) \notin \mathbb{A}_{\tilde{\boldsymbol{\Phi}}}$. This causes a dilemma: Naturally, we have the situation that all training samples $\tilde{\boldsymbol{\Phi}}$ are within the affine hull of themselves. Therefore, extrapolation in the sense of $\tilde{\boldsymbol{\phi}}(\boldsymbol{x}) \notin \mathbb{A}_{\tilde{\boldsymbol{\Phi}}}$ does not occur during training and α^*, which maximizes the LML, might not yield desirable results.

To illustrate the issue we present a simple regression problem with $n_x = n_y = 1$ and $m = 3$ samples. We investigate a NN with $n_{\tilde{\phi}} = 2$ which allows for a

graphical representation of the feature space. The NN is trained by maximizing the LML (30), yielding the optimal parameters $\boldsymbol{\Theta}^*$, which includes α^*. The predicted mean and standard deviation for the trained model using α^* can be seen in Figure 2 a). The predicted mean of the NN, in light of the sparse training data, is suitable. However, the variance shows that the prediction is overconfident. We show in Figure 2 b) and c) that this overconfidence can be tackled simply by increasing α relative to the optimal value α^*.

The reason for this effect of α on the extrapolation uncertainty can be seen by inspecting Figure 2 d)-f), where the features (recall $n_{\tilde{\phi}} = 2$) for the test ($\tilde{\boldsymbol{\phi}}$) and the training points ($\tilde{\boldsymbol{\Phi}}$) are displayed.

As desired, we have that $\mathbb{A}_{\tilde{\boldsymbol{\Phi}}} \subset \mathbb{R}^{n_{\tilde{\phi}}}$, that is, the training features lie in a subspace of $\mathbb{R}^2$, which can be seen in Figure 2 where the training samples in the feature space could be connected by a straight line. This effect can be attributed to the log det-regularization in the LML. Extrapolation thus occurs for $\tilde{\boldsymbol{\phi}} \notin \mathbb{A}_{\tilde{\boldsymbol{\Phi}}}$ and for these points the extrapolative uncertainty grows with increasing α. Importantly, by considering Definition 2, we also have extrapolation for test points that are within the convex hull of the input space, i.e. $\boldsymbol{x} \in \mathbb{C}_{\boldsymbol{X}}$. In this example, the only true interpolation points are the training samples for which increasing α has no significant effect on the predicted variance.

In Figure 3, we further investigate the effect of increasing α for the same regression problem and NN architecture as displayed in Figure 2. To quantify the quality of the predictive distribution, we use the log-predictive density (LPD) [29]:

$$\log \bar{p}(\boldsymbol{t}^{\text{test}}) = \frac{1}{m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \log p(\mathrm{t} = t_i^{\text{test}} | \mathcal{D}, \boldsymbol{\Theta}, \boldsymbol{x}^{\text{test}}), \tag{42}$$

which evaluates the logarithm of the posterior distribution of the targets (11) for all test values and computes the average thereof.

In Figure 3, we display log-predictive density (42) and the negative LML (30) for the optimal parameters $\boldsymbol{\Theta}^*$ and as a function of α. As expected, α^* minimizes the negative LML. We see, however, that the optimal value is almost at a plateau and that increasing α has only a minor effect on the LML. We also see in the top diagram of Figure 3 that the log-predictive density for the training points remains independent of α. On the other hand, we see that the log-predictive density for test points can be increased significantly by setting $\alpha = \alpha^{\max}$. Further increasing α has a negative effect as the predicted posterior distribution becomes increasingly flat.

As a consequence of this investigation we propose Algorithm 1 to obtain a NN with BLL, trained with LML maximization, and enhanced extrapolation uncertainty.

The optimal value for the scalar parameter α in Algorithm 1 can be obtained with a simple bijection approach.

