Saket Adhau

Data-Driven Control Strategies: From MPC to Reinforcement Learning

Thesis for the degree of Philosophiae Doctor
Trondheim, January 2024

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Natural Sciences
Department of Chemical Engineering

ISBN 978-82-326-1888-0 (printed version)
ISBN 987-82-326-1889-7 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2024:275

Printed by NTNU Graphic Center

To my parents

Abstract

This thesis presents a comprehensive investigation into the integration of machine learning techniques to enhance the performance and applicability of Model Predictive Control (MPC). The main focus is on addressing challenging aspects of MPC through innovative methodologies, contributing to the advancement of control strategies in complex systems. The initial focus is on solving Non-Linear Model Predictive Control (NMPC) online, which is a computationally demanding task. To mitigate this challenge and improve the real-time feasibility of NMPC, a novel supervised learning framework is proposed. The framework harnesses the potential of neural networks injected with explicit constraint knowledge, and integrating insights from Karush-Kuhn-Tucker (KKT) conditions through the use of logarithmic barrier functions in the loss function. This approach effectively approximates the complex optimization problem, providing faster solutions while maintaining a fine balance between optimality and constraint satisfaction.

In the next chapter, the thesis introduces an RL-based Economic Nonlinear MPC (ENMPC) scheme to improve closed-loop performance even when the system's model is inaccurate. This scheme employs comprehensive data-based tuning for the real system, targeting performance improvements during transient and steady-state operations. Additionally, the integration of a Real-Time Optimization (RTO) layer aids in data-based tuning of the optimal control policy. Furthermore, to address the limitations of Reinforcement Learning (RL), specifically the time and computational efforts required to learn the optimal control policy, the thesis proposes a method to approximate the action-value function from the optimal solution of the value function using nonlinear programming sensitivities. This approach significantly reduces computational efforts while maintaining comparable closed-loop performance to conventional methods.

Overall, this thesis contributes to the advancement of Model Predictive Control through the development of efficient supervised learning approximations, integration with reinforcement learning techniques, and data-based tuning strategies. The proposed methods open new avenues for enhancing control performance and overcoming computational challenges in real-world control applications.

Contents

Preface

The main motivation behind the last three years of my research is the realization that machine learning in process control is the ability to predict failures, schedule maintenance, reduce downtime and operating costs, improve efficiency by identifying bottlenecks and suboptimal operating conditions.

After completing my Master's degree in Instrumentation and Control at the College of Engineering in Pune, I worked for a short time at Aker Solutions in India. This experience provided me with valuable insight into the workflow between researchand practice and the human aspects associated with successful industrial use of control and optimization. During my time as a master's student and as a professional, I noticed that the huge chunks of valuable information is often discarded in industries. The natural question that came to me was, "How can we use this data efficiently to leverage this information to support optimization tools and improve production optimization?

With some well-motivated research questions, I decided to pursue my PhD. I was very pleased when Prof. Sigurd Skogestad (Department of Chemical Engineering, NTNU) offered me a PhD position in his Process Systems Group, NTNU. I arrived in Trondheim at the end of September 2019 and started my PhD studies. In the last four years, I have been working on different approaches for surrogate optimization using neural networks for MPC; data-driven approaches for real-time optimization, and ENMPC using reinforcement learning, which are presented in this PhD thesis.

The PhD research was carried out at the Department of Chemical Engineering, at the Norwegian University of Science and Technology (NTNU) under the supervision of Prof. Sigurd Skogestad as my main supervisor and Prof. Sébastien Gros as my co-supervisor during the period from September 2019 to January 2022. The work has been funded by the Norwegian Research Council, NTNU and major industrial partners (Project number 299585).

This thesis is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (PhD) at the Norwegian University of Science and Technology (NTNU).

Acknowledgments

First and foremost, I extend my heartfelt appreciation to Prof. Sigurd Skogestad, my thesis advisor, for granting me the privilege to contribute to his research group. His guidance, motivation, and unwavering encouragement were indispensable to the realization of this project. Prof. Skogestad's fervor for research, vast knowledge, and extensive experience have not only inspired personal and intellectual growth but also laid the foundation for this work.

I am also deeply grateful to Prof. Sébastien Gros, my co-supervisor, for his invaluable support in the latter part of my PhD journey. Sébastien not only directed me towards the right path but also provided the freedom to cultivate my own ideas. His consistent backing, genuine interest in my work, and the multitude of learning opportunities he offered have been instrumental. Working closely with Sébastien has been an honor, and I owe a significant part of the success of this project to his guidance and ideas.

A special acknowledgment goes to my fraternal brother, Vihangkumar Naik, whose constant motivation and contributions significantly impacted this work. I appreciate the fruitful collaboration, patient guidance, and the enjoyable experiences we shared in Italy and Switzerland. Vihangkumar, thank you for brightening the challenging moments of my PhD journey.

My deepest gratitude extends to Pallavi, a friend whose unwavering support, patience, and encouragement played a pivotal role in navigating this challenging yet rewarding journey. Her love has been a constant source of inspiration, providing the strength needed to overcome obstacles. I feel truly fortunate to have her as a guiding light and a constant source of motivation.

I would like to express my thanks to the members of the systems and cybernetics groups, including Allyne, Cristina, David, Lucas Ferreira, Lucas Cammann, José, Risvan, Robert, Zawadi, Dirk, Anil, Akhil, Wenqui, and Shambhu. Special appreciation goes to my mentor Prof. Dayaram Sonawane and Deepak for their motivation and unwavering support in pursuing my PhD.

I cannot overlook the importance of my furry companion, Bluebell, whose delightful presence and unwavering companionship provided moments of joy and tranquility during intense research sessions.

Lastly, I am grateful to my friends in India, particularly Utkarsh. My deepest appreciation goes to my parents and brother for their unconditional love, unwavering faith, and constant support, even in the most challenging times. This work is dedicated to them, and I am forever in debt for their invaluable contributions to my personal and academic journey.

Chapter 1

Introduction

This chapter provides a concise introduction to the motivation behind the topics explored in this thesis. It outlines the primary contributions made in this research and provides an overview of the publications presented in this work. Additionally, an outline of the thesis structure is also provided in this chapter.

1.1 Motivation

Machine learning (ML) has surged in popularity, driven by notable breakthroughs such as deep learning and the growing commercial focus on big data analytics. However, amidst this enthusiasm, skepticism from respected experts is understandable, considering past disappointments with neural networks and other AI methods. Nonetheless, recent fundamental advancements, such as the capability to train deep neural networks with numerous layers for hierarchical feature learning, could open up substantial technological and commercial prospects [1].

Model Predictive Control (MPC) utilizes an optimization-based control strategy with a receding horizon approach [2]. In MPC, a model of the real system dynamics, which may have some inaccuracies, is employed to generate an input-state sequence within a specified finite horizon. This trajectory is optimized based on a given cost function while ensuring compliance with system constraints. At each time step, the optimization problem is solved using the current system state, and the first input from the optimal solution is applied to the system. Due to the finite-horizon nature and potential discrepancies in the model, MPC typically provides a reasonable but suboptimal approximation of the optimal policy.

Due to computational constraints, the MPC scheme generally favors straightforward models. Consequently, the MPC model often lacks the necessary complexity to accurately capture the genuine system dynamics and stochastic behavior. Consequently, MPC typically provides a practical yet suboptimal approximation of the optimal policy.

In recent studies, researchers have explored the integration of Machine Learning (ML) into Model Predictive Control (MPC) to learn the system model used in the MPC scheme from data [3]. While this approach holds promise, it doesn't entirely resolve challenges associated with model inaccuracies. The effectiveness of a policy derived from an MPC system incorporating an ML-based model is inherently tied to the quality of the ML model itself. Therefore, it remains constrained by the structure and decisions made during the ML model development process.

A number of approaches to approximate optimal control law given by MPC using machine learning methods have been explored in literature. In these approaches, the control law is typically approximated using various parametric approximators often termed as imitation learning or supervised learning [3, 4, 5]. The inability to guarantee constraint satisfaction using these methods is one of the major limitations, contributing to active research in this direction.

Reinforcement Learning (RL) is a powerful paradigm in the field of machine learning that holds significant promise for various control tasks. Unlike supervised learning, RL agents learn to make sequences of decisions by interacting with an environment, receiving feedback in the form of rewards or penalties based on their actions. This trial-and-error learning approach makes RL particularly well-suited for dynamic and complex control scenarios. In industrial settings, RL is applied to optimize processes, manage resources, and enhance efficiency. For instance, in autonomous robotics, RL algorithms enable robots to learn optimal paths and actions to navigate through changing environments. In autonomous vehicles, RL can be used to develop adaptive driving policies, allowing the vehicle to respond intelligently to diverse road conditions. RL is also valuable in energy management systems, where it optimizes energy usage by learning patterns of demand and supply, thereby reducing costs and promoting sustainability. The ability of RL algorithms to learn and adapt in real-time positions them as a transformative technology in the realm of control, offering solutions to challenges that demand intelligent decision-making in dynamic, uncertain environments.

Certainly, RL proves to be a potent tool for addressing Markov Decision Process (MDP) challenges even when prior knowledge of the controlled process is lacking. Most RL methods focus on learning the optimal policy and value functions for the real system, described by an MDP, using a function approximator. It's imperative to ensure that this function approximator is sufficiently versatile to capture the optimal policy or value function of the given MDP. A prevalent choice in the RL community is to employ a Deep Neural Network (DNN) for this purpose. For example, in [6], baseline control is utilized to maintain the stability and tracking performance of an Autonomous Surface Vehicle (ASV). Simultaneously, DNN-based RL is integrated to address uncertainties and collision avoidance, showcasing the flexibility and effectiveness of using neural networks in reinforcement learning applications.

Regrettably, analyzing the closed-loop stability of a system utilizing the optimal policy derived from a Deep Neural Network (DNN) or a generic function approximation can be a complex task, as stated in [7]. Additionally, assigning meaningful initial weights to the DNN poses a significant challenge.

The concept of employing Model Predictive Control (MPC) as a function approximator for a specific Markov Decision Process (MDP) was initially introduced and substantiated in [8]. The study demonstrated that a parameterized MPC can accurately capture the optimal policy and value function of the MDP. This is achieved through modifications to the stage cost and terminal cost, even when a basic and imprecise model is utilized within the MPC framework. Subsequently, Reinforcement Learning (RL) methods like Q-learning and policy gradient techniques can be applied to adjust the parameters, aiming to attain the most favorable long-term closed-loop performance.

Despite recent publications covering the central theories related to MPC-based learning, there are lingering questions in this area. Therefore, this thesis aims to address specific application challenges by building upon previous research. These challenges involve extending prior work, applying the method to diverse engineering applications like Gas lift optimization, and highlighting its advantages in addressing complex queries within the realm of (E)MPC, RL, and MDP. The subsequent section of this chapter provides summaries of the developed theories.

To summarize, this thesis tries to address the following questions:

- **Integration of ML into MPC:** How can Machine Learning (ML) be effectively integrated into Model Predictive Control (MPC) to overcome the limitations associated with computational constraints and model inaccuracies?

- **Quality of ML Models:** What is the impact of the quality of ML models on the effectiveness of policies derived from MPC systems incorporating ML-based models? How does the ML model's structure and development decisions influence the derived policy?

- **Approximation of Optimal Control Law:** Can machine learning methods approximate the optimal control law provided by MPC, and how do these methods perform in comparison to traditional approaches?

- **Ensuring Constraint Satisfaction:** How can the challenge of guaranteeing constraint satisfaction be effectively addressed when using machine learning methods for MPC?

- **Role of RL in Control:** What is the potential role of Reinforcement Learning (RL) in addressing challenges related to Markov Decision Processes (MDP), especially in cases where prior knowledge of the controlled process is limited?

- **Optimal Policy Capture:** Can Model Predictive Control (MPC) serve as a reliable function approximator for a specific MDP, capturing the optimal policy and value function accurately?

- **Adaptive Adjustment with RL:** How effective are Reinforcement Learning (RL) methods, such as Q-learning and policy gradient techniques, in adjusting parameters to achieve favorable long-term closed-loop performance in the MPC framework?

- **Application Challenges:** How do the proposed methodologies extend to diverse engineering applications, such as gas lift optimization, and what advantages do they offer in addressing complex queries within the realms of (E)MPC, RL, and MDP?

1.2 Research Contribution and the Thesis Structure

The thesis adopts a paper-based structure, comprising five chapters. Chapter 1 outlines the thesis's motivation and objectives. The subsequent chapters delve into the research contributions accomplished during this PhD study, each detailed below.

- Constrained Neural Networks for NMPC.
  - **Saket Adhau**, Vihangkumar Naik and Sigurd Skogestad. "Constrained Neural Networks for Approximate Nonlinear Model Predictive Control". In 2021 60*th* IEEE Conference on Decision and Control (CDC) (2021), pp.295 – 300.
- RTO-RLMPC for GasLift Optimization: Practical Implementation
  - **Saket Adhau**, José Matias, Sebastien Gros, and Sigurd Skogestad."Data Driven Framework for Combining Real-Time Optimization and Economic NMPC on an Experimental Rig" In preparations.
- NLP Sensitivites for FAST RLMPC
  - **Saket Adhau**, Dirk Peter Reinhardt, Sigurd Skogestad, and Sebastien Gros. " Fast Reinforcement Learning Based MPC Using NLP Sensitivities". In IFAC World Congress 2023.
- RLMPC using Neural Networks for unknown dynamical systems
  - **Saket Adhau**, Sebastien Gros, and Sigurd Skogestad. "Reinforcement Learning based MPC with Neural Dynamical Models." Submitted to 2024 European Control Conference (ECC) (2024)

Chapter 2

Background

In this chapter, we begin by offering essential background information on Neural Networks and MDPs, a fundamental concept in Machine Learning. Neural networks play a pivotal role in Model Predictive Control (MPC) by offering adaptive and data-driven solutions to complex control problems. By leveraging historical data, neural networks can learn the intricate relationships within a system, providing accurate approximations of its dynamics.

MDP serves as a comprehensive description of real systems, ensuring their state transitions adhere to the Markov property. Subsequently, we delve into Reinforcement Learning (RL), a practical and robust technique for solving MDPs. Within this section, we explore Q-learning and policy gradient methods. Additionally, we provide an overview of Model Predictive Control (MPC) and explain the policies derived from MPC schemes. Lastly, we explore the integration of MPC and RL, introducing a pivotal theorem recently developed to delineate the primary focus of our current research.

2.1 Neural Networks

Neural networks are increasingly employed in Model Predictive Control (MPC) due to their ability to model complex, non-linear relationships in data. In the context of MPC, neural networks serve as function approximators, capturing the underlying dynamics of the system being controlled. These networks are trained on historical data to learn the relationships between input variables and system responses.

One common application of neural networks in MPC is in system modeling. Traditional MPC requires an accurate model of the system dynamics, which can be challenging to obtain, especially for complex processes. Neural networks can learn the system behavior from data, allowing MPC controllers to operate effectively even when the underlying system model is unknown or difficult to characterize.

Neural networks can also be used within the MPC framework to approximate cost functions or constraints, enabling more flexible and adaptive control strategies. By employing neural networks, MPC controllers can handle high-dimensional and non-linear systems, making them suitable for a wide range of real-world applications such as robotics, autonomous vehicles, and industrial processes.

Moreover, recent advancements in deep learning, a subset of neural networks, have further enhanced the capabilities of MPC. Deep neural networks, with multiple hidden layers, can learn intricate patterns and dependencies in data, making them well-suited for modeling complex systems and optimizing control actions.

2.1.1 Artificial Neural Networks

This subsection briefly revisits the basic principles of artificial neural networks. A feed-forward neural network is structured as a series of interconnected layers of neurons, collectively defining a mathematical function of the form $\mathcal{N} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_u}$

$$\mathcal{N}(x : \theta, M, L) = \tag{2.1}$$

$$\begin{cases} f_{L+1} \circ g_L \circ f_L \circ \cdots \circ g_1 \circ f_1(x) & \text{for } L \leq 2, \\ f_{L+1} \circ g_L \circ f_1(x), & \text{for } L = 1, \end{cases} \tag{2.2}$$

In this setup, the network takes an input x from the space $\mathbb{R}^{n_x}$ and produces an output u in $\mathbb{R}^{n_u}$. Here, M represents the number of neurons in each hidden layer, and L indicates the total number of hidden layers. When $L \leq 2$, the network is referred to as a deep neural network, and if $L = 1$, it is termed a shallow neural network. Each hidden layer comprises an affine function:

$$f_l(\xi_{l-1}) = W_l \xi_{l-1} + b_l, \tag{2.3}$$

where, $\xi_{l-1} \in \mathbb{R}^M$ is the output of the previous layer with $\xi_0 = x$. The second element of the neural network is a nonlinear activation function g_l. Rectifier linear units (ReLU) are considered as activation function, which compute the element-wise maximum between zero and the affine function of the current layer l:

$$g_l(f_l) = \max(0, f_l). \tag{2.4}$$

The parameter $\theta = \{\theta_1, \theta_2, \ldots, \theta_{L+1}\}$ contains all and biases of the affine functions of each layer,

$$\theta_l = \{W_l, b_l\} \; \forall l = 1, \ldots, L+1, \tag{2.5}$$

where the weights are

$$W_l \in \begin{cases} \mathbb{R}^{M \times n_x} \text{ if } l = 1, \\ \mathbb{R}^{M \times M} \text{ if } l = 2, \ldots, L, \\ \mathbb{R}^{n_u \times M} \text{ if } l = L+1, \end{cases} \tag{2.6}$$

and the biases are

$$b_l \in \begin{cases} \mathbb{R}^{M} \text{ if } l = 1, \ldots, L, \\ \mathbb{R}^{n_u} \text{ if } l = L+1, \end{cases} \tag{2.7}$$

2.2 Markov Decision Processes

Markov Decision Processes (MDPs) serve as a standardized framework for optimal control in discrete-time stochastic processes. In this context, the stage cost and transition probability are determined solely by the current state and input of the system. An MDP operates within defined state (S) and action (A) spaces, which can be discrete, continuous, or a combination of both. Here, ρ represents a conditional probability measure defining the system dynamics. For a specific state-action pair $(\mathbf{s}, \mathbf{a}) \in S \times A$, the subsequent state ($\mathbf{s}_+$) is distributed according to,

$$\mathbf{s}_+ \sim \rho(\cdot|\mathbf{s}, \mathbf{a}) \tag{2.8}$$

Note that (2.8) represents a generalization of classic dynamics, whether deterministic or stochastic, frequently explored within the realm of control theory, usually represented as,

$$\mathbf{s}_+ = \mathbf{F}(\mathbf{s}, \mathbf{a}, \mathbf{w}), \;\; \mathbf{w} \sim W \tag{2.9}$$

where $\mathbf{w} \in D$ is a random disturbance from distribution W and $\mathbf{F} : S \times A \times D \to S$ is a Borel-measurable function. In the special case $\mathbf{w} = 0$ yeilds deterministic dynamics. Solving an MDP is then the problem of finding the optimal policy $\boldsymbol{\pi}^\star : S \to A$ solution of:

$$\boldsymbol{\pi}^\star \in \arg\min_{\boldsymbol{\pi}} J(\boldsymbol{\pi}), \tag{2.10}$$

where $J(\boldsymbol{\pi})$ is the performance function and depends on the optimality criteria describing the MDP.

2.2.1 Discounted setting

In the discounted setting, an MDP is characterized by the triplet (L, γ, ρ), where $L : S \times A \to \mathbb{R}$ represents the stage cost, $\gamma \in (0, 1]$ is the discount factor, and the performance function $J(\boldsymbol{\pi})$ is defined as follows:

$$J(\boldsymbol{\pi}) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k L(\mathbf{s}_k, \mathbf{a}_k) \middle| \mathbf{a}_k = \boldsymbol{\pi}(\mathbf{s}_k) \right], \tag{2.11}$$

Here, the expected value operator $\mathbb{E}[\cdot]$ is computed over the (possibly) stochastic closed-loop trajectories of the system. Solving MDPs is often approached using the Bellman equations, implicitly defining the optimal value function $V^\star : S \to \mathbb{R}$ and the optimal action-value function $Q^\star : S \times A \to \mathbb{R}$ as

$$V^\star(\mathbf{s}) = \min_a Q^\star(\mathbf{s}, \mathbf{a}), \tag{2.12}$$

$$Q^\star(\mathbf{s}, \mathbf{a}) = L(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}[V^\star(\mathbf{s}_+)|\mathbf{s}, \mathbf{a}] \tag{2.13}$$

The optimal policy, denoted as $\boldsymbol{\pi}^\star(\mathbf{s})$ is determined by selecting actions that minimize $Q^\star(\mathbf{s}, \mathbf{a})$,

$$\boldsymbol{\pi}^\star(\mathbf{s}) \in \arg\min_a Q^\star(\mathbf{s}, \mathbf{a}) \tag{2.14}$$

2.3 Reinforcement Learning

As previously mentioned, solving an MDP involves determining an optimal policy that minimizes the expected cumulative cost, considering the current state. Dynamic Programming (DP) techniques are commonly employed to solve MDPs through the Bellman equations. However, solving these equations becomes computationally infeasible, especially in high-dimensional problems [9]. This challenge, known as the "curse of dimensionality" in literature, hampers the applicability of DP methods [10]. Additionally, DP necessitates precise knowledge of the transition probabilities in MDPs. In practical engineering contexts, obtaining exact transition probabilities for real systems is often unfeasible.

Reinforcement Learning (RL) emerges as a prevalent approach to overcome these challenges. Its core objective is to utilize data to approximate the optimal policy, denoted as $\boldsymbol{\pi}^\star$. RL provides effective strategies for addressing MDPs without requiring precise knowledge of the underlying state transition probability distribution, ρ. RL methods typically operate by directly approximating the optimal policy or indirectly approximating the action-value function.

The field broadly falls into two main categories: value-based methods and policy-based methods. In this context, we will elaborate on Q-learning, representing an indirect approach, and Policy Gradient methods, representing a direct approach.

2.3.1 Q-Learning

Q-learning is a widely used reinforcement learning algorithm that enables an agent to learn optimal actions in a Markov Decision Process (MDP). It operates by estimating the quality of actions through a Q-value, which represents the expected cumulative reward obtained from taking a specific action in a particular state and following an optimal policy thereafter. The algorithm iteratively updates Q-values based on the agent's experiences in the environment. Through exploration and exploitation strategies, Q-learning converges towards an optimal policy, allowing the agent to make informed decisions even in complex and dynamic environments. This approach, rooted in the principle of temporal difference learning, has been applied across various domains, showcasing its effectiveness in solving diverse reinforcement learning problems.

Q-learning, involves approximating the optimal action-value function $Q^\star$ using a parameterized function $Q_{\boldsymbol{\theta}}$. The parameters θ are iteratively adjusted using data, ensuring that the approximated Q-values, denoted as $Q_{\boldsymbol{\theta}^\star}$, closely match the optimal $Q_{\boldsymbol{\theta}} \approx Q^\star$ for the corresponding optimal parameters $\boldsymbol{\theta}^\star$.

Q-learning addresses a Least Squares (LS) problem, aiming to find the best parameters $\boldsymbol{\theta}^\star$ that accurately describe the optimal action-value function $Q^\star$.

$$\min_{\boldsymbol{\theta}} \mathbb{E}\Big[(Q_\theta(\mathbf{s}_k, \mathbf{a}_k) - Q^\star(\mathbf{s}_k, \mathbf{a}_k))^2\Big]. \tag{2.15}$$

Temporal-Difference (TD) learning provides a widely used approach to address (2.15). In this context, a fundamental TD-based learning step involves updating the parameters $\boldsymbol{\theta}$ at time instance k, in the discounted setting when $\gamma = 1$, using the following update rule:

$$\delta_k = L(\mathbf{s}_k, \mathbf{a}_k) + \gamma V_{\boldsymbol{\theta}}(\mathbf{s}_{k+1}) - Q_{\boldsymbol{\theta}}(\mathbf{s}_k, \mathbf{a}_k), \tag{2.16}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \zeta \delta_k \nabla_{\boldsymbol{\theta}}(\mathbf{s}_k, \mathbf{a}_k) \tag{2.17}$$

where the scalar $\zeta > 0$ is the learning step-size, δ_k is labeled the TD error and $V_{\boldsymbol{\theta}}$ is the parameterized value function. An approximation of the optimal policy $\boldsymbol{\pi}^\star$ can then be obtained using:

$$\hat{\boldsymbol{\pi}}^\star(\mathbf{s}) = \arg\min_a Q_{\boldsymbol{\theta}^\star}(\mathbf{s}, \mathbf{a}) \tag{2.18}$$

2.4 Model Predictive Control

Model Predictive Control (MPC) is an advanced control strategy widely used in engineering and industrial applications. Unlike traditional control methods, MPC operates by predicting the system's future behavior using a dynamic model and optimizing the control inputs over a finite prediction horizon. This predictive approach enables MPC to account for system constraints, input limitations, and desired performance criteria, making it particularly effective for complex and nonlinear systems. MPC continuously reoptimizes the control inputs based on real-time feedback, ensuring the system's optimal performance while adhering to constraints. Its ability to handle multivariable systems, deal with constraints, and adapt to changing operating conditions makes MPC a powerful and versatile control technique used in various fields such as process industries, automotive systems, robotics, and energy management. MPC is often formulated as:

$$\min_{\mathbf{x},\mathbf{a}} = T(\mathbf{x}_N) + \sum_{k=0}^{N-1} L(\mathbf{x}_k, \mathbf{u}_k) \tag{2.19}$$

$$\text{s.t} \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad \mathbf{x}_0 = s \tag{2.20}$$

$$\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \quad \mathbf{u}_k \in A \tag{2.21}$$

for a given system state $\mathbf{s}$, where N is the horizon length, T is the terminal cost, $\mathbf{f}$ is a model of the system and $\mathbf{h}$ is the mixed input-state constraint.

Equation (2.19) generates a comprehensive set of control inputs $\mathbf{u}^\star = \{\mathbf{u}_0^\star, \dots, \mathbf{u}_{N+1}^\star\}$ along with corresponding state predictions $\mathbf{x} = \{\mathbf{x}_0^\star, \dots, \mathbf{u}_N^\star\}$. However, only the initial element $\mathbf{u}_0^\star$ from the input sequence $\mathbf{u}^\star$ is implemented in the system. Upon receiving the next state $\mathbf{s}$ at the subsequent physical sampling instant, (2.19) is recalculated, producing a new sequence of $\mathbf{u}^\star$. MPC yeilds a policy,

$$\boldsymbol{\pi}_{MPC}(\mathbf{s}) = \mathbf{u}_0^\star \tag{2.22}$$

using $\mathbf{u}_0^\star$ solution from (2.19) for a given $\mathbf{s}$. Achieving this policy depends heavily on how well the MPC model $\mathbf{f}$ approximates the actual dynamics (2.8). This aspect poses a significant challenge, as accurately modeling many systems is inherently complex. Moreover, selecting the most appropriate model $\mathbf{f}$ within a modeling framework, one that optimally aligns with the data collected from the real system, is a daunting task. There's no guarantee that the model $\mathbf{f}$ that best fits the collected data is also the best model in terms of $J(\boldsymbol{\pi}_{MPC})$.

2.5 Learning based MPC

The integration of Reinforcement Learning (RL) and Model Predictive Control (MPC) offers a solution to the challenges outlined earlier. In this section, we present a key finding that supports this assertion. To understand this, it's beneficial to view MPC as a potentially localized model of the action-value function, $Q^\star$. Consider an MPC-based policy,

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{s}) = \mathbf{u}_0^\star \tag{2.23}$$

where $\mathbf{u}_0^\star$ is part of :

$$\begin{aligned} \mathbf{x}^\star, \mathbf{u}^\star = \arg\min_{\mathbf{x},\mathbf{u}} \quad & T_{\boldsymbol{\theta}}(\mathbf{x}_N) + \sum_{k=0}^{N-1} L_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k), && \text{(2.24a)} \\ \text{s.t} \quad & \mathbf{x}_{k+1} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k), \quad \mathbf{x}_0 = \mathbf{s}, && \text{(2.24b)} \\ & \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \quad \mathbf{u}_k \in A. && \text{(2.24c)} \end{aligned}$$

This MPC formulation mirrors equation (2.19), yet with a crucial difference: the cost, constraints, and dynamics within the MPC scheme are now all parameterized in $\boldsymbol{\theta}$, except for the input constraint $\mathbf{u}_k \in U$ The rationale behind this choice is explained below. The MPC-derived model of $Q^\star$ can thus be represented as:

$$Q_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) = \min_{\mathbf{x},\mathbf{u}} (2.24\text{a}) \tag{2.25}$$

$$\text{s.t.}(2.24\text{b}) - (2.24\text{c}), \quad \mathbf{u}_0 = \mathbf{a}, \tag{2.26}$$

A constraint $\mathbf{u}_0 = \mathbf{a}$ on the initial input has been introduced to (2.24). The MPC formulation (2.25) serves as a valid model of $Q^\star$ as it fulfills the relationships (2.12) and (2.14), i.e:

$$\boldsymbol{\pi}_\theta(\mathbf{s}) = \arg\min_a Q_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}), \quad V_{\boldsymbol{\theta}}(\mathbf{s}) = \min_a Q_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) \tag{2.27}$$

In this context, $V_{\boldsymbol{\theta}}(\mathbf{s})$ represents the optimal cost obtained by solving MPC equation (2.24). It is crucial to note that if the MPC parameters $\boldsymbol{\theta}$ are chosen in a way that $Q_{\boldsymbol{\theta}} = Q^\star$, the MPC scheme (2.24) yields the optimal policy $\boldsymbol{\pi}^\star$ as per equation (2.23), denoted as $\boldsymbol{\pi}_{\boldsymbol{\theta}} = \boldsymbol{\pi}^\star$. A significant question arises: how well can an MPC scheme approximate $Q^\star$, particularly in the vicinity of $\mathbf{a} = \boldsymbol{\pi}^\star(\mathbf{s})$? Moreover, $Q^\star$ is usually constructed using a discounted sum of the stage costs L, while undiscounted MPC formulations are typically preferred.

The theorem presented below addresses these concerns and serves as the key rationale for considering the MPC parametrization (2.24) in learning-based MPC. It establishes that under certain mild conditions, (2.25) can precisely model $Q^\star$, even if its predictive model (2.24a) is imprecise. Consequently, MPC formulation (2.24a) can achieve optimal closed-loop performances, even when the MPC model is inaccurate.

Theorem 2.1. *Assuming that the parameterized stage cost, terminal cost, and constraints in (2.24) can universally approximate functions with adjustable parameters $\boldsymbol{\theta}$, there exist specific parameters denoted as $\boldsymbol{\theta}$ for which the following identities hold, $\forall\gamma$:*

(i) $V_{\boldsymbol{\theta}^\star}(\mathbf{s}) = V^\star(\mathbf{s}), \forall s \in \mathcal{S}$

(ii) $\boldsymbol{\pi}_{\boldsymbol{\theta}^\star}(\mathbf{s}) = \boldsymbol{\pi}^\star(\mathbf{s}), \forall s \in \mathcal{S}$

(iii) $Q_{\boldsymbol{\theta}^\star}(\mathbf{s},\mathbf{a}) = Q^\star(\mathbf{s},\mathbf{a}), \forall \mathbf{a} \in \mathcal{A}$ *such that* $|V^\star(\mathbf{f}_{\boldsymbol{\theta}^\star}(\mathbf{s},\mathbf{a}))| < \infty$.

if the set,

$$\mathcal{S} = \left\{ \mathbf{s} \in \mathcal{S} \middle| |[V^\star(\mathbf{x}_k^\star)]| < \infty, \forall k \leq N \right\} \tag{2.28}$$

is non-empty.

