Robust NMPC of Large-Scale Systems and Surrogate Embedding Strategies for NMPC

by

Carlos Andrés Elorza Casas

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Applied Science

in

Chemical Engineering

Waterloo, Ontario, Canada, 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Non-linear model predictive control (NMPC) is a promising control algorithm due to its ability to deal with constrained multivariable problems. However, NMPC can be computationally expensive to solve due to its non-linear nature, multiple interacting process units and the presence of model uncertainty. Real-world NMPC applications also necessitate state estimation for feedback control. While robust NMPC and state estimators have been studied individually for large-scale problems, understanding their combined impact is crucial for wider NMPC adoption. Integrating tractable Machine Learning (ML) surrogates, particularly Neural Networks (NNs), into NMPC to reduce the computational load is an emerging strategy. However, embedding NN surrogates in NMPC, in a form amenable to simultaneous solution approaches, remains unresolved.

This thesis aims to address two major NMPC implementation issues. First, this work analyses the combined impact of uncertainty and state estimation on the performance of NMPC on large-scale systems. Two scenario-based robust approaches to NMPC, multi-scenario NMPC (MSc-NMPC) and multi-stage NMPC (MS-NMPC), are implemented on the benchmark Tennessee-Eastman (TE) process in closed-loop using two standard state estimation algorithms, Extended Kalman Filter (EKF) and Moving Horizon Estimation (MHE). Robust NMPC with MHE is shown to prevent constraint violation while closely tracking the set-points under process uncertainty where traditional NMPC failed. The additional computational time required to solve the robust NMPC and MHE does not cause significant delays for the sampling time considered, demonstrating their applicability to challenging large-scale industrial chemical and manufacturing processes.

This work also aims to benchmark various strategies for embedding NN surrogates in NMPC. One strategy embeds NN models as explicit algebraic constraints within the optimization framework, leveraging the auto differentiation (AD) of algebraic modelling languages (AMLs) to evaluate the

derivatives. Alternatively, the surrogate can be evaluated externally from the optimization framework, using the efficient AD of ML environments. Physics-informed NNs (PINNs) and Physics-informed Convolutional NNs (PICNNs) are used as NN surrogates due to their ability to maintain fidelity to fundamental physics laws while reducing the need for historical/process data. The study reveals that replacing mechanistic models with NN surrogates may not always offer computational advantages, even with highly nonlinear systems. Smooth activation functions provide little to no advantage over the mechanistic equations when a local non-linear program (NLP) solver is used. Moreover, the external evaluation of the NN surrogates often outperforms the embedding as algebraic constraints, likely due to the difficulty in initializing the auxiliary variables and constraints introduced with the explicit algebraic reformulations.

Acknowledgements

I would like to acknowledge my supervisors, Prof. Luis Ricardez-Sandoval and Prof. Joshua Pulsipher. I have had the opportunity to grow professionally and develop my technical skills and knowledge. Both of you are not only knowledgeable and experts in your fields, but you are also passionate about what you do, encourage your students to perform their best, and understand the situations that may arise beyond the academic/professional settings in your students' lives. Thank you for pushing me to be the best I can be. Additionally, thank you to Mahshad Valipour for her guidance during my first project.

I would also like to thank my family, in particular, my mother, Mildred Casas Fulla, for their ongoing support. Despite being a single mother, she has done everything in her power and made many sacrifices to always provide me with the tools I needed to be successful. Additionally, I would like to mention my grandparents, Jaime and Deyanira Casas, who have unfortunately both passed. They were always loving and kind, and the best role models I could have wished for.

Moreover, I wish to thank my friends. Those who have been with me for most of my life and those whom I have met along the way. Your support has been crucial for relieving the times that have been stressful and difficult. To Andres Camilo, we have known each other since first grade, you are my longest-lasting friendship and I hope that continues for a very long time. To Nicolas, thank you for being a friend since we met in high school. To Esteban, Juanse, Natalia, Sofia and Valentina, your friendship over the past year has been very valuable to me. Finally, Carlo, Connor and Mitchell, we will forever be lab group 9.

Thank God for putting all of you in my life and for giving me the opportunity to go through this journey.

Thank you to the University of Waterloo for the financial support through the Engineering Excellence Master's Fellowship and to the Natural Sciences and Research Council of Canada for funding this research.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AD	Automatic Differentiation
AML	Algebraic Modelling Language
CNN	Convolutional Neural Network
DAE	Differential Algebraic Equations
EKF	Extended Kalman Filter
FNN	Fully-Connected Feedforward NN
FS	Full-space Formulation
EF1	External Function Formulation Using *torch.func*
EF2	External Function Formulation Using *torch.autograd.functional*
IPOPT	Interior Point Optimizer
LSTM	Long-short term memory network
MHE	Moving Horizon Estimator
MIMO	Multiple Input-Multiple Output
ML	Machine Learning
MPC	Model Predictive Control
MSc-NMPC	Multiscenario NMPC
MS-NMPC	Multistage NMPC
NLP	Non-Linear Program
NMPC	Non-linear Model Predictive Control
NN	Neural Network
ODE	Ordinary Differential Equation
OMLT	Optimization and Machine Learning Toolkit
PDE	Partial differential equation
PID	Proportional, integral, derivative
PICNN	Physics-Informed CNN
PINN	Physics-Informed NN
RH	Robust Horizon
RK	Runge-Kutta Methods
RK45	Runge-Kutta Method of Order 5(4)
RMSE	Root-Mean-Squared Error
RNN	Recurrent NN
RS	Reduced-space Formulation
TE	Tennessee Eastman

List of Symbols

Symbol	Description
$\boldsymbol{A}_k$	Jacobian matrix of process model at sampling time k
A	reactor cross-sectional area
$AGSP$	agitator speed
$\boldsymbol{b}_l$	bias vector at layer l
C	Concentration, mol L^{-1}
$C_{p,s}$	specific heat capacity of component s, J mol^{-1} K^{-1}
DEN	reaction kinetics term
$\boldsymbol{e}_i$	vector of process model parameters
E_A	activation energy, J mol^{-1}
$\boldsymbol{E}_l$	dense or linear layer weight matrix at layer l
f	process model
F_q	flow rate of stream q, mol s^{-1}
F_s	molar flow rate of component s, mol s^{-1}
$F_{cw,r}$	reactor cooling water, kg s^{-1}
$F_{cw,s}$	separator cooling water, kg s^{-1}
g	function of process constraints
$\boldsymbol{H}_k$	Jacobian matrix of measurement model at sampling time k
h	measurement model function
k_{rxn}	reaction rate constant, mol g^{-1} s^{-1} or mol L^{-1} s^{-1}
$\boldsymbol{K}_k$	Kalman gain at sampling time k
K_j	Equilibrium constant of reaction j
$K_{a,s}$	adsorption constant of component s
$loss$	loss function
$\boldsymbol{L}$	matrix of weights on the output variables
M	control horizon
N	MHE sliding window length
NN_θ	neural network parametrized by θ
$\mathcal{O}$	observability matrix
$\boldsymbol{p}$	PDE model parameters
P	prediction horizon
P_s	partial pressure of component s, kPa
P_r	reactor pressure, kPag
$\boldsymbol{P}_k$	state error covariance matrix at sampling time k
p_8	compressor recycle valve position
p_9	purge valve position
p_{stm}	steam valve position
$\boldsymbol{Q}_k$	process noise covariance matrix at sampling time k
$\boldsymbol{R}_k$	measurement noise covariance matrix at sampling time k
R	ideal gas constant, J mol^{-1} K^{-1}
RR'_j	rate of reaction j, mol g^{-1} s^{-1}

$\boldsymbol{r}_l$	output of FNN layer l
$\boldsymbol{r}_{l,v}$	output of CNN layer l at spatial location v
T_r	reactor temperature, ℃
T_m	mixing zone temperature, ℃
T_s	separator temperature, ℃
T_{str}	stripper temperature, ℃
$\boldsymbol{u}^L$, $\boldsymbol{u}^U$	bounds on the manipulated variables
$\boldsymbol{u}_i^*$	vector of manipulated variables at time interval i in the NMPC prediction
$\boldsymbol{v}_k$	vector of measurement noises at sampling time k
$V_{L,r}$	reactor liquid level, m^3
$V_{L,s}$	separator liquid level, m^3
$V_{L,str}$	stripper liquid level, m^3
$\boldsymbol{W}$	matrix of weight on the input variables
$\boldsymbol{w}_k$	vector of process noises at sampling time k
$\boldsymbol{x}^L$, $\boldsymbol{x}^U$	bounds on the states
$\boldsymbol{x}_i^*$	vector of predicted state variables at time interval i
$\widehat{\boldsymbol{x}}_k$	state estimate at sampling time k
$\overline{\boldsymbol{x}}_{k-N}$	state expected value at sampling time $k-N$
$x_{i,str}$	liquid mole fraction of species i in the stripper
$\boldsymbol{y}_i$	vector of process outputs at time interval i
y_s	vapour mole fraction of component s
Greek and Letter-like Symbols	**Description**
$\mathcal{B}$	boundary conditions operator
$\Delta\boldsymbol{u}_i^*$	change in the manipulated variable at time interval i in the NMPC prediction
ΔG_j^0	Standard Gibb's free energy of reaction j, J mol^{-1}
ΔH_j	enthalpy of reaction j, J mol^{-1}
$\mathcal{F}_{\boldsymbol{z}}$	non-linear PDE differential operator
$\mathcal{I}$	initial conditions operator
φ_{k-N}	arrival cost
ν	stoichiometric coefficient
ρ_c	catalyst packed density, g cm^{-3}
σ	activation function
ω_j	weight assigned to scenario j

Quote

If you look for truth, you may find comfort in the end; if you look for comfort you will not get either comfort or truth only soft soap and wishful thinking to begin, and in the end, despair.

– C. S. Lewis

1. Introduction

Non-Linear Model Predictive Control (NMPC) has become a widely accepted control algorithm to deal with constrained multivariable problems (Biegler and Zavala, 2009; Lee, 2011; Rawlings, Mayne and Diehl, 2017; Valipour and Ricardez-Sandoval, 2021a). However, large-scale chemical processes pose an important challenge to the implementation of NMPC. The models for industrial-scale processes are often large, highly non-linear, and involve several inputs, outputs, and states (Biegler and Thierry, 2018). In addition, dynamic systems often feature several constraints, and multiple sources of uncertainty such as plant/model mismatch, unmeasured disturbances, and measurement errors that can deteriorate controller performance or even lead to constraint violation (Bemporad and Morari, 1999). Moreover, NMPC requires initial states for closed-loop feedback control; hence, the impact of a state estimator on the process performance must be considered for practical applications (Valipour and Ricardez-Sandoval, 2021a). As a result, NMPC problems are computationally expensive and can limit their application as online feedback controllers (Mesbah *et al.*, 2022). Alternative modelling approaches, such as Machine Learning (ML) surrogate models, are gaining attention due to their universal approximation properties and the fast evaluation times of trained models (Antonelo *et al.*, 2022; Esche *et al.*, 2022; Misener and Biegler, 2023; Daoutidis *et al.*, 2024).

Robust methods have been developed to explicitly deal with uncertainty in the NMPC formulation (Bemporad and Morari, 1999; Mesbah, 2016). In robust NMPC, the uncertain parameters are bounded inside an uncertainty set and the constraints must be satisfied for all possible uncertainty realizations. General robust NMPC formulations are intractable as they result in non-linear semi-infinite programs (Yanıkoğlu, Gorissen and den Hertog, 2019). Scenario-based approaches relax the requirement to be feasible over the entire uncertainty set to only a few realizations of the

uncertainty sampled from this set, resulting in tractable Non-Linear Programs (NLPs). Two notable scenario-based approaches are Multi-scenario NMPC (MSc-NMPC) and Multi-stage NMPC (MS-NMPC). MSc-NMPC considers discrete scenarios of the uncertain parameters that are time-independent (Huang and Biegler, 2009; Piceno-Díaz *et al.*, 2020). MS-NMPC uses discrete scenarios that evolve over a scenario tree where future control actions in the prediction horizon can adjust to past realizations of the uncertainty (Lucia and Engell, 2015; Tătulea-Codrean, Fischer and Engell, 2020). MS-NMPC is generally less conservative than MSc-NMPC due to its ability to adjust to past uncertainty in the prediction horizon, however, the branches in the scenario tree grow exponentially with the robust horizon, thus, the problem very quickly increases in size. Real-world chemical engineering applications generally involve multiple interacting process units with several states, control inputs and outputs, highlighting the importance of considering the challenges involved in the implementation of NMPC to those processes. The size of the computational problem inevitably grows with the size of the process model, making online implementation a challenge for large-scale systems. In addition, large process models from different sectors often involve a larger number of constraints and sources of uncertainty (Zheng, Ricardez-Sandoval and Budman, 2020; Patrón and Ricardez-Sandoval, 2024; Patrón, Toffolo and Ricardez-Sandoval, 2024; Toffolo, Meunier and Ricardez-Sandoval, 2024). Although several studies have investigated the application of NMPC to large-scale systems, there is a gap in the literature regarding the effect on performance caused by the combined presence of uncertainty and the necessary use of state estimation on large-scale systems.

As highlighted above, NMPC problems can be computationally expensive to solve. Especially, problems modelled by non-linear partial differential equations (PDEs) as they often result in models with a large number of states. Systems of PDEs modelling plug-flow reactors (PFRs) are

of particular interest because they are common problems in chemical engineering. Direct transcription is a popular approach to make PDE systems amenable to NLP solvers (Cervantes and Biegler, 2008; Biegler, 2021). Direct transcription discretizes the PDEs via finite differences or collocation to generate algebraic constraints. This approach results in a large system of non-linear algebraic equality constraints; hence, the NLP solver performs the optimization and solves the modelling equations simultaneously. To find a solution to the NMPC optimization problem, NLP solvers often require the gradients of the objective and constraints with respect to the decision variables (Luenberger and Ye, 2016). Although simultaneous approaches generate a larger optimization problem than shooting methods, the evaluation of gradients is facilitated through the auto differentiation (AD) of algebraic modelling languages (AMLs), where even second derivatives can be evaluated for relatively low cost (Griewank and Walther, 2003). NLP solvers leverage second derivatives to accelerate convergence to optimal solutions (Wächter and Biegler, 2006; Luenberger and Ye, 2016). Alternative to direct transcription, shooting methods require solving the PDE system with an integrator/simulator over the entire prediction horizon at every iteration of optimization. Shooting methods are simple to implement and robust in open-loop stable systems, however, the calculation of gradients becomes expensive in problems with many decision variables (Biegler, 2021); as a result, second derivatives are rarely calculated.

Both approaches to NMPC, direct transcription and shooting, can be computationally taxing for complex systems like PDE-constrained problems, preventing the broad application in online feedback control where the NMPC problem must be resolved at every sampling step. Model simplifications, like approximating the problem with a linearized model, can reduce the complexity of the problem; however, this also means the model may not accurately represent the actual process. Hence, in an effort to alleviate the computational burden, surrogate models based

on types of Neural Networks (NNs) are becoming a particularly popular choice to replace the mechanistic models (Daoutidis *et al.*, 2024). Physics-informed NNs (PINNs) have become a prominent choice for ML surrogates due to their ability to decrease the reliance on historical/process data while maintaining fidelity to fundamental physics laws (Antonelo *et al.*, 2022; Hao *et al.*, 2023). Data can be difficult to obtain due to the difficulty of experiments or lack of historical data, which is one major obstacle for general ML applications. However, the embedding of NN surrogates into the NMPC framework is still an open problem. Previous works have used PINNs as surrogates, but, they have employed sequential/shooting methods to solve the surrogate NMPC problem (Antonelo *et al.*, 2022; Nicodemus *et al.*, 2022; Sanyal and Roy, 2023). Given the properties of simultaneous approaches, they are major contenders for solving NMPC problems (Wächter and Biegler, 2006; Biegler and Zavala, 2009). There are two promising strategies to include NN surrogates within simultaneous solution approaches. The embedding of the NN surrogate as algebraic equations in the AML is a potential strategy (Ceccon *et al.*, 2022). This approach can take advantage of the AD from AMLs to evaluate gradients. On the other hand, the AD from ML environments is also efficient (Baydin *et al.*, 2018). Hence, embedding the ML model as an external function is another potential alternative, where the AD from the AML is bypassed, and instead, the gradients are provided to the NLP solver from the ML environment. To the author's knowledge, the different embedding strategies have not been compared in the literature highlighting the computational times and challenges in the implementation.

1.1. Research Objectives and Contributions

This thesis aims to address two major issues regarding the implementation of NMPC. First, this work aims to study the combined effect of uncertainty and the necessary use of state estimation on the performance and implementation of NMPC on large-scale chemical and manufacturing

systems. These challenges present a major obstacle to the widespread adoption of NMPC. The following objectives were proposed to address this gap in the literature:

- Apply two robust NMPC approaches (MSc-NMPC and MS-NMPC) to handle parametric uncertainty in large-scale systems.
- Leverage the challenging plant-wide TE problem, which has several interacting units, is highly non-linear and is open-loop unstable, as a representative large-scale problem.
- Evaluate the closed-loop performance of the robust NMPC with two standard state estimation algorithms: Extended Kalman Filter (EKF) and Moving Horizon Estimator (MHE).

As discussed, NMPC problems are computationally taxing. Hence, this work also aims to explore different NN surrogate embedding strategies in NMPC and benchmark their computational times and implementation challenges. A comparative analysis is currently missing in the literature comparing surrogate embedding strategies that leverage simultaneous solution approaches in the context of NMPC. The following objectives were proposed to fill-in this gap in the literature:

- Apply the embedding of NN surrogate as algebraic constraints in the AML, taking advantage of the AD from AMLs, to solve the NMPC problem.
- Implement the embedding of the NN surrogate within the AML by treating it as an external function where the Jacobian and Hessian information are evaluated outside the AML, taking advantage of the AD from the ML environment.
- Compare the challenges and limitations of the proposed embedding methods on PDE-constrained NMPC case studies, using non-linear plug-flow reactors (PFRs) of increasing complexity.

- Leverage physics-informed NNs (PINNs) and physics-informed convolutional NNs (PICNNs) to eliminate the need for historical/process data.

The contribution of the first aim outlined above is a comprehensive assessment of the challenges of simultaneously dealing with uncertainty and state estimation on a large-scale system while using comprehensive robust NMPC strategies. This work may broaden the applicability of robust NMPC and state estimation to large-scale systems and highlight the limitations of different algorithms. Regarding the second aim of this thesis, i.e., NN surrogate embedding strategies within NMPC. These strategies were benchmarked based on their computational times and implementation challenges using PINNs and PICNNs as surrogates and comparing their performance against the traditional approach. The contribution of this part of this thesis was to inform what strategies are most efficient to embed NN surrogates in NMPC formulations solved using simultaneous approaches. This work may provide insights about when using a surrogate may be beneficial in NMPC while using a direct transcription method for the solution of the NMPC formulation.

1.2. Thesis Structure

This thesis is structured as follows:

Chapter 2 provides a literature review highlighting the relevant work and background in the areas of NMPC, robust NMPC, state estimation, PINNs and ML surrogates in control. Gaps in the implementations of NMPC on large-scale systems and the use of surrogates on NMPC are identified. It is also confirmed that the contributions presented in this thesis are novel.