Algorithm 1 LML Optimization With Adapted Extrapolation Penalty α for Enhanced BLL

Require: $\mathcal{D}^{\text{train}}$, $\mathcal{D}^{\text{val}}$
Require: NN structure $(L, n_{a_l}, g_l(\cdot)\ \forall l \in \mathbb{I}_{[1,L]})$
$\boldsymbol{\Theta}^* \leftarrow \arg\min_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta}; \mathcal{D}^{\text{train}})$ ▷ solve (30)
$\alpha^{\max} \leftarrow \arg\max_{\alpha} \log \bar{p}(\boldsymbol{t}^{\text{val}})$ ▷ solve (42)

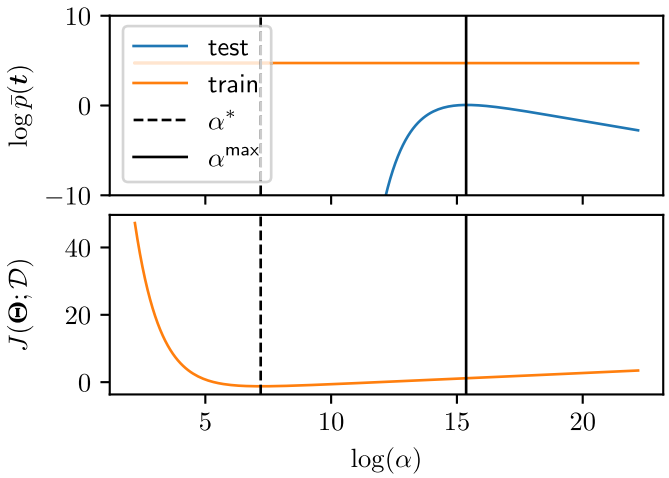

FIGURE 3. Effect of log(α) on the LML for a trained NN and the mean log-predictive density (42). The same regression problem and NN as in Figure 2 is considered.

V. THE MULTIVARIATE CASE

In general, we seek to investigate multivariate problems where, in contrast to Section II, we now have that $\boldsymbol{t} \in \mathbb{R}^{n_y}$. As before, we consider a dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{T}\}$ consisting of m samples and introduce $\boldsymbol{T} = [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m]^\top \in \mathbb{R}^{m \times n_y}$.

To obtain the setting in which Lemma 1 holds, we augment (5) for the multivariate case:

$$\boldsymbol{T} = \boldsymbol{\Phi}\boldsymbol{W} + \boldsymbol{E}, \tag{43}$$

where $\boldsymbol{E} \in \mathbb{R}^{m \times n_y}$ is the matrix of residuals of the regression model. We use the vec($\cdot$) operation as defined in [30, Defn. 11.5] and vectorize (43):

$$\text{vec}(\boldsymbol{T}) = \text{vec}(\boldsymbol{\Phi}\boldsymbol{W}) + \text{vec}(\boldsymbol{E}). \tag{44}$$

Introducing $\otimes$ as the Kronecker product, this equation can be reformulated as [30, Prop. 11.16(b)]:

$$\text{vec}(\boldsymbol{T}) = (\boldsymbol{I}_{n_y} \otimes \boldsymbol{\Phi})\text{vec}(\boldsymbol{W}) + \text{vec}(\boldsymbol{E}), \tag{45}$$

and expressed as:

$$\hat{\boldsymbol{t}} = \hat{\boldsymbol{\Phi}}\hat{\boldsymbol{w}} + \hat{\boldsymbol{\epsilon}}. \tag{46}$$

In the form of (46), Lemma 1 directly applies to the multivariate case where the prior and noise covariances from Assumption 2 and 3 now refer to:

$$\hat{\boldsymbol{\epsilon}} \sim \mathcal{N}(0, \hat{\boldsymbol{\Sigma}}_E),$$
$$\hat{\boldsymbol{w}} \sim \mathcal{N}(0, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{W}}),$$

and for which we need to consider the multivariate feature matrix $\hat{\boldsymbol{\Phi}}$.