Proof. We select the parameters such that the following holds:

$$T_{\boldsymbol{\theta}^\star}(\mathbf{s}) = V^\star(\mathbf{s}) \tag{2.29a}$$

$$L_{\theta^\star}(\mathbf{s},\mathbf{a}) = \begin{cases} Q^\star(\mathbf{s},\mathbf{a}) - V^\star(\mathbf{f}_\theta^\star(\mathbf{s},\mathbf{a})) & \text{if } \ |V^\star(\mathbf{f}_{\boldsymbol{\theta}^\star}(\mathbf{s},\mathbf{a}))| < \infty \\ \infty & \text{otherwise} \end{cases} \tag{2.29b}$$

□

The proof can be derived from the principles outlined in [8].

Theorem 2.1 asserts that, within a given Markov Decision Process (MDP), an MPC scheme, even with a potentially imprecise model, can yield the optimal value functions and policy of the original MDP. This achievement hinges on the careful selection of appropriate stage cost, terminal cost, and constraints. The theorem's applicability extends to various MPC variants, including robust MPC, stochastic MPC, and Economic MPC (EMPC), regardless of whether they are discounted or not. The assumption stated in (2.28) can be interpreted as a stability condition for $\mathbf{f}_{\boldsymbol{\theta}^\star}$ under the optimal trajectory $\mathbf{x}^\star$. Essentially, this assumption necessitates the existence of a non-empty set wherein the optimal value function $V^\star$ for the predicted trajectories $\mathbf{x}^\star$ based on the system model remains finite with a unitary probability for all initial states within this set.

To achieve the optimal parameter $\boldsymbol{\theta}^\star$, RL techniques, elaborated upon in the previous section, such as Q-learning and the policy gradient method, can be utilized to fine-tune the parameters $\boldsymbol{\theta}$ of the parameterized MPC scheme (2.24). This approach enables the approximation of the optimal parameter $\boldsymbol{\theta}^\star$.

Chapter 3

Contributions

3.1 Constrained Neural Networks for NMPC

This paper delves into the innovative application of constrained neural networks in the realm of Approximate Nonlinear Model Predictive Control (NMPC). The study explores the integration of neural networks to approximate complex nonlinear systems, focusing specifically on NMPC, a powerful control technique used in various fields. By introducing constraints into the neural network architecture, we aim to enhance the accuracy and reliability of control predictions in real-time scenarios.

The paper discusses the methodology employed, emphasizing the incorporation of constraints within the neural network model. This approach involves optimizing the network's architecture to handle specific system constraints efficiently. We also present experimental results and simulations, demonstrating the effectiveness of this proposed Constrained Neural Network approach in NMPC applications. Comparisons with traditional NMPC techniques and other neural network models have been included to showcase the advantages of their method.

Abstract

Solving Non-Linear Model Predictive Control (NMPC) online is often challenging due to the computational complexities involved. This issue can be avoided by approximating the optimization problem using supervised learning methods which comes with a trade-off on the optimality and/or constraint satisfaction. In this paper, a novel supervised learning framework for approximating NMPC is proposed, where we explicitly impart constraint knowledge within the neural networks. This knowledge is inherited by augmenting the loss function of the neural networks during the training phase with insights from KKT conditions. Logarithmic barrier functions are utilized to augment the loss function including conditions of primal and dual feasibility. The proposed framework can be applied to other machine learning based parametric approximators. This approach is easy to implement and its efficacy is demonstrated on a benchmark NMPC problem for continuous stirred tank reactor (CSTR).

Constrained Neural Networks for Approximate Nonlinear Model Predictive Control

Saket Adhau, Vihangkumar V. Naik, Sigurd Skogestad

I. INTRODUCTION

Model Predictive Control (MPC) has widespread usage in industrial applications ranging from chemical plants, petroleum refineries, aerospace, and automotive domains [1]. However, the applicability of MPC, especially non-linear MPC (NMPC) is limited due to computational complexity involved in solving the associated nonconvex optimization problem within the stipulated sampling time. Problems of this class are being investigated by control as well as machine learning communities. Especially, by using machine learning tools such as Neural Networks to approximate the optimal control law given by MPC. Such approaches encounter challenges for industrial applications where the availability of necessary data could be limited, and synthesizing the available data could be challenging. Another underlying problem is that such approximated control laws often make a trade-off between performance and constraints satisfaction. Satisfying the constraints while approximating the control law can be of paramount importance and needs to be addressed.

Related Work:

1) Approximate MPC: A number of approaches to approximate optimal control law given by MPC using machine learning methods have been explored in literature. In these approaches, the control law is typically approximated using various parametric approximators often termed as imitation learning or supervised learning [2]–[4].

This work was supported by the Research Council of Norway, under the IKTPLUSS program.

Saket Adhau and Sigurd Skogestad are with the Department of Chemical Engineering from Norwegian University of Science and Technology, 7491, Trondheim, Norway. {saket.adhau, sigurd.skogestad}@ntnu.no

Vihangkumar V. Naik is with ODYS S.r.l., Via A. Passaglia 185, Lucca, 55100, Italy. vihang.naik@odys.it

2) Constraint handling in Neural Networks: The inability to guarantee constraint satisfaction using these methods is one of the major limitations, contributing to active research in this direction. The authors in [5], provide guarantees on closed-loop constraint satisfaction and asymptotic stability. Projection of feasible control input on polytopes derived from maximal control invariant set of the system is presented in [6] and similar methods have also been shown in [7]. Furthermore, two separate networks for primal and dual problems are trained to estimate sub-optimality and probabilistic bounds are provided on the trained linear controller in [8]. In [9], the network is trained while adjusting the architecture until probabilistic bounds are satisfied. Additionally, instead of replacing the traditional MPC with approximate control law, neural networks have also been examined to assist the controller in warm starting, such as [10], [11]. The contribution in [12] present end to end learning of both system dynamics and control policies with constraint satisfaction capabilities.

Research work [13] dating back as early as $90's$, exhibit a class of neural networks for approximating general non-linear programming problems including equality and inequality constraints. The method is based on Lagrangian multipliers in optimization and provide solutions to satisfy the necessary conditions of optimality. The authors in [14] devise Lagrange type neural networks which can handle inequality constraints, whereas stability and convergence is discussed using Liapunov's approximation principle. The optimization aspects of imposing hard inequality constraints on Convolutional neural network (CNN), by using Log barrier functions along-with Lagrangian-dual optimization has been presented in [15]. In addition, [16]–[20] discuss methods from optimization theory especially KKT conditions to impose constraints on neural networks. To the best of the authors' knowledge, none of the ideas mentioned above have been studied in control domain for assisting in approximating optimal control laws.

In this paper, we present a novel constraint handling approach exploiting the domain knowledge from KKT conditions while approximating the NMPC problem. Typically, constraint handling in neural network based approximators is done at a later stage, only after the network is trained. Conversely, in our approach, primal and dual feasibility conditions are inherently passed on to the network during the training phase itself. This is achieved by augmenting the loss function with penalties on primal and dual feasibility. Hence, the overall idea is to introduce domain specific knowledge whilst training, to better leverage constraint characterization inside the network. We analyze performance of the proposed approach for approximating NMPC applied to a benchmark continuous stirred tank reactor (CSTR) problem. It is interesting to note that the proposed idea is applicable to approximate a wide range of constrained optimization problems.

The paper is organized as follows: Section II introduces background for neural networks. NMPC problem formulation, the conventional approach for NMPC approximation and the proposed approach are presented in Section III. Implementation of the proposed approach and results are discussed in Section IV while the paper concludes in Section V.

II. Background

This section introduces the loss function of neural networks to be utilized as a basis for the proposed framework. Let N be the number of training samples where, input matrix is denoted by $X = \{x_1, x_2, \ldots, x_N\}^\top \in \mathbb{R}^{N\times d}$, and output matrix is denoted by $Y = \{y_1, \ldots, y_N\}^\top \in \mathbb{R}^{N\times m}$ for the training data $\{x_i, y_i\}_{i=1}^N$. Input dimensions are denoted by d, whereas m denotes output dimensions. A fully connected feedforward network consisting of L layers indexed as $0, 1, 2, \ldots, L$ corresponding to input layer is considered. These hidden layers are composed of weight matrices $(W_k)_{k=1}^L \in \mathcal{W} := \mathbb{R}^{d\times n_1} \times \cdots \times \mathbb{R}^{n_{k-1}\times n_k} \times \cdots \times \mathbb{R}^{n_{L-1}\times m}$ where n_k denotes number of neurons on layer k, $n_0 = d$ and $n_L = m$ whereas the bias on the layers $(b_k)_{k=1}^L \in \mathcal{B} := \mathbb{R}^{n_1} \times \ldots \mathbb{R}^{n_L}$. We use nonlinear Rectified Linear Unit (ReLU), a non differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$. Let $\Psi \in \mathbb{R}^{N\times n_k}$ be the matrix containing feature vectors of layer k after application of activation function. Once the network architecture is designed, the Loss function $\Phi : \theta \to \mathbb{R}$ of the network can be defined as,

$$\Phi\Big((W_k, b_k)_{k=1}^L\Big) = \frac{1}{mN}\sum_{i=1}^{N}\sum_{j=1}^{m}\Big((\Psi_{Lj}(x_i)) - y_{ij}\Big). \tag{1}$$

In this paper, we consider, Mean Squared Error (MSE) as our loss function which is assumed to be continuously differentiable.

III. Neural network based approximate NMPC

Consider a discrete-time nonlinear system of the form,

$$x(t+1) = f(x(t), u(t)), \tag{2}$$

where the current states $x \in \mathcal{X}$, the control input $u \in \mathcal{U}$ are constrained to lie in compact sets of $\mathcal{X} \subset \mathbb{R}^{n_x}$ and $\mathcal{U} \subset \mathbb{R}^{n_u}$. The nominal model, $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is assumed to be twice differentiable in x and u.

The NMPC problem $\mathcal{P}(x(t))$ can be formulated as,

$$\min_{x(.),u(.)} \int_{t_0}^{t_0+T} \frac{1}{2}\left(||x(t)-x^{\text{ref}}(t)||_Q^2 + ||u(t)-u^{\text{ref}}(t)||_R^2\right) dt$$
$$+ ||x(t_0+T) - x^{\text{ref}}(t_0+T)||_P^2, \tag{3a}$$

subjected to following constraints for all $t \in [t_0, t_0+T]$

$$x(t_0) = \hat{x}_0, \tag{3b}$$

$$x(t+1) = f(x(t), u(t)), \tag{3c}$$

$$\underline{u} \leq u(t) \leq \overline{u}, \ \forall u \in \mathcal{U}, \tag{3d}$$

$$\underline{x} \leq x(t) \leq \overline{x}, \ \forall x \in \mathcal{X}. \tag{3e}$$

where, $T > 0$ is the prediction horizon, $x(t) \in \mathbb{R}^{n_x}$ and $u(t) \in \mathbb{R}^{n_u}$. The cost function is weighted by $Q \succeq 0, P \succeq 0$, and $R \succ 0$ and $\hat{x}_0$ is the current state estimate in (3b) at time t_0. The control input is bounded by $\underline{u}$ and $\overline{u}$ and states by $\underline{x}$ and $\overline{x}$, see [21], [22, Chapter 2]. The resulting NMPC control law is given by,

$$u^*(t) = \pi_{MPC}(x(t)). \tag{4}$$

Solving this problem online at each sampling time t is often computationally complex and may become impractical with increase in problem size.

A. *Approximate NMPC*

The problem in (3) can be approximated with neural networks using the loss function given by,

$$\mathcal{L}(x;\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\pi_{approx}(x_i;\theta) - u_i^*\right)^2. \tag{5}$$

In the above equation, $\pi_{approx}(x_i;\theta)$ is a trained policy such that $\pi_{approx} \approx \pi_{MPC}$ and u_i^* is the optimal control input or labeled dataset. The loss function $\mathcal{L}(x;\theta)$ is minimized w.r.t $\theta = \mathcal{W} \times \mathcal{B}$ which is an unconstrained optimization problem.

However, in general the nonlinear programming (NLP) problems arising in NMPC formulation contains constraints on u and/or x, such as on both in (3d) and (3e). The approximate control law π_{approx} may not mimic the behavior of the optimal policy accurately, often leading to a poor approximation, which results in constraints being not satisfied at all times. As a rule of thumb, the control input/s are saturated/clipped off for input constraint satisfaction [7], which may deteriorate the overall control performance. In order to derive a better approximation, we describe a novel supervised learning framework in the following sections.

B. Penalty methods and Logarithmic Barrier Functions

Penalty methods are often used to convert constrained optimization problem into an unconstrained optimization problem, by augmenting the objective function with a penalty when constraints are violated. However, penalty methods may not always guarantee constraint satisfaction and need precise tuning of the weight for each penalty term in the equation. If in a cost function, there are several constraints, penalty functions will not act as *barriers* at the boundaries of feasible region but rather be null. This may lead to oscillations and make the training unstable [15]. To overcome these drawbacks in penalty terms, we use Logarithmic barrier functions.

Consider the following NLP problem,

$$\min_{w} \ \psi(w), \tag{6a}$$

$$\text{s.t } g_i(w) \leq 0, \ \forall_i \in \{1, \dots, m\}, \tag{6b}$$

where the strictly feasible region $\mathcal{F}^0$ is defined by,

$$\mathcal{F}^0 \equiv \{w \in \mathbb{R}^n | g_i(w) \leq 0\}, \ \forall_i \in \{1, \dots, m\}, \tag{7}$$

and $\mathcal{F}^0$ is assumed to be non-empty. The logarithmic barrier function for the constraint, $g_i(w) \leq 0, \ \forall_i \in \{1, \dots, m\}$, denoted by $\phi(w)$ is,

$$\phi(w) = \sum_{i \in \mathcal{I}} -\log(-g_i(w)), \tag{8}$$

where $\log(.)$ is natural logarithm. The value of the function approaches ∞ as w moves towards the edge of feasible region $\mathcal{F}^0$ implying when the constraints are violated, a high penalty of ∞ would be added to the cost function[1]. In the event of no constraint violation, the barrier term would be 0. The new objective function for the constrained optimization problem (6), when converted into an unconstrained optimization problem, can be written as,

$$\min_{w, \mu^k} \ \psi(w) - \mu^k \sum_{i=1}^{m} \log(-g_i(w)), \tag{9}$$

where, μ^k is a sequence of positive scalar barrier parameters, [23, Chapter 6].

[1]In case of neural networks, where the gradient of ∞ cannot be evaluated, a penalty of high value is imposed.

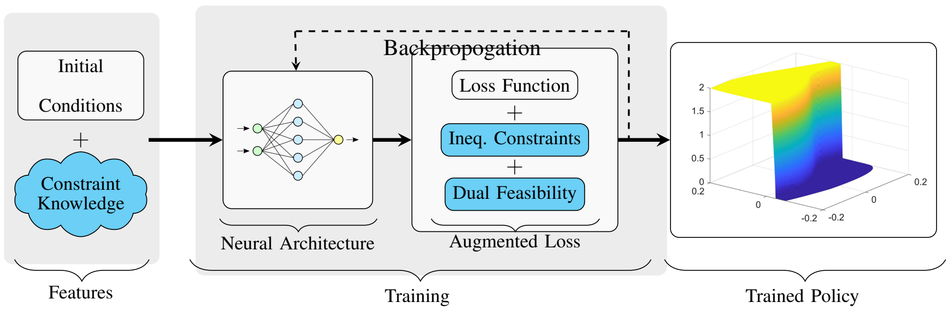

Fig. 1. Illustration of the proposed methodology for injecting constraint knowledge during the training phase for approximating constrained optimization problems.

C. Approximation of Constrained Optimization Problems

For approximating the NMPC problem (3), the main idea here is to introduce constraints (3d) and (3e) inside the neural networks during the training phase. This can be done by augmenting the loss function defined in (5), with (3d) and (3e) using logarithmic barrier function as in (9).

We start by introducing the input constraints (3d) in the loss functions (5) given by,

$$\mathcal{L}(x;\theta) = \frac{1}{N}\sum_{i=1}^{N}\Bigg(\Big\|\pi_{approx}(x_i;\theta) - u_i^*\Big\|_2^2 + \\ \mu_{\underline{u}}\Big\|f_b(\hat{u}_i,\underline{u}_i)\Big\|_2^2 + \ \mu_{\overline{u}}\Big\|f_b(\hat{u}_i,\overline{u}_i)\Big\|_2^2\Bigg),$$

where, $\mu_{\underline{u}}, \mu_{\overline{u}} > 0$ are the barrier parameters. The function f_b for lower and upper bound is defined as per (8),

$$f_b(\hat{u}_i,\underline{u}_i) : \ \underline{u}_i - \hat{u}_i \leq 0 : \quad -\log(\hat{u}_i - \underline{u}_i), \tag{10a}$$

$$f_b(\hat{u}_i,\overline{u}_i) : \ \hat{u}_i - \overline{u}_i \leq 0 : \quad -\log(\overline{u}_i - \hat{u}_i). \tag{10b}$$

Furthermore, to include state constraints (3e), a new policy $\hat{\pi}_x$ is required to be defined which will not only predict the optimal control inputs but also the next states,

$$\begin{bmatrix}\hat{u} & \hat{x}^+\end{bmatrix}^\top = \hat{\pi}_x(x_i;\theta). \tag{11}$$

The resulting loss function is formulated as,

$$\begin{aligned}\mathcal{L}_x(x;\theta) =& \frac{1}{N}\sum_{i=1}^{N}\Bigg(\Big\|\hat{\pi}_x(x_i;\theta) - z^*_{x_i}\Big\|_2^2 + \\ & \mu_{\underline{u}}\Big\|f_b(\hat{u}_i,\underline{u}_i)\Big\|_2^2 + \ \mu_{\overline{u}}\Big\|f_b(\hat{u}_i,\overline{u}_i)\Big\|_2^2 + \\ & \mu_{\underline{x}}\Big\|f_b(\hat{x}_i,\underline{x}_i)\Big\|_2^2 + \ \mu_{\overline{x}}\Big\|f_b(\hat{x}_i,\overline{x}_i)\Big\|_2^2\Bigg),\end{aligned} \tag{12}$$

where, $\mu_{\underline{x}}, \mu_{\overline{x}} > 0$, vector z^*_x consists of $[u^* \ x^+]^\top$ calculated by solving the NMPC problem $\mathcal{P}(x)$.

To better leverage constraint characterization inside the network, we also exploit the insights from KKT conditions, specifically the dual feasibility. In order to do this, for the sake of simplicity of notations, we rewrite inequality constraints in (3d) & (3e) and the dual variables associated with the inequality constraints as, $\mathcal{A} = [-u + \underline{u}, u - \overline{u}, -x + \underline{x}, x - \overline{x}]^\top \leq 0$, $\lambda = [\lambda_{\underline{u}}, \lambda_{\overline{u}}, \lambda_{\underline{x}}, \lambda_{\overline{x}}]^\top \geq 0$ respectively where, $\lambda \in \mathbb{R}^{2n_x+2n_u}$. For including dual feasibility $\lambda \geq 0$, the function f_λ to be used in loss function is defined similar to (10) as,

$$f_\lambda(\hat{\lambda}_i): \quad -\hat{\lambda}_i \leq 0: \quad -\log(\hat{\lambda}_i).$$

Next, we define a new policy $\hat{\pi}_\lambda(x_i;\theta)$, such that it predicts dual variables $\hat{\lambda}$ along-with predicted inputs $\hat{u}$ and next states $\hat{x}^+$ as in (11). Finally, we write the proposed loss function as,

$$\begin{aligned}\mathcal{L}_\lambda(x;\theta) = & \frac{1}{N}\sum_{i=1}^{N}\Bigg(\Big\|\hat{\pi}_\lambda(x_i;\theta) - z^*_{\lambda_i}\Big\|_2^2 + \\ & \mu_{\underline{u}}\Big\|f_b(\hat{u}_i,\underline{u}_i)\Big\|_2^2 + \mu_{\overline{u}}\Big\|f_b(\hat{u}_i,\overline{u}_i)\Big\|_2^2 + \\ & \mu_{\underline{x}}\Big\|f_b(\hat{x}_i,\underline{x}_i)\Big\|_2^2 + \mu_{\overline{x}}\Big\|f_b(\hat{x}_i,\overline{x}_i)\Big\|_2^2 + \mu_\lambda\Big\|f_\lambda(\hat{\lambda}_i)\Big\|_2^2\Bigg),\end{aligned} \tag{13}$$

where, $\mu_\lambda > 0$ and the vector $z^*_{\lambda_i}$ consists $[u^* \ x^+ \ \lambda^*]^\top$. Using dual variables, the idea is to ensure constraint satisfaction when feasible solution exists and help the network optimize its policy-parameters θ, while inheriting constraint knowledge. The proposed general idea for approximating constrained optimization problems is illustrated in Fig. 1.

The barrier terms are only active in the loss function when constraints are violated, reducing the excessive gradient ascent iterations and making the training less computationally expensive [15].

The detailed procedure for the proposed approach is summarized in Algorithm 1. We start with a formulated NMPC problem $\mathcal{P}(x)$, a null dataset $\mathcal{D}$ for collecting the data required and a feasible state space $\mathcal{X}$. For each instance $i = 1, \ldots, N$, the NMPC problem is solved by randomly choosing x_i from the feasible state space $\mathcal{X}$ and the dataset $\mathcal{D}$ is updated with $x_i, u_i^*, x_i^+, \lambda_i^*$. Eventually, once N samples are processed, the network is trained using the cost function described in (13).

Algorithm 1 Learning of constrained approximate NMPC.

Input: NMPC problem $\mathcal{P}(x)$, feasible set $\mathcal{X}$, null dataset $\mathcal{D}$

```
1: for $i = 1$ to $N$ do
2:     Sample $x_i \in \mathcal{X}$
3:     Collect $u_i^*$ and $x_i^+$ for every $x_i$ by solving $\mathcal{P}(x_i)$ and    $\lambda_i^*$ for every inequality constraint
4:     Update the dataset, $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_i, u_i^*, x_i^+, \lambda_i^*)\}$
5: end for
6: $\theta \leftarrow$ (13)
```

Output: $\hat{\pi}_\lambda(x_i; \theta)$.

IV. Simulation Results

As an illustrative example, we consider the Continuous Stirred Tank Reactor (CSTR) benchmark system [24] given by,

$$\begin{aligned} \dot{x}_1 &= \tfrac{1}{\beta}(1 - x_1) - c x_1 e^{-\frac{S}{x_2}}, \\ \dot{x}_2 &= \tfrac{1}{\beta}(x_f - x_2) + c x_1 e^{-\frac{S}{x_2}} - \alpha u (x_2 - x_c), \end{aligned} \tag{14}$$

where the states are product concentration x_1 and the reactant coolant temperature x_2. The feasible state space is given by $\mathcal{X} = [0.0632, 0.4632] \times [0.4519, 0.8519]$ and the coolant flow input rate u is characterized as $\mathcal{U} = [-0.7853, 1.2147]$.

The system parameters considered are $\beta = 20$, $c = 300$, $S = 5$, $x_f = 0.3947$, $x_c = 0.3816$ and $\alpha = 0.117$. The control objective is to take the system from a locally stable steady state to a locally unstable steady state point, $x^{\text{ref}} = [0.2632, 0.6519]^\top$ which makes the problem challenging. This control problem is solved using the formulation in (3). The initial conditions for NMPC formulation are $x_0 = [0.9831, 0.3918]^\top$ with a prediction horizon of $T = 60$ and sampling time of $t = 0.5s$.

Step I: Generating data for Neural Networks: We use grid-based sampling approach as seen in [9]. An uniform grid of size $(8 \cdot 10^{-3})$ consisting of $(1.6 \cdot 10^5)$ data points is created out of which around $(1.4 \cdot 10^5)$ data points were identified as feasible. Detailed procedure for generating training data as required by the proposed method is given in Algorithm 1. NMPC problem $\mathcal{P}(x)$ from (3) was solved for all $x_i \in \mathcal{X}$ using CasADi v3.5.5 for Python [25].

Step II: Learning: For comparison, we train 3 different policies namely, (i) supervised learning approach $\pi_{approx}(x_i;\theta)$ in (5); (ii) log barrier penalty function for input and state constraints $\hat{\pi}_x(x_i;\theta)$ as in (12); (iii) the proposed method $\hat{\pi}_\lambda(x_i;\theta)$ in (13).

For training of the above mentioned policies, a feedforward neural network consisting of two hidden layers of 160 and 1560 neurons respectively were used. ReLU and linear activation functions were used with Adam optimizer. The neural network architecture was implemented in Keras v2.3.0 for Python with a learning rate of $1 \cdot 10^{-3}$, batch size of 500 and 512 epochs. The barrier parameters used in the cost function are $\mu_{\underline{u}} = \mu_{\overline{u}} = 0.1$, $\mu_{\underline{x}} = \mu_{\overline{x}} = 0.1$ and $\mu_\lambda = 0.01$.

All the simulations in this paper including training of the policies, have been done on a MacBook Pro with Intel Core i7, 2.6GHz processor and 16GB of memory. The time required for training the policies is reported in Table. I. It is also observed that training time for the proposed approach $\hat{\pi}_\lambda(x_i;\theta)$ is only marginally increased.

TABLE I
COMPARISON OF TRAINING TIMES FOR VARIOUS POLICIES.

Policy	$\pi_{approx}(x_i;\theta)$	$\hat{\pi}_x(x_i;\theta)$	$\hat{\pi}_\lambda(x_i;\theta)$
Time [s]	103.54	106.22	108.58

A. Result Analysis

We use the standard NMPC results as primary reference and compare with three approaches described in Step II of Section IV. Figs. 2–4 show the experimental results of state evaluation for x_1, x_2 and control input u for a benchmark CSTR problem.

The results in Fig. 2 show that, the most commonly used method of the control input generated by the policy $\pi_{approx}(x_i;\theta)$ results in violation of constraints imposed on the coolant flow rate input. Secondly, we observed that the results denoted by $\hat{\pi}_x(x_i;\theta)$, demonstrate how the inclusion

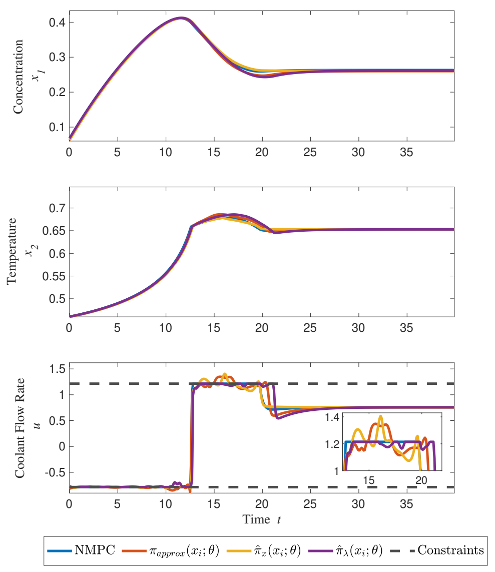

Fig. 2. CSTR benchmark: Performance comparison of states x_1, x_2 and control input u using various approaches *v.i.z* traditional supervised learning approach $\pi_{approx}(x_i;\theta)$; penalty for input and state constraints $\hat{\pi}_x(x_i;\theta)$ and the proposed method $\hat{\pi}_\lambda(x_i;\theta)$.

of input and state constraint penalties as described in the loss function (12), are not sufficient for constraints satisfaction. Though as a convention, adding saturation to the control input for the policies $\pi_{approx}(x_i;\theta)$ and $\hat{\pi}_x(x_i;\theta)$ would satisfy the constraints on coolant flow rate input, the control performance gets degraded as demonstrated in Fig. 3. Finally, the results of $\hat{\pi}_\lambda(x_i;\theta)$ are same in Fig. 2 and 3 for illustration purposes, which denotes performance of the proposed method including the dual variables, characterized by (13). From Fig. 3, it can be concluded that using $\hat{\pi}_\lambda(x_i;\theta)$ there are no input constraint violation (without saturating the control input) and it renders the best performance in terms of state evaluation against standard NMPC approach.

As observed in Fig. 4, the policies $\pi_{approx}(x_i;\theta)$ and $\hat{\pi}_x(x_i;\theta)$ violate the constraints for state x_2. Whereas using the proposed approach, state constraints are always ensured, even with a different initial states as compared to the former figures, which is further illustrated in Fig. 5. Fig 5 shows evolution of states x_1 *vs* x_2 inside the feasible region $\mathcal{X}$ considering different initial states for the NMPC as well as the proposed policy $\hat{\pi}_\lambda(x_i;\theta)$. For each of the initial conditions considered, it can be observed that the proposed approach is able to mimic the NMPC trajectories without any constraint violation.

Furthermore, the proposed method is evaluated for 5000 different trajectories. The resulting values of key performance indices (KPIs) compared with standard NMPC and maximum state constraint violations are reported in Table. II. The approximations rendered by the proposed approach $\hat{\pi}_\lambda(x_i;\theta)$, outperforms standard approach $\pi_{approx}(x_i;\theta)$ as well as $\hat{\pi}_x(x_i;\theta)$.

V. CONCLUSION

In this paper, we proposed simple yet effective approach for constraint handling for approximating nonlinear MPC problems using neural networks. The basic idea of including the domain knowledge pertaining to the underlying nonlinear optimization problem while training the neural network has been explored. Specifically, the intuitive insights of KKT conditions (primal and dual feasibility) and log barrier methods are utilized to modify the loss function of the neural networks. Thorough numerical simulations on a benchmark CSTR problem have demonstrated the ability to explicitly handle constraints on both, input as well as states. The proposed framework rendered improved performance in terms of input and state evaluation compared to the conventional neural

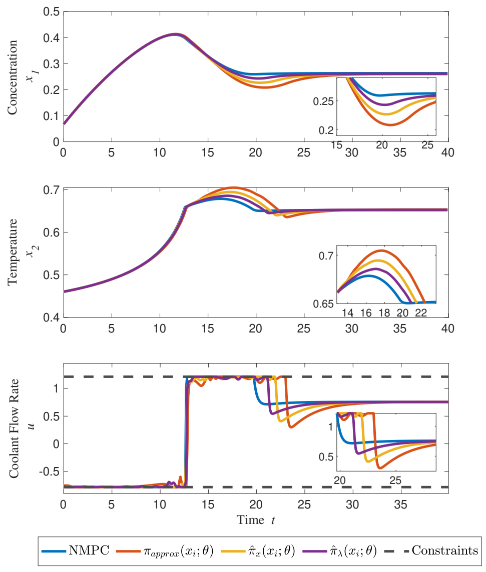

Fig. 3. CSTR benchmark: Performance comparison for states x_1, x_2 and control input u using various approaches *v.i.z* traditional supervised learning approach $\pi_{approx}(x_i;\theta)$ with clipping of control input; penalty for input and state constraints $\hat{\pi}_x(x_i;\theta)$ with input clipped and the proposed method $\hat{\pi}_\lambda(x_i;\theta)$ (without clipping of control input).