Chapter 3 presents the implementation of MSc-NMPC and MS-NMPC on the benchmark TE process. The impact on the controller performance is assessed as a result of the presence of process uncertainty when coupled with the additional layer of state estimation. To the author's knowledge,

the performance of NMPC and state estimation in the presence of uncertainty had not been studied on a large-scale system like the TE process. Outcomes from this study have been published in the literature (Elorza Casas, Valipour and Ricardez Sandoval, 2023). This paper was written entirely by myself. Dr. Valipour provided guidance on the methods implemented. It was edited by my supervisor, Prof. Ricardez-Sandoval. Permission has been granted by the publisher to include the content in this thesis.

Chapter 4 presents the benefits, challenges and limitations of different surrogate embedding strategies. This is accomplished via the use of PINNs and PICNNs as surrogate modes, applying the approach of embedding the NN surrogate as algebraic equations and the external function evaluation approach as the two major embedding strategies. To the author's knowledge, formal benchmarking has not been conducted in the literature on simultaneous optimization approaches.

Chapter 5 presents concluding remarks on the implementation of robust NMPC and state estimation on a large-scale process, and the surrogate embedding strategies for NMPC. In addition, recommendations for future work are also outlined in this chapter.

2. Background and Literature Review

Large-scale systems present a major challenge to NMPC. The presence of model uncertainty, the requirement of a state estimator and the computational costs associated with the simultaneous interactions between the state estimator, controller and uncertain process plant are aspects that must be considered for the widespread adoption of NMPC. Chapter 2.1 provides a summary of MPC/NMPC and previous work on the implementation of robust NMPC and state estimation on large-scale systems. Due to the previously discussed challenges, NMPC problems can be computationally taxing and difficult to solve which can limit their online applications (Mesbah *et al.*, 2022). As a result, modelling alternatives, such as ML surrogates, are gaining attention (Daoutidis *et al.*, 2024). In particular, PINNs are an attractive alternative modelling approach in NMPC. Chapter 2.2 provides a broad description of PINNs and some of the advances in this field followed by the extensions of PINNs for dealing with control problems and how NN surrogate models have been embedded in NMPC. The end of each subsection highlights the major findings and gaps identified from the literature review concerning each subject.

2.1. Non-linear Model Predictive Control

Non-linear Model Predictive Control (NMPC) seeks to find the optimal control actions, $\boldsymbol{u}_k^*$, at the current time interval of the process, k, based on some desired performance metric (e.g., minimization of the squared set-point tracking error of the output variables or a user-defined economic function). This is accomplished by placing the dynamic process plant model as a constraint within the optimization problem. In this work, linear MPC is referred to as just MPC, while non-linear MPC will be referred to as NMPC. MPC uses a linear model as an internal process model whereas NMPC, as the name suggests, uses a non-linear process model thus aiming to achieve more accurate predictions. As shown in Figure 1, NMPC optimizes the future control

actions, $\boldsymbol{u}$, such that the error between the predicted output, $\boldsymbol{y}$ (unfilled circle markers), and its set-point (thin dashed line) is minimized. It also shows that NMPC only predicts for a finite prediction horizon, P, and the control actions (bold dashed line in Figure 1) may be restricted to remain constant after the control horizon, M. Feedback control functions in a receding horizon manner, i.e., when the NMPC problem is solved, the control action at the current sampling interval k is implemented in the process and the dynamics of the process evolve. The prediction horizon shifts forward one sampling step. The initial condition, $\hat{\boldsymbol{x}}_k$, is updated from the process measurements (usually a state estimator is necessary), the NMPC problem is resolved to obtain the next control action, and the cycle is repeated. NMPC has some key advantages over traditional control schemes such as proportional-integral-derivate (PID) controllers. NMPC can take process limits into account as constraints in the optimization formulation (e.g., an upper limit on the operating temperature can be declared by the constraint $T \leq T^U$, which means NMPC will find control actions that prevent the prediction model from violating this constraint). Furthermore, NMPC can drive the system optimally to a user-defined objective. However, the benefits of NMPC hinge on the accuracy of the model (i.e., plant/model mismatch can deteriorate performance) and complex systems with fast dynamics limit the applications of NMPC, as the NMPC problem can become too computationally costly to solve online (Lee, 2011; Rawlings, Mayne and Diehl, 2017; Biegler and Thierry, 2018). The following is a conventional discrete-time NMPC formulation:

$$\min_{\boldsymbol{u}_i^* \in \mathbb{R}^{N_u}\ \forall i \in \{k,k+1,\dots,k+P-1\}} \sum_{i=k+1}^{k+P} \left\| \boldsymbol{y_{sp}} - \boldsymbol{y}_i^* \right\|_{\boldsymbol{L}}^2 + \sum_{i=k}^{k+M-1} \|\Delta \boldsymbol{u}_i^*\|_{\boldsymbol{W}}^2 \tag{1}$$

$$\text{s.t. } \boldsymbol{x}_{i+1}^* = f(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*) \quad \forall i \in \{k, k+1, \dots, k+P-1\} \tag{2}$$

$$\boldsymbol{y}_i^* = h(\boldsymbol{x}_i^*) \quad \forall i \in \{k, k+1, \dots, k+P\} \tag{3}$$

$$g(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*, \boldsymbol{y}_i^*) \leq 0 \quad \forall i \in \{k, k+1, \dots, k+P\} \tag{4}$$

$$\boldsymbol{x}^L \leq \boldsymbol{x}_i^* \leq \boldsymbol{x}^U \quad \forall i \in \{k, k+1, \dots, k+P\} \tag{5}$$

$$\boldsymbol{u}^L \leq \boldsymbol{u}_i^* \leq \boldsymbol{u}^U \quad \forall i \in \{k, k+1, \dots, k+P-1\} \tag{6}$$

$$\boldsymbol{u}_i^* = \boldsymbol{u}_{k+M-1}^* \quad \forall i \in \{k+M, k+M+1, \dots, k+P-1\} \tag{7}$$

$$\Delta \boldsymbol{u}_i^* = \boldsymbol{u}_i^* - \boldsymbol{u}_{i-1}^* \quad \forall i \in \{k, k+1, \dots, k+P-1\} \tag{8}$$
$$\boldsymbol{x}_k^* = \hat{\boldsymbol{x}}_k \tag{9}$$

where:
$\boldsymbol{x}_i^* \in \mathbb{R}^{N_x}, \boldsymbol{y}_i^* \in \mathbb{R}^{N_y} \quad \forall i \in \{k, k+1, \dots, k+P\}$
$\boldsymbol{L} \in \mathbb{R}^{N_y \times N_y}, \boldsymbol{W} \in \mathbb{R}^{N_u \times N_u}$
$\boldsymbol{x}^L \in \mathbb{R}^{N_x}, \boldsymbol{x}^U \in \mathbb{R}^{N_x}, \boldsymbol{u}^L \in \mathbb{R}^{N_u}, \boldsymbol{u}^U \in \mathbb{R}^{N_u}, \hat{\boldsymbol{x}}_k \in \mathbb{R}^{N_x}$
$f: \mathbb{R}^{N_x \times N_u} \to \mathbb{R}^{N_x}, h: \mathbb{R}^{N_x} \to \mathbb{R}^{N_y}, g: \mathbb{R}^{N_x \times N_u \times N_y} \to \mathbb{R}^{N_g}$

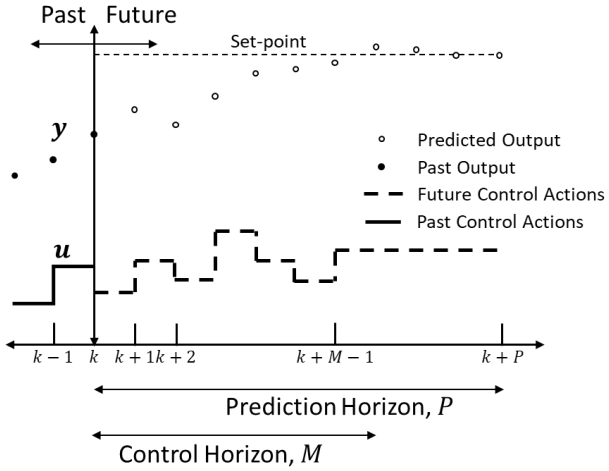


Figure 1 Depiction of the prediction horizon in NMPC.

The vector of predicted states is denoted by $\boldsymbol{x}_i^*$ whereas $\boldsymbol{y}_i^*$ denotes the process outputs. The time interval within the prediction horizon is denoted by i, which goes from the current plant time interval k to $k+P$. The scalar P denotes the prediction horizon. The term $\|\boldsymbol{v}\|_{\boldsymbol{B}}^2$ in the objective function denotes the L_2-norm of vector $\boldsymbol{v}$ with weight matrix $\boldsymbol{B}$ squared ($\|\boldsymbol{v}\|_{\boldsymbol{B}}^2 \coloneqq \boldsymbol{v}^T \boldsymbol{B} \boldsymbol{v}$). The objective function, (1), is a common performance metric for NMPC where the first term aims to minimize the set-point tracking error between the process outputs, $\boldsymbol{y}_i^*$, and their corresponding set-points, $\boldsymbol{y}_{\boldsymbol{sp}}$. The second term aims to regulate the changes in the control actions, $\Delta \boldsymbol{u}_i^*$ (defined in equation (8)), from sampling step to the next. $\boldsymbol{L}$ and $\boldsymbol{W}$ are user-defined output and input weight matrices, respectively, that assign the relative importance to each variable in the objective function.

These are typically diagonal matrices with positive elements. The set of constraints in equation (2) represents the non-linear discrete-time process model. Real-world systems are generally modelled by differential-algebraic equations (DAEs) or PDEs. To generate a tractable optimization problem, one major approach is the direct transcription of the system of DAEs/PDEs (Cervantes and Biegler, 2008; Biegler, 2021). In direct transcription, the DAEs/PDEs are discretized over the continuous domains (temporal and spatial) via methods such as finite difference or collocation, generating algebraic constraints that can be handled by algebraic modelling languages (AMLs) for simultaneous optimization. This results in the discrete-time formulation shown in (1)-(9). The set of constraints in equation (3) represents the map from states, $\boldsymbol{x}_i^*$, process outputs, $\boldsymbol{y}_i^*$. The set of inequality constraints, (4), represents any general non-linear path constraints that may be imposed on the states, control inputs or control outputs (e.g., a particular control output may be constrained to follow a particular trajectory). This highlights the flexibility of NMPC to handle physical and process constraints imposed on the system. Constraints (5) and (6) impose upper and lower limits on the states and control actions (e.g., a valve can only operate between 0 and 100% open). Constraint (7) ensures that the control actions remain constant after the control horizon, M. The state vector, $\hat{\boldsymbol{x}}_k$, is the initial condition to the NMPC problem that is provided through state feedback and the constraint (9) ensures that it is the initial state to the process model (2). In practice, the initial condition must be provided by a state estimator because the process outputs are usually limited and do not necessarily correspond directly to the process states. For example, it may only be possible to measure the temperature of a reactor and, hence, other state variables, such as concentration, must be estimated from this measurement. Since the states are not directly measured, the state estimator may introduce some error between the actual state of the process, $\boldsymbol{x}_k$, and the state that is provided to NMPC, $\hat{\boldsymbol{x}}_k$, especially if the measurements are noisy. This may

deteriorate the performance of the controller; hence, accurate state estimation schemes are necessary for NMPC. Furthermore, state estimation adds another layer of computations that must be performed online, further increasing the computational cost of NMPC.

2.1.1. NMPC Under Uncertainty

To achieve good controller performance (i.e., the controlled variables closely track their set-points while process variables remain within their operating limits), the process model, (2), must be an accurate representation of the actual process. However, sources of uncertainty may result in plant/model mismatch that hinders the performance or even results in infeasible solutions. A major source of uncertainty is parametric uncertainty (i.e., inaccurate model parameters). Stochastic and robust NMPC are the foremost approaches for handling uncertainty in NMPC. This thesis focuses primarily on robust NMPC; however, Mesbah (2016) provides an in-depth review of stochastic methods for MPC/NMPC. A general robust NMPC formulation assumes that uncertain parameters, $\boldsymbol{e}_i$, belong to a bounded compact set, $\mathcal{E}$, and process constraints must be satisfied for all possible uncertainties within the set (Mayne, 2014). The general formulation for robust NMPC can be seen in equations (10)-(18). This NMPC formulation shows that the constraints must be satisfied for all possible realizations of $\boldsymbol{e}_i$ within $\mathcal{E}$. Thus, if a solution is found, it is guaranteed that the constraints will be satisfied if the uncertainty lies within the uncertainty set. However, even if $\mathcal{E}$ is compact and bounded, there is an infinite number of possible realizations for $\boldsymbol{e}_i$. For example, assume there is only one uncertain parameter α, i.e., $\boldsymbol{e}_i = [\alpha]$, then, the uncertainty set would be the bounds around α, $\alpha \in \mathcal{E} = \{\alpha: \alpha^L \leq \alpha \leq \alpha^U\}$. There is an infinite number of possible values α can take between α^L and α^U, so, problem (10)-(18) requires the consideration of an infinite number of constraints. Thus, the optimization problem (10)-(18) is a semi-infinite

program which is intractable (Mayne *et al.*, 2011; Leyffer *et al.*, 2018; Holtorf, Mitsos and Biegler, 2019; Yanıkoğlu, Gorissen and den Hertog, 2019).

$$\min_{\boldsymbol{u}_i^* \in \mathbb{R}^{N_u}\ \forall i \in \{k,k+1,\dots,k+P-1\}} \sum_{i=k+1}^{k+P} \left\| \boldsymbol{y_{sp}} - \boldsymbol{y}_i^* \right\|_{\boldsymbol{L}}^2 + \sum_{i=k}^{k+M-1} \|\Delta \boldsymbol{u}_i^*\|_{\boldsymbol{W}}^2 \tag{10}$$

$$\text{s.t. } \boldsymbol{x}_{i+1}^* = f(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*, \boldsymbol{e}_i) \quad \forall \boldsymbol{e}_i \in \mathcal{E} \forall i \in \{k, k+1, \dots, k+P-1\} \tag{11}$$

$$\boldsymbol{y}_i^* = h(\boldsymbol{x}_i^*) \quad \forall i \in \{k, k+1, \dots, k+P\} \tag{12}$$

$$g(\boldsymbol{x}_i^*, \boldsymbol{u}_i^*, \boldsymbol{y}_i^*) \leq 0 \quad \forall \boldsymbol{e}_i \in \mathcal{E} \forall i \in \{k, k+1, \dots, k+P\} \tag{13}$$

$$\boldsymbol{x}^L \leq \boldsymbol{x}_i^* \leq \boldsymbol{x}^U \quad \forall \boldsymbol{e}_i \in \mathcal{E} \forall i \in \{k, k+1, \dots, k+P\} \tag{14}$$

$$\boldsymbol{u}^L \leq \boldsymbol{u}_i^* \leq \boldsymbol{u}^U \quad \forall i \in \{k, k+1, \dots, k+P-1\} \tag{15}$$

$$\boldsymbol{u}_i^* = \boldsymbol{u}_{k+M-1}^* \quad \forall i \in \{k+M, k+M+1, \dots, k+P-1\} \tag{16}$$

$$\Delta \boldsymbol{u}_i^* = \boldsymbol{u}_i^* - \boldsymbol{u}_{i-1}^* \quad \forall i \in \{k, k+1, \dots, k+P-1\} \tag{17}$$

$$\boldsymbol{x}_k^* = \widehat{\boldsymbol{x}}_k \tag{18}$$

where:
$\boldsymbol{x}_i^* \in \mathbb{R}^{N_x}, \boldsymbol{y}_i^* \in \mathbb{R}^{N_y} \quad \forall i \in \{k, k+1, \dots, k+P\}$
$\boldsymbol{L} \in \mathbb{R}^{N_y \times N_y}, \boldsymbol{W} \in \mathbb{R}^{N_u \times N_u}, \mathcal{E} \subset \mathbb{R}^{N_e}$
$\boldsymbol{x}^L \in \mathbb{R}^{N_x}, \boldsymbol{x}^U \in \mathbb{R}^{N_x}, \boldsymbol{u}^L \in \mathbb{R}^{N_u}, \boldsymbol{u}^U \in \mathbb{R}^{N_u}, \widehat{\boldsymbol{x}}_k \in \mathbb{R}^{N_x}$
$f\colon \mathbb{R}^{N_x \times N_u \times N_e} \to \mathbb{R}^{N_x}, h\colon \mathbb{R}^{N_x} \to \mathbb{R}^{N_y}, g\colon \mathbb{R}^{N_x \times N_u \times N_y} \to \mathbb{R}^{N_g}$

Alternative tractable formulations have been proposed that either relax the requirement of feasibility for all possible realizations or reformulate the problem in a tractable form. To name a few, min-max approaches use an open-loop formulation that calculates the control actions that minimize the worst-case cost with respect to a bounded set of parameters, $\boldsymbol{e}_i$ (Lee and Yu, 1997; Scokaert and Mayne, 1998; Alamir and Balloul, 1999; Lee, 2011). The problem is formulated as a bilevel optimization program wherein the lower level, the objective is maximized, and the uncertain parameters are the decision variables, thereby, finding the worst-case scenario for $\boldsymbol{e}_i$ in the uncertainty set. In the upper level, the objective is minimized, and decision variables are the usual control actions. As a result, this approach tends to be overly conservative since it does not consider feedback in the prediction. Min-max formulations have generally been limited to linear systems because to guarantee that the worst-case is found, strong assumptions about convexity

must be made about the non-linear system, which is difficult to guarantee for general non-linear systems (Grancharova and Johansen, 2005, 2012; Raimondo *et al.*, 2007; Limon *et al.*, 2009; Lucia and Engell, 2012; Bayer, Muller and Allgower, 2016; Jang, Lee and Biegler, 2016; Holtorf, Mitsos and Biegler, 2019). Approximations of the min-max NMPC problem have been made where the infinite set $\mathcal{E}$ is approximated by a finite set of discrete realizations of the uncertain parameters $\boldsymbol{e}_i$ (Grancharova and Johansen, 2009); however, it cannot guarantee that the finite set includes the worst case in $\mathcal{E}$. Tube-based MPC ensures closed-loop predictions lie in a tube satisfying the constraints by employing a feedback control law (Mayne *et al.*, 2011; Mayne, 2014). For linear systems, the feedback control law is parametrized by an affine law that is used to determine a polytopic set that includes the nominal trajectory for all additive uncertainties. This polytopic set acts as a tube around the nominal trajectory that defines tighter constraints ensuring that all trajectories satisfy the constraints of the MPC problem. Since tube-based MPC optimizes over a feedback control law, future control actions can adjust to past realizations of the uncertainty, making it less conservation than min-max formulations. For non-linear systems, the affine feedback law is not guaranteed to work, and determining the tightened constraints requires solving semi-infinite programs like the general robust NMPC. Methods for determining tubes for nonlinear systems have been developed, but they require satisfying assumptions about monotonicity of the dynamic system and Lipschitz continuity (Yang and Scott, 2020). Stochastic approximations of the tightened constraints have been considered; however, these do not guarantee that all possible trajectories will remain inside the tube (Mayne *et al.*, 2011). Due to these challenges, min-max and tube-based approaches are beyond the scope of this work since the underlying assumptions may be difficult to satisfy for large-scale non-linear systems.