A. SIMPLIFIED TRAINING

The multivariate settings adds significant complexity to the marginal likelihood maximization introduced in Section III. To reduce the computational complexity, we present Result 3 and two required assumptions in the following.

Assumption 5: Noise and prior, as introduced in Assumption 2 and 3, are uncorrelated for all n_y outputs and identical for all m samples and all n_ϕ features, respectively. We denote $\boldsymbol{\sigma}_e = \left[\sigma_{e,1}, \ \ldots, \ \sigma_{e,n_y}\right]$ and $\boldsymbol{\sigma}_w = \left[\sigma_{w,1}, \ \ldots, \ \sigma_{w,n_y}\right]$ and obtain:

$$\hat{\boldsymbol{\Sigma}}_E = \operatorname{diag}(\boldsymbol{\sigma}_e) \otimes \boldsymbol{I}_m, \tag{47}$$

$$\hat{\boldsymbol{\Sigma}}_W = \operatorname{diag}(\boldsymbol{\sigma}_w) \otimes \boldsymbol{I}_{n_\phi}. \tag{48}$$

We introduce $\boldsymbol{\alpha} = \frac{\boldsymbol{\sigma}_w^2}{\boldsymbol{\sigma}_e^2}$, similarly to (27), and assume the following.

Assumption 6: For all predicted outputs y_i, we have the same parameter α, i.e.:

$$\alpha_1 = \cdots = \alpha_{n_y} \overset{!}{=} \alpha. \tag{49}$$

Result 3: Let Assumption 5 and 6 hold. The scaled negative LML in (12) for the vectorized multivariate case (46) is:

$$\begin{aligned} J(\boldsymbol{\Theta}; \mathcal{D}) = \frac{n_y}{2m} &\left(m \log(2\pi) + n_\phi \log(\alpha) + \log\det(\bar{\boldsymbol{\Lambda}}_p)\right) \\ &+ \sum_{i=1}^{n_y} \left(\log(\sigma_{e,i}) + \frac{1}{2m} \sigma_{e,i}^{-2} \|\boldsymbol{t}_i - \boldsymbol{y}_i\|_2^2 \right) \\ &+ \sum_{i=1}^{n_y} \left(\frac{1}{2m} \alpha^{-1} \sigma_{e,i}^{-2} \|\bar{\boldsymbol{w}}_i\|_2^2 \right), \end{aligned} \tag{50}$$

with $\boldsymbol{\Theta} = \left\{\mathbb{W}_{L+1}, \alpha, \sigma_{e,1}, \ldots, \sigma_{e,n_y}\right\}$ and $\bar{\boldsymbol{\Lambda}}_p$ according to (29). The full precision matrix $\hat{\boldsymbol{\Lambda}}_p$ for the multivariate case can be obtained as:

$$\hat{\boldsymbol{\Lambda}}_p = \operatorname{diag}(\boldsymbol{\sigma}_e^{-2}) \otimes \bar{\boldsymbol{\Lambda}}_p. \tag{51}$$

Proof: The result follows directly from the properties of the Kronecker product [30] applied to the log-marginal likelihood in (12) with (46), as well as Assumption 5 and 6. □

Result 3 shows that the LML can be easily expressed for the multivariate case, allowing for fast and efficient NN training. This is only possible due to Assumption 5 and 6. While Assumption 5 is a natural extension of Assumption 4 for the multivariate case, Assumption 6 might be questioned. We argue again with the interpretation of α as an extrapolation penalty weight, as discussed in Section IV. Importantly, extrapolation, as defined in Definition 2, is a property of the feature space and occurs regardless of the number of outputs.