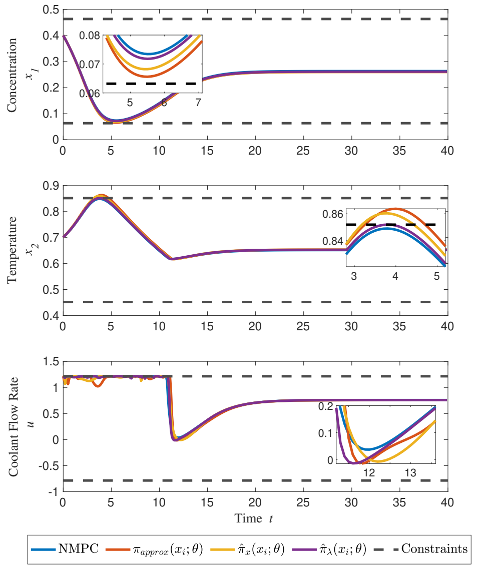

Fig. 4. Demonstration of state constraint satisfaction for CSTR benchmark: traditional supervised learning approach $\pi_{approx}(x_i;\theta)$ with clipping of control input; penalty for input and state constraints $\hat{\pi}_x(x_i;\theta)$ with clipping of control input and the proposed method $\hat{\pi}_\lambda(x_i;\theta)$ (without clipping of control input).

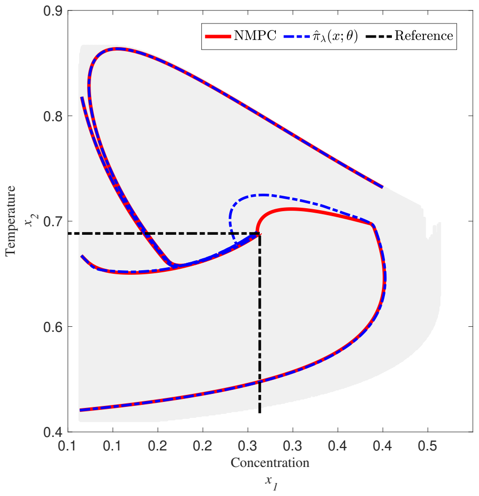

Fig. 5. CSTR benchmark: Concentration (x_1) *vs.* temperature (x_2) for randomly initiated evolution of states using proposed method against NMPC, black dashed line represents reference values.

network based method. This approach is not limited to NMPC but can be used to approximate a wide range of constrained optimization problems.

TABLE II

COMPARISON OF KPIS WITH NMPC FOR OVER 5000 TRAJECTORIES

KPI	$\pi_{approx}(x_i;\theta)$	$\hat{\pi}_x(x_i;\theta)$	$\hat{\pi}_\lambda(x_i;\theta)$
Comparison for states			
ISE	$0.148 \cdot 10^{-6}$	$8.71 \cdot 10^{-6}$	$2.56 \cdot 10^{-6}$
IAE	0.8857	0.5017	0.1528
ITSE	0.3821	0.2914	0.1566
Comparison for control input			
ISCE	$3.02 \cdot 10^{-5}$	$1.35 \cdot 10^{-5}$	$0.142 \cdot 10^{-5}$
Maximum state constraint violation			
	4.4231	3.2783	0.7739

REFERENCES

[1] S. J. Qin and T. A. Badgwell, “A survey of industrial model predictive control technology,” *Control engineering practice*, vol. 11, pp. 733–764, 2003.

[2] S. Lucia and B. Karg, “A deep learning-based approach to robust nonlinear model predictive control,” in *6th IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2018, pp. 511–516.

[3] S. Lucia, D. Navarro, B. Karg, H. Sarnago, and O. Lucia, “Deep learning-based model predictive control for resodat power converters,” *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 409–420, 2020.

[4] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, “Approximate model predictive building control via machine learning,” *Applied Energy*, vol. 218, pp. 199–216, 2018.

[5] B. Karg and S. Lucia, “Stability and feasibility of neural network-based controllers via output range analysis,” in *59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 4947–4954.

[6] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, “Approximating explicit model predictive control using constrained neural networks,” in *2018 Annual American Control Conference (ACC)*, 2018, pp. 1520–1527.

[7] J. A. Paulson and A. Mesbah, "Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction," *IEEE Control Systems Letters*, vol. 4, pp. 719–724, 2020.

[8] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari, "Large scale model predictive control with neural networks and primal active sets," *arXiv preprint arXiv:1910.10835*, 2019.

[9] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, pp. 543–548, 2018.

[10] M. Klaučo, M. Kalúz, and M. Kvasnica, "Machine learning-based warm starting of active set methods in embedded model predictive control," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 1–8, 2019.

[11] Y. Vaupel, N. C. Hamacher, A. Caspari, A. Mhamdi, I. G. Kevrekidis, and A. Mitsos, "Accelerating nonlinear model predictive control through machine learning," *Journal of Process Control*, vol. 92, pp. 261–270, 2020.

[12] J. Drgona, K. Kis, A. Tuor, D. Vrabie, and M. Klauco, "Differentiable predictive control: An MPC alternative for unknown nonlinear systems using constrained deep learning," *arXiv preprint arXiv:2011.03699*, 2020.

[13] S. Zhang and A. Constantinides, "Lagrange programming neural networks," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, pp. 441–452, 1992.

[14] Y. Huang, "Lagrange-type neural networks for nonlinear programming problems with inequality constraints," in *44th IEEE Conference on Decision and Control (CDC)*, 2005, pp. 4129–4133.

[15] H. Kervadec, J. Dolz, J. Yuan, C. Desrosiers, E. Granger, and I. B. Ayed, "Constrained deep networks: Lagrangian optimization via log-barrier extensions," *arXiv preprint arXiv:1904.04205*, 2019.

[16] P. Márquez-Neila, M. Salzmann, and P. Fua, "Imposing hard constraints on deep networks: Promises and limitations," *arXiv preprint arXiv:1706.02025*, 2017.

[17] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, pp. 25–57, 2006.

[18] Y. Nandwani, A. Pathak, P. Singla *et al.*, "A primal dual formulation for deep learning with constraints," *Advances in Neural Information Processing Systems*, pp. 12 157–12 168, 2019.

[19] S. Shalev-shwartz and S. M. Kakade, "Mind the duality gap: Logarithmic regret algorithms for online optimization," *Advances in Neural Information Processing Systems*, vol. 21, 2009.

[20] Q. Nguyen and M. Hein, "The loss surface of deep and wide neural networks," in *34th International Conference on Machine Learning*, 2017, pp. 2603–2612.

[21] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl, "Auto-generated algorithms for nonlinear model predictive control on long and on short horizons," in *52nd IEEE Conference on Decision and Control (CDC)*, 2013, pp. 5113–5118.

[22] A. Grancharova and T. A. Johansen, *Explicit nonlinear model predictive control: Theory and applications*. Springer Science & Business Media, 2012, vol. 429.

[23] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. Society for Industrial and Applied Mathematics, 2010.

[24] D. Q. Mayne and E. C. Kerrigan, "Tube-based robust nonlinear model predictive control1," in *7th IFAC Symposium on Nonlinear Control Systems*, 2007, pp. 36–41.

[25] 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*, pp. 1–36, 2019.

3.2 RTO-RLMPC for GasLift Optimization: Practical Implementation

In this paper, we present the real-world implementation of the RTO-RLMPC (Real-Time Optimization - Reinforcement Learning Model Predictive Control), a novel approach presented in the context of gas lift optimization. Gas lift optimization is a crucial process in various industrial applications, and this study demonstrates the practical application of the RTO-RLMPC technique in this specific domain. The paper provides insights into how the theoretical framework of RTO-RLMPC translates into actual experimental settings, offering valuable information about its feasibility, effectiveness, and performance in optimizing gas lift operations. By showcasing the implementation on a real gas lift pilot plant, the paper bridges the gap between theoretical concepts and practical applications, providing valuable implications for the field of process optimization.

Abstract

In this section, we present a Reinforcement Learning (RL) based Economic Nonlinear MPC (ENMPC) scheme to improve closed-loop performance of the controller when using an inaccurate model of the system at hand. The proposed scheme uses a comprehensive data-based tuning of the ENMPC scheme for the real system. The tuning targets performance improvements both through the transients and at steady-state. Moreover, we show how the Real Time Optimization (RTO) layer can be harmoniously combined with the proposed tools to further assist the data-based tuning of the optimal control policy. The method is validated on an experimental rig that emulates a subsea oil-well network. We show that the proposed method can tune the ENMPC scheme using offline measurement data from the real system. We compare the performance of the proposed framework with a naive ENMPC scheme and present the gains in terms of profit achieved.

Data Driven Framework for Combining Real-Time Optimization and Economic NMPC on an Experimental Rig

Saket Adhau, José Matias, Sébastien Gros, Sigurd Skogestad

I. Introduction

In chemical plants, deciding the best operating strategy in terms of increasing profit or reducing costs is a key way to respond to a competitive globalized market. However, defining this strategy in face of frequent changing operating conditions, different feedstocks, variable constraints and major changes in economics (such as product and feedstock prices) is challenging.

The conventional approach is to divide the decision-making process into successive layered problems, as shown in Fig. 1. Each layer deals with a problem concerning different time scales and scopes, such as daily scheduling decisions to avoid storage problems or local PID controllers rejecting process disturbances in comparison to plant-wide feedstock purchasing decisions [1]. The translation of economic objectives into operational objectives is achieved through two sequential layers: the real-time optimization (RTO) and the model predictive control (MPC) layer. The former converts business decisions into setpoints for controlled and manipulated variables, whereas the latter uses a multivariable linear dynamic model to provide real-time tracking of these setpoints. Typically, the decisions of the RTO and MPC layers are defined by an optimization problem. This hierarchical decomposition simplifies the design of the control system by making it possible to use different models and objective functions in each layer. For example, the RTO layer usually minimizes an economic cost function based on a physical nonlinear static model,

This work was supported by the Research Council of Norway, under the IKTPLUSS program.

Saket Adhau and Sigurd Skogestad are with the Department of Chemical Engineering from Norwegian University of Science and Technology, 7491, Trondheim, Norway. {saket.adhau, sigurd.skogestad}@ntnu.no

José Matias is with the Department of Chemical Engineering, KU Leuven and Sebastien Gros is with the department of Engineering Cybernetics, NTNU.

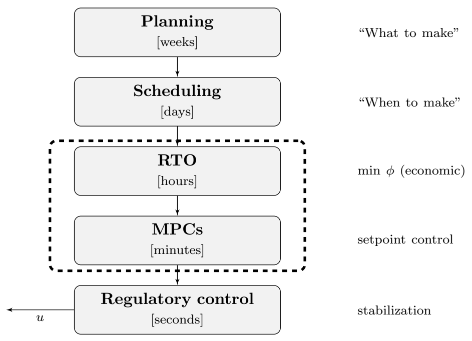

Fig. 1. Typical plant decision hierarchy for optimal plant operating strategy. Adapted from [1]

and computes setpoints to the MPC layer. The MPC layer operates on an intermediate time scale (minutes) and typically uses a dynamic linear model while minimizing a cost function that involves setpoint deviations. The regulatory PID control layer, on the other hand, works on a faster time scale (seconds) and deals with disturbance rejection and stabilization.

The main disadvantage of this approach arises when the assumption of time scale separation does not hold. For instance, a batch process is never at a steady state, making it necessary to include dynamics in the control layer. In other cases, it may be more beneficial to combine the MPC and PID layers. In such situations, economic model predictive control (EMPC) becomes an attractive solution, as it can combine all three layers.

Furthermore, the performance of model-based optimization tools such as RTO and ENMPC heavily relies on the accuracy of the system model. Plant-model mismatch can cause the controller to drive the system to a non-economical steady-state point. In the context of EMPC, data-driven approaches can be employed to adapt the EMPC model and compensate for plant-model mismatch and model uncertainties. However, it is important to note that the EMPC model may not fully capture the behavior of the real system, and fitting the data to the model may not necessarily lead to an optimal policy from the EMPC scheme.

Data-driven approaches can be an effective way to address plant-model mismatch and model uncertainties in control systems. These approaches involve using data from the system to adapt or refine the model used for control. The basic idea is to use the real-time data generated by the system to update the control model so that it better matches the actual system behavior.

One specific technique that has been used conventionally to address these issues in RTO is

Modifier Adaptation (MA), see [2]. This approach involves introducing modifier terms in the objective function and constraints of the optimization problem to compensate for the model-plant mismatch. MA has been shown to improve the performance of the optimization algorithm and reduce the sensitivity to model uncertainties and disturbances.

In recent years, researcher have explored data-driven approaches for control systems, including Reinforcement Learning (RL) based frameworks. In a series of papers, authors [3]–[5] have proposed an approach where the ENMPC scheme can be viewed as a model for the solution of the Markov Decision Process (MDP). This model consists of a policy, value function, and action-value function. Specifically, the ENMPC scheme is used to approximate the MDP action-value function, which is a departure from the classical use of Deep Neural Networks (DNN) in RL.

In this work, we propose an integrated data-driven learning strategy that builds upon the works of [3] and incorporates the concepts of RTO and Modifier Adaptation to enhance the performance of the control system both in transient and steady-state. Our goal is to achieve optimal economic performance in the presence of modeling errors and unforeseen disturbances.

The proposed framework consists of two main parts: (i) A parameterized RTO layer that uses modifier terms in the steady-state model, stage cost, and constraints, and (ii) A parameterized lower layer ENMPC scheme that uses modifier terms in the dynamic model, stage cost, terminal cost, and constraints. The RTO layer provides optimal steady-state inputs that can be directly incorporated into the lower layer ENMPC scheme as a terminal penalty. RL tools will seek to adjust the parameters shared between both the layers, to improve the performance of the RTO and the ENMPC scheme in synergy.

To evaluate the proposed approach, we implement three control strategies on an experimental rig that emulates a sub-sea oil well network. The first is a naive ENMPC approach that uses a nominal model, the second is an RL-MPC approach that uses a single parameterized ENMPC scheme as shown in [3], and the third is the proposed approach consisting of RTO and ENMPC, which we refer to as RTO-RLMPC. We compare the experimental results in terms of the overall closed-loop performance of the control policy, and the results demonstrate the effectiveness of the proposed approach in improving system performance.

The paper is organized as follows: Section II briefly introduces concepts from RTO, MA, and machine learning in process control to handle plant-model mismatch. Introduction for Reinforcement Learning in process control and fundamental results regarding RL-MPC have been presented in Section II. The proposed approach RTO-RLMPC has been presented in Sections IV. Experimental implementation of the proposed approach and results are discussed in Sections V-VI respectively. The paper concludes in Section VIII.

II. Reinforcement Learning

Machine learning models and tools have been proposed in the literature as as a combat method for plant-model mismatch. As previously mentioned, one potential cause of plant-model mismatch is the need for a deep understanding of the system to obtain models based on first-principle relationships. Conversely, machine learning-based models require only minimal understanding of the inner workings of the system during their development. If the available data of the system is rich enough, a proper representation of the system under investigation can be obtained using a data-driven paradigm [6].

In the RL framework, an agent enacts a control policy and receives only partial information about the effectiveness of its control strategy. One of the major challenges in RL is the development of an action-value function Q, that describes the value or quality of being in a particular state and making a particular control policy decision. Over time, the agent learns and refines this Q function, improving its ability to make better decisions.

Recent studies have focused exclusively on using RL to address plant-model mismatch and have used generic function approximators such as Deep Neural Networks (DNNs) for policy and value function approximations, as in [7]–[10].

One of the keys to a successful implementation of RL in control problems is the selection of an appropriate value function approximation. Using a generic function approximator for e.g. conventional deep neural networks for the policy approximation limits the ability to formally analyze and deliver certificates on the behavior of the generated optimal policy. The authors [3] propose to use a parameterized ENMPC scheme instead of a conventional DNN for policy and value function approximation in RL, so that the rich theory underlying ENMPC can be used to deliver certificates on the behavior of the policy. In the following part, we briefly illustrate the basic structure of RL.

A. Markov Decision Process

The framework of the Markov Decision Process (MDP) is not commonly used in the RTO literature. The MDP is a mathematical framework used in reinforcement learning, where an agent makes decisions in an environment to maximize a reward, considering the current state and possible actions.

In RL literature, it is common to use $\mathbf{s} \in \mathbb{R}^{n_x}$ and $\mathbf{a} \in \mathbb{R}^{n_u}$ for state-input pair. In this paper, we use $\mathbf{x}, \mathbf{u}$ to represent predicted states and inputs, whereas $\mathbf{s}, \mathbf{a}$ are used for observed states and applied actions to the true system. At each discrete time instant, a policy delivers an action to be applied to the system based on the current system state, or recent past observations, i.e. at time instant k, the action $\mathbf{a}_k$ is provided by:

$$\mathbf{a}_k = \boldsymbol{\pi}\left(\mathbf{s}_k\right) \tag{1}$$

where $\mathbf{s}_k$ is the current state of the system. Over the closed-loop trajectories of the system subject to policy $\boldsymbol{\pi}$, a stage cost

$$L\left(\mathbf{s}_k, \mathbf{a}_k\right) \in \mathbb{R} \tag{2}$$

is collected. The objective of solving the MDP is then to find the optimal policy $\boldsymbol{\pi}^\star$ that minimizes a cumulative cost:

$$\boldsymbol{\pi}^\star = \arg\min_{\boldsymbol{\pi}} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k L\left(\mathbf{s}_k, \mathbf{a}_k\right) \middle| \mathbf{a}_k = \boldsymbol{\pi}\left(\mathbf{s}_k\right)\right] \tag{3}$$

where the expected value is taken over the (possibly) stochastic closed-loop trajectories of the system, and $\gamma \in (0, 1]$ is a discount factor.

In that context, RL is a set of tools that allows one to find arbitrarily close approximations of $\boldsymbol{\pi}^\star$ without the need of building a model of the system at hand, using purely observations of the state-input trajectories and the associated stage costs [11]. For example, RL can be used to train unsupervised tasks such as autonomous driving [12], operations-research flavored optimal control [13], and robotics [14], [15].

III. Plant-model mismatch in RTO

In a typical RTO setup, process performance is optimized based on a steady-state rigorous nonlinear model based on first-principles [1]. However, developing a detailed model requires a thorough knowledge of the system, specially if the process has highly complex nonlinear behavior and, as such, plant-model mismatch and model uncertainty is very likely to occur. Therefore, the performance of this model-based approach is highly dependent on the quality of the model used. A typical formulation of an RTO problem reads as,

$$\bar{\mathbf{x}}, \bar{\mathbf{u}} \;=\; \arg\min_{\mathbf{x},\mathbf{u}} l_p(\mathbf{x}, \mathbf{u}) \tag{4a}$$

$$\text{s.t} \quad \mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{4b}$$

$$\mathbf{h}(\mathbf{x}, \mathbf{u}) \leq 0 \tag{4c}$$

The most common approach to handle plant-model mismatch is to adapt the model parameters so that the model outputs fit the available process measurements. Then, the adapted model is used to compute the new optimal operating point. This approach, called two-step RTO or model parameter adaptation (MPA) [16], [17], does not guarantee that the model will perform adequately for use within a process optimization framework [18].

For instance, if the model structure matches the plant (i.e., there is only *parametric* plant-model mismatch) and the parameters can be accurately estimated, the resulting RTO framework is able to converge to the ”true” plant optimum. However, due to the complexity of large-scale industrial processes, it is very difficult to know the true model structure, and cases with only parametric plant-model mismatch are rare in practice [19]. In this case, the solution computed by the RTO framework is likely to be suboptimal and may lead to constraint violations [20]. Since feasibility is a key aspect of any practical process optimization method, practitioners tend to introduce back-offs from the constraints, which further decreases the RTO benefits [19].

Convergence of the two-step RTO under *structural* plant–model mismatch has been discussed by, for example, [21]. Here, the authors proposed a point-wise model adequacy criterion for the RTO model. The conditions assume stationarity of the model-based optimization gradient and negative definiteness of the Hessian in the reduced space of the optimization problem. Pragmatically, however, simply adapting the parameters of a ”simplified” model is very unlikely to mitigate plant-model mismatch in rigorous mathematical terms [18], [19].

1) Modifier Adaptation (MA): Due to the limitations of the conventional two-step RTO approach in handling structural plant-model mismatch, several variants have been proposed in the literature. One such variant is Modifier Adaptation (MA) [2], which imposes stronger mathematical requirements to guarantee optimality of process operation. With MA, the resulting RTO framework is able to predict the optimality conditions of the plant optimization problem, specifically the Karush-Kuhn-Tucker conditions, and steer the operation towards the true plant optimum.

To understand the concept of the classical MA framework, we first present the formulation of the optimization problem. We assume that the plant can be represented by a steady-state map $\boldsymbol{y}_p(\boldsymbol{u}, \boldsymbol{d}_p)$, in which $\boldsymbol{u} \in \mathbb{R}^{n_u}$ are the decision variables, $\boldsymbol{d}_p \in \mathbb{R}^{n_d}$ the process disturbances, and $\boldsymbol{y}_p \in \mathbb{R}^{n_y}$ the plant measurements. However, we only have access to an approximation of the plant map $\boldsymbol{y}(\boldsymbol{u}, \boldsymbol{\theta})$, i.e. the process model parametrized by a set of parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$. In the traditional MPA formulation, we try to find the "true" plant optimum at time k by solving:

$$\min_{\boldsymbol{u} \in \mathbb{U}} \quad \phi(\boldsymbol{u}, \boldsymbol{y}(\boldsymbol{u}, \hat{\boldsymbol{\theta}}_k)) \tag{5a}$$

$$\text{s.t.} \quad \boldsymbol{g}(\boldsymbol{u}, \boldsymbol{y}(\boldsymbol{u}, \hat{\boldsymbol{\theta}}_k)) \leq \boldsymbol{0} \tag{5b}$$

in which, $\mathbb{U}$ is a the input constraint set, $\phi : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \rightarrow \mathbb{R}$ is an economic cost function, and $\boldsymbol{g} : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \rightarrow \mathbb{R}^{g}$ are the system constraints. As discussed, only adapting $\boldsymbol{\theta}$ may not suffice to deal with plant-model mismatch, and the solution of the optimization problem in Eq.(5) may not optimal (or even feasible) for the plant. Hence, in MA, instead of solely adapting the model parameters to represent the current plant state, we add modifiers to the nominal process model $\boldsymbol{y}(\cdot, \boldsymbol{\theta}_{nom})$ and solve the modified optimization problem:

$$\min_{\boldsymbol{u} \in \mathbb{U}} \phi(\boldsymbol{u}, \boldsymbol{y}(\boldsymbol{u}, \boldsymbol{\theta}_{nom})) + \boldsymbol{\lambda}_{\phi,k}^T(\boldsymbol{u} - \boldsymbol{u}_k) \tag{6a}$$

$$\text{s.t. } \boldsymbol{g}(\boldsymbol{u}, \boldsymbol{y}(\boldsymbol{u}, \boldsymbol{\theta}_{nom})) + \boldsymbol{\epsilon}_{g,k} + \boldsymbol{\lambda}_{g,k}^T(\boldsymbol{u} - \boldsymbol{u}_k) \leq \boldsymbol{0} \tag{6b}$$

where, $\boldsymbol{\epsilon}_{g,k} \in \mathbb{R}^{n_g}$ are the zeroth-order modifiers and correspond to the difference between the constraints measured in the plant and the process model predictions at time k:

$$\boldsymbol{\epsilon}_{g,k} = \boldsymbol{g}(\boldsymbol{u}_k, \boldsymbol{y}_p(\boldsymbol{u}_k, \boldsymbol{d}_{p,k})) - \boldsymbol{g}(\boldsymbol{u}_k, \boldsymbol{y}(\boldsymbol{u}_k, \boldsymbol{\theta}_{nom})) \tag{7}$$

whereas the first-order modifiers represent the differences between the plant and model gradients at time k:

$$(\boldsymbol{\lambda}_{\phi,k}^T) = \frac{\partial \phi(\boldsymbol{u}, \boldsymbol{y}_p)}{\partial \boldsymbol{u}}\bigg|_{\boldsymbol{u}_k, \boldsymbol{d}_{p,k}} - \frac{\partial \phi(\boldsymbol{u}, \boldsymbol{y})}{\partial \boldsymbol{u}}\bigg|_{\boldsymbol{u}_k, \boldsymbol{\theta}_{nom}} \tag{8a}$$

$$(\boldsymbol{\lambda}_{g,k}^T) = \frac{\partial \boldsymbol{g}(\boldsymbol{u}, \boldsymbol{y}_p)}{\partial \boldsymbol{u}}\bigg|_{\boldsymbol{u}_k, \boldsymbol{d}_{p,k}} - \frac{\partial \boldsymbol{g}(\boldsymbol{u}, \boldsymbol{y})}{\partial \boldsymbol{u}}\bigg|_{\boldsymbol{u}_k, \boldsymbol{\theta}_{nom}} \tag{8b}$$

It has been shown that, under some mild assumptions regarding model $\boldsymbol{y}(\cdot, \boldsymbol{\theta}_{nom})$ adequacy, the MA scheme is able to reach the "true" plant optimum by iteratively solving the problem in Eq.(6), see [22] for details. However, the convergence to the true plant optimum may be slow. For dealing with these issues, some authors (for instance, [23] and [24]) studied the effect of solving the MA optimization problem above while also updating the model parameters to improve the performance of the iterative MA scheme, that is using $\hat{\boldsymbol{\theta}}_k$ instead of $\boldsymbol{\theta}_{nom}$.

Apart from the rate of convergence, the practical implementation of MA can be challenging since the computation of the first-order modifiers need plant gradients estimates, which may require a series of time-consuming experimental measurements in order to evaluate them.

A. *Integration of RTO and MPC*

In the RTO approach, only steady-state optimization is taken into account. As shown in Fig. 1, RTO systems send their optimal decisions to a lower level MPC, which is responsible for setpoint control. This two-level implementation strategy presents numerous difficulties that limit the achievable flexibility and overall economic benefit [25]. For instance, RTO strategies usually use a nonlinear stationary system representation whereas MPC generally relies on linear dynamic models. These two different model paradigms may present inconsistencies such as different steady-state gains, which may affect the system abilities to handle unmeasured disturbances, decreasing the stability of the RTO system [26].

To address this issue, a single-layer strategy combining RTO and MPC has been proposed. Initial work in this field (e.g., [27]–[29]) focused primarily on solving the economic optimization problem at the MPC level by considering an objective function that combines plant economics and control tracking, balanced with appropriate weights. Here, the optimal steady-state conditions of the plant are still computed using a rigorous nonlinear process model, while the trajectory to be followed is predicted using a linear dynamic model. The biggest challenge of these preliminary works was to provide a workable solution of the resulting control/optimization problem at each sampling step.

In this paper, our focus is on utilizing an ENMPC scheme to handle transients. Addressing model uncertainties and stochasticity within the ENMPC layer remains a less-explored area. We aim to leverage recent theories that combine ENMPC and Reinforcement Learning to optimize these transients in the ENMPC layer [3].f

1) Economic Nonlinear MPC: Recent advances in the development of efficient large-scale NLP solvers [30] together with strategies for fast control move updates based on NLP sensitivity information [31] and closed-loop stability and convergence theory [32] allowed the development and deployment of economic nonlinear MPCs (ENMPC). The optimization problem associated with ENMPC is generally designed based on a nonlinear dynamic model describing the plant, and the objective function could be based purely on economics [33], or a hybrid between cost and control performance [34]. The ENMPC optimization problem can be formulated as:

$$\min_{\boldsymbol{u}\in\mathbb{U}} \quad \sum_{k=0}^{N-1} \Phi\left(\boldsymbol{u}_k, \boldsymbol{y}_k\right) + V^{\mathrm{f}}\left(\boldsymbol{y}_N\right) \tag{9a}$$

$$\text{s.t.}$$

$$\boldsymbol{x}_{k+1} = F\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}_{nom}\right) \qquad k = 0, \dots N-1 \tag{9b}$$

$$\boldsymbol{y}_k = H\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}_{nom}\right) \qquad k = 0, \dots N-1 \tag{9c}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}(0) \tag{9d}$$

$$G\left(\boldsymbol{u}_k, \boldsymbol{y}_k\right) \leq 0, \qquad k = 0, \dots N-1 \tag{9e}$$

$$G^{\mathrm{f}}(\boldsymbol{y}_N) \leq 0 \tag{9f}$$

where, $\boldsymbol{x} \in \mathbb{R}^{n_x}$ are the system states, $F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x}$ and $H : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_x}$ are the nonlinear dynamic model of the system. $G : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_g}$ and $\Phi : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}$ are the discrete-time version of the constraints and objective function. V^{f} and G^{f} are the terminal costs and constraints. N is the prediction horizon and $x(0)$ is the states at the current time.

Note that tracking the optimal steady set points (in an economical sense) and computing the input trajectory as in Eq. (9) can lead to significantly differences operation strategies, even for nominal cases [35]. To ensure that Economic NMPC converges to steady state, typically one has to enforce terminal costs and constraints to the optimization problem above.

Nevertheless, enforcing convergence to a steady-state is not sufficient to handle plant-model mismatch. Similarly to the steady-state case, the use of a fixed dynamic nominal model is typically insufficient to drive the plant to optimality, even with more frequent feedback from the plant to compensate for the model inaccuracies [36]. In order to deal with this issue, several authors proposed blending ideas from Modifier Adaptation and Economic MPC (e.g., [37], [38], and [39]). Unfortunately, the proposed frameworks still require estimation of plant gradients.

2) ENMPC as a Function Approximator: In this section we explain how to approximate the optimal policy and value functions $\pi_\star, V_\star, Q_\star$ by adjusting the stage cost and constraints in NMPC, even when using an incorrect model of the system.