Another relatively recent approach is Multi-stage NMPC (MS-NMPC). MS-NMPC uses discrete scenarios that evolve over a scenario tree where future control actions in the prediction horizon can adjust to past realizations of the uncertainty (Lucia and Engell, 2015; Tătulea-Codrean, Fischer and Engell, 2020). A finite number of discrete realizations of the uncertain parameters, $\boldsymbol{e}^j$, are sampled from the uncertainty set $\mathcal{E}$ (i.e., $\boldsymbol{e}^j \in \{\boldsymbol{e}^1, \boldsymbol{e}^2, \dots, \boldsymbol{e}^{N_r}\} \subset \mathcal{E}$). The uncertainty evolves over time by branching over each combination of $\boldsymbol{e}^j$. The scenario tree starts from the root node that is the initial state $\hat{\boldsymbol{x}}_k$, which branches to N_r scenarios; one for each $\boldsymbol{e}^j$. Then, at the next sampling step, each branch further branches for each $\boldsymbol{e}^j$, creating a total of $N_r \times N_r$ branches. The tree continues to branch until the robust horizon (RH), after which the uncertain parameter is held constant for the remaining of the prediction horizon. A depiction of the branching scenario tree can be seen in Figure 5 in chapter 3. The MS-NMPC tends to be the least conservative approach since the branching tree structure allows future control actions to adapt to past observations within the prediction horizon. However, the number of discrete scenarios grows exponentially with the length of the robust horizon (the total number of branches would be N_r^{RH}), which can make it computationally expensive. Shortening the RH tends to make MS-NMPC more conservative. Multi-scenario NMPC (MSc-NMPC) also considers discrete scenarios of the uncertain parameters but the parameters are assumed to remain constant over the entire prediction horizon (Huang and Biegler, 2009; Rasoulian and Ricardez-Sandoval, 2015; Patrón and Ricardez-Sandoval, 2020a; Piceno-Díaz *et al.*, 2020). The MSc-NMPC is equivalent to MS-NMPC with the scenario tree only branching at the first sampling interval (i.e., the robust horizon is one). The mathematical formulations of MS-NMPC and MSc-NMPC and their detailed description are presented in Chapter 3, section 3.2. Other approaches such as Offset-Free NMPC attempt to correct for unmeasured disturbances by either adding state disturbance or output disturbance term to the

model which can eliminate the steady-state set-point tracking error (Huang, Biegler and Patwardhan, 2010; Das and Mhaskar, 2014; Pannocchia, Gabiccini and Artoni, 2015; Tatjewski and Ławryńczuk, 2020; Jalanko *et al.*, 2021). Lyapunov-based NMPC formulates appropriate constraints in the optimization problem such that the controller inherits the stability and robustness properties of the Lyapunov-based controller (Christofides, Liu and Muñoz De La Peña, 2011; Zhang and Liu, 2013; Das and Mhaskar, 2014; Ellis, Liu and Christofides, 2017). However, there currently are no methods for constructing Lyapunov functions for general non-linear systems (Zhang and Liu, 2013). These last two approaches are beyond the scope of this work. Table 1 summarizes the robust NMPC approaches discussed above.

Table 1 Summary of robust NMPC approaches.

Method	Description	Advantages	Drawbacks	Sources
Min-max	Uses a closed-loop formulation that calculates the control actions that minimize the worst-case cost with respect to a bounded set of parameters, $\boldsymbol{e}_i$.	• Guarantees constraint satisfaction in all uncertain scenarios. • Minimizes the worst-case objective.	• Strong assumptions about the convexity of the problem must be made. • Assumptions for general non-linear systems are hard to verify. • Overly conservative.	(Lee and Yu, 1997; Scokaert and Mayne, 1998; Alamir and Balloul, 1999; Grancharova and Johansen, 2005)
Tube-based	Ensures closed-loop predictions lie in a tube satisfying the constraints by employing a feedback control law	• Less conservative by considering feedback in the formulation. • Guarantees constraint satisfaction in all uncertain scenarios.	• Non-linear systems result in semi-infinite programs. • Stochastic approximation does not guarantee all trajectories will remain inside the tube.	(Mayne *et al.*, 2011; Mayne, 2014)

MS-NMPC	Uses discrete scenarios that evolve over a scenario tree where future control actions in the prediction horizon can adjust to past realizations of the uncertainty. Tends to be less conservative.	• Less conservative by considering feedback in the formulation. • Tractable for NLP solvers, even in non-linear cases.	• Scenario tree grows exponentially with the robust horizon. • Becomes expensive due to model inflation.	(Lucia and Engell, 2015; Tătulea-Codrean, Fischer and Engell, 2020)
MSc-NMPC	Considers discrete scenarios of the uncertain parameters but the parameters are assumed to remain constant over the prediction horizon.	• Tractable for NLP solvers, even in non-linear cases. • Less expensive to solve than MS-NMPC.	• Tends to be more conservative than MS-NMPC.	(Huang and Biegler, 2009; Piceno-Díaz *et al.*, 2020).
Off-set free NMPC	Attempts to correct for unmeasured disturbances by either adding a state disturbance or output disturbance term to the model which can eliminate the steady-state error. Tends to be less conservative and easy to implement.	• Eliminates steady-state set-point tracking error. • Similar computational costs as nominal NMPC.	• Does not directly consider the uncertain parameters. • An observer may need to be used to estimate the disturbance terms. • The joint system of states and disturbance must be observable.	(Huang, Biegler and Patwardhan, 2010; Das and Mhaskar, 2014; Pannocchia, Gabiccini and Artoni, 2015; Tatjewski and Ławryńczuk, 2020; Jalanko *et al.*, 2021)
Lyapunov-based	Formulates appropriate constraints in the optimization problem such that the controller inherits the stability and robustness properties of the Lyapunov-based controller.	• Guarantees Lyapunov stability. • Slightly more expensive than the nominal NMPC.	• Does not directly consider the uncertain parameters. • Methods to formulate Lyapunov functions for general nonlinear	(Christofides, Liu and Muñoz De La Peña, 2011; Zhang and Liu, 2013; Das and Mhaskar, 2014; Ellis, Liu and Christofides, 2017)

			systems are not available.	

2.1.2. State Estimation

As mentioned above, the initial state, $\hat{\boldsymbol{x}}_k$, must be provided as the initial condition for the NMPC problem. However, the number of process measurements is limited and does not necessarily correspond to the states of the process. Hence, state estimators seek to compute the states from the measured variables (or process outputs). There are two major approaches for model-based state estimation of non-linear systems: iterative approaches and optimization-based approaches. Iterative approaches use the estimate from the previous sampling interval plus the measurements at the current sampling interval to recursively compute the current estimate (Valipour and Ricardez-Sandoval, 2021b). Optimization-based approaches embed the process model in an optimization problem that minimizes process and measurement errors (Zavala and Biegler, 2009). In this thesis, two state estimation strategies are explored: Extended Kalman Filter (EKF) (an iterative approach) and Moving Horizon Estimator (MHE) (an optimization-based approach) as these have been widely studied approaches for state estimation in non-linear systems.

Extended Kalman Filter (EKF) is widely accepted as the standard approach in the industry to tackle estimation for non-linear processes (Welch and Bishop, 2006). EKF is a two-step iterative approach that uses a stochastic form of the process model where the process and measurement noises are assumed to follow normal distributions. The main drawback of EKF is that it does not take into account process constraints (Valipour and Ricardez-Sandoval, 2021b). To overcome this limitation, the Moving Horizon Estimator (MHE) has been introduced as an optimization-based estimation scheme that can consider process constraints. This makes it a powerful estimation framework for constrained non-linear applications. MHE considers a sliding window involving a

finite number of past measurements and provides an optimal estimation of the unknown states by minimizing the process and measurement noises (Segovia *et al.*, 2019; Valipour and Ricardez-Sandoval, 2021a). Figure 2 shows a depiction contrasting the sliding window looking into the past used in MHE against the prediction horizon used in NMPC. MHE minimizes the error between the estimated measurements (square markers) and the past plant measurements (filled circle markers) by treating the past states as decision variables. It only considers a sliding window of N sampling steps looking into the past. The past control actions are known; thus, they appear as fixed parameters in the MHE formulation. The state at time k (the last sampling step in the window) is passed as the initial condition to NMPC. At every sampling step, the window shifts forward one step like NMPC, thus, the MHE problem must be resolved at every step. Chapter 3 shows the mathematical formulations of EKF and MHE.

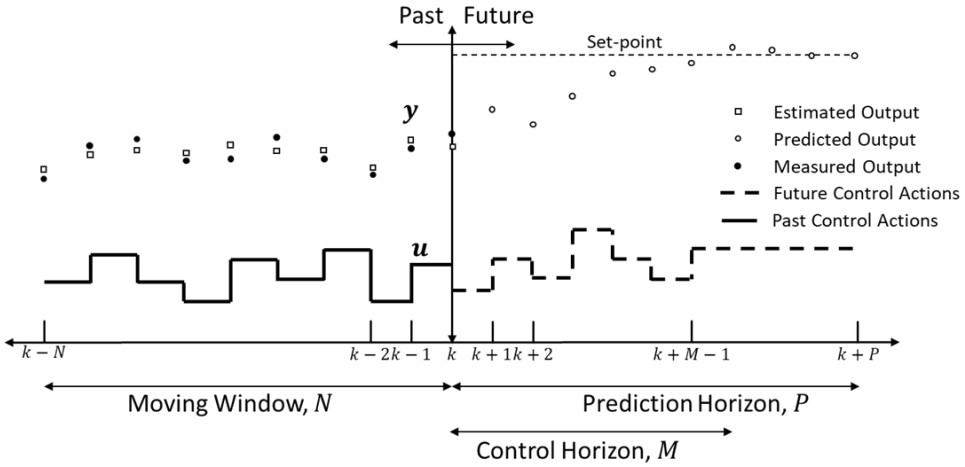


Figure 2 Depiction contrasting the sliding window in MHE against the prediction horizon in NMPC.

Other state estimation strategies for non-linear systems include the Unscented Kalman Filter which utilizes the unscented transform to calculate the statistics of a Gaussian random variable after a non-linear transformation (Vinoth Upendra and Prakash, 2013). Particle filters draw samples from

the process plant and integrate them with the process model to handle non-Gaussian distributions and non-linearities (Arulampalam *et al.*, 2002; Fuchigami, Niina and Takada, 2020). These methods overcome some of the challenges for non-Gaussian noises that are present in the EKF, but, as a result, they are more computationally taxing. The EKF has also been more extensively studied. In this work, it was assumed that noises follow Gaussian distributions, thus, these two approaches are beyond the scope of this work.

2.1.3. Robust NMPC and State Estimation on Large-Scale Systems

A few studies have investigated the application of NMPC to large-scale industrial systems. Huang et al. (2009) successfully implemented an advanced-step NMPC for an air separation unit of 320 states. The typical NMPC implementation took 120-220 seconds to solve, but the advanced-step NMPC reduces online computations to 1 second. Lopez-Negrete et al. (2013) applied NMPC to a detailed distillation column model and steam generation in a power plant. Their advanced-step formulation successfully reduces computational time by 2-3 orders of magnitude and the use of exact derivatives significantly reduced the computational costs. Toffolo et al. (2024) and Patrón et al. (2024) applied an NMPC and economic NMPC, respectively, to a chemical looping combustion process with 75 states for the oxidation stage and 200 states for the reduction stage. Both used a pseudo-homogeneous model to offset the NMPC computational cost of their rigorous multi-scale model. Despite these efforts, those studies have not considered the effect of uncertainty or state estimation. Fewer have considered the impact of the interaction between NMPC and state estimation. Biegler and Zavala (2009) applied an NMPC with Moving Horizon Estimator (MHE) as the state estimator to a polyethylene process with 350 state variables and differential algebraic equations. The advanced-step MHE implemented in that work successfully reduced the MHE computational costs. Patrón and Ricardez (2020b, 2022) applied NMPC with MHE and Kalman

Filter as state estimators to a CO_2 capture system with 116 states. Moreover, Patrón and Ricardez (2020a) also implemented an MSc-NMPC to a CO_2 absorber unit with 80 states but did not consider state estimation in that work. They showed the successful implementation of state estimation and NMPC as well as the robust NMPC on large-scale problems, simultaneously applying real-time optimization to determine the NMPC set-points. Other works have studied the implementation of scenario-based robust NMPC, but generally, it has been performed on small-case studies involving a relatively low number of states (Tătulea-Codrean, Fischer and Engell, 2020; Piceno-Díaz *et al.*, 2020; Kummer, Nagy and Varga, 2020; Lucia *et al.*, 2014, 2017; Puschke and Mitsos, 2018; Skupin *et al.*, 2022; Subramanian, Lucia and Engel, 2015; Thangavel *et al.*, 2018; Thangavel, Paulen and Engell, 2020; Thombre *et al.*, 2021; Zheng, Ricardez-Sandoval and Budman, 2020; Santander, Elkamel and Budman, 2019; Holtorf, Mitsos and Biegler, 2019). For instance, Holtorf et al. (Holtorf, Mitsos and Biegler, 2019) implemented an MS-NMPC with online-generated scenario trees for a semi-batch polymerization process with seven states. Kummer, Nagy and Varga (2020) applied an MS-NMPC to a semi-batch Williams-Otto process with four states and used Extended Kalman Filter (EKF) as the state estimator. Piceno-Diaz et al. (2020) implemented an MSc-NMPC for an anaerobic digester with seven states. Lucia et al. (2014) applied an economic MS-NMPC to a polymerization reaction with eight states. In practice, for large-scale problems, both state estimation and uncertainty will be present as obstacles to the implementation of NMPC. There is a gap in the literature highlighting the challenges that arise regarding controller performance and computational costs from the interactions between state estimation, uncertainty and robust NMPC; especially regarding the fact that uncertainty will be present in both the NMPC and state estimator models. This work will attempt to address this gap

by studying the challenging Tennessee-Eastman (TE) problem. A brief introduction to this plant is provided next.

The benchmark TE challenge is a plant-wide process control problem introduced by Downs and Vogel (1993). It is an open-loop unstable MIMO system, with several process units, which makes it a desirable problem to test large-scale state estimation and control applications. Jockenhövel, Biegler and Wächter (2003) described the implementation of an NMPC to the TE process whereas Tătulea-Codrean, Fischer and Engell (2020) used an economic MS-NMPC. However, those studies did not consider the effect of state estimation. Ricker and Lee (1995b, 1995a) applied NMPC and EKF to the TE process and analysed the performance. Kraus et al. (2006) and Kuhl et al. (2011) applied an MHE state estimator; the control schemes used to stabilize the TE process were not described. The process models describing the operation of the TE process used in the aforementioned studies and several other works, which include control and state estimation applications, often exhibit significant variations. Some works neglected the energy balances which are important to consider temperature and energy consumption effects (Ricker and Lee, 1995b, 1995a; Zheng, 1998); others neglected dynamic mole balances in parts of the process such as the stripper (Jockenhövel, Biegler and Wächter, 2003; Kraus *et al.*, 2006; Kühl *et al.*, 2011; Tătulea-Codrean, Fischer and Engell, 2020). The TE problem and the corresponding process model are described in more detail in Chapter 3.

2.1.4. Summary

Robust NMPC has become a popular way to deal with uncertainty in non-linear systems. Several formulations have been made available in the literature, such as min-max NMPC, tube-based NMPC, offset-free NMPC, and scenario-based approaches. However, implementations on large-scale systems are limited. In addition to uncertainty, practical implementations of NMPC require

an additional layer of state estimation. Many state estimation algorithms have been developed for non-linear systems, such as EKF and MHE. However, there is a gap in the literature highlighting the challenges and limitations of the combined effect of uncertainty and state estimation on a large industrial-scale system. Real-world chemical engineering applications such as the benchmark TE process generally involve multiple interacting process units with several states, control inputs and outputs, highlighting the importance of considering the challenges involved in implementing NMPC to such processes. Chapter 3 attempts to address this gap by presenting the implementation of two robust NMPC controllers, MS-NMPC and MSc-NMPC, together with the state estimation algorithms, EKF and MHE, on the TE process.

2.2. Physics-Informed Neural Networks

Physics-Informed NNs (PINNs) leverage mechanistic models, based on fundamental physics laws, to alleviate the need for historical/process data to train NNs (Raissi, Perdikaris and Karniadakis, 2019; Hao *et al.*, 2023). The seminal work by Raissi et al. (2019) on PINNs aimed to solve PDEs of the following form:

$$\frac{\partial \boldsymbol{x}}{\partial t}(t, \boldsymbol{z}) = \mathcal{F}_{\boldsymbol{z}}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p})(t, \boldsymbol{z}) \quad t \in \mathcal{T}, \boldsymbol{z} \in \Omega \subset \mathbb{R}^{N_z} \tag{19}$$

$$\mathcal{B}(\boldsymbol{x}, \boldsymbol{u})(t, \boldsymbol{z}) = 0 \quad t \in \mathcal{T}, \boldsymbol{z} \in \partial\Omega \tag{20}$$

$$\mathcal{I}(\boldsymbol{x})(t, \boldsymbol{z}) = 0 \quad t = t_0, \boldsymbol{z} \in \Omega \tag{21}$$

where $\boldsymbol{x}: \mathcal{T} \times \Omega \to \mathcal{X}$ ($\mathcal{X} \subset \mathbb{R}^{N_{sv}}$) are the state variables of the PDEs, the solution of which is a function of the time, $t \in \mathcal{T} = [t_0, t_f]$, and spatial, $\boldsymbol{z} \in \Omega$, domains (i.e., $\boldsymbol{x}(t, \boldsymbol{z})$). $\mathcal{F}_{\boldsymbol{z}}$ represents a non-linear differential operator with respect to the spatial domain. $\mathcal{B}$ represents the boundary condition operator and $\mathcal{I}$ the initial condition. The control actions, $\boldsymbol{u}: \mathcal{T} \to \mathcal{U}$ ($\mathcal{U} \subset \mathbb{R}^{N_u}$), are time-dependent inputs that the user can manipulate; these may appear either at the boundary condition,

$\mathcal{B}$ (e.g., inlet flow to a PFR) or as forcing terms within the PDE equations, $\mathcal{F}_{\boldsymbol{z}}$ (e.g., a heat generation term in the heat equation). The PDE modelling parameters are represented by $\boldsymbol{p}$; these may be constants (e.g., the reaction rate constant in an isothermal PFR) or dependent on the state variables, $\boldsymbol{p}(\boldsymbol{x})$ (e.g., the heat diffusivity is temperature-dependent when solving the heat equation). PINNs approximate the solution of the PDEs via an NN parametrized by θ (i.e., the weights and biases of the NN), as shown in equation (22). The NN takes as inputs the independent variables of the PDEs and returns the solution of the PDEs as the outputs. This is depicted in Figure 3 where it is shown that a NN takes the independent variables, t and $\boldsymbol{z}$, and returns the solution of the PDE as its prediction.

$$\boldsymbol{x}(t, \boldsymbol{z}) \approx NN_{\theta}(t, \boldsymbol{z}) \tag{22}$$

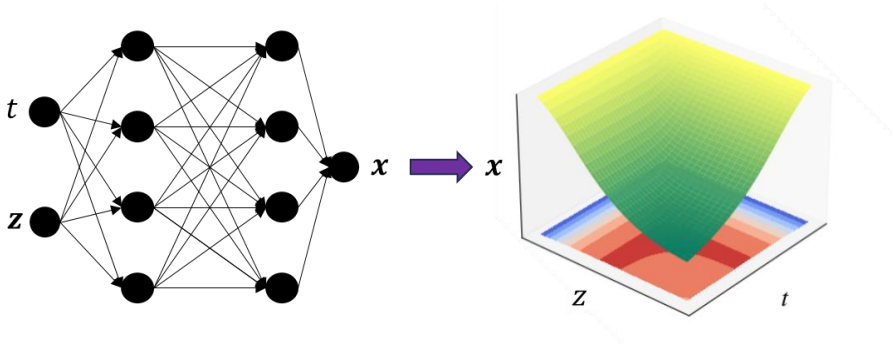


Figure 3 Depiction of PINN predicting the solution to PDE.