Apart from simplifying the LML in (50), Assumption 5 and 6 also yield a simplified computation of the predictive distribution. The outputs are uncorrelated due to Assumption 5 and we can obtain independent covariance matrices:

$$\boldsymbol{\Sigma}_{\hat{y}_i}^{\mathrm{NN}}(\boldsymbol{x}) = \boldsymbol{\phi}^\top \boldsymbol{\Lambda}_{p,i}^{-1} \boldsymbol{\phi}, \quad \text{with: } \boldsymbol{\Lambda}_{p,i} = \sigma_{e,i}^{-2} \bar{\boldsymbol{\Lambda}}_p, \tag{52}$$

where we have the same $\bar{\boldsymbol{\Lambda}}_p$ for all outputs due to Assumption 6. Therefore, the complexity of evaluating the predictive distribution scales negligibly with the number of outputs. This is a major advantage of NNs with BLL, in comparison to GPs, where it is common to fit an independent model for each output.

VI. BAYES BY BACKPROP

As a comparative baseline to our proposed method, we also employ variational inference with the *Bayes by Backprop* [1], [21] method to train a full BNN, that is, a neural network in which all weights follow a probability distribution.

A. BACKGROUND

For the formulation of the NN with BLL we require the posterior distribution of the weights in the last layer in (7). As a main difference, we now state the posterior distribution for all weights of the NN, that is:

$$p(\mathbb{W}_{L+1}|\mathcal{D}, \boldsymbol{\Theta}) = \frac{p(\mathcal{D}|\mathbb{W}_{L+1}, \boldsymbol{\Theta}) p(\mathbb{W}_{L+1}|\boldsymbol{\Theta})}{p(\mathcal{D}|\boldsymbol{\Theta})}. \tag{53}$$

In contrast to NNs with BLL, the exact posterior distribution is intractable and we resort to variational inference. To this end, we introduce a surrogate distribution $q(\mathbb{W}_{L+1})$ and minimize the Kullback-Leibler (KL) divergence between the surrogate and the true posterior:

$$\begin{aligned} &D_{\mathrm{KL}}(q(\mathbb{W}_{L+1}) \parallel p(\mathbb{W}_{L+1}|\mathcal{D}, \boldsymbol{\Theta})) \\ &\quad = \mathbb{E}_{q(\mathbb{W}_{L+1})}\left[\log \frac{q(\mathbb{W}_{L+1})}{p(\mathbb{W}_{L+1}|\mathcal{D}, \boldsymbol{\Theta})}\right]. \end{aligned} \tag{54}$$

Minimizing the KL-divergence yields the surrogate distribution of the weights that approximates the true posterior distribution. As proposed in [21], we choose the surrogate distribution as:

$$q(\operatorname{vec}(\boldsymbol{W}_l)) = \mathcal{N}(\boldsymbol{\mu}_{w,l}, \operatorname{diag}(\boldsymbol{\sigma}_{w,l}^2)) \quad \forall l \in \mathbb{I}_{[1,L+1]}, \tag{55}$$

with trainable parameters $\boldsymbol{\mu}_{w,l}$ and $\boldsymbol{\sigma}_{w,l}$. Consequently, each weight of the neural network is now represented with two parameters, which makes Bayes by Backprop a tractable method even for larger models.

For the concrete implementation of training the BNN with Bayes by Backprop, we refer to [1]. In particular, we use Bayes by Backprop in combination with empirical Bayes to obtain the optimal parameters $\boldsymbol{\Theta} = \{\boldsymbol{\sigma}_w, \boldsymbol{\sigma}_e\}$ which, similarly to the NN with BLL, include the prior variances of the weights in the last layer, that is, $\boldsymbol{\sigma}_w$ and the noise covariances $\boldsymbol{\sigma}_e$. Following the discussion in [21], we assume a fixed prior for the weights in the previous layers.