We formulate the parameterized ENMPC scheme, which can be used as a function approximator in RL. We consider the parameterization of the value function $V_\star$ using the following

ENMPC scheme parameterized by $\boldsymbol{\theta}$,

$$\begin{aligned} V_\theta(\mathrm{s}) = \min_{\boldsymbol{u},\boldsymbol{x},\boldsymbol{\sigma}} \quad & \lambda_\theta(\boldsymbol{x_0}) + \gamma^N \left(V_\theta^{\mathrm{f}}(\boldsymbol{x_N}) + w_{\mathrm{f}}^\top \boldsymbol{\sigma}_{\mathbf{N}}\right) \\ & + \sum_{k=0}^{N-1} \gamma^k \left(\ell_\theta(\boldsymbol{x_k}, \boldsymbol{u_k}) + w^\top \boldsymbol{\sigma_k}\right) && \text{(10a)} \\ \text{s.t.} \quad & \boldsymbol{x_{k+1}} = f_\theta\left(\boldsymbol{x_k}, \boldsymbol{u_k}\right), \quad \boldsymbol{x_0} = \mathrm{s}, && \text{(10b)} \\ & g\left(\boldsymbol{u_k}\right) \leq 0, && \text{(10c)} \\ & h_\theta\left(\boldsymbol{x_k}, \boldsymbol{u_k}\right) \leq \boldsymbol{\sigma_k}, \quad h_\theta^{\mathrm{f}}(\boldsymbol{x_N}) \leq \boldsymbol{\sigma_N}. && \text{(10d)} \end{aligned}$$

where ℓ_θ, V_θ^{f} are the parameterized stage cost and terminal cost respectively whereas f_θ, h_θ contain the model and constraint function parameterization. The extra cost λ_θ is to bridge the gap between ENMPC stability theory and RL tools. For a detailed explanation on the function of λ_θ, the reader is directed to [3, Section V]. The parameterized value function in (10) is a classic ENMPC formulation if $\gamma = 1$ and $\lambda_\theta = 0$, see [40].

Note that we will use $\boldsymbol{x}, \boldsymbol{u}$ for the ENMPC predicted states in the rest of this paper. The slack variables $\boldsymbol{\sigma}_{\mathbf{k}}$ are used for constraint relaxation with w, w_{f} being the weights for relaxation.

The parameterized ENMPC gives the policy,

$$\pi_\theta(\mathrm{s}) = \boldsymbol{u}_{\mathbf{0}}^\star, \tag{11}$$

where $u_0^\star$ is the first element of the input sequence $\boldsymbol{u}_{\mathbf{0}}^\star, \ldots, \boldsymbol{u}_{\boldsymbol{N-1}}^\star$ of the solution of (10) for a given state s. The action-value function $Q_\theta(\mathrm{s}, \mathbf{a})$ is given as,

$$\begin{aligned} Q_\theta(\mathrm{s}, \mathbf{a}) = \min_{\boldsymbol{u},\boldsymbol{x}} \quad & (10) && \text{(12a)} \\ \text{s.t.} \quad & (10\text{b}) - (10\text{d}), && \text{(12b)} \\ & \boldsymbol{u_0} = \mathbf{a}. && \text{(12c)} \end{aligned}$$

The proposed parameterization satisfies the fundamental equalities underlying the Bellman equations [41]. More specifically, one can verify that the policy $\pi_\theta(\mathrm{s})$, the value function $V_\theta(\mathrm{s})$ and the action-value function $Q_\theta(\mathrm{s}, \mathbf{a})$ satisfy,

$$\pi_\theta(\mathrm{s}) = \arg\min_{\mathbf{a}} Q_\theta(\mathrm{s}, \mathbf{a}), \quad V_\theta(\mathrm{s}) = \min_{\mathbf{a}} Q_\theta(\mathrm{s}, \mathbf{a}). \tag{13}$$

The goal of RL is to find the parameters $\boldsymbol{\theta}$ that maximizes the closed loop performance under the policy $\pi_\theta(\mathrm{s})$.

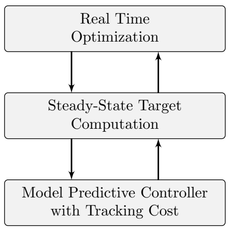
Real Time Optimization
Steady-State Target Computation
Model Predictive Controller with Tracking Cost

Fig. 2. Typical combination of RTO and MPC layers.

IV. Proposed Two-Step Iterative Learning

Using a parameterized ENMPC scheme with modifications in the stage cost, terminal cost and constraints, in theory it is possible to generate an optimal policy for the incorrect model. However, the ENMPC scheme proposed in [3] is focused on performance improvements during transients and steady state is not considered. Therefore, the ENMPC may converge to a non-economically steady-state point or to a different steady-state than desired at the same time [42].

In a conventional hierarchical control structure that combines an RTO with economic cost and an MPC with tracking cost, the two layers, RTO and MPC are combined so that the upper layer provides setpoints that are tracked by the lower layer MPC, see Fig. 2 [1], [43]–[45].

In order to achieve an offset-free MPC, we propose to include parameterized modifier terms in the RTO layer such that the attained equilibrium is the true optimum for the plant. Additionally, we also parameterize the lower layer economic MPC such that RL will attempt to tune the parameters from the MPC layer as well as the RTO layer subsequently. The proposed RL learning framework is able to compensate for unknown disturbances and reach the true optimum for the plant.

To reduce the complexity of the proposed framework, one could argue that the optimal control policy can be obtained by simply using a parameterized upper layer RTO instead of the economic MPC and using a tracking MPC in the lower layer in a more traditional sense. Similarly one may also argue that, instead of having an economic NMPC, a parameterized tracking formulation of the MPC such as (10) can be used with appropriate modifications in a traditional setup of RTO and MPC. In both cases, learning may introduce inconsistencies between the two simulation models (RTO and MPC), and the complete structure may still suffer from poor closed loop performance.

Another naive approach would be to tune both the layers separately, i.e. generate an optimal policy for the RTO scheme as well as another optimal policy for the tracking MPC using RL. This approach would make the framework extremely complex and the time required to learn both the policies would be substantial. Furthermore, the simulation models in RTO and the tracking MPC would still be inconsistent and it would be difficult to track the setpoints even in the absence of any kind of plant-model mismatch.

A. *Parameterized RTO with Modifier terms*

In this section, we explain the proposed method on the modifier-adaptation technique. For a detailed discussion of modifier-adaptation, we refer the interested readers to [2], [22].

Consider a process described by a discrete-time nonlinear model,

$$\mathbf{x}_{k+1} = \mathbf{f}_\theta(\mathbf{x}_k, \mathbf{u}_k), \tag{14a}$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k),$$

where $\mathbf{u}_k \in \mathbb{R}^{n_u}$ denotes the decision variables; $\mathbf{y}_k \in \mathbb{R}^{n_y}$ are the measured output variables; and $\mathbf{x}_k \in \mathbb{R}^{n_x}$ are the states at time step k respectively. The model is parameterized by a set of time-varying parameters represented by $\boldsymbol{\theta} \in \mathbb{R}^{n_m}$. The model equations are represented by $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_m} \to \mathbb{R}^{n_x}$ and $\mathbf{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$.

We will now consider the parameterization of the steady-state optimization scheme parameterized by $\boldsymbol{\theta}$ with some additional modifier terms as,

$$\mathbf{u}^\star_{k+1}, \mathbf{y}^\star_{k+1} = \arg\max_{\mathbf{u}} \;\; \lambda_\theta(\mathbf{y}) + \bar{\Phi}(\mathbf{y}, \mathbf{u}) + (\lambda^\phi_\theta)^\top_k (\mathbf{u} - \mathbf{u}_k) \tag{15a}$$

$$\text{s.t.} \quad \mathbf{y} = \mathbf{f}_\theta(\mathbf{x}, \mathbf{u}), \tag{15b}$$

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}}, \tag{15c}$$

$$\mathbf{g}_\theta(\mathbf{y}, \mathbf{u}) + (\boldsymbol{\epsilon}^{\mathbf{g}}_\theta)_k + (\lambda^{\mathbf{g}}_\theta)^\top_k (\mathbf{u} - \mathbf{u}_k) \leq 0, \tag{15d}$$

where $\bar{\Phi} : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}$ is the economic objective function, $\mathbf{g}_\theta : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_c}$ describes the vector of parameterized non-linear constraints, the bounds on the input are denoted separately in (15c). Note that $\bar{\Phi}$ and $\mathbf{g}$ are not directly dependent on the parameters $\boldsymbol{\theta}$, but implicitly via the process outputs $\mathbf{y}$ governed by the model (14).

The first-order modifier terms $(\lambda^\phi_\theta)^\top$ and $(\lambda^{\mathbf{g}}_\theta)^\top$ and the zero-order correction term $\boldsymbol{\epsilon}^{\mathbf{g}}_\theta$ included in the constraint function of the optimization problem to adjust the model cost and constraint

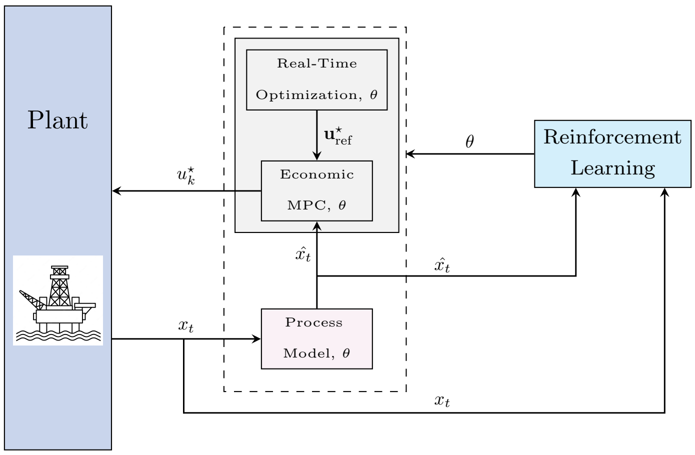

Fig. 3. Schematic representation of the integration of reinforcement learning with real-time optimization and economic MPC.

functions due to the presence of uncertainty. Note that, if the cost and constraints are the *known* functions of the inputs $\mathbf{u}$, then the corresponding correction terms are zero, since no model adaptation is required.

The above mentioned terms closely represent the correction terms used in conventional modifier adaptation approaches, wherein the terms are evaluated using the plant gradient information as described in [21], [46], [47]. The plant gradients $\frac{\partial \bar{\Phi}_p}{\partial \mathbf{u}}\mathbf{u}_k$ and $\frac{\partial \mathbf{g}_p}{\partial \mathbf{u}}\mathbf{u}_k$ are *assumed* to be available at $\mathbf{u}_k$ where $(\cdot)_p$ is used for variables associated with the plant. However, RL does not require the plant gradient estimation to adjust these correction terms and can learn simply by observing the transient input-output data.

At the k^{th} RTO iteration, the next optimal inputs $\mathbf{u}^{\star}_{k+1}$ are computed by solving (15) which can directly enter (10) as a terminal penalty. While several formulations of the economic MPC are possible, in this particular work, we use a terminal equality constraint to achieve asymptotic stability, see [48].

B. $Q-$Learning for ENMPC

The classical $Q-$ Learning is a simple yet powerful tool to adjust the parameters $\boldsymbol{\theta}$ in the parameterized ENMPC scheme (10). In its simplest form, $Q-$ Learning minimizes the temporal-difference error,

$$\tau_k = L(\mathbf{s}_k, \mathbf{a}_k) + \gamma V_\theta(\mathbf{s}_{k+1}) - Q_\theta(\mathbf{s}_k, \mathbf{a}_k), \tag{16a}$$

where,

$$L(\mathbf{s}_k, \mathbf{a}_k) = \ell_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k) + w^\top \max\left(0, h_\theta\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right)\right), \tag{16b}$$

where $\boldsymbol{u}_k$ is selected according to the NMPC policy $\pi_\theta(\mathbf{s})$ and the parameter update reads as,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\tau_k \nabla_\theta Q_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{16c}$$

where $\alpha > 0$ is the step size or the learning rate used commonly in stochastic gradient-based approaches.

The evaluation of the gradient $\nabla_\theta Q_\theta$ requires one to compute the sensitivities of the action-value function (12) and is briefly discussed in [3]. The RL parameters $\boldsymbol{\theta}$ impact Q_θ directly via the ENMPC scheme and indirectly via the RTO scheme in (15), by modifying the optimal input $\mathbf{u}^\star$ which enters as a terminal constraint in the ENMPC scheme. The gradient $\nabla_\theta Q_\theta$ associated to the proposed RTO-RLMPC scheme is given by the following total derivative:

$$\frac{\partial Q_\theta}{\partial \theta} = \frac{\partial Q_\theta}{\partial \theta} + \frac{\partial Q_\theta}{\partial \mathbf{u}^\star}\frac{\partial \mathbf{u}^\star}{\partial \theta} \tag{17}$$

In the next section, we briefly review on how to compute these sensitivities for (12) and (15).

C. Sensitivities for the RTO-RLMPC Scheme

To compute the sensitivities, we denote the Lagrange function associated with the ENMPC Problem (12) and the RTO Problem (15) as $\mathcal{L}_\theta$ and $\mathcal{L}_\theta^\diamond$ respectively:

$$\mathcal{L} = \Phi_\theta + \lambda^\top H_\theta + \mu^\top G_\theta, \tag{18a}$$

$$\mathcal{L}^\diamond = \bar{\Phi}_\theta + \bar{\lambda}^\top \bar{H}_\theta + \bar{\mu}^\top \bar{G}_\theta, \tag{18b}$$

where Φ, $\bar{\Phi}$ are the cost of the ENMPC and RTO problems, and $\lambda, \bar{\lambda}$ are the Lagrange multipliers associated with the equality constraints $H_\theta, \bar{H}_\theta$. Variables $\mu, \bar{\mu}$ are the Lagrange multipliers for the inequality constraints $G_\theta, \bar{G}_\theta$. Furthermore, we also introduce $\mathbf{y} = \{\mathbf{p}, \boldsymbol{\lambda}, \boldsymbol{\mu}\}$ and $\bar{\mathbf{y}} = \{\bar{\mathbf{p}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\mu}}\}$ as the vector of the primal-dual variable associated to the ENMPC and RTO problems, where $\mathbf{p}, \bar{\mathbf{p}}$ denote the vector of primal variables. The sensitivities of the ENMPC scheme can be computed as described in [3] as,

$$\frac{\partial Q_\theta}{\partial \theta} = \frac{\partial \mathcal{L}_\theta(\mathbf{y}^\star)}{\partial \theta}, \tag{19}$$

where $\mathbf{y}^\star$ is the primal-dual solution vector of (12).

The sensitivity $\frac{\partial \mathbf{u}^\star}{\partial \theta}$ associated to the RTO scheme in (15) can be obtained using the Implicit Function Theorem (IFT) on the Karush Kuhn Tucker (KKT) conditions underlying the parametric Nonlinear Parametric Programming (NLP). Assuming that the linear independence constraint qualification (LICQ), second order sufficient condition (SOSC), and strict complementarity (SC) hold at $\bar{\mathbf{y}}^\star$ [49], then,

$$\frac{\partial \bar{\mathbf{y}}^\star}{\partial \theta} = -\frac{\partial R_\theta}{\partial \bar{\mathbf{y}}}^{-1} \frac{\partial R_\theta}{\partial \theta}\bigg|_{\mathbf{u}^\star}, \tag{20a}$$

where

$$R_\theta = \begin{bmatrix} \nabla_{\bar{\mathbf{p}}} \bar{\mathcal{L}}_\theta \\ \bar{H}_\theta \\ \boldsymbol{\mu}^{\star\top} \bar{G}_\theta \end{bmatrix} = 0, \tag{20b}$$

are the KKT conditions associated to the parameterized RTO in (15). Since $\mathbf{u}^\star$ is a part of $\bar{\mathbf{y}}^\star$, the sensitivity of the associated RTO problem ,$\frac{\partial \mathbf{u}^\star}{\partial \theta}$ as required in (17) can be extracted from the matrix $\frac{\partial \bar{\mathbf{y}}^\star}{\partial \theta}$.

V. Case Study: Subsea Oil-Well Network

The goal of a subsea oil drilling network is to extract natural oil and gas trapped beneath the geological structures of the seabed. This oil and gas can be extracted by drilling a network of wells into the seafloor and transporting the extracted oil and gas to the seafloor through long vertical pipes called risers. The pressure in the reservoir is usually sufficient to move the fluids to the surface. However, if the pressure is not sufficient to lift the fluids to the surface, artificial gas lift methods such as electric submersible pumps, subsea boosting stations, or gas lift methods can be used. Gas lift is a commonly used method because of its robust design and relatively low operating costs.

In a typical gas-lift approach, a specific compressed gas is injected into the bottom of the well via the annulus to reduce fluid mixture density [50]. The gas injection leads to a decrease in Hydrostatic pressure drop, increasing the flow from the reservoir. However, if too much of the gas is injected, it will increase the frictional pressure drop, creating a negative effect on the flow rate. At one point, this counter effect of the frictional pressure drop will dominate the effect of the positive hydrostatic pressure drop, decreasing the flow from the reservoir [51]. Thus, there is an optimal gas lift injection rate that corresponds to the maximum oil production. In addition,

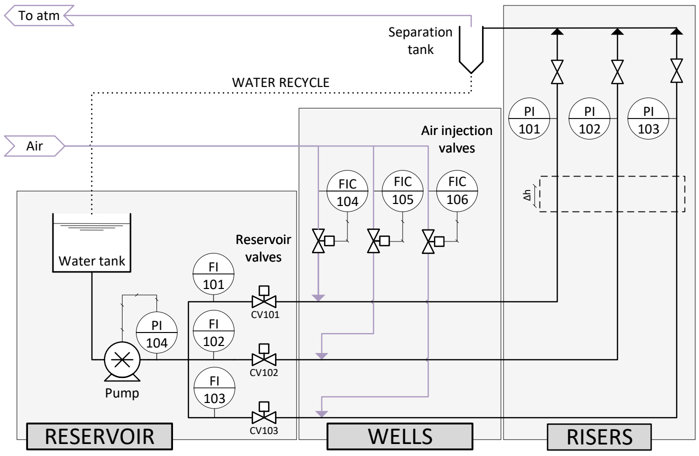

Fig. 4. Experimental schematic adapted from [52]. The input to the system is the gas injection flowrates (FI104, FI105, and FI106) and the measurements are the well top pressures (PI101, PI102 and PI103), the pump outlet pressure (PI104), and the liquid flowrates (FI101, FI102, and FI103). The reservoir valve openings (CV101, CV102, and CV103) act as system disturbances.

the availability of lift gas may be limited by the gas processing capacity at the surface, and this resource must be optimally distributed among the various wells.

Thus, the objective is to find the optimal gas lift injection rate for each well so that the total oil production is maximized, taking into account gas availability constraints.

A. *Experimental rig*

To experimentally validate the proposed method, we use a laboratory scale experimental rig emulating the subsea oil production network used in [52]. For simplicity, the experimental rig uses air and water as a substitute for oil and gas. The choice of working fluids does not affect the gas lift phenomenon. In addition, air and water have been used to replace oil and gas in experimental oil rigs representing subsea oil wells, see e.g [53].

A simplified flowsheet of the rig is depicted in Fig. 4. The system consists of three sections: a reservoir, wells, and the riser section.

The reservoir consists of a stainless steel tank with $200\,\mathrm{L}$ to replicate oil storage in subsea. A centrifugal pump with a controller is also provided to regulate the pump rotation. The outlet pressure (PI104) is kept constant at $0.3\,\mathrm{bar}$ for all experiments.. The setup contains three control valves (CV101, CV102, CV103) whose openings can be manipulated in order to represent disturbances from reservoir, for instance to emulate pressure oscillations or reservoir depletion. This reservoir setup will produce liquid and it's output flow will range from $2\,\mathrm{L\,min^{-1}}$ to $15\,\mathrm{L\,min^{-1}}$. Three flow meters (FI101, FI102, FI103) are located just before the reservoir valves to measure the outflow rates.

The second section consists of wells represented by three flexible hoses having inner diameter of $2\,\mathrm{cm}$ and length of $1.5\,\mathrm{m}$. Air is injected into the wells approximately $10\,\mathrm{cm}$ just after the reservoir valves, with the use of three air flow controllers (FIC104, FIC105, FIC106). The flow controllers work within the range of $1\,\mathrm{sL/min}$ to $5\,\mathrm{sL/min}$.

The riser consists of three vertical pipes having inner diameter $2\,\mathrm{cm}$ and $2.2\,\mathrm{m}$ in length are placed orthogonal to the wells. The pressure in the top is measured using the indicators (PI101, PI102, PI103) followed by three manual valves. A detailed description of the experimental setup can be found in [52].

B. Process Modeling and Model parameterization

The first step in designing a control problem using the proposed methods is to obtain steady-state and dynamic models for the experimental rig. A detailed first principles model was designed in [52].

The proposed methodology, in theory can adapt to structural mismatch given there are some parameters which can be updated. The selection of adjustable parameters should be the ones that can be updated such that the most significant process disturbance are reflected in the model. Since the main disturbances in our experimental rig are associated with the reservoir valves (CV101, CV102 and CV103), it is important to adjust parameters associated with the behavior of these valves. We select the following adjustable parameters in the model,

$$\boldsymbol{\theta}_m = \left[\theta_{res,1}, \theta_{res,2}, \theta_{res,3}, \theta_{top,1}, \theta_{top,2}, \theta_{top,3}\right]^\top , \tag{21}$$

where $\theta_{res,i}$ is the reservoir valve constant and $i = 1, 2, 3$ denote the well number.

VI. Experimental Setup

The objective of this paper is to capture the optimal policy $\pi_\star$ based on an inaccurate model by observing the transient input-output data of the real system. We implement the proposed learning framework for the experimental oil-rig setup. We also present some guidelines and recommendations for the practical implementation of RTO-RLMPC.

A. Dataset

Our experimental dataset is obtained by observing real system dynamics with a sampling time of $T_s = 10\,\mathrm{s}$. The transient input-output dataset consists of 1.0×10^3 data points with each episode of approximately $20\,\mathrm{min}$ having about 1000 data points. The generation of the optimal control policy is based on the closed-loop observation of the dynamics of the model and thus no extra measurement of the real system is required. To improve the generalization across dynamic modes, the data set was chosen such that it encompasses most of the dynamics and disturbances of the real system.

B. Problem Formulation

The optimal operation of the experimental rig is achieved by maximizing the "oil" revenue, while still considering for gas availability constraints and bounds on the gas lift flowrates. The economic objective function J is chosen as [52]:

$$J = 20\,Q_{l,1} + 10\,Q_{l,2} + 30\,Q_{l,3}\,, \tag{22}$$

where Q_l are the liquid flowrates of wells $1, 2$ and 3. For illustrative purposes, it has been assumed that the three wells produce different valued hydrocarbons, reflected by different weights in the cost function. The input vector is given by,

$$\mathbf{u} = \begin{bmatrix} Q_{gl,1} & Q_{gl,2} & Q_{gl,3} \end{bmatrix}^\top, \tag{23}$$

where Q_{gl} are the injected gas flowrates of wells $1, 2$ and 3 respectively. The constraint sets g in (10) and g in (15) are composed by the gas lift injection Q_g lower and upper bounds ($Q_{g,min} = 1\,\mathrm{sL/min}$ and $Q_{g,max} = 5\,\mathrm{sL/min}$) and the gas availability constraints ($Q_{g,1} + Q_{g,2} + Q_{g,3} \leq 7.5\,\mathrm{sL/min}$). The initial value of the inputs is specified as $Q_{g,1} = Q_{g,2} = Q_{g,3} = 2.5\,\mathrm{sL/min}$.

C. Offline Learning

For comparison and analysis, we generate 2 different policies, (i) using the standard RLMPC approach in (10) presented in [3]; (ii) using the proposed RTO-RLMPC approach. For learning the policies, we use the discount factor $\gamma = 0.99$. To introduce a rich exploration, we perturb the optimal feedback as follows,

$$a = \mathrm{sat}(u_0^\star + e, u_l, u_b), \quad e \sim \mathcal{N}(0, 0.1) \tag{24}$$

where $\mathrm{sat}(\cdot, u_l, u_b)$ saturates the input between its lower and upper bounds. The learning concludes when delta error τ in (16a) average outs to zero or max episodes are met. Since the parameters $\boldsymbol{\theta}$ are only locally valid, sometimes they may show aggressive behavior. In this case, different step sizes α shall be used for different parameters or first order filters. The nominal value of the step size was set to $\alpha = 10^{-4}$.

VII. Results

In the previous section, we presented the procedure for learning the optimal policy using the transient input-output data and RL. Before implementing the learned policies on the experimental rig, we initially validated the controller using a high fidelity dynamic model developed in Matlab. The reader is directed to our source code available on GitHub for detailed model source code and parameters.[1]

We begin by implementing an ENMPC using a nominal model, followed by implementing learned control policies RLMPC and RTO-RLMPC on the experimental rig. To compare the performance of the proposed system, we perform experiments each of $20\,\mathrm{min}$. To simulate a disturbance in the process, we vary the opening of valves CV101, CV102, and CV103. The pump outlet pressure P_{pump} is kept constant. The disturbance scenario emulates declining production of the oil wells over time. The bottom-most pane of Fig. 5 depicts the designed disturbance scenario.

The top 3 panes in Fig. 5 shows the profile of manipulated variable Q_{gl}. At the beginning of the experiment i.e. from $t = 0\,\mathrm{min}$ to $t = 6\,\mathrm{min}$, well 3 is prioritized (evident by sudden increase in gas lift flowrate) over well 1 due to larger weight on J. The gas injection in well 2 is kept minimum ($Q_{g,min} = 1\,\mathrm{sL/\,min}$) due to lower values both in terms of valve opening (CV102) and weights on J.

In the next time segment $t = 6\,\mathrm{min}$ to $t = 14\,\mathrm{min}$, the opening of the valve CV101 is gradually decreasing, involuntarily making the vale opening CV103 $>$ CV101, which in turn further increasing the gas injection in well 3 and decreasing in well 1. At the end, the gas injection in well 2 starts to rise as both the valve openings CV101, CV103 $<$ CV102, but still with $Q_{g,3} > Q_{g,1} > Q_{g,2}$.

The profiles in Fig. 5 confirm that all the methods ,*v.i.z.* naive ENMPC, RLMPC, RTO-RLMPC follow the expected behavior. RTO-RLMPC, however, is able to capture more dynamics and is

[1]https://github.com/Process-Optimization-and-Control/ProductionOptRig

more responsive to changes in disturbances. We also note that RLMPC is slower to adapt the inputs in the face of disturbances as compared to ENMPC, which could be due to the RL parameters $\boldsymbol{\theta}$ not being properly tuned.

Fig. 7 shows the constraint satisfaction $(Q_{g,1} + Q_{g,2} + Q_{g,3} \leq 7.5\,\mathrm{sL/min})$ for all the three methods. In general, due to process and measurement noise, the actual system is noisy. In such a noisy environment, tightly controlling the hard constraints is a challenging task. RTO-RLMPC is still able to improve the plant performance without any significant constraint violations.

Comparison of the optimal cost: Fig. 6 shows the comparison of the profit obtained using RL with the naive ENMPC approach. We use a $60\,\mathrm{s}$ moving average for smoothing the profiles to reduce the noise and jitters in the measurements. As expected, RLMPC and RTO-RLMPC have a better performance in terms of cost than the naive ENMPC. We also compute the total difference. We observed that RLMPC and RTO-RLMPC increased the obtained profit by approximately 4.73% and 8.60% respectively when compared with ENMPC.

These studies may not fully capture the scale, complexity, and real-world variabilities present in industrial processes. The equipment and instrumentation used in experimental setups are typically smaller in scale and less sophisticated than those found in industrial settings, where large-scale and complex processes require advanced instrumentation and control strategies. Experimental studies may overlook external environmental factors, while industrial process control must consider the impact of weather, temperature variations, and other environmental conditions. Additionally, safety considerations in experimental studies may be less stringent compared to the strict adherence to safety regulations and standards in industrial settings. Long-term stability, integration with legacy systems, and optimization considering economic and operational constraints are crucial aspects that experimental studies may not fully address. Bridging these gaps is essential for researchers and practitioners to ensure the successful implementation of process control in real industrial environments, considering the complexities and challenges unique to large-scale, dynamic, and unpredictable industrial systems.

VIII. Conclusion

In this paper, we have explored the combination of RTO and economic NMPC in the context of RL. We proposed a framework to merge concepts from modifier adaptation and show that the combined RTO and ENMPC scheme can generate an optimal policy for the real system, even if the underlying model is inaccurate, with modifications in stage cost and constraint function.

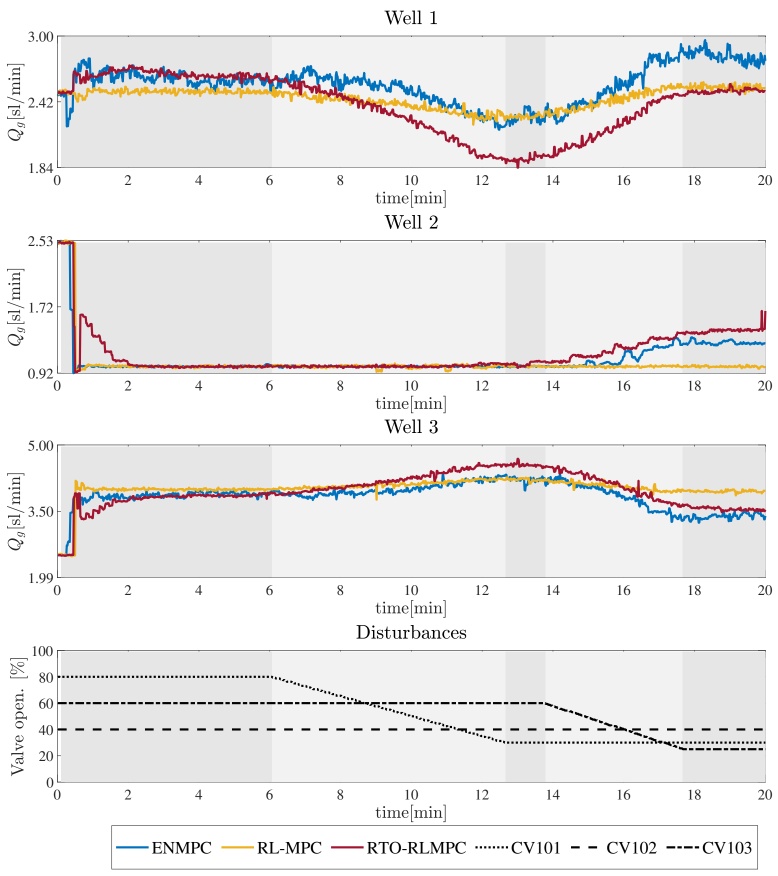

Fig. 5. Gas lift flowrate Q_g in each well for Economic MPC, RLMPC and RTO-RLMPC. Disturbance profile used for simulation is shown in the bottom most pane which depicts declining oil production in wells 1 and 3.

Specifically, we propose to combine two main elements: a tailored formulation of RTO consisting of modifier terms, and a parameterized ENMPC scheme to be solved at each NMPC iteration. We suggest using modifier terms in the RTO scheme to reach the true optimum, providing optimal solution which can be directly given as a terminal penalty in the lower layer ENMPC. Our approach eliminates the crucial element of steady-state plant gradient estimation in MA.

The main contribution of this paper also regards to the experimental validation of the proposed method, motivating the use of RL for potential implementation in industries. Our paper confirmed the previous in-silico findings, experimentally showing that the RL-MPC proposed in [3] is able to

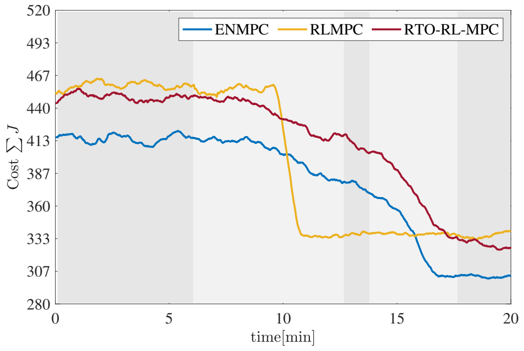

Fig. 6. Comparison of average profit from the experimental rig using various approaches *v.i.z.* ENMPC, RLMPC, RTO-RLMPC.