PINNs can only accurately predict the solution to the PDEs if the NN parameters, θ, have been sufficiently optimized. In typical supervised deep-learning tasks, one would like to train a model that takes the independent variables, $\mathcal{i}$, as inputs and predicts the dependent variables, $\mathcal{o}$, as outputs, (i.e., $\mathcal{o} = NN_{\theta}(\mathcal{i})$). To accomplish this task, one may have an input-to-output labelled data set, $\{\mathcal{i}^d, \mathcal{o}^d\}_{d=1}^{N_{data}}$. In the context of PINNs, the labelled data set corresponds to historical/process data or experimental data. Then, one can define what is known as a loss function,

as the prediction error from the NN as depicted in equation (23). The NN parameters can then be obtained by minimizing the loss function with respect to the network parameters θ, as seen in problem (24). Hence, supervised learning is similar to performing model regression. Several algorithms have been developed to perform this optimization such as stochastic gradient descent (SGD) or ADAM (Amari, 1993; Kingma and Ba, 2017). Note that even if the best possible parameters are achieved, i.e., problem (24) is solved to global optimality, the NN might not be able to make accurate predictions due to having a suboptimal structure (e.g., number of layers, or number of neurons in each layer) (Hammerstrom, 1993).

$$loss = \frac{1}{N_{data}} \sum_{d=1}^{N_{data}} \|NN_\theta(i^d) - o^d\|^2 \tag{23}$$

$$\theta = \underset{\theta}{\operatorname{argmin}}\, loss \tag{24}$$

Learning tasks often require large amounts of data for accurate predictions (Siemers, Feldmann and Bajorath, 2022). However, it may be difficult to obtain sufficient data, as it may require a significant number of experiments and historical/process data may not be available (Antonelo *et al.*, 2022; Hao *et al.*, 2023). PINNs overcome the requirement for labelled data by incorporating the physics equations as terms in the loss function. One may define the operator $\not{f}$ to describe the residuals of the PDEs in equation (19), i.e.,

$$\not{f}(\boldsymbol{x})(t, \boldsymbol{z}) = \frac{\partial \boldsymbol{x}}{\partial t}(t, \boldsymbol{z}) - \mathcal{F}_{\boldsymbol{z}}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p})(t, \boldsymbol{z}) \tag{25}$$

Therefore, the following loss can be defined which is used to train the PINN:

$$loss = loss_{\not{f}} + \beta loss_{\mathcal{B}} + \gamma loss_{\mathcal{I}} \tag{26}$$

where:

$$loss_{\mathbb{f}} = \frac{1}{N_{\mathbb{f}}} \sum_{d=1}^{N_{\mathbb{f}}} \left\| \mathbb{f}(NN_\theta)\left(t_{\mathbb{f}}^d, \boldsymbol{z}_{\mathbb{f}}^d\right) \right\|^2 \tag{27}$$

$$loss_{\mathcal{B}} = \frac{1}{N_{\mathcal{B}}} \sum_{d=1}^{N_{\mathcal{B}}} \left\| \mathcal{B}(NN_\theta)\left(t_{\mathcal{B}}^d, \boldsymbol{z}_{\mathcal{B}}^d\right) \right\|^2 \tag{28}$$

$$loss_{\mathcal{I}} = \frac{1}{N_{\mathcal{I}}} \sum_{d=1}^{N_{\mathcal{I}}} \left\| \mathcal{I}(NN_\theta)\left(t_0, \boldsymbol{z}_{\mathcal{I}}^d\right) \right\|^2 \tag{29}$$

where $loss_{\mathbb{f}}$ (equation (27)) describes the error due to the PDEs in equation (19); $loss_{\mathcal{B}}$ (equation (28)) and $loss_{\mathcal{I}}$ (equation (29)) describe the errors due to the boundary and initial conditions, respectively. Recall that the operator $\mathcal{B}$ defined the boundary conditions in equation (20) whereas the operator $\mathcal{I}$ specifies the initial conditions in equation (21). The data points $\left\{t_{\mathbb{f}}^d, \boldsymbol{z}_{\mathbb{f}}^d\right\}_{d=1}^{N_{\mathbb{f}}}$ are randomly sampled from the time, $\mathcal{T}$, and spatial domains, Ω. The data points $\left\{t_{\mathcal{B}}^d, \boldsymbol{z}_{\mathcal{B}}^d\right\}_{d=1}^{N_{\mathcal{B}}}$ are sampled from the time domain and the boundary of the spatial domain, $\partial\Omega$. The points $\left\{\boldsymbol{z}_{\mathcal{I}}^d\right\}_{d=1}^{N_{\mathcal{I}}}$ are sampled from the spatial domain. As can be seen, it is not necessary to have output data to evaluate the loss. It is only necessary to generate random points in the domains of the PDE equations. The differential terms in $\mathbb{f}$ can be evaluated via automatic differentiation (AD) by applying the chain rule on compositions of functions which most ML packages support (Baydin *et al.*, 2018; Raissi, Perdikaris and Karniadakis, 2019). A typical NN structure involves the sequential composition of affine/linear and nonlinear functions. AD keeps track of the symbolic rules of differentiation (i.e., chain rule, product rule, power rule, etc.) without generating symbolic expressions. Hence, AD is computationally efficient at evaluating exact derivatives. For example, the term $\frac{\partial \boldsymbol{x}}{\partial t}(t, \boldsymbol{z})$ in equation (25) can be evaluated by backpropagating the gradient of the NN

output with respect to the input (i.e., $\frac{\partial NN_\theta}{\partial t}(t, \boldsymbol{z})$). Backpropagation means evaluating the gradient backwards from the output, by applying the chain rule on a NN, to obtain the exact partial derivatives with respect to the inputs (Baydin *et al.*, 2018). The parameters β and γ assign relative importance to each of the loss terms which may have different magnitudes. These can be user-defined, but they may require tuning to achieve accurate predictions; trial-and-error may be tedious and does not guarantee satisfactory results, thus, more sophisticated methods have been developed such as Lagrange-dual approaches (Fioretto *et al.*, 2020; Eydenberg *et al.*, 2022) and learning rate annealing (Wang, Teng and Perdikaris, 2020). From this point, the loss can be minimized using traditional ML training algorithms like SGD and ADAM. If labelled data is available from past historical data or experimental measurements, e.g., $\left\{t_{data}^d, \boldsymbol{z}_{data}^d, \boldsymbol{x}_{data}^d\right\}_{d=1}^{N_{data}}$, it can also be included as an additional term in the loss function, i.e.,

$$loss_{data} = \frac{1}{N_{data}} \sum_{d=1}^{N_{data}} \left\| NN_\theta\left(t_{data}^d, \boldsymbol{z}_{data}^d\right) - \boldsymbol{x}_{data}^d \right\|^2 \tag{30}$$

Since their introduction to the community by Raissi et al. (2017a, 2017b, 2019), many extensions and alternative approaches have been developed. To name a few, Eydenberg et al. (2022) perform loss reweighting by applying a Lagrange dual formulation, Nabian et al. (2021) proposed an importance sampling approach to better select training points, Sirignano and Spiliopoulos (2018) use numerical differentiation to deal with higher order PDEs, whereas Gao et al. (2021) used CNNs as alternative ML architectures. PINNs can be used to solve a broad number of problems given its flexibility and universal approximation properties of NNs. As a result, PINNs have found widespread engineering applications, e.g., Zhang (2022) applied CNNs to Darcy flow in reservoirs, Mao et al. (2020) used PINNs to model high-speed flows, Cai et al. (2021) used PINNs to solve

heat transfer problems, and Chaffart et al. (2024) employed PINNs to capture multiscale stochastic thin-film deposition.

One should point out some of the difficulties that arise from this methodology. For instance, training strongly depends on many user-defined hyperparameters (number of hidden layers, number of neurons in each layer, selection of training algorithm, initialization of weight and biases, etc.). PINNs also add challenges of their own, e.g., the selection of the loss weighting parameters, β and γ, and the selection of the number of training points required for every loss term are non-trivial user decisions that directly impact convergence. Also, additional loss weighting terms may be required for PDEs of different scales. For instance, suppose a non-isothermal PFR has concentration and temperature as state variables. If the PDE describing the energy balance has terms of different magnitude than the PDE describing the mole balance, the training algorithm may focus on the larger term. Problem-specific challenges may also be present, like stiffness or high-order derivatives. These are still open challenges for PINNs. Hao et al. (2023) make a comprehensive review of several extensions of PINNs and current methodologies available.

Moreover, the trained NN only approximates a unique solution to the PDEs. The time profile of the control inputs, $\boldsymbol{u}(t)$, had to be known *a priori* to evaluate the loss (26). If the initial conditions, boundary conditions, parameters within the PDE equations, or the control profile change, a new solution must be obtained by retraining the PINN. This means that this formulation is not suitable for generating surrogate models for PDE-constrained feedback NMPC which requires updating the initial conditions and changing the control actions over time. The following section describes how PINNs have been extended to deal with feedback control problems.

2.2.1. PINNs in Feedback Process Control

As mentioned above, PINNs in their basic formulation as proposed by Raissi et al. (2019) cannot be directly applied to solve control problems. PINNs have primarily been used to solve single instances of PDEs, i.e., find the solution for a given set of boundary and initial conditions. In a problem where there is a control input in a dynamical system, the time profile of this input must be known ahead of time. To find a solution for a new initial condition and control profile, the PINN must be retrained. However, a few extensions have been proposed to apply NN surrogates in control problems. For instance, Barry-Straume et al. (2022) solve the optimal control problem by minimizing the Lagrangian of the system as the loss. They trained three NNs: one for the state variables, one for the manipulated variables and one for the Lagrange multipliers (co-states). On the other hand, Mowlavi and Nabi (2023) solved the optimal control problem by adding the objective of the PDE-constrained optimal control problem to the PINN training loss. They trained two NNs: one for the state variables and one for the manipulated variables. However, in both works, the trained models only learn the solution to the open-loop optimal control problem. Thus, to apply their methods in a closed-loop feedback scheme, the model must be retrained at each time interval using the new initial conditions, i.e., the current states of the plant, thus making this approach computationally taxing or even intractable for large-scale systems. In addition, neither of these works considered process constraints in their optimal control problem.

As an alternative, other works have trained models that learn the system dynamics by constructing models that map states and control actions at a current sampling time step to states at the next sampling step. The trained model is then embedded as a surrogate in an NMPC formulation. Chen et al. (2019) used an input convex recurrent NN (RNN) to learn the system dynamics and embedded the model into NMPC by evaluating the gradients via backpropagation. Zheng et al.

(2023) also employed a physics-informed RNN structure. The trained RNN was used as a surrogate model to control non-linear systems subjected to Gaussian measurement noise. The derivative information (Jacobian and Hessian) was provided to IPOPT (an NLP solver) via finite difference approximations; however, they did not report the computational time to solve the NMPC problem. Alhajeri et al. (2022) implemented Long-Short Term Memory (LSTM) networks as surrogate models. They applied a Monte Carlo dropout technique for training the LSTM; the strategy used to embed the surrogate model within NMPC, and the computational time to solve the NMPC problem were not reported. Antonelo et al. (2022) trained a PINN that accepts time, t, as a continuous input, the current states, $\boldsymbol{x}_k$, and the current control actions, $\boldsymbol{u}_k$. The PINN predicts the states at any time within the sampling interval $t \in [0, \Delta T]$ using a fully-connected feedforward deep NN (FNN). The output of the PINN can be provided as a new input to the NN model to make predictions over longer time horizons. To the author's knowledge, a shooting approach was used to numerically solve the PINN embedded within the NMPC framework. Other works have implemented the same numerical approach in different case studies (Nicodemus *et al.*, 2022; Sanyal and Roy, 2023). Their implementations were restricted to small ODE-constrained problems (e.g., a multilink manipulator of four states, and a quadrotor drone of 13 states). Gokhale et al. (2022) proposed a similar formulation, except that they did not include time as an input to the network and instead used a loss function that is discretized over the time domain. However, their PINN was not embedded within an NMPC problem. Zhang (2022) formulated a physics-informed CNN for modelling time-dependent PDEs. The model maps the initial condition, as N_z-dimensional input (N_z is the number of spatial dimensions), to the next time step as the output. That formulation does not consider control inputs; however, they could be added as additional inputs to the network. Table 2 summarises the above literature involving PINNs and NMPC.

Table 2 Summary of literature involving PINNs and NMPC.

Source	Description	Advantages	Drawbacks
Barry-Straume et al. (2022)	• Solved optimal control problems by minimizing the Lagrangian of the system. • Lagrangian is used as the loss of the PINN. • Trained three NNs: one for the states, one for the manipulated variables and one for the Lagrange multipliers.	• Find the solution to the PDE-constrained optimal control problem.	• Only learns the solution to the open-loop optimal control problem. • Updating the initial condition would require retraining. • Does not consider process constraints.
Mowlavi and Nabi (2023)	• Solved the PDE-constraint optimal control problem. • Add the control problem objective to the PINN loss. • Trained two NNs: one for the states and one for the manipulated variables.	• Find the solution to the PDE-constrained optimal control problem. • Formulation only involved two NNs.	
Chen et al. (2019)	• Used input-convex RNN to learn the system dynamics as a state-space model. • Used backpropagation to evaluate derivative information for NMPC optimization. • Data-driven models using labelled data were considered (not PINN models).	• Solution found by the NMPC is global due to convexity.	• Optimization algorithm used to solve the NMPC problem is not clear. • Input-convex structure may restrict the class of problem on which the model can be trained.
Zheng et al. (2023)	• Trained physics-informed RNNs to learn the system dynamics. • Model is treated as a discrete-time state-space model.	• Integrated measurement data and physics-informed losses.	• Used finite differences to approximate derivative information for the optimizer.
Alhajeri et al. (2022)	• Trained physics-informed LSTMs under noisy data to learn the system dynamics. • Model is treated as a discrete-time state-space model.	• Reduced over-fitting in the presence of noisy data by employing a drop-out technique.	• The model embedding method and computational times were not reported.

Antonelo et al. (2022)	• Trained a PINN that takes time as continuous input, as well as the current states and control actions. • Outputs the state over a sampling interval. • PINN is treated as a discrete-time state-space model to predict over longer horizons and to be used in NMPC.	• Evaluates exact derivatives with respect to time to evaluate loss. • Showed slight improvement in computational cost.	• Solved the NMPC problem using a shooting approach. • Case studies were restricted to small ODE problems.
Gokhale et al. (2022)	• Used a PINN that accepts the current states and control actions as inputs. • Outputs the states at the next sampling step. • Derivative terms in the loss function are discretized over time.	• Formulated a PINN amenable to control problems.	• Did not embed their model within NMPC. • Restricted to small ODE problems.
Zhang (2022)	• Formulated a physics-informed CNN to model time-dependent PDEs. • Leverages the structure of CNNs to deal with the multidimensional nature of PDEs.	• Physics-informed CNN can handle changing initial conditions.	• Did not consider changing control inputs in their formulation.

When embedding the ML model within NMPC, the study performed by Antonelo et al. (2022) has shown slight computational advantages over the implementation of the analytic/mechanistic model within the shooting optimization approach using Runge-Kutta (RK) methods as the ODE integrator. Nevertheless, optimization approaches where the modelling equations are solved simultaneously have become popular in NMPC (Wächter and Biegler, 2006; Biegler and Zavala, 2009). One particular approach by which NN surrogates could be made amenable to simultaneous solution approaches is through the embedding of the NN structure as explicit algebraic constraints within the algebraic modelling language (AML) (Ceccon *et al.*, 2022). Several formulations have been proposed depending on the type of activation functions. For smooth activations, like tanh or

sigmoid, full-space (FS) and reduced-space (RS) formulations have been proposed (Schweidtmann and Mitsos, 2019). These formulations are amenable to local NLP solvers, like IPOPT. Therefore, they can take advantage of the AD from the AML to evaluate the gradients of the optimization problem. For non-smooth activations, like rectified linear units (ReLU), formulations involving mixed-integer variables like BigM or ReLU-partition have been proposed (Anderson *et al.*, 2019; Tsay *et al.*, 2021). These two formulations involve adding variables like $m \in \{0,1\}$ in the optimization problem which can only take the discrete-integer values of 0 and 1. Problems involving integer variables can be challenging to solve within short time spans because, in the worst-case, branch-and-bound algorithms might have to check all possible combinations of the integer values (Hillier and Lieberman, 2015). Since the ReLU constraint repeats at every neuron at every hidden layer in the NN, the number of integer variables required to represent a full NN may also be very large. Thus, solving mixed-integer programs is not desirable for online feedback control applications. The ReLU-complementarity formulation converts the ReLU activations to complementarity constraints (Yang, Balaprakash and Leyffer, 2022), which are amenable to NLP solvers (i.e., they can be expressed as nonlinear equalities/inequalities). However, NLP solvers usually make assumptions about the optimization problem, like satisfaction of constraint qualification. NLP solvers struggle with complementarity constraints due to degeneracy because they fail to meet the constraint qualification assumption (Thierry and Biegler, 2020). The FS and RS formulations are described in more detail in Chapter 4. Moreover, ML environments can efficiently evaluate derivatives. Hence, evaluating the NN surrogate as an external function and providing the NLP solver with the evaluation of the gradients is another potential approach is to embed it within the AML. In this approach, the NLP solver passes the current iteration of the decision variables to the ML environment. The AD in the ML environment can apply the chain

rule over the layers of the NN to evaluate the Jacobian and Hessian matrices of the surrogate model. The NLP can then use these gradients to update the decision variables in an optimal direction until a solution is found. Note that this approach requires smooth activation functions because NLP solvers, like IPOPT, generally assume that the constraints and objective are differentiable to second order derivatives. This approach is also described in more detail in Chapter 4. Hence, there is a gap in literature comparing the different approaches by which NN surrogates can be embedded to NMPC models, highlighting the challenges of each approach and the impact on computational time.