B. EVALUATION

The predictive distribution of the trained BNN can be evaluated as:

$$p(\mathbf{t}|\mathcal{D}, \boldsymbol{\Theta}, \boldsymbol{x}) = \mathbb{E}_{q(\mathbb{W}_{L+1})}\left[p(\mathbf{t}|\boldsymbol{x}, \mathbb{W}_{L+1})\right]. \tag{56}$$

In contrast to predictive distribution of the NN with BLL in (11), expression (56) has no analytical solution. Instead,

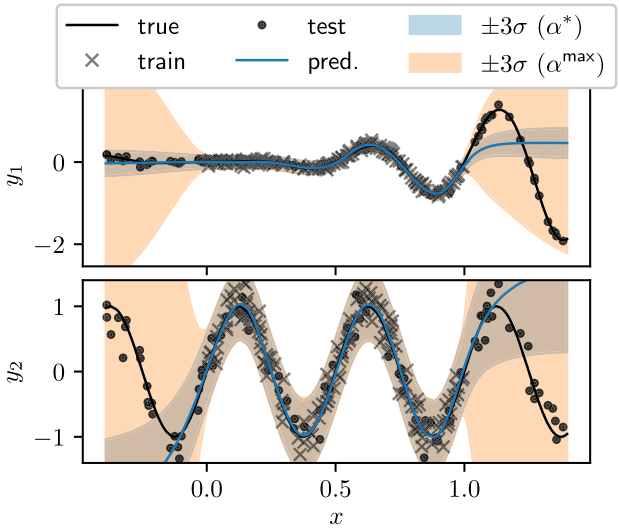

FIGURE 4. Neural network with Bayesian last layer: Mean and standard deviation for two outputs with unknown noise levels. Comparison of the distributions with optimal α^* and $\alpha^{\max}$ from Algorithm 1.

TABLE 1. Simulation study in Figure 4 and 5: Comparison of log-predictive density (LPD) for a NN with BLL (minimized LML), BLR with NN features (minimized MSE) and a BNN (trained with variational inference).

		LPD (↑ is better)	
		train	test
NN w. BLL	α^*	0.42	-18.14
NN w. BLL	$\alpha^{\max}$	0.41	**-1.83**
BLR w. NN features	α^*	0.29	-19.20
BLR w. NN features	$\alpha^{\max}$	0.29	-1.95
BNN w. VI [21]		0.00	-2.30

we resort to Monte Carlo sampling of the weights from the surrogate distribution and obtain the predictive distribution by averaging over the samples, that is:

$$p(\mathbf{t}|\mathcal{D}, \boldsymbol{\Theta}, \boldsymbol{x}) \approx \frac{1}{N}\sum_{i=1}^{N} p(\mathbf{t}|\boldsymbol{x}, \mathbb{W}_{L+1}^{(i)}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{N}(\bar{\mathbf{y}}^{(i)}, \boldsymbol{\Sigma}_E), \tag{57}$$

with $\mathbb{W}_{L+1}^{(i)} \sim q(\mathbb{W}_{L+1})$. Equation (57) yields a Gaussian mixture model and also allows for the computation of the log-predictive density, similarly to (42).

VII. SIMULATION STUDY

In this section, we investigate our proposed method in comparison to BLR with features from a trained NN, and a full BNN trained with Bayes by Backprop. The code to reproduce the presented results is available online[1].

[1]https://github.com/4flixt/2023_Paper_BLL_LML

We investigate data from a function with $n_x = 1$ input and $n_y = 2$ outputs, which is shown in Figure 4. Both outputs exhibit non-linear behavior with different noise-levels, in particular $\sigma_1 = 0.05$ and $\sigma_2 = 0.2$. Both noise variances are assumed to be unknown.

We train a NN with BLL using our proposed Algorithm 1. For the study, a suitable structure with $L = 2$, $n_{\tilde{\phi}_1} = n_{\tilde{\phi}_2} = n_{\tilde{\phi}_3} = 20$ and $g_1(\cdot) = g_2(\cdot) = g_3(\cdot) = \tanh(\cdot)$ is determined using trial and error. To maximize the LML, we use the Adam [31] optimizer and employ early-stopping to avoid overfitting.