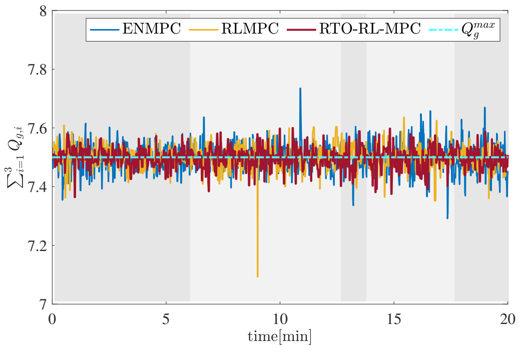

Fig. 7. Constraint satisfaction: $Q_{g,1}+Q_{g,2}+Q_{g,3} \leq 7.5\,\mathrm{sL/min}$ for ENMPC, RLMPC and RTO-RLMPC from the experimental rig

improve the closed loop performance in-spite of having plant-model mismatch. We also validated the ability of the proposed approach on an experimental rig, to converge to the true optimum with an improved performance of over 8.60% as compared to the standard ENMPC when using a nominal model.

References

[1] M. L. Darby, M. Nikolaou, J. Jones, and D. Nicholson, "RTO: An overview and assessment of current practice," *Journal of Process control*, vol. 21, pp. 874–884, 2011.

[2] A. Marchetti, B. Chachuat, and D. Bonvin, "Modifier-adaptation methodology for real-time optimization," *Industrial & Engineering Chemistry Research*, vol. 48, pp. 6022–6033, 2009.

[3] S. Gros and M. Zanon, “Data-Driven Economic NMPC Using Reinforcement Learning,” *IEEE Transactions on Automatic Control*, vol. 65, pp. 636–648, 2019.

[4] S. Gros, M. Zanon, and A. Bemporad, “Safe reinforcement learning via projection on a safe set: How to achieve optimality?” *IFAC-PapersOnLine*, vol. 53, pp. 8076–8081, 2020.

[5] M. Zanon and S. Gros, “Safe reinforcement learning using robust MPC,” *IEEE Transactions on Automatic Control*, vol. 66, pp. 3638–3652, 2021.

[6] D. Bonvin, C. Georgakis, C. Pantelides, M. Barolo, M. Grover, D. Rodrigues, R. Schneider, and D. Dochain, “Linking models and experiments,” *Industrial & Engineering Chemistry Research*, vol. 55, pp. 6891–6903, 2016.

[7] P. Petsagkourakis, I. Sandoval, E. Bradford, D. Zhang, and E. del Rio-Chanona, “Reinforcement learning for batch bioprocess optimization,” *Computers & Chemical Engineering*, vol. 133, p. 106649, 2020.

[8] ——, “Constrained reinforcement learning for dynamic optimization under uncertainty,” *IFAC-PapersOnLine*, vol. 53, pp. 11 264–11 270, 2020.

[9] M. Mowbray, P. Petsagkourakis, A. D. R. Chanona, and D. Zhang, “Safe chance constrained reinforcement learning for batch process optimization and control,” *32nd European Symposium on Computer Aided Process Engineering*, vol. 51, pp. 1039–1044, 2022.

[10] J. W. Kim, B. J. Park, T. H. Oh, and J. M. Lee, “Model-based reinforcement learning and predictive control for two-stage optimal control of fed-batch bioreactor,” *Computers & Chemical Engineering*, vol. 154, p. 107465, 2021.

[11] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, “Reinforcement learning for control: Performance, stability, and deep approximators,” *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018.

[12] M. O’Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, “Scalable end-to-end autonomous vehicle testing via rare-event simulation,” *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, p. 9849–9860, 2018.

[13] D. P. Bertsekas and J. N. Tsitsiklis, “Neuro-dynamic programming: an overview,” *Proceedings of* 1995 *34th IEEE Conference on Decision and Control*, vol. 1, pp. 560–564, 1995.

[14] M. P. Deisenroth, G. Neumann, and J. Peters, “A survey on policy search for robotics,” *Foundations and Trends in Robotics*, vol. 2, pp. 388–403, 2013.

[15] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 2013.

[16] C. Y. Chen and B. Joseph, “On-line optimization using a two-phase approach: An application study,” *Industrial & engineering chemistry research*, vol. 26, pp. 1924–1930, 1987.

[17] S.-S. Jang, B. Joseph, and H. Mukai, “On-line optimization of constrained multivariable chemical processes,” *AIChE Journal*, vol. 33, pp. 26–35, 1987.

[18] B. Chachuat, B. Srinivasan, and D. Bonvin, “Adaptation strategies for real-time optimization,” *Computers & Chemical Engineering*, vol. 33, pp. 1557–1567, 2009.

[19] A. D. Quelhas, N. J. C. de Jesus, and J. C. Pinto, “Common vulnerabilities of RTO implementations in real chemical processes,” *The Canadian Journal of Chemical Engineering*, vol. 91, pp. 652–668, 2013.

[20] H. Yoo, H. E. Byun, D. Han, and J. H. Lee, “Reinforcement learning for batch process control: Review and perspectives,” *Annual Reviews in Control*, vol. 52, pp. 108–119, 2021.

[21] J. F. Forbes and T. E. Marlin, “Model accuracy for economic optimizing controllers: The bias update case,” *Industrial & Engineering Chemistry Research*, vol. 33, pp. 1919–1929, 1994.

[22] A. G. Marchetti, G. François, T. Faulwasser, and D. Bonvin, “Modifier adaptation for real-time optimization—methods and applications,” *Processes*, vol. 4, p. 55, 2016.

[23] A. Ahmad, W. Gao, and S. Engell, “A study of model adaptation in iterative real-time optimization of processes with uncertainties,” *Computers & Chemical Engineering*, vol. 122, pp. 218–227, 2019.

[24] J. Matias and J. Jäschke, “Online model maintenance in real-time optimization methods,” *Computers & Chemical Engineering*, vol. 145, p. 107141, 2021.

[25] J. Kadam, M. Schlegel, W. Marquardt, R. Tousain, D. Van Hessem, J. Van Den Berg, and O. Bosgra, “A two-level strategy of integrated dynamic optimization and control of industrial processes—a case study,” *Computer Aided Chemical Engineering*, vol. 10, pp. 511–516, 2002.

[26] W. Yip and T. Marlin, “The effect of model fidelity on real-time optimization performance,” *Computers & Chemical Engineering*, vol. 28, pp. 267–280, 2004.

[27] M. T. De Gouvêa and D. Odloak, “One-layer real time optimization of lpg production in the FCC unit: procedure, advantages and disadvantages,” *Computers & Chemical Engineering*, vol. 22, pp. S191–S198, 1998.

[28] A. C. Zanin, M. T. De Gouvea, and D. Odloak, “Integrating real-time optimization into the model predictive controller of the FCC system,” *Control Engineering Practice*, vol. 10, pp. 819–831, 2002.

[29] G. De Souza, D. Odloak, and A. C. Zanin, “Real time optimization (RTO) with model predictive control (MPC),” *Computers & Chemical Engineering*, vol. 34, pp. 1999–2006, 2010.

[30] L. T. Biegler, “An overview of simultaneous strategies for dynamic optimization,” *Chemical Engineering and Processing: Process Intensification*, vol. 46, pp. 1043–1053, 2007.

[31] L. T. Biegler and V. M. Zavala, “Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization,” *Computers & Chemical Engineering*, vol. 33, pp. 575–582, 2009.

[32] J. B. Rawlings, D. Angeli, and C. N. Bates, “Fundamentals of economic model predictive control,” *Proceedings of* 2012 *54th IEEE Conference on Decision and Control*, pp. 3851–3861, 2012.

[33] R. Amrit, J. B. Rawlings, and D. Angeli, “Economic optimization using model predictive control with a terminal cost,” *Annual Reviews in Control*, vol. 35, pp. 178–186, 2011.

[34] M. Ellis and P. D. Christofides, “Integrating dynamic economic optimization and model predictive control for optimal operation of nonlinear process systems,” *Control Engineering Practice*, vol. 22, pp. 242–251, 2014.

[35] R. Amrit, J. B. Rawlings, and L. T. Biegler, “Optimizing process economics online using model predictive control,” *Computers & Chemical Engineering*, vol. 58, pp. 334–343, 2013.

[36] D. Bonvin and B. Srinivasan, “On the role of the necessary conditions of optimality in structuring dynamic real-time optimization schemes,” *Computers & Chemical Engineering*, vol. 51, pp. 172–180, 2013.

[37] T. Faulwasser and G. Pannocchia, “Toward a unifying framework blending real-time optimization and economic model predictive control,” *Industrial & Engineering Chemistry Research*, vol. 58, pp. 13 583–13 598, 2019.

[38] M. Vaccari, D. Bonvin, F. Pelagagge, and G. Pannocchia, “Offset-free economic MPC based on modifier adaptation: Investigation of several gradient-estimation techniques,” *Processes*, vol. 9, p. 901, 2021.

[39] E. Oliveira-Silva, C. de Prada, and D. Navia, “Dynamic optimization integrating modifier adaptation using transient measurements,” *Computers & Chemical Engineering*, vol. 149, p. 107282, 2021.

[40] J. B. Rawlings and R. Amrit, “Optimizing process economic performance using model predictive control,” *Springer Berlin Heidelberg*, pp. 119–138, 2009.

[41] D. P. Bertsekas, “Dynamic programming and suboptimal control: A survey from ADP to MPC*,” *European Journal of Control*, vol. 11, pp. 310–334, 2005.

[42] M. Vaccari and G. Pannocchia, “A modifier-adaptation strategy towards offset-free economic MPC,” *Processes*, vol. 5, 2017.

[43] S. Engell, “Feedback control for optimal process operation,” *Journal of process control*, vol. 17, pp. 203–219, 2007.

[44] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[45] A. G. Marchetti, A. Ferramosca, and A. H. González, "Steady-state target optimization designs for integrating real-time optimization and model predictive control," *Journal of Process Control*, vol. 24, pp. 129–145, 2014.

[46] B. Chachuat, A. Marchetti, and D. Bonvin, "Process optimization via constraints adaptation," *Journal of Process Control*, vol. 18, pp. 244–257, 2008.

[47] A. G. Marchetti, T. de Avila Ferreira, S. Costello, and D. Bonvin, "Modifier adaptation as a feedback control scheme," *Industrial & Engineering Chemistry Research*, vol. 59, pp. 2261–2274, 2020.

[48] D. Angeli, R. Amrit, and J. B. Rawlings, "On average performance and stability of economic model predictive control," *IEEE transactions on automatic control*, vol. 57, pp. 1615–1626, 2011.

[49] J. Nocedal and S. J. Wright, *Numerical optimization*, 1st ed. Springer, 1999.

[50] A. BenAmara, "Gas lift: Past and Future," *SPE Middle East Artificial Lift Conference and Exhibition*, pp. 420–425, 2016.

[51] A. Hernandez, *Fundamentals of gas lift engineering: Well design and troubleshooting*. Gulf Professional Publishing, 2016.

[52] J. Matias, J. P. Oliveira, G. A. Le Roux, and J. Jäschke, "Steady-state real-time optimization using transient measurements on an experimental rig," *Journal of Process Control*, vol. 115, pp. 181–196, 2022.

[53] E. Jahanshahi, C. J. Backi, and S. Skogestad, "Anti-slug control based on a virtual flow measurement," *Flow Measurement and Instrumentation*, vol. 53, pp. 299–307, 2017.

3.3 NLP Sensitivites for FAST RLMPC

This section proposes a comprehensive approach to improve the computational efficiency of Reinforcement Learning (RL) based Model Predictive Controller (MPC). Although MPC will ensure controller safety and RL can generate optimal control policies, combining the two requires substantial time and computational effort, particularly for larger data sets. In a typical RL-based MPC and Q learning workflow, two not-so-different MPC problems must be evaluated at each RL iteration, i.e. one for the action-value and one for the value function, which is time-consuming and prohibitively expensive in terms of computations. We employ nonlinear programming (NLP) sensitivities to approximate the action-value function using the optimal solution from the value function, reducing computational time. The proposed approach can achieve comparable performance to the conventional method but with significantly lower computational time. We demonstrate the proposed approach on two examples: Linear Quadratic Regulator (LQR) problem and Continuously Stirred Tank Reactor (CSTR).

Reinforcement Learning Based Economic NMPC using NLP-Sensitivities

Saket Adhau, Dirk Reinhardt, Sigurd Skogestad, Sébastien Gros

I. Introduction

Model predictive control (MPC) has been widely used for the control of dynamic systems due to its ability to optimize future control actions based on a mathematical model of the system [1]. However, uncertainties in model parameters, measurement errors, stochasticity, unmodeled dynamics, and incomplete knowledge of the system can introduce errors, leading to suboptimal control performance. To address these challenges, researchers have proposed combining data-driven and model-based control. In particular, [2], proposed an approach that leverages data-driven techniques to enhance the performance of MPC in complex systems with nonlinear dynamics and uncertainties, without relying solely on a precise mathematical model of the system. This approach will hereafter be termed as RL based MPC or simply RL-MPC.

The approach proposed by [2] can achieve good control performance for complex systems with uncertain and nonlinear dynamics, but it is often computationally inefficient due to the iterative nature of updating the policy parameters. In their approach, an MPC scheme is used to support the parametrization to approximate the action-value function, which requires iterative updates of the policy parameters to converge to an optimal solution. This iterative process can be time-consuming and computationally expensive, especially for systems with high-dimensional state and action spaces. The usual workflow in this approach is to evaluate two optimal control problems (OCP) in a single RL iteration, as both the value function and action-value function

This work was supported by the Research Council of Norway, under the IKTPLUSS program (Project number 299585)

Saket Adhau and Sigurd Skogestad are with the Department of Chemical Engineering, Norwegian University of Science and Technology, 7491, Trondheim, Norway. (e-mail: {saket.adhau, sigurd.skogestad}@ntnu.no)

S. Gros and Dirk Reinhardt are with the Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway (e-mail: sebastien.gros@ntnu.no)

are parameterized using an MPC scheme. As a result, this increases the computational burden, making the approach less efficient for real-time control applications. Additionally, RL algorithms require large amounts of data to train to an acceptable level of performance, further increasing the computational cost.

To mitigate these challenges, we propose a method to reduce the computational burden by approximating the action-value function using nonlinear programming (NLP) sensitivities derived from the optimal solution of the value function. By exploiting these sensitivities, we only need to evaluate one optimal control problem (OCP) in a single RL iteration, instead of two, resulting in a significant reduction of computational effort and time without compromising control performance. This approach is particularly well-suited for complex real-time control applications. We demonstrate our approach on two benchmark methods commonly used in the field of control engineering. We show that even using NLP sensitivities for action-value function approximation can provide comparable control performance to traditional RL methods. This highlights the potential of our method to improve the computational efficiency of RL-based control strategies.

The paper is organized as follows: In Section II, we provide a brief overview of parametric nonlinear programming and sensitivity analysis. Section III introduces background on RL-based MPC. The proposed approach for approximating the OCP associated with the action-value function is presented in Section IV, followed by implementations and numerical examples in Section V. Finally, in Section VI, we conclude the paper.

II. NLP Sensitivities

Nonlinear model predictive control (NMPC) is a widely used control technique in various fields due to its ability to handle nonlinear and non-convex systems. However, it is also known to pose significant computational challenges due to the need to solve a sequence of online optimization problems repeatedly. Additionally, the presence of constraints and nonlinearities can further exacerbate these challenges [3]. To tackle this problem, various real-time NMPC strategies have been proposed, including explicit MPC for e.g. [4], neighboring extremals for e.g. [5], Newton-type controllers for e.g. [6], and controllers based on NLP sensitivities for e.g. [7], [8].

The authors [8] and [9] suggest the use of NLP sensitivities to predict the future state of the plant based on the current control action. This allows for the approximation of the future optimal control problem (OCP) in advance and was later used to solve MPC problems with an economic objective function by [10]. Alternatively, the real-time iteration scheme proposed by [5] approximates the next control law by solving a few quadratic programming (QP) problems, which are considerably faster to evaluate. This real-time iteration (RTI) scheme was also adapted as a function approximator for RL by [11]. This integration of RTI scheme has enabled the implementation of the approach proposed in [2] to a wide range of real-time systems. In the upcoming section, we will delve into a concise explanation of the theory behind NLP sensitivities.

A. Parametric Nonlinear Optimization Problem

Parametric NLP involves solving a nonlinear optimization problem where the objective function and constraints depend on a set of parameters. In this context, we consider a general parametric NLP with both equality and inequality constraints, which can be written as:

$$\mathscr{P}(p) = \min_{\boldsymbol{w}} \quad F(\boldsymbol{w}, p), \tag{1a}$$

$$s.t. \quad c_i(\boldsymbol{w}, p) = 0, \quad \forall i \in \mathcal{E}, \tag{1b}$$

$$g_i(\boldsymbol{w}, p) \leq 0, \quad \forall i \in \mathcal{I}, \tag{1c}$$

where the decision variables are denoted by $\boldsymbol{w} \in \mathbb{R}^{n_{\boldsymbol{w}}}$ and the parameter vector by $p \in \mathbb{R}^{n_p}$. Further, $F : \mathbb{R}^{n_{\boldsymbol{w}}} \times \mathbb{R}^{n_p} \to \mathbb{R}$ is the scalar cost, $\mathcal{I}$ denotes the set of indices of inequality constraints, and $\mathcal{E}$ denotes the set of indices of equality constraints.

The Lagrangian function $\mathcal{L}$ for the problem $\mathscr{P}(p)$ is given by:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}, p) := F(\boldsymbol{w}, p) + \boldsymbol{\lambda}^\top c(\boldsymbol{w}, p) + \boldsymbol{\mu}^\top g(\boldsymbol{w}, p), \tag{2}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the Lagrangian multipliers corresponding to the respective equality and inequality constraints. Suppose that $F(\cdot,\cdot)$, and $g(\cdot,\cdot)$ are continuously differentiable and $\boldsymbol{w}^\star$ is a local optimizer. If the linear independence constraint qualification (LICQ) condition holds at $\boldsymbol{w}^\star$, then the first-order optimality or Karush-Kuhn-Tucker (KKT) conditions for $\mathscr{P}(p)$ are given by

$$\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star, p) = 0, \tag{3a}$$

$$c_i(\boldsymbol{w}^\star, p) = 0, \quad \forall i \in \mathcal{E}, \tag{3b}$$

$$g_i(\boldsymbol{w}^\star, p) \leq 0, \quad \forall i \in \mathcal{I}, \tag{3c}$$

$$\boldsymbol{\mu}_i^\star \geq 0, \quad \forall i \in \mathcal{I}, \tag{3d}$$

$$\boldsymbol{\mu}_i^\star g_i(\boldsymbol{w}^\star, p) = 0, \quad \forall i \in \mathcal{I}. \tag{3e}$$

A point $\boldsymbol{z}^\star(p) := [\boldsymbol{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star]$ of primal-dual variables satisfying the KKT conditions (3) for a given initial parameter p, is called a KKT point for p. If the parameter p is clear from the context, we simply write $\boldsymbol{z}^\star$ instead of $\boldsymbol{z}^\star(p_0)$.

The set of indices of active constraints is denoted by $\mathcal{I}_{\mathbb{A}}$, and is defined as $\mathcal{I}_{\mathbb{A}} = \{i \in \mathcal{I} : g_i(\boldsymbol{w}^\star) = 0\} \cup \mathcal{E}$. However, for a local minimizer of (1) to be a KKT point, a constraint qualification is required [12]. Specifically, the LICQ condition must hold at $\boldsymbol{w}$, which means that the vectors $\{\nabla g_i(\boldsymbol{w})\}_{i \in \mathcal{I}_{\mathbb{A}}}$ are linearly independent. Moreover, the strict complementarity condition holds if $\boldsymbol{\mu}_i > 0$ for all $i \in \mathcal{I}_{\mathbb{A}}$.

This set of KKT conditions is typically expressed as a system of nonlinear equations and inequalities in the primal and dual variables, and is used to solve parametric NLP problems numerically, using optimization algorithms such as sequential quadratic programming (SQP) and interior point methods.

$$\varphi(\boldsymbol{z}^\star(p), p) = \begin{bmatrix} \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{z}^\star(p), p) \\ c(\boldsymbol{w}^\star, p) \\ \boldsymbol{\mu}^{\star\top} g(\boldsymbol{w}^\star, p) \end{bmatrix} = 0. \tag{4}$$

Theorem 1: Let $F(\cdot,\cdot)$, $c(\cdot,\cdot)$ and $g(\cdot,\cdot)$ of the parametric NLP problem $\mathscr{P}(p)$ be twice continuous differentiable in a neighborhood of the KKT point $\boldsymbol{z}^\star(p_0)$. Further, assume strict complementarity (SC), linear independence constraint qualification (LICQ), and second order sufficient condition (SOSC) hold for the solution vector $\boldsymbol{z}^\star(p_0)$. Then,

- $\boldsymbol{z}^\star(p_0)$ is an isolated local minimizer of $\mathscr{P}(p_0)$ and the associated Lagrangian multipliers $\boldsymbol{\lambda}^\star$ are unique.
- For parametric perturbations Δp in the neighborhood of p_0, there exits a unique, continuous and differentiable vector function $\boldsymbol{z}^\star(p_0 + \Delta p)$ which is a local minimizer satisfying SOSC and LICQ for $\mathscr{P}(p_0 + \Delta p)$.

- For parametric perturbations Δp in the neighborhood of p_0, the set of active constraints remain unchanged.

See [13].

The above results allow us to apply the implicit function theorem to (4) at $\boldsymbol{z}^\star(p_0)$, such that:

$$\frac{\partial}{\partial p}\varphi(\boldsymbol{z}^\star(p), p)\bigg|_{p=p_0} = \frac{\partial \varphi}{\partial \boldsymbol{z}}\frac{\partial \boldsymbol{z}^\star}{\partial p} + \frac{\partial \varphi}{\partial p} = 0\,. \tag{5}$$

Since the nominal solution satisfies both the SOSC and LICQ conditions, the KKT matrix (4) at the KKT point $\boldsymbol{z}^\star(p_0)$ is non-singular [12], and hence, can be used to calculate the sensitivity matrix from (5). The first-order estimates of the solution for neighboring problems can then be obtained as:

$$\tilde{\boldsymbol{z}}(p) \approx \boldsymbol{z}^\star(p_0) + \frac{\partial \boldsymbol{z}^\star(p_0)}{\partial p}\Delta p\,, \tag{6}$$

where $\tilde{\boldsymbol{z}}(p)$ is the approximate primal-dual solution of the optimization problem $\mathscr{P}(p_0+\Delta p)$. This is a computationally efficient approach, as the cost of solving this linear system is typically much lower than that of solving $\mathscr{P}(p_0 + \Delta p)$ through a conventional optimization routine, especially when the number of decision variables is large, provided the solution $\boldsymbol{z}^\star(p_0)$ is available. This makes sensitivity analysis a powerful tool in optimizing and controlling complex systems.

Similarly, the first order Taylor expansion of the optimal cost for the problem $\mathscr{P}(p_0 + \Delta p)$ can be expressed as

$$F(\boldsymbol{z}(p)) \approx F(\boldsymbol{z}^\star(p_0)) + \frac{\partial F(\boldsymbol{z}^\star(p_0))}{\partial p}\Delta p\,, \tag{7}$$

where the first order sensitivity derivative of the objective function is given by [7]

$$\frac{\partial}{\partial p}F\left(\boldsymbol{z}^\star(p), p\right)\bigg|_{p=p_0} = \frac{\partial}{\partial p}\mathcal{L}(\boldsymbol{z}^\star(p_0)), \tag{8}$$

and the second order Taylor expansion of the optimal cost is given as

$$\begin{aligned} F(\boldsymbol{z}(p)) \approx F(\boldsymbol{z}^\star(p_0)) &+ \frac{\partial F(\boldsymbol{z}^\star(p_0))}{\partial p}\Delta p \\ &+ \frac{1}{2}(\Delta p)^\top \frac{\partial^2 F(\boldsymbol{z}^\star(p_0))}{\partial p^2}\Delta p. \end{aligned} \tag{9}$$

III. Reinforcement Learning Background

Consider a real system having state transition dynamics described by a Markov Process (MP) with state $\boldsymbol{s}$ and action $\boldsymbol{a}$ and a state transition $\boldsymbol{s}, \boldsymbol{a} \rightarrow \boldsymbol{s}_+$ described by a probability density

$$\mathbb{P}[\boldsymbol{s}_+ | \boldsymbol{s}, \boldsymbol{a}] \,. \tag{10}$$

To model the system as a Markov Decision Process (MDP), we augment this model with a stage cost function $\ell(\boldsymbol{s}, \boldsymbol{a})$ and a discount factor $0 \leq \gamma \leq 1$.

Let us consider a deterministic policy that delivers the control action $\boldsymbol{a} = \pi(\boldsymbol{s})$, giving a Markov Chain distribution τ^{π}. The ultimate goal of RL is to find the best policy $\pi^\star$ by evaluating the cumulative cost of the policy π, i.e. by solving,

$$\pi^\star := \arg\min_{\pi} J(\pi) := \mathbb{E}_{\tau^\pi}\left[\sum_{k=0}^{\infty} \gamma^k \ell\left(\boldsymbol{s}_k, \pi(\boldsymbol{s}_k)\right)\right] . \tag{11}$$

The optimal action-value function $Q^\star$, value function $V^\star$, and optimal policy $\pi^\star(\boldsymbol{s})$ associated to the MDP, are defined by the Bellman equations [14]:

$$Q^\star\left(\boldsymbol{s}, \boldsymbol{a}\right) = \ell\left(\boldsymbol{s}, \boldsymbol{a}\right) + \gamma \mathbb{E}\left[V^\star(s_+) \mid \boldsymbol{s}, \boldsymbol{a}\right], \tag{12a}$$

$$V^\star\left(\boldsymbol{s}\right) = Q^\star\left(\boldsymbol{s}, \pi^\star\left(\boldsymbol{s}\right)\right) = \min_{a} Q^\star\left(\boldsymbol{s}, \boldsymbol{a}\right). \tag{12b}$$

Various RL methods have been proposed in the literature to generate the optimal policy $\pi^\star$. However, in this paper, we will be specifically focusing on the classical $Q-$-learning algorithm.

A. *ENMPC as a Function Approximator*

The use of an ENMPC scheme as a generic function approximator for RL has been proposed by [2]. The authors, have demonstrated that an ENMPC scheme can support the parametrization of the action-value function Q_θ, V_θ and the policy π_θ. The central result is that with some modifications in the stage cost, terminal cost, and constraints, ENMPC scheme can deliver optimal control policy even when the model underlying the MPC scheme is incorrect. We briefly summarize the main results of [2] in the following.

Let us use a parameter vector $\boldsymbol{\theta}$ and consider the following parametrization of an ENMPC scheme to approximate the value function:

$$V_\theta(\boldsymbol{s}) = \min_{\boldsymbol{x},\boldsymbol{u}} \ \lambda_\theta(\boldsymbol{x_0}) + \gamma^N V_\theta^{\mathrm{f}}(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} \gamma^k \ell_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{13a}$$

$$s.t. \quad \boldsymbol{x}_0 = \boldsymbol{s}, \tag{13b}$$

$$\boldsymbol{x}_{k+1} = \mathbf{f}_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{13c}$$

$$\boldsymbol{g}(\boldsymbol{u}_k) \leq 0, \tag{13d}$$

$$\boldsymbol{h}_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k) \leq 0, \quad \boldsymbol{h}_\theta^{\mathrm{f}}(\boldsymbol{x}_N) \leq 0, \tag{13e}$$

where the stage and terminal cost $\ell_\theta, V_\theta^{\mathrm{f}}$, the system dynamics $\mathbf{f}_\theta$ and the constraints $\boldsymbol{h}_\theta, \boldsymbol{h}_\theta^{\mathrm{f}}$ and the storage cost λ_θ are parametric functions of $\boldsymbol{\theta}$. To ensure feasibility of (13), [2] propose to use an exact relaxation of the state-dependent constraints. We define the policy

$$\pi_\theta(\boldsymbol{s}) = \boldsymbol{u}^\star_, , \tag{14}$$

where $\boldsymbol{u}_0^\star$ is the first element of the optimal input sequence $\boldsymbol{u}_0^\star, \ldots, \boldsymbol{u}_{N-1}^\star$ solution of (13) for a given state $\boldsymbol{s}$. Further, we define the action-value function $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ as

$$Q_\theta(\boldsymbol{s}, \boldsymbol{a}) = \min_{\boldsymbol{u},\boldsymbol{x}} \quad (13a), \tag{15a}$$

$$\text{s.t.} \quad (13b) - (13e), \tag{15b}$$

$$\boldsymbol{u}_0 = \boldsymbol{a}, \tag{15c}$$

Note that the two problems i.e. (13) and (15) differ only with an additional constraint $\boldsymbol{u}_0 = \boldsymbol{a}$ in (15). The proposed parametrization satisfies the fundamental equalities underlying the bellman equations, i.e.,

$$\pi_\theta(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}} \ Q_\theta(\boldsymbol{s}, \boldsymbol{a}), \quad V_\theta(\boldsymbol{s}) = \min_{\boldsymbol{a}} \ Q_\theta(\boldsymbol{s}, \boldsymbol{a}). \tag{16}$$

Additionally, the Lagrange function underlying the parameterized ENMPC scheme in (15) is given as:

$$\begin{aligned} \mathcal{L}_\theta(\boldsymbol{y}) =& \lambda_\theta(\boldsymbol{x}_0) + \gamma^N V_\theta^{\mathrm{f}}(\boldsymbol{x}_N) + \boldsymbol{\chi}_0^\top(\boldsymbol{x}_0 - \boldsymbol{s}) + \boldsymbol{\mu}_N^\top \boldsymbol{h}_\theta^{\mathrm{f}}(\boldsymbol{x}_N) \\ &+ \sum_{k=0}^{N-1} \boldsymbol{\chi}_{k+1}^\top \left(\boldsymbol{f}_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k) - \boldsymbol{x}_{k+1}\right) + \boldsymbol{\nu}_k^\top \boldsymbol{g}_\theta(\boldsymbol{u}_k) \\ &+ \gamma^k \ell_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{\mu}_k^\top \boldsymbol{h}_\theta(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{\zeta}^\top(\boldsymbol{u}_0 - \boldsymbol{a}), \end{aligned} \tag{17}$$

where $\boldsymbol{\chi}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\zeta}$ are the multipliers associated to the constraints (15b)-(15c), and $\boldsymbol{y} = (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\chi}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\zeta})$ are the primal-dual variables associated to (15). Note that for $\boldsymbol{\zeta} = 0$, $\mathcal{L}_\theta(\boldsymbol{y})$ is the Lagrange function associated to the problem (13), or $V_\theta(\boldsymbol{s})$. Using the results in (8), we observe that,

$$\frac{\partial}{\partial \theta} V_\theta(\boldsymbol{s}) = \frac{\partial}{\partial \theta} \mathcal{L}_\theta(\boldsymbol{y}^\diamond), \tag{18}$$

where $\boldsymbol{y}^\diamond$ is the primal-dual solution of (13).