2.2.2. Summary

NMPC problems are computationally expensive to solve, particularly for large-scale applications. Hence, alternative modelling approaches, such as ML surrogates, have recently gained attention. Various formulations of control-oriented PINNs have been proposed in the literature. Some attempt to solve the optimal control problem by directly minimizing the Lagrangian of the system in the loss function. The main drawback is that the solution is unique to the initial condition used; hence, it cannot be used in feedback control because updating the initial condition would require retraining. Other studies have used various NN structures, such as RNNs, LSTMs and CNNs, that map current states and control actions to states at the next sampling interval. The resulting model can then act as a surrogate state-space model in NMPC. However, there is a gap in the literature comparing how these surrogates can be embedded within the NMPC framework, highlighting the challenges and computational costs. Given the rise in popularity of simultaneous optimization approaches in NMPC, embedding strategies that take advantage of these optimization methods could be potential avenues. Chapter 4 attempts to address this gap by benchmarking multiple embedding strategies. The FS and RS formulations are used to directly formulate the ML model

as mathematical expressions in the optimization program. AD from ML environments was used to evaluate the model externally from the optimization program, thereby, solving the NMPC problem in an external function approach.

3. Multi-Scenario and Multi-Stage Robust NMPC with State Estimation: Application on the Tennessee-Eastman Process

As discussed in Chapter 2, uncertainty and state estimation are inevitable challenges to the implementation of NMPC in large-scale systems. Plant/model mismatch and inaccurate state estimation cause controller performance to deteriorate. There is a gap in the literature studying the combined effect of uncertainty, state estimation and robust NMPC, highlighting the challenges, effects on performance and computational costs. To address this gap, in this study, two scenario-based robust NMPC formulations, MS-NMPC and MSc-NMPC, are applied to consider parametric uncertainty in the TE process model. The performance of these two controllers is assessed when coupled with the additional layer of state estimation, with an unconstrained estimator such as EKF and a constrained estimator such as MHE. In addition, the model used in this study was based on the original Fortran model by Downs and Vogel (1993) and the parameters were taken from it, which considers the complete mole and energy balances in each unit of the process. It is expected that the implementation of the full model as considered in this study provides a more realistic representation of such an industrial-scale process.

The study is structured as follows: Section 3.1 briefly introduces the TE process model. Section 3.2 introduces MS-NMPC and MSc-NMPC. Section 3.3 introduces the idea of state estimation and the EKF and MHE formulations. Section 3.4 presents the results of the computational experiments under different simulation scenarios and controller/state estimator combinations. Finally, section 3.5 summarizes the major findings of this work.

3.1. Tennessee-Eastman Model

The Tennessee-Eastman (TE) process is a benchmark plant model initially proposed by Downs and Vogel (1993) for control strategies and process monitoring. It is based on a real industrial

process, but the names of components have been modified with generic names. It is modelled by five process units: a two-phase reactor, a phase separator, a stripper, a reactor-feed mixing zone, and a compressor. Figure 4 presents a flowsheet of the TE process. There are eight chemical components involved, A to H. Components A to C are non-condensable gases and components D to H may exist in both liquid and gaseous phases. Component B is an inert component. The reactions occur in the gas phase of the reactor (Downs and Vogel, 1993). Products G and H are desirable and component F is a by-product. This process is open-loop unstable since the reactions shown in (31)-(34) are exothermic, and irreversible, and their rates are highly sensitive to temperature changes. Thus, this system is considered a highly non-linear plant (Jockenhövel, Biegler and Wächter, 2003). Moreover, an efficient control strategy is required to stabilize the system and maintain it within the corresponding operating limits.

$$\mathrm{A}(g) + \mathrm{C}(g) + \mathrm{D}(g) \rightarrow \mathrm{G}(liq) \quad (31)$$
$$\mathrm{A}(g) + \mathrm{C}(g) + \mathrm{E}(g) \rightarrow \mathrm{H}(liq) \quad (32)$$
$$\mathrm{A}(g) + \mathrm{E}(g) \rightarrow \mathrm{F}(liq) \quad (33)$$
$$3\mathrm{D}(g) \rightarrow 2\mathrm{F}(liq) \quad (34)$$

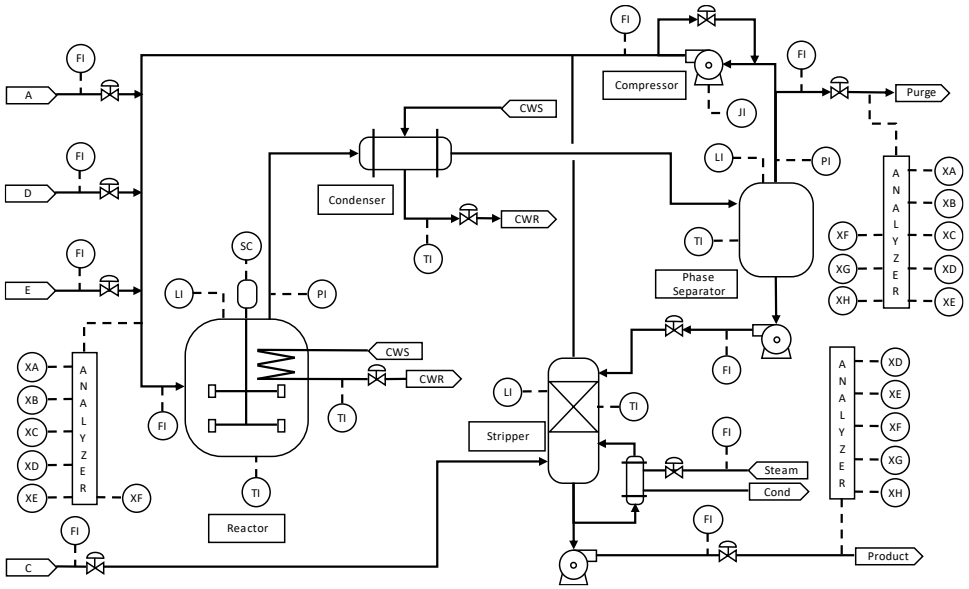


Figure 4 Flowsheet of the Tennessee-Eastman process.

The original TE model has been rewritten in Pyomo to make use of this optimization platform. The model was validated against the original Fortran model by providing the same inputs to both models and observing that the output responses were the same. The model consists of 38 ordinary differential equations (ODEs); hence, 38 differential process states are considered in the state estimation. The states correspond to variables such as the molar hold-ups of each species in each of the major units of the process, the energy balance of each of the units and the energy balance of the cooling streams. It also includes 91 nonlinear algebraic expressions for evaluating pressures, liquid levels, heat transfer rates, among other physical quantities. Additionally, there are 12 manipulated variables and 35 measured variables. From the measured variables, only two are directly measured states, i.e., the remaining states must be estimated. Downs and Vogel (1993) state that some of the composition measurements have a sampling rate of 0.1 h and others of 0.25 h. In this work, it was assumed a constant sampling period of 0.1 h for all measurements, and the measurements are available without delay. Note that this has been a common assumption in practically all previous studies involving the TE process, e.g., (Ricker and Lee, 1995b, 1995a; Zheng, 1998; Jockenhövel, Biegler and Wächter, 2003; Kraus *et al.*, 2006; Kühl *et al.*, 2011; Kumar, Patwardhan and Noronha, 2020; Tătulea-Codrean, Fischer and Engell, 2020). There is some variability in the exact choice of the sampling period, with some opting for 0.028 h (100 s) and others using 0.25 h.

3.2. Multi-stage and Multi-scenario Robust NMPC

Multi-stage Robust NMPC (MS-NMPC) represents the uncertainty by a tree of discrete scenarios that evolve by branching at each predicted sampling interval (Lucia and Engell, 2015; Tătulea-Codrean, Fischer and Engell, 2020). In this formulation, future control actions can take into consideration future state feedback (i.e. the controller considers that in future stages, the state

vector will be updated as the uncertainty becomes realized, hence, the control actions can adjust to the uncertainty from earlier stages in the prediction horizon). Thus, it is a less conservative approach compared to multi-scenario NMPC (MSc-NMPC). The evolution of states in the prediction horizon of MS-NMPC is depicted in Figure 5. It shows that the initial state, $\boldsymbol{x}_k^*$, branches to three discrete realizations of the uncertain parameters, $\boldsymbol{e}_k$. This results in three possible states in stage two which subsequently branch again at each stage for each realization of the uncertain parameters, resulting in the tree structure shown in the figure. If the tree continues to branch to the end of the prediction horizon, P, the problem size grows exponentially with the length of the prediction horizon. Thus, branching in the scenario tree is often considered to a time length shorter than the prediction horizon, this is referred to as the robust horizon (RH). After the end of the RH, the values of the uncertain parameters are held constant in each branch, as shown in Figure 5. The MS-NMPC formulation is as follows:

$$\min_{\boldsymbol{u}^{*j}_i \in \mathbb{R}^{N_u}\ \forall i \in \{k,k+1,\dots,k+P-1\} \forall j \in J} \sum_{n=1}^{N_s} \omega_n \left(\sum_{i=k+1}^{k+P} \left\| \boldsymbol{y_{sp}} - \left(\boldsymbol{y}^{*j}_i\right)_n \right\|_{\boldsymbol{L}}^2 + \sum_{i=k}^{k+M-1} \|(\Delta \boldsymbol{u}^*_i)_n\|_{\boldsymbol{W}}^2 \right) \tag{35}$$

$$\text{s.t. } \boldsymbol{x}^{*j}_{i+1} = f\left(\boldsymbol{x}^{*p(j)}_i, \boldsymbol{u}^{*j}_i, \boldsymbol{e}_i^{r(j)}\right) \quad \forall i \in \{k, k+1, \dots, k+P-1\}\ \forall j \in J \tag{36}$$

$$\boldsymbol{y}^{*j}_i = h\left(\boldsymbol{x}^{*j}_i\right) \quad \forall i \in \{k, k+1, \dots, k+P\}\ \forall j \in J \tag{37}$$

$$g\left(\boldsymbol{x}^{*j}_i, \boldsymbol{u}^{*j}_i, \boldsymbol{y}^{*j}_i\right) \leq 0 \quad \forall i \in \{k, k+1, \dots, k+P\}\ \forall j \in J \tag{38}$$

$$\boldsymbol{x}^L \leq \boldsymbol{x}^{*j}_i \leq \boldsymbol{x}^U \quad \forall i \in \{k, k+1, \dots, k+P\}\ \forall j \in J \tag{39}$$

$$\boldsymbol{u}^L \leq \boldsymbol{u}^{*j}_i \leq \boldsymbol{u}^U \quad \forall i \in \{k, k+1, \dots, k+P-1\}\ \forall j \in J \tag{40}$$

$$\boldsymbol{u}^{*j}_i = \boldsymbol{u}^{*j}_{k+M-1} \quad \forall i \in \{k+M, k+M+1, \dots, k+P-1\}\ \forall j \in J \tag{41}$$

$$\Delta \boldsymbol{u}^{*j}_i = \boldsymbol{u}^{*j}_i - \boldsymbol{u}^{*j}_{i-1} \quad \forall i \in \{k, k+1, \dots, k+P-1\}\ \forall j \in J \tag{42}$$

$$\boldsymbol{u}^{*j}_k = \boldsymbol{u}^{*l}_k \text{ if } \boldsymbol{x}^{*p(j)}_k = \boldsymbol{x}^{*p(l)}_k \quad \forall i \in \{k+M, k+M+1, \dots, k+P-1\}\ \forall j, l \in J \tag{43}$$

$$\boldsymbol{x}^*_k = \widehat{\boldsymbol{x}}_k \tag{44}$$

where:

$\boldsymbol{x}^{*j}_i \in \mathbb{R}^{N_x}, \boldsymbol{y}^{*j}_i \in \mathbb{R}^{N_y}, \boldsymbol{e}^j_i \in \mathbb{R}^{N_e}\ \forall i \in \{k, k+1, \dots, k+P\}\ \forall j \in J$

$\boldsymbol{L} \in \mathbb{R}^{N_y \times N_y}, \boldsymbol{W} \in \mathbb{R}^{N_u \times N_u}$

$\boldsymbol{x}^L \in \mathbb{R}^{N_x}, \boldsymbol{x}^U \in \mathbb{R}^{N_x}, \boldsymbol{u}^L \in \mathbb{R}^{N_u}, \boldsymbol{u}^U \in \mathbb{R}^{N_u}, \widehat{\boldsymbol{x}}_k \in \mathbb{R}^{N_x}$

$f\colon \mathbb{R}^{N_x \times N_u \times N_e} \to \mathbb{R}^{N_x}, h\colon \mathbb{R}^{N_x} \to \mathbb{R}^{N_y}, g\colon \mathbb{R}^{N_x \times N_u \times N_y} \to \mathbb{R}^{N_g}$

The vector of predicted states at time interval i is denoted by $\boldsymbol{x}_i^*$ whereas $\boldsymbol{y}_i^*$ denotes the process outputs. The term $\|\boldsymbol{v}\|_{\boldsymbol{B}}^2$ denotes the L_2-norm of vector $\boldsymbol{v}$ with weight matrix $\boldsymbol{B}$ squared ($\|\boldsymbol{v}\|_{\boldsymbol{B}}^2 \coloneqq \boldsymbol{v}^T\boldsymbol{B}\boldsymbol{v}$). $\boldsymbol{L}$ and $\boldsymbol{W}$ are the output and input weight matrices, respectively, in the objective function. The scalar M denotes the length of the control horizon. The state vector, $\hat{\boldsymbol{x}}_k$, is the initial condition to the NMPC problem that is provided through state feedback. In practice, the initial condition must be the estimate provided by a state estimator (Valipour and Ricardez-Sandoval, 2021a). In the process model, equation (36), the term $\boldsymbol{x}^{*p(j)}_k$ represents the dependence of the states at the next sampling interval on the parent state by the branching in the scenario tree. The superscript in this variable $p(j)$ is a function that makes reference of the uncertainty realization j to the parent node $p(j)$ or state $\boldsymbol{x}^{*p(j)}_k$. The function $r(j)$ makes reference to the considered uncertainty realization of $\boldsymbol{e}_i^{r(j)}$ in the branch of the scenario tree (e.g. $\boldsymbol{x}^{*6}_{k+2} = f\left(\boldsymbol{x}^{*p(6)}_{k+1}, \boldsymbol{u}^{*6}_{k+1}, \boldsymbol{e}^{r(6)}_{k+1}\right) = f\left(\boldsymbol{x}^{*2}_{k+1}, \boldsymbol{u}^{*6}_{k+1}, \boldsymbol{e}^{3}_{k+1}\right)$ as shown in Figure 5). The realization, $r(j)$, of the uncertainty at the stage i is one of N_r total possible combinations of the uncertain parameters (i.e., $\boldsymbol{e}_i^{r(j)} \in \left\{\boldsymbol{e}_i^1, \boldsymbol{e}_i^2, \dots, \boldsymbol{e}_i^{N_r}\right\}$) Here, J is defined as the set of the index j in the scenario tree. The non-anticipativity constraints, (43), imply that the control actions cannot anticipate the future realization of the uncertainty; hence, control actions stemming from the same node must be equal (e.g., in the tree shown in Figure 5, $\boldsymbol{u}^{*1}_k = \boldsymbol{u}^{*2}_k = \boldsymbol{u}^{*3}_k$, $\boldsymbol{u}^{*4}_{k+2} = \boldsymbol{u}^{*5}_{k+2} = \boldsymbol{u}^{*6}_{k+2}$, …). Let S_n denote the n^{th} scenario which constitutes the path from the root node, $\boldsymbol{x}_k^1$, to the end of one of the leaves and it contains all the inputs $\left(\boldsymbol{u}^{*j}_i\right)_n$, states $\left(\boldsymbol{x}^{*j}_i\right)_n$ and outputs $\left(\boldsymbol{y}^{*j}_i\right)_n$ in between (e.g., the sequence of states $\boldsymbol{x}^{*1}_k \rightarrow \boldsymbol{x}^{*1}_{k+1} \rightarrow \boldsymbol{x}^{*2}_{k+2} \rightarrow \boldsymbol{x}^{*2}_{k+3} \rightarrow \boldsymbol{x}^{*2}_{k+4}$ constitute scenario $n = 2$ in Figure 5). To summarize the indexes, the subscript i refers to the sampling interval within the prediction horizon; the subscript k is the current sampling interval in the process whereas the superscript j refers to the branch of

the scenario tree. Moreover, the subscript n refers to a full scenario in the tree starting from the root node to the end of one of the leaves. The objective function in (35) is given by the weighted sum over each of the scenarios, where each scenario is assigned a non-negative weight ω_n, such that $\sum_{n=1}^{N_s} \omega_n = 1$. More detailed descriptions of MS-NMPC can be found elsewhere (Lucia and Engell, 2015; Lucia *et al.*, 2017).

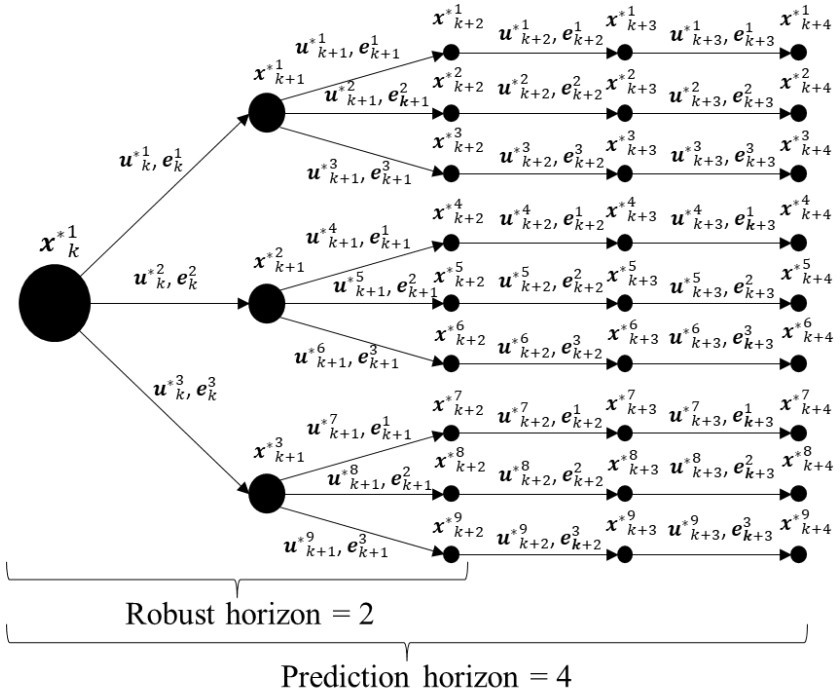


Figure 5 MS-NMPC scenario tree representation of the evolution of the uncertainty.