The results of the simulation study are shown in Figure 4. We visualize mean and standard deviation of the predicted distribution obtained from the NN with BLL. A focus of the investigation is the comparison of the distributions obtained with optimal value α^* and $\alpha^{\max}$. Both distributions suitably describe the training data and capture the behavior of the unknown function in the interpolation regime. The estimated noise variance $\sigma_1 = 0.051$ and $\sigma_2 = 0.17$ is close to the true value. However, only the distribution with tuned extrapolative uncertainty, that is, with $\alpha^{\max}$, is suitable to describe the extrapolation regime. To quantify the improvement, we compute the log-predictive density of the NN with BLL for α^* and $\alpha^{\max}$, and present the results in Table 1. For comparison, we train a NN with the same structure by minimizing the mean-squared-error and then use the learned features for BLR as in previous works [4], [24]. The second step of Algorithm 1, that is, updating α subsequently with validation data, can also be applied in this setting and the resulting performance metric is also shown in Table 1.

The results in Table 1 show that updating α, as proposed in Algorithm 1, significantly improves the log-predictive density (42) of the test data, both for BLL and BLR. We see that the NN with BLL, which is trained according to Algorithm 1, achieves overall the best results with respect to the log-predictive density.

A. COMPARISON TO BAYES BY BACKPROP

As a final step of our evaluation, we compare the NN with BLL to a full BNN trained with Bayes by Backprop, as described in Section VI. We investigate the same neural network architecture as described in the previous subsection and, as a main difference, now have distributed weights in all layers. We determine a fixed prior distribution for the weights in all but the last layer as:

$$p(\mathrm{vec}(\boldsymbol{W}_{l,0})) = \mathcal{N}(0, 0.5\boldsymbol{I}) \quad \forall l \in \mathbb{I}_{[1,L]}. \tag{58}$$

Setting the mean of the prior distribution to zero is justified by employing batch normalization [32] after each variational layer. As for the NN with BLL, the prior variance of the weights in the last layer is also learned.

Training the BNN with Bayes by Backprop is performed as outlined in Section VI and using the Adam optimizer. The resulting predictive distribution is obtained by sampling

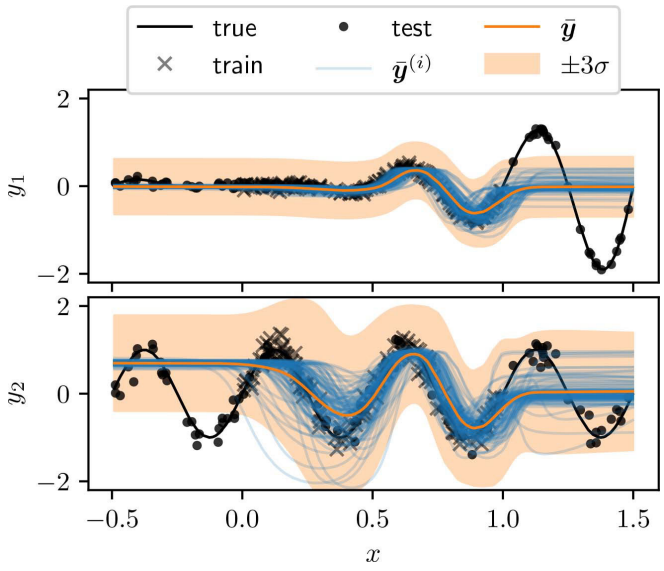

FIGURE 5. Variational inference for a BNN with Bayes by Backprop. Sampled ($N = 100$) predictive distribution with (57).