B. $Q-$Learning for ENMPC

In $Q-$learning, the action-value function is parameterized as $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$, where $\boldsymbol{\theta}$ is a vector of parameters. The classical $Q-$learning algorithm updates these parameters based on temporal differences, with instantaneous policy updates.

$$\delta_k = \ell(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma \min_{\boldsymbol{a}_{k+1}} Q_\theta\left(\boldsymbol{s}_{k+1}, \boldsymbol{a}_{k+1}\right) - Q_\theta\left(\boldsymbol{s}_k, \boldsymbol{a}_k\right), \tag{19a}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \delta_k \nabla_\theta Q_\theta\left(\boldsymbol{s}_k, \boldsymbol{a}_k\right), \tag{19b}$$

where the scalar $\alpha > 0$ is the learning rate.

IV. Sensitivity Analysis

To use ENMPC as a function approximator in RL using the $Q-$learning method, one needs to evaluate $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$, $V_\theta(\boldsymbol{s})$ in (19a) and its parametric sensitivities, i.e $\nabla_\theta Q_\theta$ in (19b). The terms $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$, $V_\theta(\boldsymbol{s})$ are conventionally obtained by evaluation of two parametric nonlinear programming problems, i.e. (13), and (15), of which, both need to be solved at each iteration. The evaluation of these two NLP problems can be computationally demanding and on a closer observation, (13) and (15) differ only with a single constraint, i.e. $\boldsymbol{u}_0 = \boldsymbol{a}$. Hence, $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ can be approximated with the help of parametric NLP sensitivities using the optimal solution from (13). Approximation of $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ from the optimal value function $V_\theta(\boldsymbol{s})$, avoiding the evaluation of one extra NLP, and reducing the computational time required for a single RL iteration.

A. Approximation of $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ and $\nabla_\theta Q_\theta$ using NLP Sensitivities

In this section, we detail on how to approximate the action-value function $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ and its gradient $\nabla_\theta Q_\theta$ from the optimal solution of (13) using the results presented in Section II-A.

More specifically, we propose to solve problem (13) to full convergence and avoid solving (15). Using (9), the second order Taylor expansion of the optimal value function can be written as

$$\begin{aligned} Q_\theta(\boldsymbol{s},\boldsymbol{a}) \approx V_\theta(\boldsymbol{s}) + \frac{\partial Q_\theta(\boldsymbol{s},\boldsymbol{a})}{\partial \boldsymbol{a}}\Delta\boldsymbol{a}\bigg|_{\boldsymbol{a}=\pi_\theta(\boldsymbol{s})} \\ + \frac{1}{2}(\Delta\boldsymbol{a})^\top \frac{\partial^2 Q_\theta(\boldsymbol{s},\boldsymbol{a})}{\partial \boldsymbol{a}^2}\Delta\boldsymbol{a}\bigg|_{\boldsymbol{a}=\pi_\theta(\boldsymbol{s})}, \end{aligned} \tag{20}$$

where $\Delta\boldsymbol{a} = \boldsymbol{a} - \pi_\theta(\boldsymbol{s})$ and $\pi_\theta(\boldsymbol{s})$ is the policy (14). Note that the first order Taylor expansion cannot be used because the term,

$$\frac{\partial Q_\theta(\boldsymbol{s},\boldsymbol{a})}{\partial \boldsymbol{a}}\bigg|_{\boldsymbol{a}=\pi_\theta(\boldsymbol{s})} = \frac{\partial}{\partial \boldsymbol{a}}\mathcal{L}_\theta(\boldsymbol{y}^\star) = 0, \tag{21}$$

where $\mathcal{L}_\theta(\boldsymbol{y}^\star)$ and $\boldsymbol{y}^\star$ are associated to the problem (15).

Next, we detail on how to approximate the gradient of $Q_\theta(\boldsymbol{s},\boldsymbol{a})$ as required in (19b) using Taylor expansion:

$$\frac{\partial}{\partial\theta}Q_\theta(\boldsymbol{s},\boldsymbol{a}) \approx \frac{\partial}{\partial\theta}V_\theta(\boldsymbol{s}) + \frac{\partial^2 Q_\theta(\boldsymbol{s},\boldsymbol{a})}{\partial\theta\partial\boldsymbol{a}}(\boldsymbol{a}-\pi(\boldsymbol{s}))\bigg|_{\pi(\boldsymbol{s})}, \tag{22}$$

where,

$$\begin{aligned} \frac{\partial^2 Q_\theta(\boldsymbol{s},\boldsymbol{a})}{\partial\theta\partial\boldsymbol{a}} &= \frac{\mathrm{d}}{\mathrm{d}\theta}\frac{\partial Q_\theta(\boldsymbol{s},\boldsymbol{a})}{\partial\boldsymbol{a}} = \frac{\mathrm{d}}{\mathrm{d}\theta}\frac{\partial \mathcal{L}_\theta(\boldsymbol{y}^\star)}{\partial\boldsymbol{a}} \\ &= \frac{\partial^2 \mathcal{L}_\theta(\boldsymbol{y}^\star)}{\partial\theta\partial\boldsymbol{a}} + \frac{\partial^2 \mathcal{L}_\theta(\boldsymbol{y}^\star)}{\partial\boldsymbol{a}\partial\boldsymbol{y}}\frac{\partial\boldsymbol{y}}{\partial\theta}. \end{aligned} \tag{23}$$

V. SIMULATIONS

In this section, we demontsrate the capabilities of our proposed method by applying it to two benchmark problems. We apply the approach to a simple LQR case study and an example from the process industry, namely continuously stirred tank reactor (CSTR). These simulations serve to illustrate the potential of our approach for practical applications in economic MPC.

A. LQR case

Consider the trivial linear dynamics and quadratic stage cost of a regulator LQR problem.

$$s_{k+1} = 2s_k - a_k, \quad L(s,a) = s^2 + a^2 + 4sa \ngeq 0 \tag{24}$$

The optimal steady-state, denoted as $(s_s, a_s) = (0, 0)$, can be easily calculated for the given linear dynamics and quadratic stage cost. Using the Riccati equation, the exact optimal value function and policy can be given as:

$$V^{*}(s) = 11.7446s^2, \quad \pi^{\star}(s) = 1.6861s$$

The parameterized stage cost $\hat{\ell}_\theta$, the terminal cost T_θ and the storage function λ_θ are selected as

$$\lambda_\theta(s) = \theta_1 s^2, \quad T_\theta(s) = \theta_2 s^2, \tag{25a}$$

$$\hat{\ell}_\theta(s) = \begin{bmatrix} s \\ a \end{bmatrix}^\top \begin{bmatrix} \theta_3 & \theta_4 \\ \theta_5 & \theta_6 \end{bmatrix} \begin{bmatrix} s \\ a \end{bmatrix}, \tag{25b}$$

where the vector of parameters $\boldsymbol{\theta} = \theta_1, \ldots, \theta_6$ can be adjusted using $Q-$learning. We initialize these parameters as $\boldsymbol{\theta}_0 = [0.1, 1, 1, 0, 0, 0]^\top$. For our simulations, we set the prediction horizon to $N = 5$, learning rate to $\alpha = 5e^{-3}$ and the discount factor to $\gamma = 1$.

The iterative updates of the parameters $\boldsymbol{\theta}$ using $Q-$learning are compared between two methods: solving $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ to full convergence and approximating it using NLP sensitivities, as shown in Fig.1. The results indicate that the update steps are similar for both methods. Furthermore, Fig.2 demonstrates that $Q-$learning can capture the optimal value function even when $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ and $\nabla_\theta Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ are approximated using NLP sensitivities.

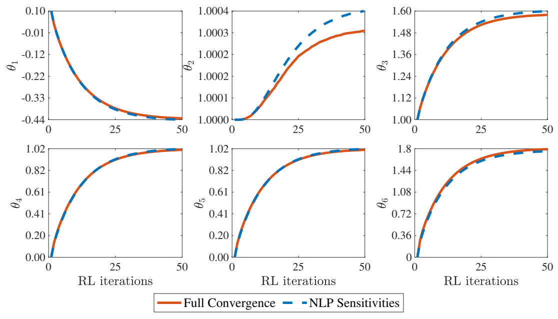

Fig. 1. Comparison of the iterative update of parameter vector $\boldsymbol{\theta}$ using $Q-$learning when the action-value function $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ is solved until full convergence vs. when it is approximated using NLP sensitivities. The plots show the progress of the parameter updates over time for both methods. Note that the scales are different.

B. CSTR Case Study

We demonstrate the proposed method using the isothermal Van de Vusse reaction in a continuous stirred tank reactor as an illustrative example [15], [16]. This model includes both material and enthalpy balances, and is described by the following equations:

$$\dot{c_A} = r_A(c_A, T) + [c_{in} - c_A]u_1 \tag{26a}$$

$$\dot{c_B} = r_B(cA, c_B, T) - c_B u_1, \tag{26b}$$

$$\dot{T} = h(c_A, c_B, T) + \varphi[T_c - T] + [T_{in} - T]u_1, \tag{26c}$$

$$\dot{T}_c = \beta[T - T_c] + \varsigma u_2, \tag{26d}$$

with $c_A(t), c_B(t)$ the concentrations of A and B respectively, $T(t)$ and $T_c(t)$ the temperatures in the reactor and in the cooling jacket. The reaction rates r_A and r_B, and the reaction enthalpy contribution h are described by

$$r_A(c_A, T) = -\, k_1(T)c_A - k_2(T)c_A^2, \tag{27}$$

$$r_B(c_A, c_B, T) = \quad k_1(T)[c_A - c_B], \tag{28}$$

$$\begin{aligned} h(c_A, c_B, T) = &-\vartheta[k_1(T)[c_A \Delta H_{AB} + c_B \Delta H_{BC}] \\ &+ k_2(T)c_A^2 \Delta H_{AD}]\,, \end{aligned} \tag{29}$$

with the functions $k_1(T)$ and $k_2(T)$ of the Arrhenius type:

$$k_i(T) = k_{i0} \exp\left(\frac{-E_i}{T + 273.15}\right), \quad i = 1, 2. \tag{30}$$

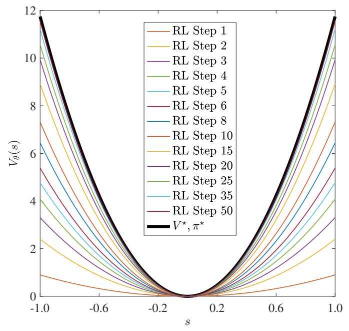

Fig. 2. Convergence of the approximated value function to the optimal $V^\star$ using the proposed approach..

The above mentioned concentrations and temperatures constitute the state $x = [c_A, c_B, T, T_c]$ with dynamics given by (26a)–(26d). The control inputs $u = [u_1, u_2]$ are normalized input flow rate $u_1(t) > 0$ through the reactor and the cooling power $u_2(t) < 0$ applied in the cooling jacket. The model parameters are summarized in Table I.

TABLE I
MODEL PARAMETERS AND THE CONSIDERED SETPOINTS OF THE CSTR SYSTEM [15].

φ	$= 30.8285\,\mathrm{h}^{-1}$	β	$= 86.688\,\mathrm{h}^{-1}$
ϑ	$= 3.556 \times 10^{-4}\,\mathrm{m}^3\mathrm{K/kJ}$	ς	$= 0.1\,\mathrm{K\,kJ}^{-1}$
k_{10}	$= (1.287) \times 10^{12}\,\mathrm{h}^{-1}$	E_1	$= 9758.3$
k_{20}	$= (9.043) \times 10^{6}\,\mathrm{m}^3/\mathrm{mol\,h}$	E_2	$= 8560$
ΔH_{AB}	$= 4.2\,\mathrm{kJ\,mol}^{-1}$		
ΔH_{BC}	$= -11\,\mathrm{kJ\,mol}^{-1}$		
ΔH_{AD}	$= -41.85\,\mathrm{kJ\,mol}^{-1}$		
c_{in}	$= 5100 \pm 600\,\mathrm{mol/m}^3$	T_{in}	$= 104.9\,\mathrm{K}^{-1}$
c_{A_0}	$= 3517.5\,\mathrm{mol/m}^3$	c_{B_0}	$= 740\,\mathrm{mol/m}^3$
T_0	$= 87\,\mathrm{K}$	T_{c_0}	$= 79.8\,\mathrm{K}$

We consider the problem of maximizing the production rate of c_B:

$$F = -c_B u_1, \tag{31}$$

and the ENMPC scheme is designed as depicted in (13), with $N = 12\,\mathrm{min}$ and sampling time of $T_s = 6\,\mathrm{s}$,

$$\min_{x,u} \quad \lambda_\theta(x_0) + \gamma^N V_\theta^{\mathrm{f}}(x_N + \omega^\top \sigma_{\mathrm{f}}) \tag{32a}$$

$$+ \sum_{k=0}^{N-1} \gamma^k \left(F(x_k, u_k) + \omega^\top \sigma_k \right) \tag{32b}$$

$$s.t. \quad x_{k+1} = f_\theta(x_k, u_k), \quad x_0 = s, \tag{32c}$$

$$g(u_k) \leq 0, \qquad h_\theta(x_k) \leq \sigma_k. \tag{32d}$$

In this work, for formulating the MPC problem, we use CasADi [17] with IPOPT as the Nonlinear Programming (NLP) solver. For learning the optimal policy, we use $Q-$learning with a learning rate of $\alpha = 10^{-3}$. The discount factor, was set to $\gamma = 0.99$. All the simulations have been performed on a MacBook Pro with Intel Core i7 running at 2.6 GHz and 16 GB of memory.

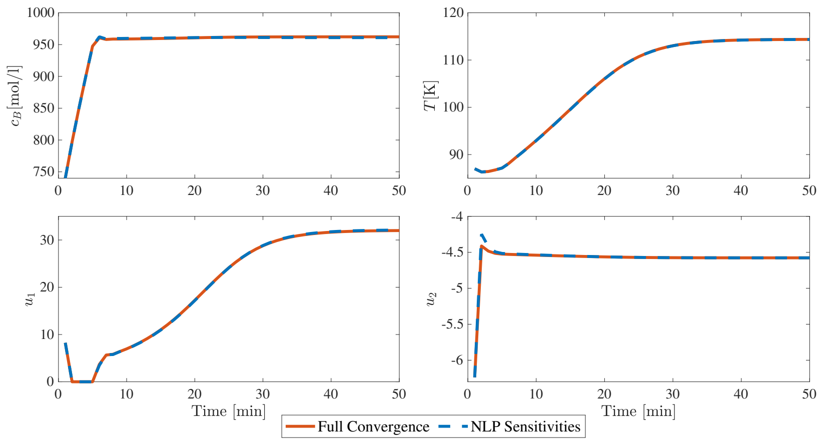

Fig. 3. Performance comparison of two approaches for states c_B, T, and control inputs u_1, u_2. The first approach involves solving (13) until full convergence, while the second approach approximates (13) using NLP sensitivities

We conducted a closed-loop simulation to compare the performance of the proposed method with the conventional approach, wherein both optimization problems are solved. The experimental results of state evaluation for c_B, T, and the control inputs u_1 and u_2 are depicted in Figure 3. As observed, the performance of both methods is remarkably similar, thereby establishing that the proposed approach would not impede the closed-loop performance.

Table II presents the computational time required to solve problem (15) to full convergence and to approximate problem (13) using NLP sensitivities in a single RL iteration. Notably, the proposed approach substantially reduces the average CPU time needed for a single RL iteration, without compromising the average cost.

TABLE II

COMPARISON OF AVERAGE COST AND COMPUTATION TIME FOR THE CSTR SYSTEM WHEN SOLVING (13) UNTIL FULL CONVERGENCE AND WHEN APPROXIMATING IT USING NLP SENSITIVITIES.

CSTR	Average Cost	Average CPU time [s]
Full Convergence	28463	9.57
NLP Sensitivities	28943	3.48

VI. Conclusion

In this paper, we have presented a simple yet effective approach to enhance the computational efficiency of RL-based MPC by utilizing NLP sensitivities. Our proposed method approximates the action-value function by exploiting similarities between consecutive nonlinear programming problems and leveraging NLP sensitivities. Through numerical simulations on two benchmark problems, we have demonstrated that our approach is capable of significantly reducing computational time without compromising controller performance. Our results suggest that the proposed framework offers a promising direction for future research aimed at further improving the computational efficiency of RL-based MPC.

References

[1] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, pp. 733–764, 2003.

[2] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, pp. 636–648, 2019.

[3] L. O. Santos, P. A. Afonso, J. A. Castro, N. M. Oliveira, and L. T. Biegler, "On-line implementation of nonlinear MPC: an experimental case study," *Control Engineering Practice*, vol. 9, pp. 847–857, 2001.

[4] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, pp. 3–20, 2002.

[5] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on Control and Optimization*, vol. 43, pp. 1714–1736, 2005.

[6] M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder, "Nominal stability of real-time iteration scheme for nonlinear model predictive control," *IEE Proceedings-Control Theory and Applications*, vol. 152, pp. 296–308, 2005.

[7] C. Büskens and H. Maurer, "Online optimization of large scale systems, chapter sensitivity analysis and real-time optimization of parametric nonlinear programming problems," *Berlin Heidelberg, Berlin, Heidelberg*, pp. 3–16, 2001.

[8] V. M. Zavala, C. D. Laird, and L. T. Biegler, "Fast implementations and rigorous models: Can both be accommodated in NMPC?" *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 18, pp. 800–815, 2008.

[9] V. M. Zavala and L. T. Biegler, "The advanced-step NMPC controller: Optimality, stability and robustness," *Automatica*, vol. 45, pp. 86–93, 2009.

[10] J. Jäschke, X. Yang, and L. T. Biegler, "Fast economic model predictive control based on NLP-sensitivities," *Journal of Process Control*, vol. 24, pp. 1260–1272, 2014.

[11] M. Zanon, V. Kungurtsev, and S. Gros, "Reinforcement learning based on real-time iteration NMPC," *IFAC-PapersOnLine*, vol. 53, pp. 5213–5218, 2020.

[12] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[13] A. V. Fiacco, *Introduction to sensitivity and stability analysis in nonlinear programming*. Academic press, 1983, vol. 165.

[14] D. P. Bertsekas, "Dynamic programming and optimal control, volume 1 of optimization and computation series," *Athena Scientific, Belmont, MA, USA, 3rd edition. Cited on*, p. 2, 2005.

[15] H. Chen, A. Kremling, and F. Allgöwer, “Nonlinear predictive control of a benchmark CSTR,” *Proceedings of 3rd European control conference*, pp. 3247–3252, 1995.

[16] K. Klatt and S. Engell, “Kontinuierlicher rührkesselreaktor mit neben–und folgereaktionen,” *Nichtlineare Regelung–Methoden, Werkzeuge, Anwendungen, VDI–Berichte*, pp. 101–108, 1993.

[17] 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, pp. 1–36, 2019.

3.4 RLMPC using Neural Networks for unknown systems

This section presents an end-to-end learning approach to developing a Nonlinear Model Predictive Control (NMPC) policy, which does not require an explicit first-principles model and assumes that the system dynamics are either unknown or partially known. The paper proposes the use of available measurements to identify a nominal Recurrent Neural Network (RNN) model to capture the nonlinear dynamics, which includes constraints on the state variables and input. To address the issue of suboptimal control policies resulting from simply fitting the model to the data, the paper uses Reinforcement learning (RL) to tune the NMPC scheme and generate an optimal policy for the real system. The approach's novelty lies in the use of RL to overcome the limitations of the nominal RNN model and generate a more accurate control policy. The paper discusses the implementation aspects of initial state estimation for RNN models and integration of neural models in MPC. The presented method is demonstrated on a classic benchmark control problem: cascaded two tank system (CTS).

Reinforcement Learning based MPC with Neural Dynamical Models

Saket Adhau, Sébastien Gros, Sigurd Skogestad

Abstract

This paper presents an end-to-end learning approach to developing a Nonlinear Model Predictive Control (NMPC) policy, which does not require an explicit first-principles model and assumes that the system dynamics are either unknown or partially known. The paper proposes the use of available measurements to identify a nominal Recurrent Neural Network (RNN) model to capture the nonlinear dynamics, which includes constraints on the state variables and input. To address the issue of suboptimal control policies resulting from simply fitting the model to the data, the paper uses Reinforcement learning (RL) to tune the NMPC scheme and generate an optimal policy for the real system. The approach's novelty lies in the use of RL to overcome the limitations of the nominal RNN model and generate a more accurate control policy. The paper discusses the implementation aspects of initial state estimation for RNN models and integration of neural models in MPC. The presented method is demonstrated on a classic benchmark control problem: cascaded two tank system (CTS).

I. Introduction

Model Predictive Control (MPC) is a widely used control strategy in various fields such as process control, robotics, and autonomous systems [1]. The success of MPC depends on the availability of a mathematical model that predicts the future behavior of the system and optimizes control actions accordingly [2]. However, uncertainties in model parameters, measurement errors, stochiasticity, unmodeled dynamics, and incomplete knowledge of the system can introduce errors, leading to suboptimal control performance. To address these challenges, the authors of [3]

This work was supported by the Research Council of Norway, under the IKTPLUSS program (Project number 299585)

Saket Adhau and Sigurd Skogestad are with the Department of Chemical Engineering, Norwegian University of Science and Technology, 7491, Trondheim, Norway. (e-mail: {saket.adhau, sigurd.skogestad}@ntnu.no)

Sebastien Gros is with the Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway (e-mail: sebastien.gros@ntnu.no)

proposed combining data-driven and model-based control to enhance MPC performance. This approach enables the control of complex systems with nonlinear dynamics and uncertainties, without relying on an accurate mathematical model of the system.

Although the approach presented in [3] can address model uncertainties, it still assumes the availability of some form of mathematical model to define the state and action spaces. In real-world scenarios, a detailed mathematical model or complete knowledge of the system dynamics may not be available, and only input-output data may be accessible. In such cases, techniques such as system identification [4] or data-driven approaches [5] can be used to estimate the system's dynamics and construct a model that can be utilized for control purposes.

In this paper, we propose to construct a “domain-aware” neural network model for modeling the unknown system dynamics. The purpose of this model is to replace the classic dynamic model used in [3]. The neural network can handle complex nonlinear systems where defining a state space may be difficult. During the training phase, we include constraints on both the state variables and input of the model to ensures that it remains physically meaningful and consistent with the real system. By constraining the state variables, we can ensure that the model remains within a physically feasible region of operation, and by constraining the input, we ensure that the control inputs generated by the model are within the range of physically feasible inputs. This dynamical model will then be integrated into an RL-based MPC framework proposed in [3] to build an optimal MPC policy with focus on closed loop performance. Our aim is to improve the closed loop performance of the system by using this neural network model, which provides more flexibility and adaptability compared to a classic dynamic model.

The initial state estimation is addressed using a past window of both measurements and outputs. The measured outputs are sequentially fed to the model instead of the predicted ones. We present discussions on adaptation of the neural model in case of model mismatch, and also the integration of neural dynamical models in MPC pipeline. We analyze the performance of the proposed approach on a classical benchmark problem: Cascaded Two tank system (CTS). The proposed method is highly sample efficient with only one requirement: a sufficiently excited time-series data set of the system dynamics.

The paper is organized into several sections. Section I provides an introduction to the problem and motivation for the research. Section II reviews the relevant literature and presents the background information on the RL-based MPC approach. Section III describes the approach to system identification using neural networks. In Section IV, we present the implementation of our proposed approach and the results of our experiments. Finally, Section V concludes the paper.

II. Background

Obtaining an accurate mathematical model to represent the dynamics of a physical system can be a daunting task, particularly for complex systems with nonlinear dynamics and uncertainties. A well-performing model should be able to capture the essential dynamics of the system and accurately predict its behavior in response to control inputs. Additionally, the model should be inexpensive to evaluate and differentiate, especially for real-time MPC applications. This necessitates that the model be of appropriate complexity, neither too simple to overlook important dynamics nor too complex to become computationally intractable.

While System identification is often considered an essential step in developing an MPC strategy, its necessity depends on the specific application and the characteristics of the system being controlled. Numerous techniques have been developed for system identification, ranging from classical methods to more advanced data-driven approaches [6].

Classical system identification methods include black-box modeling techniques such as ARX, ARMAX, and state-space modeling [7]. These methods do not require an explicit understanding of the underlying system dynamics to model the system, and can be effective for complex nonlinear systems. However, they still require some knowledge of the system and its behavior, specifically the inputs and outputs of the system and their relationship to each other. As a result, these methods may not be suitable for systems with limited or poor quality input-output data.

Neural dynamical models are a powerful tool for modeling nonlinear dynamics as they can capture complex and nonlinear relationships between input and output data without relying on a mathematical model [8]. Unlike traditional modeling techniques, these models can handle nonlinearities and interactions between system variables. RNNs are a type of neural network that can maintain information from previous inputs, making them well-suited for time series data and increasingly popular for tackling nonlinear problems [9], [10]. With their ability to capture the dynamic behavior of a system, neural dynamical models can be useful for system identification and control in a wide range of applications.

Recently, researchers [11] proposed an approach for modeling the dynamics of complex systems using RNNs, demonstrating the potential of this method in accurately identifying and controlling nonlinear systems. Another study [12] utilized constrained block-nonlinear neural networks to identify and model nonlinear systems. The authors demonstrated that this method can effectively capture the dynamics of a wide range of systems, and can be used to design control strategies that achieve improved performance compared to traditional approaches.

A. Reinforcement Learning

Consider a problem described by Markov Decision Process (MDP) with potentially stochastic state transition dynamics denoted by,

$$\mathbb{P}[\boldsymbol{s}_+|\boldsymbol{s},\boldsymbol{a}]\,, \tag{1}$$

where $\boldsymbol{s},\boldsymbol{a}$ is the current state-action pair and $\boldsymbol{s}_+$ is the subsequent state and $\mathbb{P}$ is the conditional probability. The state-action space $\boldsymbol{s}\in\mathcal{S},\boldsymbol{a}\in\mathcal{A}$ is assumed to be continuous and $\mathbb{P}$ is a probability density, but the theory proposed here is valid in general. The notation used for state transitions in (1) is standard in MDP literature, whereas the control literature commonly employs $\boldsymbol{s}_+=f(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\zeta})$ where $\boldsymbol{\zeta}$ is a stochastic variable and f is a possibly nonlinear function.

Let us consider a stage cost $L(\boldsymbol{s}_k,\boldsymbol{a}_k)$ associated to the MDP can take the form of:

$$L(\boldsymbol{s}_k,\boldsymbol{a}_k)=\ell(\boldsymbol{s}_k,\boldsymbol{a}_k)+\mathcal{I}_\infty(\boldsymbol{h}(\boldsymbol{s}_k,\boldsymbol{a}_k))\,, \tag{2}$$

where $\ell(\boldsymbol{s}_k,\boldsymbol{a}_k)$ captures the cost given to different input-output pairs, and the constraints

$$\boldsymbol{h}(\boldsymbol{s}_k,\boldsymbol{a}_k)\leq 0\,, \tag{3}$$

capture undesirable states and inputs, and infinite values are given to L when (3) is violated. Additionally, we use the indicator function,

$$\mathcal{I}_\infty(\boldsymbol{x})=\begin{cases}\infty & \text{if } \boldsymbol{x}_i>0 \text{ for some } i\\ 0 & \text{otherwise.}\end{cases} \tag{4}$$

With the addition of a discount factor $0<\gamma\leq 1$, given (1) and (2), any optimal discounted policy $\boldsymbol{\pi}_\star$ minimizes the expected total discounted cost,

$$J(\boldsymbol{\pi})=\mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k L(\boldsymbol{s}_k,\boldsymbol{a}_k)\middle|\boldsymbol{a}_k=\boldsymbol{\pi}(\boldsymbol{s}_k)\right]\,, \tag{5}$$

where the expected value $\mathbb{E}[\cdot]$ is taken over the (possibly) stochastic state transition dynamics (1) in closed loop with policy $\boldsymbol{\pi}$.

The optimal action-value function $Q_\star$, value function $V_\star$, and optimal policy $\boldsymbol{\pi}_\star(\boldsymbol{s})$ associated to the MDP, are defined by the Bellman equations [13]:

$$Q_\star\left(\boldsymbol{s},\boldsymbol{a}\right) = L\left(\boldsymbol{s},\boldsymbol{a}\right) + \gamma\mathbb{E}\left[V_\star(\boldsymbol{s}_+)\,|\,\boldsymbol{s},\boldsymbol{a}\right], \tag{6a}$$

$$V_\star\left(\boldsymbol{s}\right) = Q_\star\left(\boldsymbol{s},\pi_\star\left(\boldsymbol{s}\right)\right) = \min_{a}\, Q_\star\left(\boldsymbol{s},\boldsymbol{a}\right). \tag{6b}$$

Throughout the paper we will assume that the associated stage cost L, and the discount factor γ yield a well posed problem, i.e. the value function defined by (6) are well posed, and finite over some non-empty sets.

We then consider a model of the real system having state transition dynamics

$$\mathbb{P}[\hat{\boldsymbol{s}}_+|\boldsymbol{s},\boldsymbol{a}] \tag{7}$$

which typically do not match (1) perfectly. We now consider a modified stage cost defined as:

$$\hat{L}(\boldsymbol{s},\boldsymbol{a}) = \begin{cases} Q_\star(\boldsymbol{s},\boldsymbol{a}) - \gamma\mathcal{V}^+(\boldsymbol{s},\boldsymbol{a}) & \text{if } |\mathcal{V}^+(\boldsymbol{s},\boldsymbol{a})| < \infty \\ \infty & \text{otherwise} \end{cases} \tag{8}$$

where $\mathcal{V}^+(\boldsymbol{s},\boldsymbol{a}) = \mathbb{E}[V_\star(\hat{\boldsymbol{s}}_+)|\boldsymbol{s},\boldsymbol{a}]$ where the expectation is taken over the distribution (7). We now briefly reiterate the central theorem in [3], stating that under some condition, the policy $\boldsymbol{\pi}_\star$ that minimizes the stage cost L for the true dynamics (1) can also be generated using the model (7) combined with the stage cost $\hat{L}$. The approach here used is to bypass the difficult evaluation of (8), and replacing it by learning $\hat{L}$ directly from the data.