The MS-NMPC can be reduced to MSc-NMPC when RH is set to one. Accordingly, MSc-NMPC considers the uncertainty in the model parameters by finding control actions that are feasible under multiple discrete realizations (Piceno-Díaz *et al.*, 2020). The realization of the uncertain parameter remains constant throughout the prediction horizon in the given scenario. Since MSc-NMPC cannot adapt to past realizations of the uncertain parameter, it is a more conservative approach compared to MS-NMPC. However, the advantage over MS-NMPC, with a robust horizon greater

than or equal to two, is that it requires less computational effort for an equal number of possible realizations, thus making this approach computationally attractive for large-scale applications. Nonetheless, MSc-NMPC is still more computationally expensive than the standard NMPC formulation, since doubling the number of scenarios, doubles the number of state variables, algebraic variables, and constraints and equations involving the states. Notice that if one reduces the number of scenarios, N_s, to one, the formulation reduces to the standard NMPC formulation thus using only the nominal values for the uncertain parameters. As shown in Figure 6, the scenario tree for MSc-NMPC shows that the tree only branches on the first node. The MSc-NMPC formulation is as follows:

$$\min_{\boldsymbol{u}_i^* \in \mathbb{R}^{N_u}\ \forall i \in \{k,k+1,\dots,k+P-1\}} \sum_{j=1}^{N_s} \omega_j \left(\sum_{i=k+1}^{k+P} \left\| \boldsymbol{y_{sp}} - \boldsymbol{y}^{*j}_i \right\|^2_{\boldsymbol{L}} + \sum_{i=k}^{k+M-1} \|\Delta \boldsymbol{u}_i^*\|^2_{\boldsymbol{W}} \right) \tag{45}$$

$$\text{s.t. } \boldsymbol{x}^{*j}_{i+1} = f\left(\boldsymbol{x}^{*j}_i, \boldsymbol{u}_i^*, \boldsymbol{e}^j\right) \quad \forall i \in \{k, k+1, \dots, k+P-1\}\ \forall j \in \{1,2,\dots,N_s\} \tag{46}$$

$$\boldsymbol{y}^{*j}_i = h\left(\boldsymbol{x}^{*j}_i\right) \quad \forall i \in \{k, k+1, \dots, k+P\}\ \forall j \in \{1,2,\dots,N_s\} \tag{47}$$

$$g\left(\boldsymbol{x}^{*j}_i, \boldsymbol{u}_i^*, \boldsymbol{y}^{*j}_i\right) \le 0 \quad \forall i \in \{k, k+1, \dots, k+P\}\ \forall j \in \{1,2,\dots,N_s\} \tag{48}$$

$$\boldsymbol{x}^L \le \boldsymbol{x}^{*j}_i \le \boldsymbol{x}^U \quad \forall i \in \{k, k+1, \dots, k+P\}\ \forall j \in \{1,2,\dots,N_s\} \tag{49}$$

$$\boldsymbol{u}^L \le \boldsymbol{u}_i^* \le \boldsymbol{u}^U \quad \forall i \in \{k, k+1, \dots, k+P-1\} \tag{50}$$

$$\boldsymbol{u}_i^* = \boldsymbol{u}_{k+M-1}^* \quad \forall i \in \{k+M, k+M+1, \dots, k+P-1\} \tag{51}$$

$$\Delta \boldsymbol{u}_i^* = \boldsymbol{u}_i^* - \boldsymbol{u}_{i-1}^* \quad \forall i \in \{k, k+1, \dots, k+P-1\} \tag{52}$$

$$\boldsymbol{x}^{*j}_k = \widehat{\boldsymbol{x}}_k \quad \forall j \in \{1,2,\dots,N_s\} \tag{53}$$

where:
$\boldsymbol{x}^{*j}_i \in \mathbb{R}^{N_x}, \boldsymbol{y}^{*j}_i \in \mathbb{R}^{N_y}\ \forall i \in \{k, k+1, \dots, k+P\} \forall j \in \{1,2,\dots,N_s\}$
$\boldsymbol{L} \in \mathbb{R}^{N_y \times N_y}, \boldsymbol{W} \in \mathbb{R}^{N_u \times N_u}$
$\boldsymbol{e}^j \in \mathbb{R}^{N_e}, \boldsymbol{x}^L \in \mathbb{R}^{N_x}, \boldsymbol{x}^U \in \mathbb{R}^{N_x}, \boldsymbol{u}^L \in \mathbb{R}^{N_u}, \boldsymbol{u}^U \in \mathbb{R}^{N_u}, \widehat{\boldsymbol{x}}_k \in \mathbb{R}^{N_x}$
$f\colon \mathbb{R}^{N_x \times N_u \times N_e} \to \mathbb{R}^{N_x}, h\colon \mathbb{R}^{N_x} \to \mathbb{R}^{N_y}, g\colon \mathbb{R}^{N_x \times N_u \times N_y} \to \mathbb{R}^{N_g}$

Each scenario, which corresponds to a realization of the uncertainty, is denoted by the superscript j. Note that the model takes the same initial condition for all realizations and the manipulated variables are not indexed by the superscript j, which means the same control actions must be

feasible for all scenarios. Additionally, non-negative weight ω_j is assigned to each scenario in the objective function such that $\sum_{j=1}^{N_s} \omega_j = 1$.

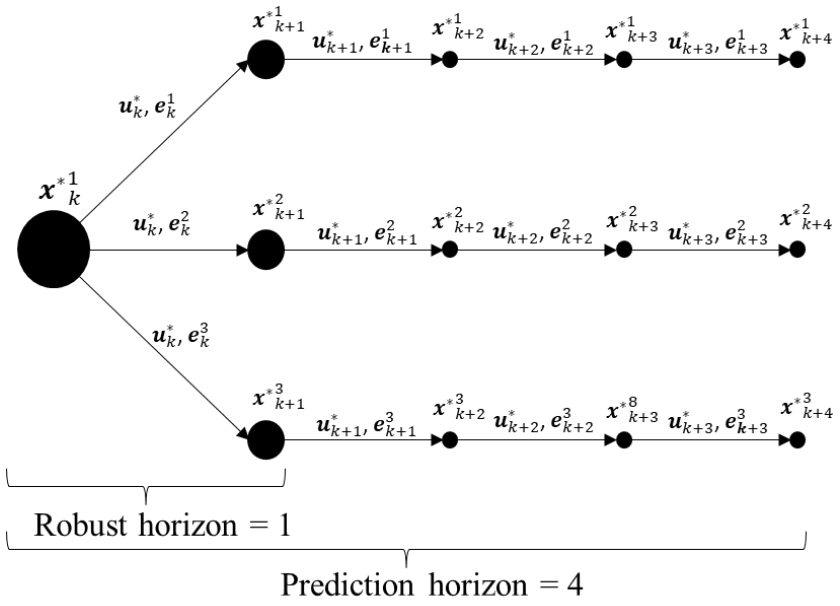


Figure 6 Scenario tree for a RH of one. This is equivalent to the MSc-NMPC formulation.

3.3. State Estimation

This section provides an overview of two common model-based state estimation strategies, the Extended Kalman filter (EKF) and the Moving Horizon Estimator (MHE). These strategies were used in this study to have access to the states that cannot be measured online and use them to provide the initial condition for the different NMPC strategies considered in this work.

3.3.1. Extended Kalman Filter

The extended Kalman filter (EKF) is one of the most widely applied state estimation algorithms for nonlinear systems (Vinoth Upendra and Prakash, 2013). Some industrial-scale studies include the application of EKF to a Gasification system (Valipour, Toffolo and Ricardez-Sandoval, 2021), to a thermal power plant (Prasad *et al.*, 1999) and multiple works have applied EKF to the TE

process (Ricker and Lee, 1995b; Yan and Ricker, 1995; Ming Yan and Ricker, 1997; Golshan, Boozarjomehry and Pishvaie, 2005; Golshan, Pishvaie and Bozorgmehry Boozarjomehry, 2008; Vinoth Upendra and Prakash, 2013; Kumar Kottakki, Bhushan and Bhartiya, 2017; Mukherjee *et al.*, 2021). The Kalman filter is an optimal estimator for linear stochastic systems with normal distributions on process and measurement noise. The EKF expands on the Kalman filter to nonlinear systems by linearizing over the current estimate of the states. This estimator uses the non-linear model in the prediction step, and it updates the Jacobian matrices at every sampling step, which provides a significant improvement in accuracy over the Kalman filter; in addition, its two-step formulation makes it very efficient to compute online and straightforward to implement. The limitations of EKF are that the distributions may not be normal after passing through the nonlinear transformations of the process model, it does not take into account process constraints and it uses a linear approximation to propagate the error over time (Welch and Bishop, 2006). This could result in large estimation errors or even infeasible estimates which ultimately results in loss of performance, suboptimal control actions or possibly destabilization of the system. EKF assumes a model of the following form:

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{w}_{k-1} \tag{54}$$

$$\boldsymbol{y}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{55}$$

where $\boldsymbol{w}_k \in \mathbb{R}^{N_x}$ and $\boldsymbol{v}_k \in \mathbb{R}^{N_y}$ are the process and measurement noises, respectively. The algorithm follows two general steps. First, in the time update step, Eq. (56)-(57), the process model is used to make an *a priori* estimation of the current state, $\hat{\boldsymbol{x}}_k^-$, based on the previous state estimate, $\hat{\boldsymbol{x}}_{k-1}$, by assuming $\boldsymbol{w}_k = \boldsymbol{0}$. Similarly, an *a priori* estimation of the state error covariance matrix, $\boldsymbol{P}_k$, is made.

Time update:

$$\widehat{\boldsymbol{x}}_k^- = f(\widehat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}) \quad (56)$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}_k \boldsymbol{P}_{k-1} \boldsymbol{A}_k^T + \boldsymbol{Q}_{k-1} \quad (57)$$

In the measurement update step, the *a posteriori* estimates of the states and the error covariance matrix are made by correcting with the plant output measurements via the Kalman gain, $\boldsymbol{K}_k$, i.e.,

Measurement update:

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1} \quad (58)$$

$$\widehat{\boldsymbol{x}}_k = \widehat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k \left(\boldsymbol{y}_k - h(\widehat{\boldsymbol{x}}_k^-) \right) \quad (59)$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_k^- \quad (60)$$

where:

$$\boldsymbol{A}_k = \left. \frac{\partial f}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\widehat{\boldsymbol{x}}_{k-1}}, \boldsymbol{H}_k = \left. \frac{\partial h}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}=\widehat{\boldsymbol{x}}_k^-}$$

$$\boldsymbol{K}_k \in \mathbb{R}^{N_x \times N_y}, \boldsymbol{A}_k \in \mathbb{R}^{N_x \times N_x}, \boldsymbol{H}_k \in \mathbb{R}^{N_y \times N_x}$$

$$\boldsymbol{Q}_k \in \mathbb{R}^{N_x \times N_x}, \boldsymbol{P}_k \in \mathbb{R}^{N_x \times N_x}, \boldsymbol{R}_k \in \mathbb{R}^{N_y \times N_y}$$

One of the computational challenges of EKF is that the Jacobian matrices, $\boldsymbol{A}_k$ and $\boldsymbol{H}_k$, must be evaluated at every sampling step. This is particularly challenging for large-scale problems as the number of partial derivatives that must be evaluated increases quadratically with the number of states.

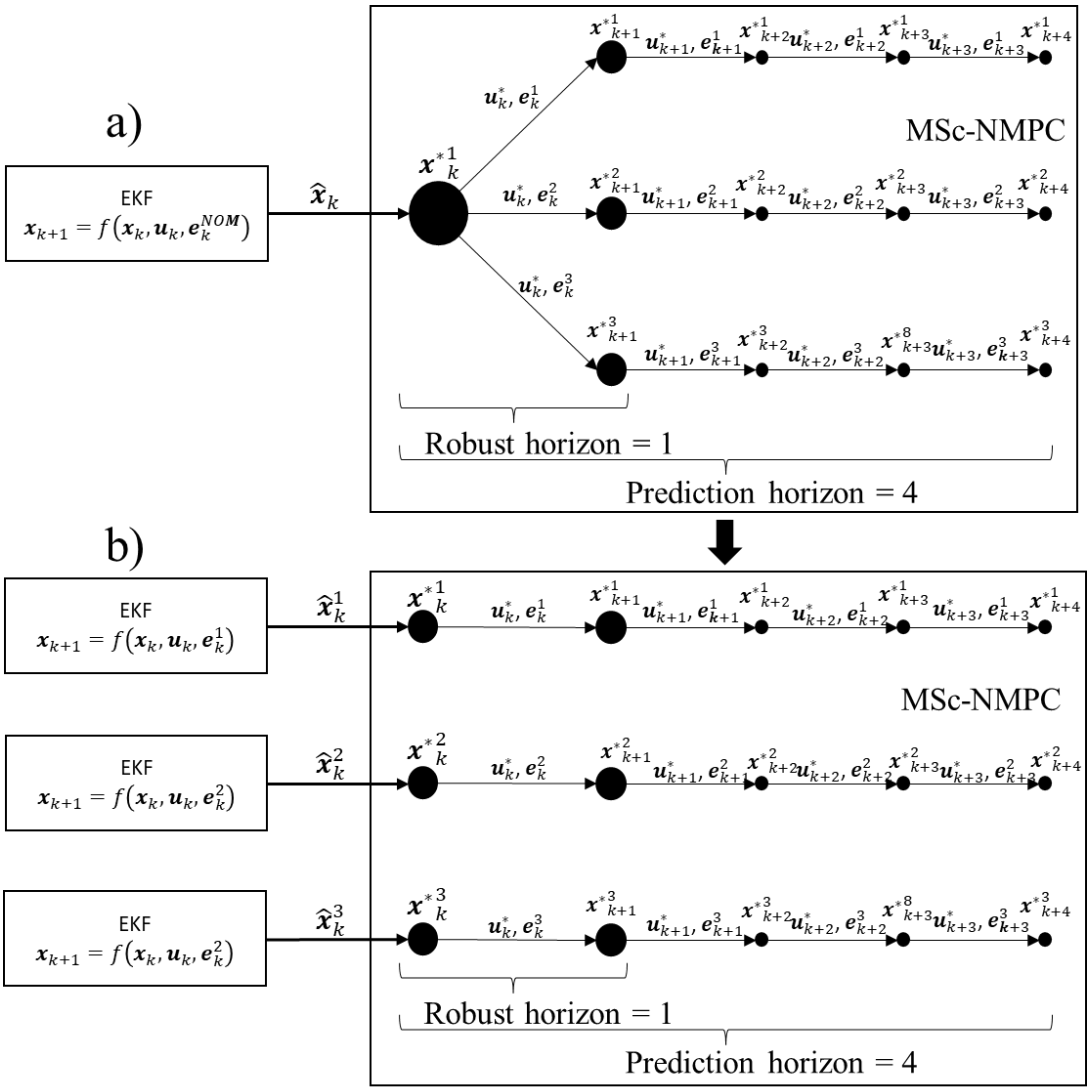


Figure 7 a) EKF using the nominal value of the uncertain parameters to provide a single initial condition to MSc-NMPC; b) 1-to-1 EKF providing a different initial condition to the corresponding scenario in MSc-NMPC.

In an attempt to improve the robustness of the EKF estimate, in this study, a MSc-NMPC/EKF combination that considers the uncertain scenarios used in MSc-NMPC was implemented. It is referred to as 1-to-1 EKF because multiple versions of the EKF are evaluated at each sampling time, each one with the same parameter values used in the MSc-NMPC scenarios. Then, each version of the EKF provides the initial condition to the corresponding scenario in MSc-NMPC. Figure 7 is a schematic that shows a) the typical arrangement where a single version of EKF uses

the nominal value of the uncertain parameter to provide a single initial condition to MSc-NMPC, and b) how 1-to-1 EKF provides a unique initial condition to the corresponding scenario in MSc-NMPC.

3.3.2. Moving Horizon State Estimation

The moving horizon estimator (MHE) finds estimates for the unknown states at the current time interval k by minimizing the estimates of the process and measurement noises using the non-linear process model. The main advantage of MHE over EKF is that it can directly consider process constraints in the formulation of the optimization problem. One limitation of the MHE scheme is that it must be solved online at each sampling instance generating additional computational costs. This could be particularly challenging for large-scale problems since the model may also be complex (Zavala and Biegler, 2009). The MHE formulation is as follows (Segovia *et al.*, 2019; Valipour and Ricardez-Sandoval, 2021a):

$$\min_{\substack{\widehat{\boldsymbol{x}}_i\in\mathbb{R}^{N_x}\ \forall i\in\{k-N,k-N+1,\dots,k\}\\ \boldsymbol{w}_i\in\mathbb{R}^{N_x}\ \forall i\in\{k-N,k-N+1,\dots,k-1\}\\ \boldsymbol{v}_i\in\mathbb{R}^{N_y}\ \forall i\in\{k-N-1,k-N+1,\dots,k\}}} \sum_{i=k-N}^{k-1} \|\boldsymbol{w}_i\|^2_{\boldsymbol{Q}_i^{-1}} + \sum_{i=k-N+1}^{k} \|\boldsymbol{v}_i\|^2_{\boldsymbol{R}_i^{-1}} + \varphi_{k-N} \tag{61}$$

$$\text{s.t. } \widehat{\boldsymbol{x}}_{i+1} = f(\widehat{\boldsymbol{x}}_i, \boldsymbol{u}_i) + \boldsymbol{w}_i \quad \forall i \in \{k-N, k-N+1, \dots, k-1\} \tag{62}$$

$$\boldsymbol{y}_i = h(\widehat{\boldsymbol{x}}_i) + \boldsymbol{v}_i \quad \forall i \in \{k-N+1, k-N+2, \dots, k\} \tag{63}$$

$$g(\widehat{\boldsymbol{x}}_i, \boldsymbol{u}_i, \boldsymbol{y}_i) \leq 0 \quad \forall i \in \{k-N, k-N+1, \dots, k\} \tag{64}$$

$$\boldsymbol{x}^L \leq \widehat{\boldsymbol{x}}_i \leq \boldsymbol{x}^U \quad \forall i \in \{k-N, k-N+1, \dots, k\} \tag{65}$$

$$\varphi_{k-N} = \|\widehat{\boldsymbol{x}}_{k-N} - \overline{\boldsymbol{x}}_{k-N}\|^2_{\boldsymbol{P}_{k-N}^{-1}} \tag{66}$$

where:

$\widehat{\boldsymbol{x}}_i \in \mathbb{R}^{N_x}\ \forall i \in \{k-N, k-N+2, \dots, k\}$
$\boldsymbol{y}_i \in \mathbb{R}^{N_y}, \boldsymbol{v}_i \in \mathbb{R}^{N_y}, \boldsymbol{R}_i \in \mathbb{R}^{N_y \times N_y}\ \forall i \in \{k-N+1, k-N+2, \dots, k\}$
$\boldsymbol{w}_i \in \mathbb{R}^{N_x}, \boldsymbol{u}_i \in \mathbb{R}^{N_u}, \boldsymbol{Q}_i \in \mathbb{R}^{N_x \times N_x}\ \forall i \in \{k-N, k-N+1, \dots, k-1\}$
$\boldsymbol{x}^L \in \mathbb{R}^{N_x}, \boldsymbol{x}^U \in \mathbb{R}^{N_x}, \overline{\boldsymbol{x}}_{k-N} \in \mathbb{R}^{N_x}, \varphi_{k-N} \in \mathbb{R}$
$f\colon \mathbb{R}^{N_x \times N_u} \to \mathbb{R}^{N_x}, h\colon \mathbb{R}^{N_x} \to \mathbb{R}^{N_y}, g\colon \mathbb{R}^{N_x \times N_u \times N_y} \to \mathbb{R}^{N_g}$

MHE considers a finite time window, N, of past measurements. In this problem, the estimates of the states, $\widehat{\boldsymbol{x}}_i$, and the estimates of the noises, $\boldsymbol{w}_i$ and $\boldsymbol{v}_i$, are decision variables; and the previously observed measurements, $\boldsymbol{y}_i$, and the past control actions, $\boldsymbol{u}_i$, are fixed parameters. The noises in the objective function are weighted by their corresponding covariance. The term φ_{k-N} is referred to as *arrival cost* which is meant to consider past information that is not within the finite horizon window. The expected value, $\overline{\boldsymbol{x}}_{k-N}$, and the state error covariance matrix, $\boldsymbol{P}_{k-N}$, are typically approximated using the *a posteriori* estimate from methods like EKF (Valipour and Ricardez-Sandoval, 2021a). Only the last state estimate in the finite horizon, $\widehat{\boldsymbol{x}}_k$, is passed as the initial condition to NMPC. If the estimation horizon is short and the arrival cost will play significant role in the MHE estimation (Valipour and Ricardez-Sandoval, 2021a). In other words, it is highly dependent on the quality of the estimate of $\overline{\boldsymbol{x}}_{k-N}$ and $\boldsymbol{P}_{k-N}$. The role of the arrival cost can be neglected by increasing the size of the estimation window; however, this also increases the size of the problem, making it more computationally demanding.