$N = 100$ weights from the surrogate distribution and displayed in Figure 5. We show the mean values $\bar{y}^{(i)}$ that are obtained with the sampled weights $\mathbb{W}_{L+1}^{(i)} \sim q(\mathbb{W}_{L+1})$. Additionally, we compute mean and variance of the resulting Gaussian mixture model in (57) and display them in the same figure. Qualitatively, the resulting distribution suitably describes the training data. However, the predicted variance does not increase significantly in the extrapolation regime. For a quantitative comparison, we compute the log-predictive density of the Gaussian mixture model and report the results in Table 1. For test data, the BNN trained with Bayes by Backprop achieves a higher LPD than the NN with BLL and the BLR with NN features, both with optimal value α^*. However, the LPD of these methods significantly exceeds the results of the BNN after tuning α with additional validation data, as proposed in Algorithm 1. With the resulting $\alpha^{\max}$, the overall highest LPD on test data is achieved with the NN with BLL. For a BNN trained with Bayes by Backprop it is not possible to modify α to calibrate the extrapolation behavior of the model. Furthermore, it is necessary to sample weights from the surrogate posterior to obtain the predictive distribution, adding significant complexity to this method. This is in contrast to the NN with BLL, where the predictive distribution can be computed analytically.

VIII. CONCLUSION

Neural networks with Bayesian last layer are an attractive compromise between tractability and expressiveness in the field of Bayesian neural networks. As a main contribution of this work, we propose an efficient algorithm for training neural networks with Bayesian last layer, based on a reformulation of the marginal likelihood. Importantly, we show that our reformulation satisfies the conditions of optimality for the same argument as the original formulation. The reformulated marginal likelihood can be efficiently minimized with gradient descent. In future work, we seek to extend our method to also enable stochastic gradient descent.

In comparison to Bayesian linear regression with features from a previously trained neural network, we find that training on the log-marginal likelihood shows advantages in a presented simulation study. Our second contribution, an algorithm to tune the extrapolative uncertainty also shows excellent results in the simulation study. To derive this algorithm, we contribute a discussion on the relationship of the Bayesian last layer covariance and a proposed metric to quantify extrapolation. This metric intuitively relates to the definition of extrapolation which is based on the convex hull. Our proposed method also compares favorably to a full Bayesian neural network trained with Bayes by Backprop. Important advantages are the overall ease of implementation, the possibility to tune the extrapolative uncertainty, and an analytical expression for the predictive distribution.

Especially the last aspect is pivotal for our future work, the identification of probabilistic dynamic system models that can be used for safe control decisions.

REFERENCES

[1] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, "Hands-on Bayesian neural networks—A tutorial for deep learning users," *IEEE Comput. Intell. Mag.*, vol. 17, no. 2, pp. 29–48, May 2022.

[2] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Inf. Fusion*, vol. 76, pp. 243–297, Dec. 2021.

[3] J. Harrison, A. Sharma, R. Calandra, and M. Pavone, "Control adaptation via meta-learning dynamics," in *Proc. Workshop Meta-Learn. (NeurIPS)*, 2018, pp. 1–6.

[4] C. D. McKinnon and A. P. Schoellig, "Meta learning with paired forward and inverse models for efficient receding horizon control," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3240–3247, Apr. 2021.

[5] K. P. Wabersich and M. N. Zeilinger, "Nonlinear learning-based model predictive control supporting state and input dependent model uncertainty estimates," *Int. J. Robust Nonlinear Control*, vol. 31, no. 18, pp. 8897–8915, Dec. 2021.

[6] J. Dong, "Robust data-driven iterative learning control for linear-time-invariant and Hammerstein–Wiener systems," *IEEE Trans. Cybern.*, vol. 53, no. 2, pp. 1144–1157, Feb. 2023.

[7] C. D. McKinnon and A. P. Schoellig, "Learning probabilistic models for safe predictive control in unknown environments," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 2472–2479.

[8] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 6, pp. 2736–2743, Nov. 2020.

[9] C. E. Rasmussen, *Gaussian Processes in Machine Learning*. Cham, Switzerland: Springer, 2003.

[10] M. Lazaro-Gredilla and A. R. Figueiras-Vidal, "Marginalized neural network mixtures for large-scale regression," *IEEE Trans. Neural Netw.*, vol. 21, no. 8, pp. 1345–1351, Aug. 2010.

[11] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4405–4423, Nov. 2020.

[12] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep Kernel Learning," in *Proc. 19th Int. Conf. Artif. Intell. Statist.*, 2016, pp. 370–378.