Let us consider the parametrization of value function $V_\star$ using the following ENMPC scheme parameterized using θ:

$$V_\theta(\boldsymbol{s}) = \min_{\boldsymbol{x},\boldsymbol{u}}\ \lambda_\theta(\boldsymbol{x}) + \gamma^N V_\theta^{\mathrm{f}}(\boldsymbol{x}_N) + \sum_{k=0}^{N-1}\gamma^k\ell_\theta(\boldsymbol{x}_k,\boldsymbol{u}_k) \tag{9a}$$

$$s.t.\quad \boldsymbol{x}_0 = \boldsymbol{s}, \tag{9b}$$

$$\boldsymbol{x}_{k+1} = \mathbf{f}_\theta(\boldsymbol{x}_k,\boldsymbol{u}_k), \tag{9c}$$

$$\boldsymbol{g}(\boldsymbol{u}_k) \leq 0, \tag{9d}$$

$$\boldsymbol{h}_\theta(\boldsymbol{x}_k,\boldsymbol{u}_k) \leq 0, \quad \boldsymbol{h}_\theta^{\mathrm{f}}(\boldsymbol{x}_N) \leq 0, \tag{9e}$$

where the ENMPC scheme (9) holds a model parametrization f_θ, a constraint parametrization $\boldsymbol{h}_\theta$, a parametrization of the stage cost ℓ_θ and terminal cost V_θ^{f} with the storage function λ_θ. While the pure input constraints are fixed $\boldsymbol{g}(a) \leq 0$ are arguably fixed, the mixed constraints in the ENMPC scheme ought to be modified, in order to capture the domain where $\hat{L}(\boldsymbol{s}, \boldsymbol{a})$ is finite. Additionally, a relaxed version of L and of mixed constraints is considered to avoid infinite penalties in case of constraint violations.

Further, we define the action-value function $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ as

$$
\begin{aligned}
Q_\theta(\boldsymbol{s}, \boldsymbol{a}) = \min_{\boldsymbol{u}, \boldsymbol{x}} \quad & (9a)\,, && \text{(10a)}\\
\text{s.t.} \quad & (9b) - (9e)\,, && \text{(10b)}\\
& \boldsymbol{u}_0 = \boldsymbol{a}\,. && \text{(10c)}
\end{aligned}
$$

The proposed parametrization trivially satisfies the fundamental equalities underlying the Bellman equations, i.e.,

$$
\pi_\theta(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}} \; Q_\theta(\boldsymbol{s}, \boldsymbol{a}), \quad V_\theta(\boldsymbol{s}) = \min_{\boldsymbol{a}} \; Q_\theta(\boldsymbol{s}, \boldsymbol{a}). \tag{11}
$$

B. Q−Learning for ENMPC

In the $Q-$learning algorithm, the action-value function is represented by $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$, where $\boldsymbol{\theta}$ is a vector of parameters. The classical approach to $Q-$learning involves updating these parameters based on temporal differences δ, which are computed from the difference between the predicted and actual rewards received from taking a particular action in a given state. This approach is coupled with instantaneous policy updates, where:

$$
\begin{aligned}
\delta_k &= \ell(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma \min_{\boldsymbol{a}_{k+1}} Q_\theta\left(\boldsymbol{s}_{k+1}, \boldsymbol{a}_{k+1}\right) - Q_\theta\left(\boldsymbol{s}_k, \boldsymbol{a}_k\right), && \text{(12a)}\\
\boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \alpha \delta_k \nabla_\theta Q_\theta\left(\boldsymbol{s}_k, \boldsymbol{a}_k\right), && \text{(12b)}
\end{aligned}
$$

where the scalar $\alpha > 0$ is the learning rate. For further clarification such as the gradient calculation in the RL-based MPC scheme, readers are directed to the paper by [3].

C. Need for a Model

In the approach presented above, a model is needed to formulate the state and action spaces, which are essential for training the RL agent. While the agent can learn the optimal policy from data, it still requires a model to operate within a well-defined state and action space. Therefore, even though the RL-based MPC scheme can handle model mismatch or uncertainties, a model is still required to formulate the state and action spaces, as well as to evaluate the performance of the learned policy.

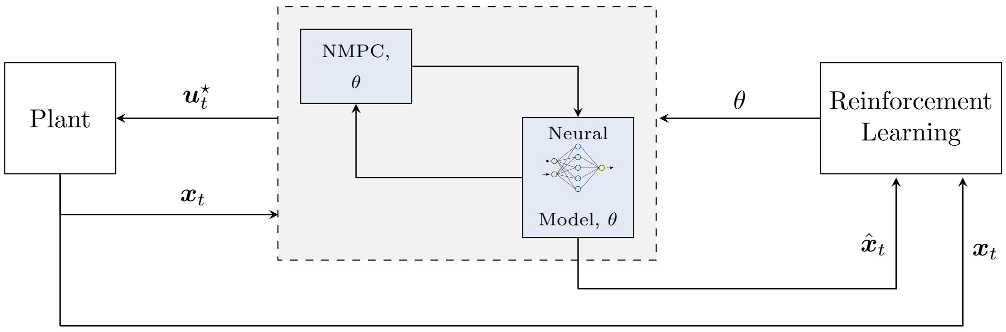

Fig. 1. Illustration depicting the proposed methodology that employs nonlinear neural dynamical models, constructed using measurement data, within a RL based MPC framework, to attain an optimal control policy.

III. Recurrent Neural Networks

The use of RNNs for modeling dynamical systems in MPC settings has several advantages over classical system identification methods. RNNs excel at capturing complex and nonlinear relationships between input and output data, making them well-suited for time series data and nonlinear problems.

A. Constrained system identification using RNNs

Consider a discrete-time dynamical system, denoted by $\mathcal{S}$ which takes input values $u \in \mathbb{R}^{n_u}$ and produces output values $y \in \mathbb{R}^{n_y}$. At any given time step k, the system's output may depend on all previous input samples, i.e., it exhibits memory,

$$y_k = \mathcal{S}(u_k, u_{k-1}, \ldots, u_0). \tag{13}$$

Assuming that the system $\mathcal{S}$ can be adequately represented by a nonlinear dynamical model, denoted by $\mathcal{M}$:

$$\hat{y}_k = \mathcal{M}(u_k, u_{k-1}, \ldots, u_0; \theta) \tag{14}$$

where $\theta \in \mathbb{R}^{n_\theta}$ is a parameter vector to be determined.

We use bold symbols $\mathbf{u}$ and $\mathbf{y}$ to denote the sequence of N input and output samples in a dataset $\mathcal{D}$.

In this paper, we assume that the model has the following state-space representation:

$$\boldsymbol{x}_{k+1} = \mathcal{F}(\boldsymbol{x}_k, \boldsymbol{u}_k; \theta_{\mathcal{F}}) \tag{15a}$$

$$\hat{\boldsymbol{y}}_k = \mathcal{G}(\boldsymbol{x}_k, \boldsymbol{u}_k; \theta_{\mathcal{G}}), \tag{15b}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ is the unknown system state, $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is the control input at time k. In this paper, $\mathcal{F}$ and $\mathcal{G}$ are the state-update and output mappings respectively, both parameterized by $\theta = \theta_{\mathcal{F}}, \theta_{\mathcal{G}}$. In particular, $\mathcal{F}$ and $\mathcal{G}$ are neural networks, and the overall model (15) is a RNN.

The data is assumed to be available in the form of input-output tuples, denoted by

$$\mathcal{D} = \{(\mathbf{y}_k^i, \mathbf{u}_k^i), (\mathbf{y}_{k+1}^i, \mathbf{u}_{k+1}^i), \ldots, (\mathbf{y}_{k+N}^i, \mathbf{u}_{k+N}^i)\}, i \in \mathbb{N}_1^n \tag{16}$$

where n is the number of sample trajectories with $\mathbb{N}$ time steps. The primary aim is to learn a constrained neural equivalent of the unknown system dynamics, given input-output time-series dataset (16) obtained by observing the system.

B. System identification loss

The neural state space dynamics (15) are trained on the sampled input-output trajectories (16) using the loss function described below:

$$\begin{aligned} &\mathcal{L}_\phi(\mathbf{Y}^{\text{true}}, \mathbf{Y}, \bar{\mathbf{Y}}, \underline{\mathbf{Y}}|\theta) = \\ &\frac{1}{nN}\sum_{i=1}^{n}\sum_{k=1}^{N}\Bigg(\Big|\Big|\mathbf{y}_k^{\text{true},i} - \mathbf{y}_k^i\Big|\Big|_2^2 + Q_y\Big|\Big|p(\mathbf{y}_k^i, \underline{\mathbf{y}}_{k+1}^i)\Big|\Big|_2^2 + \\ &+Q_y\Big|\Big|p(\mathbf{y}_k^i, \bar{\mathbf{y}}_{k+1}^i)\Big|\Big|_2^2 + Q_u\Big|\Big|p(f_u(\mathbf{u}_k^i), \underline{f}_u)\Big|\Big|_2^2 + \\ &Q_u\Big|\Big|p(f_u(\mathbf{u}_k^i), \bar{f}_u)\Big|\Big|_2^2\Bigg), \end{aligned} \tag{17}$$

where k represents the time step of the prediction horizon N, whereas i is the batch index of n sampled trajectories. The tracking loss is given by the first term, taken as two norm over the residual vector between true $\mathbf{Y}^{\text{true}} = \{\mathbf{Y}_1^{\text{true},i}, \ldots, \mathbf{Y}_N^{\text{true},i}\}$ and predicted $\mathbf{Y} = \{\mathbf{Y}_1^i, \ldots, \mathbf{Y}_N^i\}$ output trajectories over N step. The second term is optional and is used to promote smoothening of the trajectories of dynamic models. Furthermore, the third and fourth terms impose constraints on the output trajectories using penalty functions. We apply inequality constraints on predictions during the training phase itself, making the unconstrained optimization problem amenable to the gradient-based optimization methods used in deep learning. We use the following penalty functions for time-varying lower and upper bounds $\bar{\mathbf{y}}_k, \underline{\mathbf{y}}_k$:

$$\underline{p}(\mathbf{y}_k, \underline{\mathbf{y}}_k) = \max(0, -\mathbf{y}_k + \underline{\mathbf{y}}_k)\,, \tag{18a}$$

$$\bar{p}(\mathbf{y}_k, \bar{\mathbf{y}}_k) = \max(0, \mathbf{y}_k - \bar{\mathbf{y}}_k)\,, \tag{18b}$$

The penalty functions shown in (18) are easy to implement and can be modified according the specific requirements as shown in [14]. The iterative gradient-based optimization algorithms, like Adam [15], can be used to minimize the loss. The required derivatives computation is accomplished using standard reverse-mode Automatic Differentiation (AD) algorithms and software [16]. As standard practice is followed for nominal model training, we will not discuss it in more detail.

C. *RNN initial state estimation*

The estimation of initial state in an RNN model is very important, especially for our application of MPC [17]. The initial state can affect the state transition predictions and thus needs to be estimated accurately. Extending the ideas used in [11] we denote the predicted output $\hat{\boldsymbol{y}}_k$ which is a part of the state:

$$\boldsymbol{x}_{k+1} = \mathcal{F}(\boldsymbol{x}_k, \hat{\boldsymbol{y}}_k, \boldsymbol{u}_k; \theta_{\mathcal{F}}) \tag{19a}$$

$$\hat{\boldsymbol{y}}_{k+1} = \mathcal{G}(\boldsymbol{x}_{k+1}, \hat{\boldsymbol{y}}_k, \boldsymbol{u}_k; \theta_{\mathcal{G}}), \tag{19b}$$

$$\text{for } k = 0, 1, \ldots, N_c - 1. \tag{19c}$$

While the predictions are performed using (19), the initial state is estimated using a window of the past data $\{\boldsymbol{y}_{k-1}, \ldots, \boldsymbol{y}_{k-N_c}, \boldsymbol{u}_{k-1}, \ldots, \boldsymbol{u}_{k-N_c}\}$, where the measured outputs $\boldsymbol{y}_{k-i}$ are sequentially fed to the model instead of the predicted ones for N_c steps, and $\boldsymbol{x}_{k-N_c}$ is set to zero. In particular, the state estimation is performed by opening the output prediction loop for the first N_c steps:

$$\boldsymbol{x}_{k+1} = \mathcal{F}(\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{u}_k; \theta_{\mathcal{F}}) \tag{20a}$$

$$\hat{\boldsymbol{y}}_{k+1} = \mathcal{G}(\boldsymbol{x}_{k+1}, \boldsymbol{y}_k, \boldsymbol{u}_k; \theta_{\mathcal{G}}), \tag{20b}$$

D. *Using RNN models in MPC scheme*

In MPC, the control law is typically obtained by solving an optimization problem that involves the system model. If a neural network is chosen as the system model, it is crucial for it to

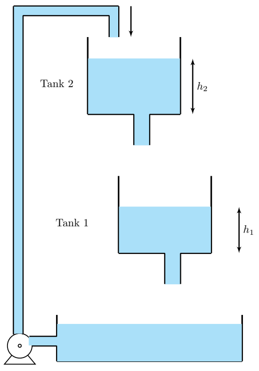

Fig. 2. Schematics of the cascaded two-tank system.

be differentiable to calculate gradients for the optimization problem. This is because most optimization algorithms typically used in MPC rely on the gradient information to find the optimal control inputs.

Fortunately, most neural network architectures used in practice, such as feedforward networks and recurrent networks, are differentiable and can be used in MPC. However, there are additional challenges when using neural network-based models in MPC. The choice of network architecture and the selection of appropriate regularization methods are some of the important considerations. Despite these challenges, neural network-based models can still be a powerful tool for improving the performance of MPC systems when carefully designed and implemented.

The detailed procedure for the proposed method is illustrated in Fig. 1. The method begins by constructing a neural model of the system using available measurement data. This model is then utilized in RL based MPC scheme to optimize the closed loop performance. By employing the RL-MPC scheme, we are able to modify the cost function as well as model (if required) to adapt and learn from the system's behavior in real-time, leading to improved control performance.

IV. Case Study

To demonstrate the effectiveness of neural network-based models in an RL-based MPC approach, we utilize the Cascaded Tanks System (CTS) depicted in Fig. 2 with descriptions in [18].

The CTS is a control system for fluid level that involves two tanks with free outlets, supplied by a pump. The controlled pump transfers water from a bottom reservoir to the upper tank. The water from the upper tank flows into the lower tank via a small opening and subsequently into the reservoir via another small opening. The system input, denoted as u, is the water flow from the bottom reservoir to the upper tank. Meanwhile, the state variables h_1 and h_2 represent the water levels in the upper and lower tanks, respectively.

The identification of the CTS system poses a significant challenge due to several factors. The system's hard saturation nonlinearity, combined with its weakly nonlinear behavior during regular operation, makes the identification process complex. Moreover, the overflow from the upper to the lower tank introduces input-dependent process noise, which further complicates the problem.

DataSet: The experimental dataset was obtained from the collection [1] of public benchmarks widely used in system identification. These datasets are commonly used as a benchmark for testing the accuracy and robustness of system identification methods. The training and test datasets contain 1024 points each, collected at a constant sampling time $\Delta t = 5\,\mathrm{s}$.

Metrics: We use Root Mean Square Error (RMSE) to evaluate the performance index as it is suggested in the description of the benchmark problem [18]:

Neural Network: The proposed method utilizes the PyTorch Deep Learning (DL) framework [16] for training neural network models with the adaptation of RNN based system identification module from [11]. The network comprises of one hidden layer with 128 neurons, utilizing Leaky ReLU as the activation function. Gradient-based optimization is carried out using the Adam optimizer [15], with the learning rate parameter set to 10^{-3}. The batch size is set to 64, and $n = 10000$ iterations are carried out to ensure convergence to a cost function plateau. The training process takes approximately 400 seconds.

The neural networks' weight parameters are initialized with random Gaussian variables with zero mean and a standard deviation of 10^{-4}, while the bias terms are initialized to zero, These values are observed to be effective in aiding the convergence of the model during the training process.

RL and MPC: In this work, for formulating the MPC problem, we use CasADi [19] with IPOPT as the Nonlinear Programming (NLP) solver [20]. For learning the optimal policy, we use $Q-$learning with a learning rate of $\alpha = 10^{-3}$. The discount factor, was set to $\gamma = 0.99$.

[1]http://www.nonlinearbenchmark.org

To enable the use of arbitrary neural network models trained in PyTorch with CasADi, we use the implementation [2] from [21]. This implementation allows seamless integration of neural network models trained in PyTorch into the CasADi framework, enabling the use of deep learning techniques in control problems. All the simulations have been performed on a Macbook Pro with Intel Core i7 running at $2.6\,\mathrm{GHz}$ and $16\,\mathrm{GB}$ of memory.

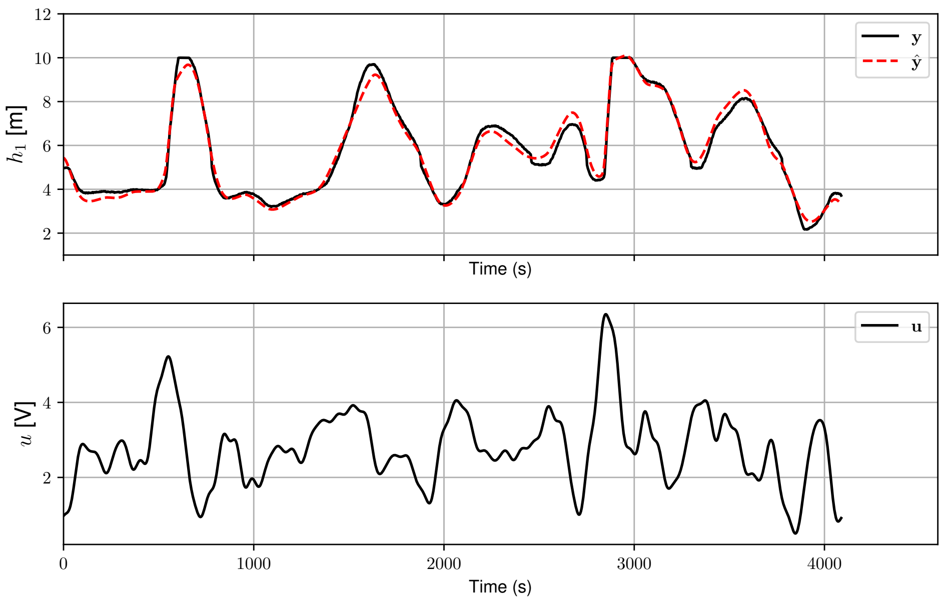

Fig. 3. CTS Benchmark: Open-loop trajectories of the trained neural model (red) and real system (black).

A. *Result Analysis:*

Fig. 3 displays the time trajectories of both the true and model output. Due to visualization constraints, only a subset of the test dataset is presented. Notably, the fitted model accurately characterizes the system dynamics with a high degree of precision. The neural loop training achieved an RMSE of 0.2912 indicating a satisfactory level of accuracy in the model's predictions. Although our proposed approach has resulted in better RMSE results compared to the state-of-the-art black-box nonlinear identification methods used for this benchmark [22]–[24], it is

[2]https://github.com/TUM-AAS/ml-casadi

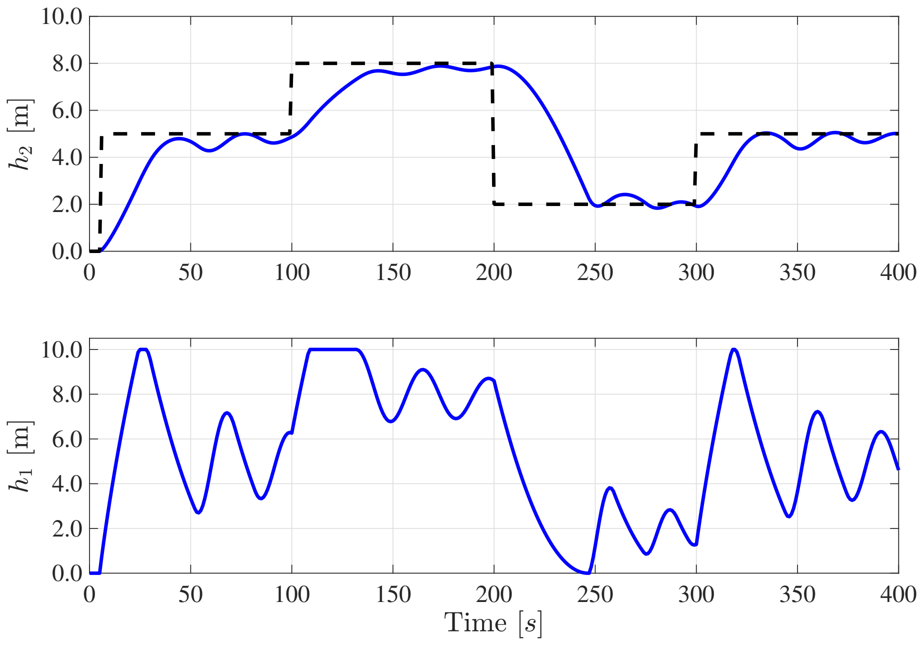

Fig. 4. CTS Benchmark: Simulated closed-loop control trajectories demonstrating RL based MPC with neural dynamical model. The upper pane depicts the reference (dashed black) alongside the controlled state h_2, while the lower panel shows the measured state h_1.

important to note that the main goal of this approach is not to obtain a highly accurate model of the system. Rather, the primary objective is to develop a model that can effectively function within the RL-MPC framework. The RL-MPC approach can then fine-tune the control policy as required.

Fig. 4 depicts the simulated closed loop trajectories of RL based MPC for the CTS benchmark problem. Our intention is to showcase that the proposed approach is capable of achieving the desired results. It is important to note that these results are presented solely for the purpose of demonstrating the capabilities of the approach and are not compared to any other technique.

V. Conclusion

In many real-world scenarios, obtaining a completely accurate model of a system can be challenging, and often measurement data is only available. In this paper, we propose a method that utilizes this measurement data to build a nonlinear neural dynamical model of the system.

The developed neural model may not be completely accurate, but we show that it is still possible to achieve optimal control policies using an RL-based MPC framework. RL-MPC uses a combination of reinforcement learning and model predictive control to learn and improve the overall closed loop performance even when the model is incomplete or inaccurate.

This approach has significant potential for practical applications in various fields where accurate models are difficult to obtain or not available. The use of neural networks and RL-based MPC can offer a robust and adaptive control strategy that can learn and improve based on the system's feedback.

References

[1] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, pp. 733–764, 2003.

[2] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[3] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, pp. 636–648, 2019.

[4] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society A*, vol. 474, p. 20180335, 2018.

[5] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Transactions on Automatic Control*, vol. 66, pp. 1702–1717, 2020.

[6] J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented road map," *IEEE Control Systems Magazine*, vol. 39, pp. 28–99, 2019.

[7] E.-W. Bai and F. Giri, "Introduction to block-oriented nonlinear systems," *Block-oriented Nonlinear System Identification*, pp. 3–11, 2010.

[8] S. Chen, S. A. Billings, and P. Grant, "Non-linear system identification using neural networks," *International journal of control*, vol. 51, pp. 1191–1214, 1990.

[9] T. W. Chow and Y. Fang, "A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics," *IEEE transactions on industrial electronics*, vol. 45, pp. 151–161, 1998.

[10] J. de Jesús Rubio and W. Yu, "Nonlinear system identification with recurrent neural networks and dead-zone kalman filter algorithm," *Neurocomputing*, vol. 70, pp. 2460–2466, 2007.

[11] M. Forgione, A. Muni, D. Piga, and M. Gallieri, "On the adaptation of recurrent neural networks for system identification," *arXiv preprint arXiv:2201.08660*, 2022.

[12] E. Skomski, S. Vasisht, C. Wight, A. Tuor, J. Drgoňa, and D. Vrabie, "Constrained block nonlinear neural dynamical models," *2021 American Control Conference (ACC)*, pp. 3993–4000, 2021.

[13] D. P. Bertsekas, "Dynamic programming and optimal control, volume 1 of optimization and computation series," *Athena Scientific, Belmont, MA, USA, 3rd edition*, vol. 2, 2005.

[14] S. Adhau, V. V. Naik, and S. Skogestad, "Constrained neural networks for approximate nonlinear model predictive control," *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 295–300, 2021.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[17] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.

[18] M. Schoukens and J. P. Noël, "Three benchmarks addressing open challenges in nonlinear system identification," *IFAC-PapersOnLine*, vol. 50, pp. 446–451, 2017.

[19] 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, pp. 1–36, 2019.

[20] 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, pp. 25–57, 2006.

[21] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, pp. 2397–2404, 2023.

[22] R. Relan, K. Tiels, A. Marconato, and J. Schoukens, "An unstructured flexible nonlinear model for the cascaded water-tanks benchmark," *IFAC-PapersOnLine*, vol. 50, pp. 452–457, 2017.

[23] G. Birpoutsoukis, P. Z. Csurcsia, and J. Schoukens, "Efficient multidimensional regularization for volterra series estimation," *Mechanical Systems and Signal Processing*, vol. 104, pp. 896–914, 2018.

[24] A. Svensson and T. B. Schön, "A flexible state–space model for learning nonlinear dynamical systems," *Automatica*, vol. 80, pp. 189–199, 2017.

Chapter 4

Discussions

4.1 Conclusion

In this concluding chapter, the thesis's pivotal contributions are summarized and discussed in depth, offering insights into the multifaceted challenges faced in the dynamic field of control and process systems engineering.

The contemporary fervor surrounding machine learning and artificial intelligence has galvanized researchers and industry professionals in the domain. While several applications have shown rapid progress, the most intellectually stimulating challenge lies in devising conceptual frameworks to navigate the intricate landscape of online process optimization.

Addressing the limitations of data-centric machine learning, we introduced innovative methods to enhance control strategies. In Section 3.1, a powerful technique was proposed to handle constraints in approximating nonlinear Model Predictive Control (MPC) problems using neural networks. This method harnessed domain-specific knowledge, leveraging the intuitive understanding of KKT conditions and log barrier methods to refine the loss function of neural networks.

A paradigm shift from neural networks to Reinforcement Learning (RL) in control tasks was explored, emphasizing dynamic learning and decision-making processes. Transitioning to RL, systems could directly learn optimal policies from interactions with their environments, rendering explicit models unnecessary. This approach not only adeptly handled complexity and uncertainty but also empowered autonomous systems to refine strategies continuously, making it an enticing choice for real-world applications.

In Section 3.2, we delved into the fusion of Real-Time Optimization (RTO) and Economic Nonlinear Model Predictive Control (ENMPC) within the realm of Reinforcement Learning (RL). Our innovative framework, blending concepts from modifier adaptation, showcased the capacity of combined RTO and ENMPC to generate optimal policies for real systems, even amidst inaccurate underlying models. By integrating modifier terms into the RTO scheme, we eliminated the need for steady-state plant gradient estimation in Modifier Adaptation (MA), streamlining the process significantly.

The limitations of the research at this stage primarily revolve around the implementation practicalities in legacy control systems and the sustainability of control schemes over time. Legacy control systems often favor simple structures for ease of implementation and maintenance, posing a challenge for the adoption of more advanced intelligent control approaches. The complexity introduced by sophisticated methods, such as those proposed in this research, may encounter resistance from traditional control engineers who are accustomed to simpler structures. To address this, future work should focus on demonstrating the practical implementation of these advanced control schemes in legacy systems, emphasizing their ease of integration and long-term sustainability. Bridging the gap between traditional and intelligent control methods will be crucial for widespread adoption and acceptance within the control engineering community.

A novel approach to augment the computational efficiency of RL-based MPC was detailed in Section 3.3. This method, utilizing Nonlinear Programming (NLP) sensitivities, approximated the action-value function by capitalizing on the similarities between consecutive nonlinear programming problems. Through meticulous numerical simulations, we substantiated the substantial reduction in computational time without compromising controller performance.

Lastly, in Section 3.4, we tackled the challenge of imperfect system modeling by leveraging available measurement data to construct a nonlinear neural dynamical model. Despite potential imprecision, our research demonstrated the feasibility of achieving optimal control policies through an RL-based Model Predictive Control (MPC) framework. By combining reinforcement learning and predictive control, our approach enabled learning and enhancement of closed-loop performance, even in the face of incomplete or imprecise models.

These contributions collectively underscore the thesis's innovation and relevance, offering promising directions for future research endeavors in the realm of control and process systems engineering.

4.2 Future work

The current research has paved the way for future investigations, acknowledging certain limitations and potential extensions. The following areas represent promising avenues for extending and refining the proposed methodology:

Exploration of Hybrid AI Approaches:

Future work could delve into exploring hybrid AI approaches that combine the strengths of dynamic neural networks (DNNs) with other artificial intelligence techniques for online modeling and control of complex and unknown nonlinear systems. Investigate how hybrid models, integrating DNNs with complementary AI methods, can enhance the adaptability, robustness, and efficiency of the proposed framework. This exploration may include considering hybrid architectures, ensemble methods, or incorporating domain-specific knowledge to further improve the performance of the system in real-world applications.

Enhanced Integration with Lower Layer Controllers:

Future work could involve a comprehensive re-evaluation of the proposed RTO-RLMPC framework when enabling intelligent online control at the lower control layers. Consider adaptive tuning strategies or hierarchical control architectures that optimize the collaboration between the higher-level RTO-RLMPC and the lower-level controllers. This exploration aims to enhance the adaptability and stability of the overall control system in diverse operational scenarios.

Neural Networks to Model Unknown Systems

Utilizing neural networks to model unknown systems and employing Reinforcement Learning (RL) techniques to fine-tune these networks can be a prominent approach. This method involves using neural networks as function approximators to represent complex, non-linear relationships within unknown dynamic systems. Through RL algorithms like Q-learning or policy gradient methods, these networks can be adjusted and optimized based on real-time feedback, allowing them to adapt and approximate the underlying system dynamics. This powerful combination of neural networks and RL not only enables accurate modeling of intricate systems but also facilitates the development of intelligent control strategies and decision-making processes in real-world applications.

Real-Time Implementation and Validation of RL-MPC: Bridging the Gap Between Theory and Practice

Implementing and validating RL-based Model Predictive Control (MPC) in real-time settings is a crucial next step. This involves deploying the developed RL-MPC algorithms on actual systems, such as robots or industrial processes, and assessing their performance in real-world scenarios. Real-time implementation and validation are essential to ensure the effectiveness and reliability of the proposed techniques in practical applications. The conducted research has established a foundation for forthcoming investigations and refinements of the RTO-RLMPC approach. Promising avenues for extending the proposed methodology include exploring its adaptability to diverse domains and industries beyond gas lift optimization. Additionally, strategies to integrate a broader spectrum of constraints, encompassing processing capacity, and stability requirements, should be investigated. The ongoing enhancement of learning algorithms, particularly reinforcement learning techniques within the RTO-RLMPC framework, remains a crucial focus. This involves examining advanced methods and alternative algorithms to bolster learning efficiency and adaptability to various system dynamics. Addressing real-time implementation challenges, such as computational requirements, is essential for practical applications. Furthermore, considering the impact of uncertainties in the system model and proposing robust methods to handle them is imperative. Comparative studies against existing control methods will contribute to a comprehensive understanding, highlighting the strengths and weaknesses of the RTO-RLMPC approach. Future plans should encompass experimental validation in a real gas lift or oil well network, evaluating practical implementation aspects and anticipating challenges and benefits. Exploring interdisciplinary collaboration with experts in control and petroleum engineering will enrich perspectives for addressing complex challenges in the oil and gas industry. Scalability considerations for larger or more complex systems are critical, and strategies for user-friendly implementation, potentially through software interfaces or frameworks, can facilitate broader adoption by practitioners. These future directions aim to build on the current research, contributing to the ongoing evolution of intelligent control strategies in dynamic and complex systems.