3.4. Results

This section presents the results of implementing the robust NMPC schemes discussed previously, coupled with state estimation, on the TE process. Figure 8 illustrates the closed-loop feedback control framework considered in this work. First, the performance of a selected MS-NMPC is evaluated and compared to MSc-NMPC and standard NMPC under various scenarios of the uncertain parameters in the plant assuming full access to the states (i.e., the online measurement is available for all the state variables). This was repeated with the inclusion of state estimation to evaluate the impact that estimating the initial states may have on the performance of a robust NMPC. Next, the impact of the RH on the MS-NMPC performance was evaluated and compared to MSc-NMPC and the standard NMPC. Finally, a closed-loop simulation using MSc-NMPC and

MHE as state estimator under the additional impact of set-point changes (i.e., changes in the mode of operation of the plant) and disturbances was assessed.

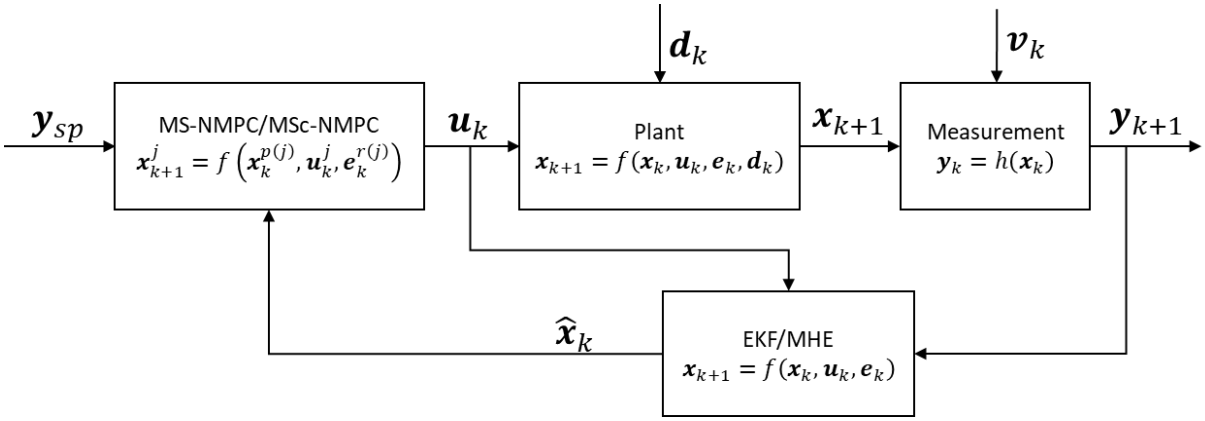


Figure 8 Block diagram representation of the closed-loop feedback control framework.

Table 3 Controlled variables present in the NMPC objective function along with their nominal set-points and weights.

m	**Controlled Variable,** y_m	**Nominal Set-point**	**Weight,** L_m
1	Mole fraction of B in purge, $y_{B,s}$	0.138	209
2	Mole fraction of G in product, $x_{G,str}$	0.537	17.3
3	Mole fraction of H in product, $x_{H,str}$	0.438	26.0
4	Product flow rate, F_{11}	58.6 mol s^{-1}	11.6E−4
5	Recycle flow rate, F_8	333.4 mol s^{-1}	9.00E−6
6	Reactor temperature, T_r	120.4 ℃	38.7E−5
7	Mixing zone temperature, T_m	86.1 ℃	11.6E−5
8	Separator temperature, T_s	80.1 ℃	16.0E−5
9	Stripper temperature, T_{str}	65.7 ℃	17.4E−5
10	Reactor liquid level, $V_{L,r}$	16.55 m^3	18.2E−4
11	Separator liquid level, $V_{L,s}$	4.88 m^3	20.9E−3
12	Separator liquid level, $V_{L,str}$	4.43 m^3	25.5E−3
13	Reactor pressure, P_r	2705 kPag	12.7E−7

The selected uncertain parameters for the TE process were the pre-exponential factors of the reaction kinetics of the first two reactions, i.e. $\boldsymbol{e}_k = [\alpha_1 \quad \alpha_2]^T$. The two selected parameters are expected to have a severe impact on the performance of the TE process due to the high-temperature sensitivity of the exothermic reactions. These parameters were also used in the work by Tatulea-

Codrean et al. (2020) where an economic MS-NMPC was implemented. The TE plant has process and safety-based limitations; these were taken from Downs and Vogel (1993) and were formulated as constraints in the NMPC problems as shown in equations (67)-(71), in addition to the saturation limits on the manipulated variables. Moreover, constraints were also included on the limits of the variable regions without physical meaning, such as molar hold-ups must be non-negative and molar fractions between zero and one. The prediction horizon, P, and control horizon, M, were equal and set to 6 h. Jockenhovel et al. (2003) selected horizons of 5 h and they explained that it is too short for the process to reach steady state, but they selected it due to the computational burden of using a long horizon with NMPC. Other works have used even shorter horizons of 0.55 h (Tătulea-Codrean, Fischer and Engell, 2020) and 1 h (Ricker and Lee, 1995a). The objective function can generally be written as shown in equation (72), where the first term represents the error between the controlled variable y_{m_i} and its set-point y_{sp} weighted by L_m, y_{m_i} being element m in $\boldsymbol{y}_i$ at sampling interval i. The second term represents the magnitude of the changes in the manipulated variables, u_{l_i}, weighted by R_l, u_{l_i} representing element l in $\boldsymbol{u}_i$ at sampling interval i. Table 3 and Table 4 summarize the controlled variables and manipulated variables included in the objective function along with their corresponding weights, respectively. The NMPC presented by Jockenhovel et al. (2003) for the TE process was utilized as a starting point. Then, a trial-and-error approach was employed to fine-tune the NMPC weights. Throughout the study, the focus was on the set-point tracking performance of the reactor pressure because it is a safety critical variable due to the open-loop unstable reactor. In addition to the reactor pressure, the analysis in this study also focuses on the set-point tracking performance of the mixing-zone temperature since it is a controlled variable in the objective function, but it is not a measured variable, which means NMPC must solely rely on the state estimates to predict its behaviour.

$$T_r \leq 150\ ℃ \tag{67}$$
$$P_r \leq 2895\ \text{kPag} \tag{68}$$
$$11.8\ \text{m}^3 \leq V_{L,r} \leq 21.3\ \text{m}^3 \tag{69}$$
$$3.3\ \text{m}^3 \leq V_{L,s} \leq 9.0\ \text{m}^3 \tag{70}$$
$$3.5\ \text{m}^3 \leq V_{L,str} \leq 6.6\ \text{m}^3 \tag{71}$$
$$J = \sum_{i=k}^{k+P} \left(\sum_{m=1}^{N_y} L_m \left(y_{m_{sp}}{}_i - y_{m}{}_i \right)^2 \right) + \sum_{i=k}^{k+P-1} \left(\sum_{l=1}^{N_u} R_l \Delta u_l^2{}_i \right) \tag{72}$$

Table 4 Manipulated variables along with their nominal operating values and weights in the objective function.

l	**Manipulated Variable,** u_l	**Nominal Value**	**Weight,** R_l
1	A feed, F_1	3.11 mol s^{-1}	0.207
2	D feed, F_2	31.8 mol s^{-1}	19.8E−4
3	E feed, F_3	27.2 mol s^{-1}	27.2E−4
4	A and C feed, F_4	116 mol s^{-1}	14.9E−5
5	Compressor recycle valve, p_8	22.2%	40.5E−4
6	Purge valve, p_9	40.1%	12.5E−4
7	Separator liquid product, F_{10}	72.1 mol s^{-1}	38.6E−5
8	Product flow rate, F_{11}	58.6 mol s^{-1}	58.2E−5
9	Agitator speed, $AGSP$	50.0%	40.0E−5
10	Reactor cooling water, $F_{cw,r}$	25.9 kg s^{-1}	29.7E−4
11	Separator cooling water, $F_{cw,s}$	13.7 kg s^{-1}	10.6E−3
12	Steam valve, p_{stm}	47.4%	44.4E−6

The closed-loop framework presented in Figure 8 was implemented in Pyomo 6.4.2 with Python 3.9.12 on a Windows 10 computer equipped with 32.0 GB RAM, Intel® Core™ i9-10980HK CPU @ 2.40 GHz 3.10 GHz. The ODEs that model the TE process were discretized by a backward finite difference such that the dynamic optimization problems may be converted to large-scale non-linear programs (NLPs). It was assumed that the discretization step size was equal to the sampling period of 0.1 h. This results in a large-scale NLP with 14,456 variables for the standard NMPC formulation, i.e., no uncertainty considered. All NLPs were solved using IPOPT.

EKF requires the evaluation of the Jacobian matrixes of the system at each time interval. For this purpose, the *differentiate* function from *pyomo.core.expr* was used. It automatically converts Pyomo expressions to Sympy and uses the symbolic differentiator. To make the online implementation more efficient, the symbolic Jacobians only have to be computed once before the simulation, for each iteration hereafter, it is only necessary to evaluate the symbolic expression at the current values. The MHE model was discretized in the same way as the NMPC formulation. The moving window was set to 4 h ($N = 40$); since the window is long, the arrival cost was not considered. This results in an NLP with 10,812 variables. The first time MHE is solved in the closed-loop framework, an initial guess must be provided for all the variables. To generate a reasonable initial guess, the model can be simulated in closed-loop with NMPC, assuming full-state knowledge, until steady-state. Since in the first 4 h of using MHE, the moving window overlaps with times earlier than the first sampling instance, in which there are not any measurements available, all measurements before that time were assumed to be the nominal value of the measurement complemented with randomly sampled measurement noise.

The set-point tracking performance of a given controlled variable was measured with the root-mean-squared error (RMSE) defined in equation (73), where N_{sim} is the number of sampling instances in the simulation and y_{j_k} represents the controlled variable j at sampling instance k.

$$RMSE = \left(\frac{1}{N_{sim}} \sum_{k=0}^{N_{sim}} \left(y_{j_{sp}} - y_{j_k} \right)^2 \right)^{0.5} \tag{73}$$

3.4.1. Robust NMPC Performance without State Estimation

NMPC, MSc-NMPC and MS-NMPC were compared assuming full access to the states considering uncertainty in the reaction kinetics of the first two reactions. In other words, the state estimator was not considered, and the states were passed directly to the controller. Different realizations of the uncertain parameters were used in the plant to simulate multiple scenarios that may cause a plant-model mismatch. A 5% uncertainty was considered, i.e. values up to 5% above and 5% below the nominal value of the parameters. As it will be shown below, this range of uncertainty was found to have a severe impact on the process performance. In the MSc-NMPC, two levels of the uncertain parameters were used which resulted in four discrete scenarios. Let $\boldsymbol{e}^{NOM} = [\alpha_1^{NOM} \quad \alpha_2^{NOM}]^T$ represent the nominal values of the uncertain parameter, then, the four scenarios used in MSc-NMPC are summarized in Table 5. For MS-NMPC, two variable combinations were considered, one where both are at the highest value and one where both are at the lowest, i.e. $\boldsymbol{e}_i^{r(j)} \in \{1.05\boldsymbol{e}^{NOM}, 0.95\boldsymbol{e}^{NOM}\}$, and the two combinations were varied over a RH of three, which resulted in eight scenarios; the scenario tree for this configuration can be seen in Figure 9. The variables in the NLPs were initialized with the nominal steady-state values. For standard NMPC, the uncertain variables were left at the nominal value, i.e., $\boldsymbol{e}^{NOM}$. These three controllers were tested in closed-loop assuming three different cases, i.e., S1, the plant was run at the nominal case, i.e. $\boldsymbol{e}_k = \boldsymbol{e}^{NOM}$; S2, both uncertain parameters were fixed at 5% below their nominal value, i.e. $\boldsymbol{e}_k = 0.95\boldsymbol{e}^{NOM}$; and S3, the kinetics of reaction one was run at 5% above the nominal value and, for reaction two, at 5% below, i.e. $\boldsymbol{e}_k = [1.05\alpha_1^{NOM} \quad 0.95\alpha_2^{NOM}]^T$. For all cases, the initial states of the plant were 1% above the nominal steady-state value to introduce a dynamic response. In general, for all cases, if the initial states provided to the controller are within the feasible range, the standard NMPC can find optimal solutions relatively fast, within 4 seconds on average at every

sampling instance. Finding an optimal solution to a robust NMPC can take significantly more time, 1 to 5 h depending on the number of uncertain scenarios considered. However, the online solution time can be significantly reduced by using the solution at the previous sampling time to initialize primal and dual variables in IPOPT for the next sampling time as a warm start. This was set up via the *warm_start_init_point* option in IPOPT. The RMSE of some selected controlled variables and computational time for each controller in each of the three cases are summarized in Table 6.

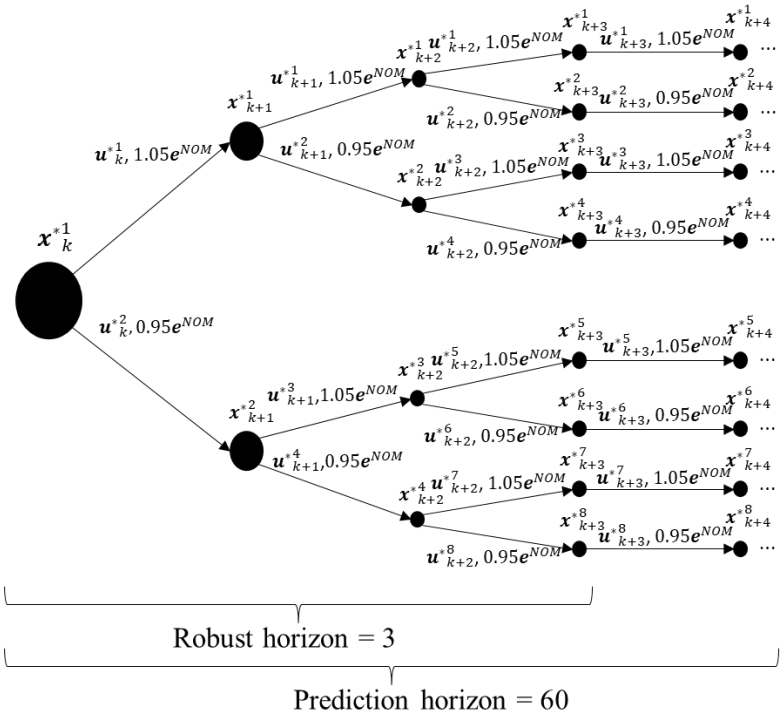


Figure 9 MS-NMPC scenario tree implemented in this study.

The closed-loop responses of the three controllers on the reactor pressure and the mixing-zone temperature for cases S1, S2 and S3 are shown in Figure 10. In the nominal case, S1, Figure 10 a) and b), where there is not any mismatch between the plant and the NMPC model, it can be observed that the controlled variables quickly reach the set points without any offset; as expected, NMPC (blue line) has the best performance of all controllers in this case. Both MS-NMPC and MSc-

NMPC exhibit an offset between the controlled variable and the set-point. These controllers must balance between all the scenarios considered in their respective formulations and remain feasible, as a result, one must sacrifice some performance, which is the main drawback of using a robust controller (Patrón and Ricardez-Sandoval, 2020a). Offset-free NMPC formulations are potential methods to correct for uncertainty while overcoming the set-point tracking error (Skupin *et al.*, 2022). However, when the uncertainty realization used in the plant is changed to S2, the process using the NMPC becomes unstable and violates the process shutdown constraint limits. As shown in Figure 10 c), the pressure goes above the shut-down limit from equation (68). When the realized values of the reaction kinetics are below the nominal values, like in S2, the reaction rates in the process are lower than what NMPC predicts; since the reactants are gaseous and the products condensable, lower reaction rates lead to higher pressures in the reactor because there is more material in the gas phase. This mismatch causes NMPC to predict that the pressure will quickly decrease back to its set-point value and thus, it does not take any action to counter the high pressure that is observed in the process. If NMPC could detect a problem within the prediction horizon, an alarm could be activated to warn that the process is approaching the constraint limits. However, as mentioned the NMPC predicted that the process would return to the set-point within the prediction horizon, thus, such a feature would not be possible with the nominal NMPC when there is uncertainty in the plant. The fact that NMPC could fail in such scenarios serves as the main motivation to use robust NMPC in the TE process. Under model uncertainty, a well-tuned NMPC may result in a closed-loop unstable response that violates the process constraints. Even in other scenarios where NMPC does not result in constraint violation, as observed for S3, Figure 10 e) and f), there is a significant deterioration in performance with large offsets and long settling times. This also highlights the difficulties in controlling the TE plant; an uncertainty of only 5% was

sufficient to cause the plant/model mismatch to become significant due to the highly non-linear nature of this process.

Table 5 Scenarios considered within MSc-NMPC

j	$\boldsymbol{e}^j$
1	$[1.05\alpha_1^{NOM} \quad 1.05\alpha_2^{NOM}]^T$
2	$[0.95\alpha_1^{NOM} \quad 0.95\alpha_2^{NOM}]^T$
3	$[1.05\alpha_1^{NOM} \quad 0.95\alpha_2^{NOM}]^T$
4	$[0.95\alpha_1^{NOM} \quad 1.05\alpha_2^{NOM}]^T$

Table 6 Summary of the robust NMPC performance under multiple scenarios with full state access.