[13] H. Liu, Y.-S. Ong, X. Jiang, and X. Wang, "Deep latent-variable kernel learning," *IEEE Trans. Cybern.*, vol. 52, no. 10, pp. 10276–10289, Oct. 2022.
[14] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[15] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3866–3878, Sep. 2020.
[16] L. Dong, J. Yan, X. Yuan, H. He, and C. Sun, "Functional nonlinear model predictive control based on adaptive dynamic programming," *IEEE Trans. Cybern.*, vol. 49, no. 12, pp. 4206–4218, Dec. 2019.
[17] J. Sarangapani, "System identification using discrete-time neural networks," in *Neural Network Control of Nonlinear Discrete-Time Systems*, 1st ed. Boca Raton, FL, USA: CRC Press, 2006, pp. 443–466.
[18] F. Fiedler, A. Cominola, and S. Lucia, "Economic nonlinear predictive control of water distribution networks based on surrogate modeling and automatic clustering," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 16636–16643, 2020.
[19] F. Fiedler and S. Lucia, "Model predictive control with neural network system model and Bayesian last layer trust regions," in *Proc. IEEE 17th Int. Conf. Control Autom. (ICCA)*, Jun. 2022, pp. 141–147.
[20] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 880–887.
[21] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1613–1622.
[22] J. Watson, J. A. Lin, P. Klink, J. Pajarinen, and J. Peters, "Latent derivative Bayesian last layer networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 1198–1206.
[23] C. M. Bishop, *Pattern Recognition and Machine Learning*. Cham, Switzerland: Springer, 2006.
[24] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams, "Scalable Bayesian optimization using deep neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2171–2180.
[25] R. Balestriero, J. Pesenti, and Y. LeCun, "Learning in high dimension always amounts to extrapolation," 2021, *arXiv:2110.09485*.
[26] J. M. Bernardo and A. F. Smith, *Bayesian Theory*, vol. 405. Hoboken, NJ, USA: Wiley, 1994.
[27] M. Fazel, H. Hindi, and S. P. Boyd, "Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices," in *Proc. Amer. Control Conf.*, Jun. 2003, pp. 2156–2162.
[28] W. Dong, G. Shi, X. Li, Y. Ma, and F. Huang, "Compressive sensing via nonlocal low-rank regularization," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3618–3632, Aug. 2014.
[29] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard, "Most likely heteroscedastic Gaussian process regression," in *Proc. 24th Int. Conf. Mach. Learn.*, Jun. 2007, pp. 393–400.
[30] George A. F. Seber," *A Matrix Handbook for Statisticians*. Hoboken, NJ, USA: Wiley, 2008.
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

FELIX FIEDLER received the M.Sc. degree in chemical engineering from TU Berlin University, Germany, in 2018. He was Ph.D. candidate at TU Berlin University, HEIBRiDS Graduate School program, until 2020. In the same year, he joined TU Dortmund University, Germany, where he is currently pursuing the Ph.D with the Chair of Process Automation Systems. He is also the current lead developer of the open-source do-mpc toolbox which enables rapid non-linear MPC developments in Python (www.do-mpc.com). His research interests include model predictive control under uncertainty, system identification, and machine learning.

SERGIO LUCIA (Member, IEEE) received the M.Sc. degree in electrical engineering from the University of Zaragoza, Spain, in 2010, and the Dr.-Ing. degree in optimization and automatic control from TU Dortmund University, Germany, in 2014. He joined Otto-von-Guericke Universität Magdeburg and visited the Massachusetts Institute of Technology as a Postdoctoral Fellow. From 2017 to 2020, he was an Assistant Professor with TU Berlin. Since 2020, he has been a Professor with TU Dortmund University and the Head of the Chair of Process Automation Systems. His research interests include decision-making under uncertainty, distributed control, and the interplay between machine learning techniques and control theory. He is currently an Associate Editor of the *Journal of Process Control*.