Investigation of MPC Failure Detection and Adaptation Mechanisms:

In future work, a thorough exploration of advanced techniques for detecting and isolating sources of failures within the Model Predictive Control (MPC) framework should be undertaken. This includes a comprehensive study of various potential failure sources such as external disturbances, model mismatches, sensor failures, drift, and offsets. Develop intelligent algorithms or monitoring systems capable of identifying deviations from expected system behavior in real-time. Investigate strategies to differentiate between disturbances and model inaccuracies to enable targeted corrective actions. Explore adaptive MPC algorithms that dynamically adjust control strategies in response to identified failures, ensuring continued optimal performance despite changing conditions. This research aims to enhance the robustness and reliability of MPC systems by proactively addressing and adapting to potential sources of failure in a real-world operational context.

References

[1] J. H. Lee, J. Shin, and M. J. Realff, "Machine learning: Overview of the recent progresses and implications for the process systems engineering field," *Computers and Chemical Engineering*, vol. 114, pp. 111–121, 2018.

[2] J. B. Rawlings and R. Amrit, "Optimizing process economic performance using model predictive control," *Springer Berlin Heidelberg*, pp. 119–138, 2009.

[3] S. Lucia and B. Karg, "A deep learning-based approach to robust nonlinear model predictive control," in *6th IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2018, pp. 511–516.

[4] S. Lucia, D. Navarro, B. Karg, H. Sarnago, and O. Lucia, "Deep learning-based model predictive control for resodat power converters," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 409–420, 2020.

[5] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate model predictive building control via machine learning," *Applied Energy*, vol. 218, pp. 199–216, 2018.

[6] Q. Zhang, W. Pan, and V. Reppa, "Model-reference reinforcement learning for collision-free tracking control of autonomous surface vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8770–8781, 2021.

[7] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[8] S. Gros and M. Zanon, "Data-Driven Economic NMPC Using Reinforcement Learning," *IEEE Transactions on Automatic Control*, vol. 65, pp. 636–648, 2019.

[9] D. Bertsekas, *Dynamic programming and optimal control.* Athena scientific, 2012, vol. 1.

[10] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality.* John Wiley & Sons, 2007, vol. 703.

[11] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, pp. 733–764, 2003.

[12] R. Negenborn, B. De Schutter, M. Wiering, and J. Hellendoorn, "Experience-based model predictive control using reinforcement learning," in *Proceedings of the 8th TRAIL Congress, November 2004, Rotterdam, The Netherlands.* Citeseer, 2004.

[13] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *2012 IEEE International Conference on Robotics and Automation.* IEEE, 2012, pp. 279–284.

[14] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, "Approximating explicit model predictive control using constrained neural networks," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 1520–1527.

[15] J. A. Paulson and A. Mesbah, "Approximate closed-loop robust model predictive control with guaranteed stability and constraint satisfaction," *IEEE Control Systems Letters*, vol. 4, pp. 719–724, 2020.

[16] B. Karg and S. Lucia, "Stability and feasibility of neural network-based controllers via output range analysis," in *59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 4947–4954.

[17] M. Klaučo, M. Kalúz, and M. Kvasnica, "Machine learning-based warm starting of active set methods in embedded model predictive control," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 1–8, 2019.

[18] S. Zhang and A. Constantinides, "Lagrange programming neural networks," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, pp. 441–452, 1992.

[19] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, pp. 543–548, 2018.

[20] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari, "Large scale model predictive control with neural networks and primal active sets," *arXiv preprint arXiv:1910.10835*, 2019.

[21] H. Kervadec, J. Dolz, J. Yuan, C. Desrosiers, E. Granger, and I. B. Ayed, "Constrained deep networks: Lagrangian optimization via log-barrier extensions," *arXiv preprint arXiv:1904.04205*, 2019.

[22] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, pp. 25–57, 2006.

[23] Y. Liu, J. Ding, and X. Liu, "IPO: Interior-point policy optimization under constraints," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 4940–4947.

[24] C. Zeng and H. Zhang, "A logarithmic barrier method for proximal policy optimization," *arXiv preprint arXiv:1812.06502*, 2018.

[25] Y. Huang, “Lagrange-type neural networks for nonlinear programming problems with inequality constraints,” in *44th IEEE Conference on Decision and Control (CDC)*, 2005, pp. 4129–4133.

[26] Y. Nandwani, A. Pathak, P. Singla *et al.*, “A primal dual formulation for deep learning with constraints,” *Advances in Neural Information Processing Systems*, pp. 12 157–12 168, 2019.

[27] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[28] X. Zhang, M. Bujarbaruah, and F. Borrelli, “Safe and near-optimal policy learning for model predictive control using primal-dual neural networks,” in *2019 American Control Conference (ACC).* IEEE, 2019, pp. 354–359.

[29] Q. Nguyen and M. Hein, “The loss surface of deep and wide neural networks,” in *34th International Conference on Machine Learning*, 2017, pp. 2603–2612.

[30] S. Shalev-shwartz and S. M. Kakade, “Mind the duality gap: Logarithmic regret algorithms for online optimization,” *Advances in Neural Information Processing Systems*, vol. 21, 2009.

[31] J. Drgona, K. Kis, A. Tuor, D. Vrabie, and M. Klauco, “Differentiable predictive control: An MPC alternative for unknown nonlinear systems using constrained deep learning,” *arXiv preprint arXiv:2011.03699*, 2020.

[32] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes.* Society for Industrial and Applied Mathematics, 2010.

[33] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016.

[34] D. Q. Mayne and E. C. Kerrigan, “Tube-based robust nonlinear model predictive controll,” in *7th IFAC Symposium on Nonlinear Control Systems*, 2007, pp. 36–41.

[35] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl, “Auto-generated algorithms for nonlinear model predictive control on long and on short horizons,” in *52nd IEEE Conference on Decision and Control (CDC)*, 2013, pp. 5113–5118.

[36] A. Grancharova and T. A. Johansen, *Explicit nonlinear model predictive control: Theory and applications.* Springer Science & Business Media, 2012, vol. 429.

[37] 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*, pp. 1–36, 2019.

[38] U. Von Luxburg and B. Schölkopf, “Statistical learning theory: Models, concepts, and results,” in *Handbook of the History of Logic.* Elsevier, 2011, pp. 651–706.

[39] B. Amos and J. Z. Kolter, “Optnet: Differentiable optimization as a layer in neural networks,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.

[40] P. Márquez-Neila, M. Salzmann, and P. Fua, “Imposing hard constraints on deep networks: Promises and limitations,” *arXiv preprint arXiv:1706.02025*, 2017.

[41] T. Parisini, M. Sanguineti, and R. Zoppoli, “Nonlinear stabilization by receding-horizon neural regulators,” *International Journal of Control*, pp. 341–362, 1998.

[42] Y. Vaupel, N. C. Hamacher, A. Caspari, A. Mhamdi, I. G. Kevrekidis, and A. Mitsos, “Accelerating nonlinear model predictive control through machine learning,” *Journal of Process Control*, vol. 92, pp. 261–270, 2020.

[43] A. Ahmad, W. Gao, and S. Engell, “A study of model adaptation in iterative real-time optimization of processes with uncertainties,” *Computers & Chemical Engineering*, vol. 122, pp. 218–227, 2019.

[44] J. Matias and J. Jäschke, “Online model maintenance in real-time optimization methods,” *Computers & Chemical Engineering*, vol. 145, p. 107141, 2021.

[45] M. T. De Gouvêa and D. Odloak, “One-layer real time optimization of lpg production in the FCC unit: procedure, advantages and disadvantages,” *Computers & Chemical Engineering*, vol. 22, pp. S191–S198, 1998.

[46] W. Yip and T. Marlin, “The effect of model fidelity on real-time optimization performance,” *Computers & Chemical Engineering*, vol. 28, pp. 267–280, 2004.

[47] W. S. Yip and T. E. Marlin, “Designing plant experiments for real-time optimization systems,” *Control Engineering Practice*, vol. 11, pp. 837–845, 2003.

[48] T. Backx, O. Bosgra, and W. Marquardt, “Integration of model predictive control and optimization of processes: Enabling technology for market driven process operation,” *IFAC Proceedings Volumes*, pp. 249–260, 2000.

[49] S. E. Sequeira, M. Graells, and L. Puigjaner, “Real-time evolution for on-line optimization of continuous processes,” *Industrial & Engineering Chemistry Research*, pp. 1815–1825, 2002.

[50] L. T. Biegler and V. M. Zavala, “Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization,” *Computers & Chemical Engineering*, vol. 33, pp. 575–582, 2009.

[51] L. T. Biegler, “An overview of simultaneous strategies for dynamic optimization,” *Chemical Engineering and Processing: Process Intensification*, vol. 46, pp. 1043–1053, 2007.

[52] M. Ellis and P. D. Christofides, “Integrating dynamic economic optimization and model predictive control for optimal operation of nonlinear process systems,” *Control Engineering Practice*, vol. 22, pp. 242–251, 2014.

[53] J. B. Rawlings, D. Angeli, and C. N. Bates, "Fundamentals of economic model predictive control," *Proceedings of* 2012 *54th IEEE Conference on Decision and Control*, pp. 3851–3861, 2012.

[54] R. Amrit, J. B. Rawlings, and L. T. Biegler, "Optimizing process economics online using model predictive control," *Computers & Chemical Engineering*, vol. 58, pp. 334–343, 2013.

[55] D. Bonvin and B. Srinivasan, "On the role of the necessary conditions of optimality in structuring dynamic real-time optimization schemes," *Computers & Chemical Engineering*, vol. 51, pp. 172–180, 2013.

[56] D. Bonvin, C. Georgakis, C. Pantelides, M. Barolo, M. Grover, D. Rodrigues, R. Schneider, and D. Dochain, "Linking models and experiments," *Industrial & Engineering Chemistry Research*, vol. 55, pp. 6891–6903, 2016.

[57] A. Toumi, S. Engell, M. Diehl, H. G. Bock, and J. Schlöder, "Efficient optimization of simulated moving bed processes," *Chemical Engineering and Processing: Process Intensification*, vol. 46, pp. 1067–1084, 2007.

[58] L. Biegler, "Technology advances for dynamic real-time optimization," *Computer Aided Chemical Engineering*, vol. 27, pp. 1–6, 2009.

[59] L. Biegler, X. Yang, and G. Fischer, "Advances in sensitivity-based nonlinear model predictive control and dynamic real-time optimization," *Journal of Process Control*, vol. 30, pp. 104–116, 2015.

[60] M. Morari, Y. Arkun, and G. Stephanopoulos, "Studies in the synthesis of control structures for chemical processes: Part i: Formulation of the problem. process decomposition and the classification of the control tasks. analysis of the optimizing control structures," *AIChE Journal*, vol. 26, pp. 220–232, 1980.

[61] S. Engell, "Feedback control for optimal process operation," *IFAC Proceedings Volumes*, vol. 39, pp. 13–26, 2006.

[62] M. Vaccari and G. Pannocchia, "A modifier-adaptation strategy towards offset-free economic MPC," *Processes*, vol. 5, 2017.

[63] R. Amrit, J. B. Rawlings, and D. Angeli, "Economic optimization using model predictive control with a terminal cost," *Annual Reviews in Control*, vol. 35, pp. 178–186, 2011.

[64] M. Diehl, R. Amrit, and J. B. Rawlings, "A lyapunov function for economic optimizing model predictive control," *IEEE Transactions on Automatic Control*, vol. 56, pp. 703–707, 2010.

[65] D. Angeli, R. Amrit, and J. B. Rawlings, "On average performance and stability of economic model predictive control," *IEEE transactions on automatic control*, vol. 57, pp. 1615–1626, 2011.

[66] M. A. Müller, D. Angeli, and F. Allgöwer, "On necessity and robustness of dissipativity in economic model predictive control," *IEEE Transactions on Automatic Control*, vol. 60, pp. 1671–1676, 2014.

[67] M. Zanon and S. Gros, "A new dissipativity condition for asymptotic stability of discounted economic MPC," *Automatica*, vol. 141, pp. 110–287, 2022.

[68] S. Gros and M. Zanon, "A new dissipativity condition for asymptotic stability of discounted economic MPC," *Automatica*, 2022.

[69] M. Zanon, S. Gros, and M. Palladino, "Stability-constrained markov decision processes using MPC," *Automatica*, vol. 143, p. 110399, 2022.

[70] P. Roberts, "An algorithm for steady-state system optimization and parameter estimation," *International Journal of Systems Science*, vol. 10, pp. 719–734, 1979.

[71] P. D. Roberts and T. Williams, "On an algorithm for combined system optimisation and parameter estimation," *Automatica*, vol. 17, pp. 199–209, 1981.

[72] P. Roberts, "Coping with model-reality differences in industrial process optimisation—a review of integrated system optimisation and parameter estimation (ISOPE)," *Computers in Industry*, vol. 26, pp. 281–290, 1995.

[73] W. Gao and S. Engell, "Iterative set-point optimization of batch chromatography," *Computers & Chemical Engineering*, vol. 29, pp. 1401–1409, 2005.

[74] P. Tatjewski, "Iterative optimizing set-point control–the basic principle redesigned," *IFAC Proceedings Volumes*, vol. 35, pp. 49–54, 2002.

[75] A. G. Marchetti, G. François, T. Faulwasser, and D. Bonvin, "Modifier adaptation for real-time optimization—methods and applications," *Processes*, vol. 4, p. 55, 2016.

[76] S. Gros, "An analysis of the directional-modifier adaptation algorithm based on optimal experimental design," *Processes*, vol. 5, 2017.

[77] ——, "Dual-mode batch-to-batch optimization as a markov decision process," *Industrial & Engineering Chemistry Research*, vol. 58, pp. 13 780–13 791, 2019.

[78] A. Marchetti, B. Chachuat, and D. Bonvin, "Real-time optimization of continuous processes via constraints adaptation," *IFAC Proceedings Volumes*, vol. 40, pp. 45–50, 2007.

[79] C. Y. Chen and B. Joseph, "On-line optimization using a two-phase approach: An application study," *Industrial & engineering chemistry research*, vol. 26, pp. 1924–1930, 1987.

[80] Y. Z. Friedman, "What's wrong with unit closed loop optimization?" *Hydrocarbon Processing-Section 1*, vol. 74, pp. 107–116, 1995.

[81] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, pp. 255–260, 2015.

[82] H. Yoo, H. E. Byun, D. Han, and J. H. Lee, "Reinforcement learning for batch process control: Review and perspectives," *Annual Reviews in Control*, vol. 52, pp. 108–119, 2021.

[83] A. Marchetti, B. Chachuat, and D. Bonvin, "Modifier-adaptation methodology for real-time optimization," *Industrial & Engineering Chemistry Research*, vol. 48, pp. 6022–6033, 2009.

[84] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from ADP to MPC*," *European Journal of Control*, vol. 11, pp. 310–334, 2005.

[85] A. BenAmara, "Gas lift: Past and Future," *SPE Middle East Artificial Lift Conference and Exhibition*, pp. 420–425, 2016.

[86] T. Faulwasser and G. Pannocchia, "Toward a unifying framework blending real-time optimization and economic model predictive control," *Industrial & Engineering Chemistry Research*, vol. 58, pp. 13 583–13 598, 2019.

[87] M. Vaccari, D. Bonvin, F. Pelagagge, and G. Pannocchia, "Offset-free economic MPC based on modifier adaptation: Investigation of several gradient-estimation techniques," *Processes*, vol. 9, p. 901, 2021.

[88] E. Oliveira-Silva, C. de Prada, and D. Navia, "Dynamic optimization integrating modifier adaptation using transient measurements," *Computers & Chemical Engineering*, vol. 149, p. 107282, 2021.

[89] J. Matias, J. P. Oliveira, G. A. Le Roux, and J. Jäschke, "Steady-state real-time optimization using transient measurements on an experimental rig," *Journal of Process Control*, vol. 115, pp. 181–196, 2022.

[90] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious nmpc with gaussian process dynamics for autonomous miniature race cars," *2018 European Control Conference (ECC)*, pp. 1341–1348, 2018.

[91] S. Gros, M. Zanon, and A. Bemporad, "Safe reinforcement learning via projection on a safe set: How to achieve optimality?" *IFAC-PapersOnLine*, vol. 53, pp. 8076–8081, 2020.

[92] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Transactions on Automatic Control*, vol. 66, pp. 3638–3652, 2021.

[93] D. Krishnamoorthy, B. Foss, and S. Skogestad, "Real-time optimization under uncertainty applied to a gas lifted well network," *Processes*, vol. 4, p. 52, 2016.

[94] H. Yoo, H. E. Byun, D. Han, and J. H. Lee, "Reinforcement learning for batch process control: Review and perspectives," *Annual Reviews in Control*, vol. 52, pp. 108–119, 2021.

[95] S. J. Qin and L. H. Chiang, "Advances and opportunities in machine learning for process data analytics," *Computers & Chemical Engineering*, vol. 126, pp. 465–473, 2019.

[96] T. Bikmukhametov and J. Jäschke, "Combining machine learning and process engineering physics towards enhanced accuracy and explainability of data-driven models," *Computers & Chemical Engineering*, vol. 138, p. 106834, 2020.

[97] S.-S. Jang, B. Joseph, and H. Mukai, "On-line optimization of constrained multivariable chemical processes," *AIChE Journal*, vol. 33, pp. 26–35, 1987.

[98] J. O. Matias and G. A. Le Roux, "Real-time optimization with persistent parameter adaptation using online parameter estimation," *Journal of Process Control*, vol. 68, pp. 195–204, 2018.

[99] Y. Sawaragi, T. Takamatsu, K. Fukunaga, E. Nakanishi, and H. Tamura, "Dynamic version of steady state optimizing control of a distillation column by trial method," *Automatica*, vol. 7, pp. 509–516, 1971.

[100] W. Bamberger and R. Isermann, "Adaptive on-line steady-state optimization of slow dynamic processes," *Automatica*, vol. 14, pp. 223–230, 1978.

[101] K. Lee and W.-K. Lee, "On-line optimizing control of a nonadiabatic fixed bed reactor," *AIChE journal*, vol. 31, pp. 667–675, 1985.

[102] A. Bhattacharya and B. Joseph, "On-line optimization of chemical processes," *1982 American Control Conference*, pp. 334–337, 1982.

[103] R. McFarlane and D. Bacon, "Empirical strategies for open-loop on-line optimization," *The Canadian Journal of Chemical Engineering*, vol. 67, pp. 665–677, 1989.

[104] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018.

[105] M. O'Kelly, A. Sinha, H. Namkoong, J. Duchi, and R. Tedrake, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, p. 9849–9860, 2018.

[106] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," *Proceedings of* 1995 *34th IEEE Conference on Decision and Control*, vol. 1, pp. 560–564, 1995.

[107] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, pp. 388–403, 2013.

[108] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 2013.

[109] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, "Machine learning-based predictive control of nonlinear processes. part I: Theory," *AIChE Journal*, vol. 65, 2019.

[110] ——, "Machine-learning-based predictive control of nonlinear processes. part II: Computational implementation," *AIChE Journal*, vol. 65, 2019.

[111] M. Lee and S. Park, "A new scheme combining neural feedforward control with model-predictive control," *AIChE Journal*, vol. 38, pp. 193–200, 1992.

[112] V. Venkatasubramanian, "The promise of artificial intelligence in chemical engineering: Is it here, finally?" *AIChE Journal*, vol. 65, pp. 466–478, 2019.

[113] D. M. Himmelblau, "Applications of artificial neural networks in chemical engineering," *Korean journal of chemical engineering*, vol. 17, pp. 373–392, 2000.

[114] J. H. Lee, J. Shin, and M. J. Realff, "Machine learning: Overview of the recent progresses and implications for the process systems engineering field," *Computers & Chemical Engineering*, vol. 114, pp. 111–121, 2018.

[115] E. Baum and F. Wilczek, "Supervised learning of probability distributions by neural networks," *Neural Information Processing Systems*, 1987.

[116] T. M. Jean Saint-Donat, Naveen Bhat, "Neural net based model predictive control," *International Journal of Control*, vol. 54, pp. 1453–1468, 1991.

[117] B. Ydstie, "Forecasting and control using adaptive connectionist networks," *Computers & Chemical Engineering*, vol. 14, pp. 583–599, 1990.

[118] P. Petsagkourakis, I. Sandoval, E. Bradford, D. Zhang, and E. del Rio-Chanona, "Reinforcement learning for batch bioprocess optimization," *Computers & Chemical Engineering*, vol. 133, p. 106649, 2020.

[119] ——, "Constrained reinforcement learning for dynamic optimization under uncertainty," *IFAC-PapersOnLine*, vol. 53, pp. 11 264–11 270, 2020.

[120] J. W. Kim, B. J. Park, T. H. Oh, and J. M. Lee, "Model-based reinforcement learning and predictive control for two-stage optimal control of fed-batch bioreactor," *Computers & Chemical Engineering*, vol. 154, p. 107465, 2021.

[121] M. Mowbray, P. Petsagkourakis, A. D. R. Chanona, and D. Zhang, "Safe chance constrained reinforcement learning for batch process optimization and control," *32nd European Symposium on Computer Aided Process Engineering*, vol. 51, pp. 1039–1044, 2022.

[122] S. Engell, "Feedback control for optimal process operation," *Journal of process control*, vol. 17, pp. 203–219, 2007.

[123] M. L. Darby, M. Nikolaou, J. Jones, and D. Nicholson, "RTO: An overview and assessment of current practice," *Journal of Process control*, vol. 21, pp. 874–884, 2011.

[124] A. G. Marchetti, A. Ferramosca, and A. H. González, "Steady-state target optimization designs for integrating real-time optimization and model predictive control," *Journal of Process Control*, vol. 24, pp. 129–145, 2014.

[125] B. Chachuat, A. Marchetti, and D. Bonvin, "Process optimization via constraints adaptation," *Journal of Process Control*, vol. 18, pp. 244–257, 2008.

[126] B. Chachuat, B. Srinivasan, and D. Bonvin, "Adaptation strategies for real-time optimization," *Computers & Chemical Engineering*, vol. 33, pp. 1557–1567, 2009.

[127] J. F. Forbes and T. E. Marlin, "Model accuracy for economic optimizing controllers: The bias update case," *Industrial & Engineering Chemistry Research*, vol. 33, pp. 1919–1929, 1994.

[128] A. G. Marchetti, T. de Avila Ferreira, S. Costello, and D. Bonvin, "Modifier adaptation as a feedback control scheme," *Industrial & Engineering Chemistry Research*, vol. 59, pp. 2261–2274, 2020.

[129] E. Jahanshahi, C. J. Backi, and S. Skogestad, "Anti-slug control based on a virtual flow measurement," *Flow Measurement and Instrumentation*, vol. 53, pp. 299–307, 2017.

[130] A. D. Quelhas, N. J. C. de Jesus, and J. C. Pinto, "Common vulnerabilities of RTO implementations in real chemical processes," *The Canadian Journal of Chemical Engineering*, vol. 91, pp. 652–668, 2013.

[131] J. Kadam, M. Schlegel, W. Marquardt, R. Tousain, D. Van Hessem, J. Van Den Berg, and O. Bosgra, "A two-level strategy of integrated dynamic optimization and control of industrial processes—a case study," *Computer Aided Chemical Engineering*, vol. 10, pp. 511–516, 2002.

[132] A. C. Zanin, M. T. De Gouvea, and D. Odloak, "Integrating real-time optimization into the model predictive controller of the FCC system," *Control Engineering Practice*, vol. 10, pp. 819–831, 2002.

[133] G. De Souza, D. Odloak, and A. C. Zanin, "Real time optimization (RTO) with model predictive control (MPC)," *Computers & Chemical Engineering*, vol. 34, pp. 1999–2006, 2010.

[134] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning.* Springer, 2006, vol. 4.

[135] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. John Wiley & Sons, Inc., 1994.

[136] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design.* Nob Hill Publishing Madison, WI, 2017, vol. 2.

[137] J. Nocedal and S. J. Wright, *Numerical optimization*, 1st ed. Springer, 1999.

[138] A. Hernandez, *Fundamentals of gas lift engineering: Well design and troubleshooting.* Gulf Professional Publishing, 2016.

[139] V. M. Zavala and L. T. Biegler, "The advanced-step NMPC controller: Optimality, stability and robustness," *Automatica*, vol. 45, pp. 86–93, 2009.

[140] V. M. Zavala, C. D. Laird, and L. T. Biegler, "Fast implementations and rigorous models: Can both be accommodated in NMPC?" *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 18, pp. 800–815, 2008.

[141] H. Chen, A. Kremling, and F. Allgöwer, "Nonlinear predictive control of a benchmark CSTR," *Proceedings of 3rd European control conference*, pp. 3247–3252, 1995.

[142] K. Klatt and S. Engell, “Kontinuierlicher rührkesselreaktor mit neben–und folgereaktionen,” *Nichtlineare Regelung–Methoden, Werkzeuge, Anwendungen, VDI–Berichte*, pp. 101–108, 1993.

[143] M. Zanon and S. Gros, “Safe reinforcement learning using robust MPC,” *IEEE Transactions on Automatic Control*, vol. 66, pp. 3638–3652, 2020.

[144] M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder, “Nominal stability of real-time iteration scheme for nonlinear model predictive control,” *IEE Proceedings-Control Theory and Applications*, vol. 152, pp. 296–308, 2005.

[145] 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, pp. 1–36, 2019.

[146] W.-H. Chen, D. J. Ballance, and J. O’Reilly, “Model predictive control of nonlinear systems: Computational burden and stability,” *IEE Proceedings-Control Theory and Applications*, vol. 147, pp. 387–394, 2000.

[147] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, “The explicit linear quadratic regulator for constrained systems,” *Automatica*, vol. 38, pp. 3–20, 2002.

[148] M. Zanon, V. Kungurtsev, and S. Gros, “Reinforcement learning based on real-time iteration NMPC,” *IFAC-PapersOnLine*, vol. 53, pp. 5213–5218, 2020.

[149] L. O. Santos, P. A. Afonso, J. A. Castro, N. M. Oliveira, and L. T. Biegler, “On-line implementation of nonlinear MPC: an experimental case study,” *Control Engineering Practice*, vol. 9, pp. 847–857, 2001.

[150] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Advances in neural information processing systems*, vol. 12, 1999.

[151] C. Büskens and H. Maurer, “Online optimization of large scale systems, chapter sensitivity analysis and real-time optimization of parametric nonlinear programming problems,” *Berlin Heidelberg, Berlin, Heidelberg*, pp. 3–16, 2001.

[152] J. Jäschke, X. Yang, and L. T. Biegler, “Fast economic model predictive control based on NLP-sensitivities,” *Journal of Process Control*, vol. 24, pp. 1260–1272, 2014.

[153] M. Diehl, H. G. Bock, and J. P. Schlöder, “A real-time iteration scheme for nonlinear optimization in optimal feedback control,” *SIAM Journal on Control and Optimization*, vol. 43, pp. 1714–1736, 2005.

[154] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[155] D. Bertsekas, *Reinforcement learning and optimal control.* Athena Scientific, 2019.

[156] F. Allgöwer and A. Zheng, *Nonlinear model predictive control.* Birkhäuser, 2012, vol. 26.

[157] A. V. Fiacco, *Introduction to sensitivity and stability analysis in nonlinear programming.* Academic press, 1983, vol. 165.

[158] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.

[159] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Transactions on Automatic Control*, vol. 66, pp. 1702–1717, 2020.

[160] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society A*, vol. 474, p. 20180335, 2018.

[161] S. Sawant, A. S. Anand, D. Reinhardt, and S. Gros, "Learning-based mpc from big data using reinforcement learning," *arXiv preprint arXiv:2301.01667*, 2023.

[162] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, pp. 2397–2404, 2023.

[163] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, pp. 1327–1349, 2021.

[164] J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented road map," *IEEE Control Systems Magazine*, vol. 39, pp. 28–99, 2019.

[165] E.-W. Bai and F. Giri, "Introduction to block-oriented nonlinear systems," *Block-oriented Nonlinear System Identification*, pp. 3–11, 2010.

[166] S. Chen, S. A. Billings, and P. Grant, "Non-linear system identification using neural networks," *International journal of control*, vol. 51, pp. 1191–1214, 1990.

[167] T. W. Chow and Y. Fang, "A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics," *IEEE transactions on industrial electronics*, vol. 45, pp. 151–161, 1998.

[168] R. Al Seyab and Y. Cao, "Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation," *Journal of Process Control*, pp. 568–581, 2008.

[169] W. Yu, "Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms," *Information sciences*, vol. 158, pp. 131–147, 2004.

[170] J. de Jesús Rubio and W. Yu, "Nonlinear system identification with recurrent neural networks and dead-zone kalman filter algorithm," *Neurocomputing*, vol. 70, pp. 2460–2466, 2007.

[171] M. Forgione, A. Muni, D. Piga, and M. Gallieri, "On the adaptation of recurrent neural networks for system identification," *arXiv preprint arXiv:2201.08660*, 2022.

[172] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[173] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[174] S. Adhau, V. V. Naik, and S. Skogestad, "Constrained neural networks for approximate nonlinear model predictive control," *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 295–300, 2021.

[175] E. Skomski, S. Vasisht, C. Wight, A. Tuor, J. Drgoňa, and D. Vrabie, "Constrained block nonlinear neural dynamical models," *2021 American Control Conference (ACC)*, pp. 3993–4000, 2021.

[176] J. Drgoňa, K. Kiš, A. Tuor, D. Vrabie, and M. Klaučo, "Differentiable predictive control: Deep learning alternative to explicit model predictive control for unknown nonlinear systems," *Journal of Process Control*, vol. 116, pp. 80–92, 2022.

[177] R. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[178] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," *International conference on machine learning*, pp. 2555–2565, 2019.

[179] D. Masti and A. Bemporad, "Learning nonlinear state-space models using deep autoencoders," *2018 IEEE Conference on Decision and Control (CDC)*, pp. 3862–3867, 2018.

[180] A. Tuor, J. Drgona, and D. Vrabie, "Constrained neural ordinary differential equations with stability guarantees," *arXiv preprint arXiv:2004.10883*, 2020.

[181] M. Schoukens and J. P. Noël, "Three benchmarks addressing open challenges in nonlinear system identification," *IFAC-PapersOnLine*, vol. 50, pp. 446–451, 2017.

[182] R. Relan, K. Tiels, A. Marconato, and J. Schoukens, "An unstructured flexible nonlinear model for the cascaded water-tanks benchmark," *IFAC-PapersOnLine*, vol. 50, pp. 452–457, 2017.

[183] G. Birpoutsoukis, P. Z. Csurcsia, and J. Schoukens, "Efficient multidimensional regularization for volterra series estimation," *Mechanical Systems and Signal Processing*, vol. 104, pp. 896–914, 2018.

[184] A. Svensson and T. B. Schön, “A flexible state–space model for learning nonlinear dynamical systems,” *Automatica*, vol. 80, pp. 189–199, 2017.