Case	Variable	NMPC	MSc-NMPC	MS-NMPC
S1 Nominal	**RMSE P_r (kPag)**	0.215	13.1	13.9
	RMSE T_m (°C)	0.00204	0.669	0.519
	RMSE $x_{G,str}$	1.03E-5	3.32E-3	3.61E-3
	RMSE $x_{H,str}$	7.21E-6	2.97E-3	2.72E-3
S2 $\alpha_1 = 0.95$, $\alpha_2 = 0.95$	**RMSE P_r (kPag)**	Unstable	39.8	44.3
	RMSE T_m (°C)	Unstable	0.997	1.36
	RMSE $x_{G,str}$	Unstable	3.94E-3	4.69E-3
	RMSE $x_{H,str}$	Unstable	5.54E-3	5.89E-3
S3 $\alpha_1 = 1.05$, $\alpha_2 = 0.95$	**RMSE P_r (kPag)**	66.7	33.5	36.4
	RMSE T_m (°C)	2.01	0.745	1.02
	RMSE $x_{G,str}$	5.63E-3	3.71E-3	4.31E-3
	RMSE $x_{H,str}$	7.96E-3	5.41E-3	5.57E-3
NMPC Solve Time (s)		3.06	21.4	55.7

In scenarios where NMPC does not perform well, both MS-NMPC and MSc-NMPC can more closely track the set points of the process, prevent constraint violation and provide significantly better performance compared to NMPC. In S2, the robust controllers, MS-NMPC (green line) and MSc-NMPC (orange line), prevented constraint violation and maintained the closed-loop response under an acceptable operation. This can be seen in the reactor pressure and mixing zone temperature response in Figure 10 c) and d) and the response of other important controlled variables with constraint limits such as the reactor temperature a) and the separator liquid level b) in Figure 11. Further looking at the graphical responses in Figure 10 and Figure 11, the

performance of both robust controllers is similar for all controlled variables in all cases. Furthermore, Figure 11 also shows that the product quality in S2, i.e. the mole fractions of G, Figure 11 c), and H, Figure 11 d), in the product stream, remains close to its objective when a robust controller is used. Figure 12 shows the control profiles of two selected manipulated variables: the purge valve position Figure 12 a), and the reactor cooling water Figure 12 b) in S2. NMPC does not take control actions to reject the increase in the reactor pressure because the model mismatch results in inaccurate predictions.

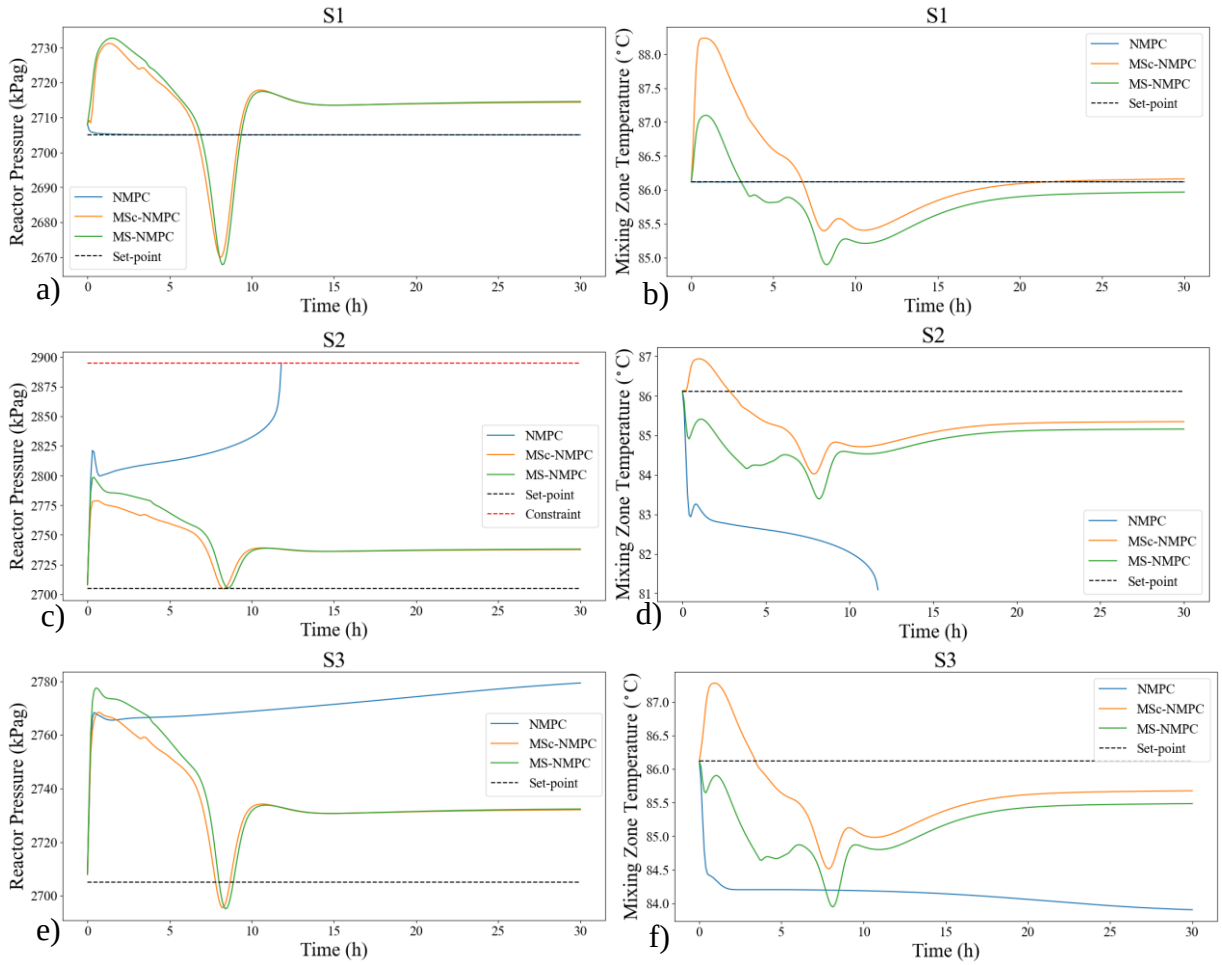


Figure 10 Simulation of robust NMPC under various realizations of the uncertain parameters assuming full state access. Reactor pressure and mixing zone response in cases S1, a) and b); S3, c) and d); and S4, e) and f). Note that the nominal NMPC response in a) and b) overlaps with the set-point for most of the simulation time.

Both MSc-NMPC and MS-NMPC can maintain the process within the operational limits while closely tracking the set point. In S3, Figure 10 e) and f), the response to NMPC is not closed-loop unstable but the simulation time was not enough to reach steady state and the offset from the controller set point is significant. The robust controllers showed similar performance in S3 as in the previous two cases. The RMSE, listed in Table 6, of a given controlled variable, would suggest that MSc-NMPC performed slightly better in some cases, but worse in others compared to MS-NMPC. For example, the RMSE of the mixing zone temperature is slightly higher for MSc-NMPC in S1, 0.669 ℃, compared to MS-NMPC, 0.519 ℃. However, it is lower in the other cases, like S2 where MSc-NMPC has an RMSE of 0.997 ℃ and MS-NMPC is 1.36 ℃. Similar observations can be made with the other controlled variables shown in Figure 11.

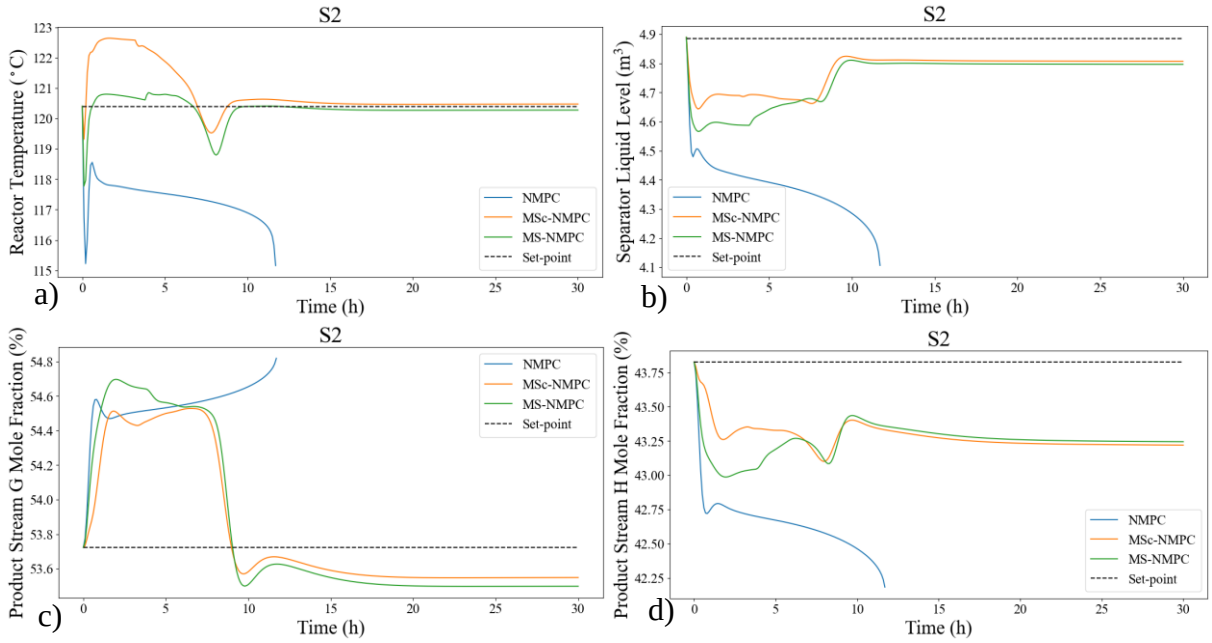


Figure 11 Reactor temperature a), separator liquid level b), and product stream mole fractions of G, c), and H, d), in S2.

The computation times reported in Table 6 are the average online solve times for each controller. It shows that MS-NMPC takes the longest to solve at every sampling instance. This is primarily

due to the size of the problem because, with a RH of three, MS-NMPC involves eight scenarios. Recall, the standard NMPC has 14,456 variables; in addition, MSc-NMPC has 51,308 variables and MS-NMPC has 110,951. The size of the problem is proportional to the number of scenarios considered. The computation time could be reduced if the RH is reduced, however, as discussed in section 3.4.2, the controller will become more conservative. Additionally, MS-NMPC, with RH of one, yields identical performance to MSc-NMPC when the same combinations of discrete uncertain variable realizations are considered. Hence, the selection of the controller is dependent on how much conservativeness the user is willing to accept as well as how sensitive the process may be to uncertainty. The control architecture used in this work is centralized; depending on the size of the problem a decentralized architecture may be necessary to alleviate the computational burden (Anderson, Ellis and Christofides, 2015). However, for the selected sampling time (0.1 h) in this problem, the computation times were found to be reasonable, thus, decentralized approaches were not considered and are beyond the scope of this work.

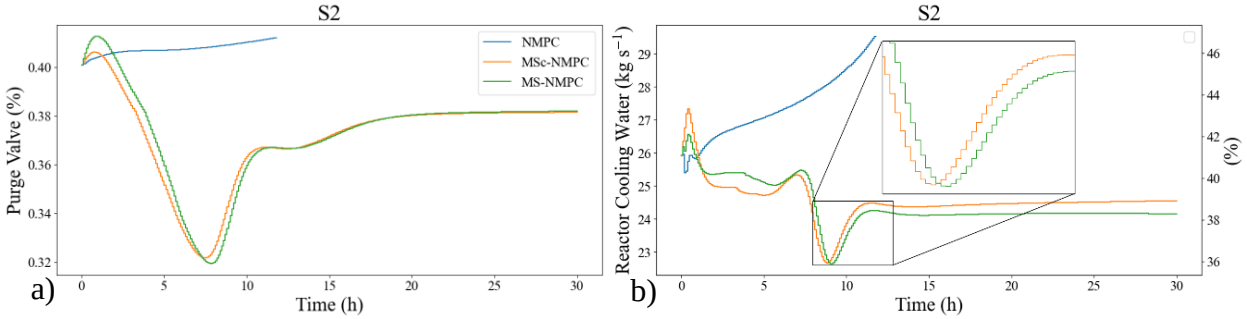


Figure 12 Control profiles of the purge valve a), and the reactor cooling water b), in S2.

3.4.2. Robust NMPC/State Estimation Closed-loop Performance

The same set of cases as in the previous section was considered in this section, except that instead of providing the current states directly from the plant to the controllers, the framework shown in Figure 8 was implemented, where a state estimator takes the current measurements from the plant

and provides the estimates as the initial condition to the controller. The observability of the system was evaluated by verifying that $\det(\mathcal{O}^T\mathcal{O})$ is different from zero, where $\mathcal{O}$ is the observability matrix of the linearized system. The two state estimators studied in this work (i.e., EKF and MHE) use the TE process model to generate their estimates, and when there is uncertainty present in model parameters, the quality of the state estimates is affected. That in turn, affects the quality of the control actions since the solution that NMPC finds is optimal to the estimated state values but not necessarily to the current state of the plant. For standard EKF and MHE, it was assumed that the uncertain parameter remains at the nominal value. In an attempt to improve the robustness of the estimate, the 1-to-1 EKF described in section 3.3.1 was also considered as a state estimator where each scenario used MSc-NMPC is also in a corresponding version of EKF. A schematic representing the interaction between 1-to-1 EKF and MSc-NMPC is shown in Figure 7 b). In the plant and the internal models used for estimation, it was assumed a zero-mean Gaussian process noise, $\boldsymbol{w}_k$, with standard deviation of 0.1% the nominal magnitude of the corresponding state; and zero-mean Gaussian measurement noise, $\boldsymbol{v}_k$, with the standard deviation of 0.1% of the corresponding measurement (e.g., the nominal reactor pressure is 2705 kPag, hence, the measurement noise has a standard deviation of 2.705 kPag).

The cases presented here are, S1, the nominal case, i.e. $\boldsymbol{e}_k = \boldsymbol{e}^{NOM}$; S3, the kinetics of the first reaction 5% above and the second 5% below the nominal value, i.e. $\boldsymbol{e}_k = [1.05\alpha_1^{NOM} \quad 0.95\alpha_2^{NOM}]^T$; and S4, both uncertain variables are 5% above their nominal value, i.e. $\boldsymbol{e}_k = 1.05\boldsymbol{e}^{NOM}$. The combination of controller and state estimator tested in this work were NMPC/EKF, MSc-NMPC/EKF, MSc-NMPC/1to1EKF, MSc-NMPC/MHE and MS-NMPC/EKF. Figure 13 shows the reactor pressure and the mixing zone temperature in all three cases tested in

this section. Table 7 summarizes the calculated RMSE and the computational times of the controllers and state estimators.

Table 7 Summary of the robust NMPC performance under multiple scenarios with state estimation.

Case	Variable	NMPC/ EKF	MS-NMPC/ EKF	MSc-NMPC/ EKF	MSc-NMCP/ MHE	MSc-NMPC/ 1-to-1 EKF
S1 Nominal	**RMSE P_r (kPag)**	3.61	13.7	12.9	12.9	Unstable
	RMSE T_m (°C)	0.148	0.525	0.674	0.678	Unstable
	RMSE $x_{G,str}$	5.58E-4	3.74E-3	3.37E-3	3.37E-3	Unstable
	RMSE $x_{H,str}$	5.67E-4	2.77E-3	3.02E-3	3.03E-3	Unstable
S3 $\alpha_1 = 1.05$, $\alpha_2 = 0.95$	**RMSE P_r (kPag)**	98.5	45.7	41.5	32.5	32.3
	RMSE T_m (°C)	2.96	1.27	0.840	0.783	0.827
	RMSE $x_{G,str}$	8.77E-3	4.94E-3	4.33E-3	3.83E-3	3.26E-3
	RMSE $x_{H,str}$	1.14E-2	6.61E-3	6.33E-3	7.26E-3	5.56E-3
S4 $\alpha_1 = 1.05$, $\alpha_2 = 1.05$	**RMSE P_r (kPag)**	39.4	Unstable	Unstable	15.2	Unstable
	RMSE T_m (°C)	0.897	Unstable	Unstable	0.819	Unstable
	RMSE $x_{G,str}$	2.63E-3	Unstable	Unstable	3.39E-3	Unstable
	RMSE $x_{H,str}$	3.47E-3	Unstable	Unstable	2.48E-3	Unstable
NMPC Solve Time (s)		3.06	55.7	21.4		
State Estimation Time (s)		0.171			2.99	0.534

As shown in Table 7, in case S1, NMPC/EKF has the best performance because the NMPC and the EKF model perfectly match the plant model. This is also observed in the blue line Figure 13 a) and b). However, note that because NMPC/EKF is less conservative compared to the other controllers, the effect of the added process/measurement noise is more amplified compared to the other controllers. This is most evident in the reactor pressure response in case S1 in Figure 13 a).

As mentioned above, MSc-NMPC and MS-NMPC leave an offset between the controlled variable and the set-point because they must balance all the scenarios considered in their formulations. The main observation is that MSc-NMPC/EKF and MS-NMPC/EKF may not be able to control the TE process in all cases when EKF is used as the state estimator. In cases such as S4, the mismatch between the EKF model and the plant causes the estimated initial state to fall into an infeasible region. Figure 14 shows the estimation performance on the reactor molar hold-up of D and the hold-up of G in the stripper in cases S1 and S4. Note that the dashed and solid lines of the same colour represent the estimated states and the true states, respectively. In case S4 in Figure 14 c), it is observed that MS-NMPC/EKF (green line) and MSc-NMPC/EKF (orange line) made the estimate of the molar hold-up of D below zero kmol. In almost all cases where the plant response became unstable, it was because the reactor molar hold-up of D was estimated to be below zero by EKF. At sampling instances where EKF made infeasible state estimates, neither of the two robust controllers could find feasible solutions, and the solver either converged to an infeasible point or reached maximum iterations. Then, the implemented control action resulted in destabilizing the process, hence, the simulation stops earlier as shown in Figure 13 e) and f) and in Figure 14 c) and d). Figure 15 shows the control profiles of the purge valve, Figure 15 a), and the reactor cooling water, Figure 15 b), for case S4. For the cases where the robust controllers were engaged with EKF, i.e., MS-NMPC/EKF and MSc-NMPC/EKF, the state estimator returned overly aggressive control actions with the valves moving between their upper and lower saturation limits for a few iterations thus leading to an unstable closed-loop operation, as shown in Figure 15 a). The same condition was observed in cases S1 and S4 when 1-to-1 EKF was used, as one or more of the scenarios considered in the MSc-NMPC/1to1EKF (purple line) returned unrealistic estimates, as shown in Figure 14 a) and c). However, in S3, where all the scenarios used in 1-to-1 EKF made

feasible estimates, the closed-loop performance was observed to be better than the standard EKF as reflected by the RMSE values reported in Table 7. Thus, 1-to-1 EKF can improve the set-point tracking performance of MSc-NMPC, however, it does not resolve the problem that EKF may also return unrealistic estimates in some instances. Moreover, in cases where there is model mismatch between the plant and the EKF model, but EKF does not make infeasible estimates, a large gap between the true and predicted states is observed, as depicted in S4 in Figure 14 c) and d) in NMPC/EKF (blue line). This further adds to the large offset observed with NMPC/EKF in Figure 13 c), d), e) and f).

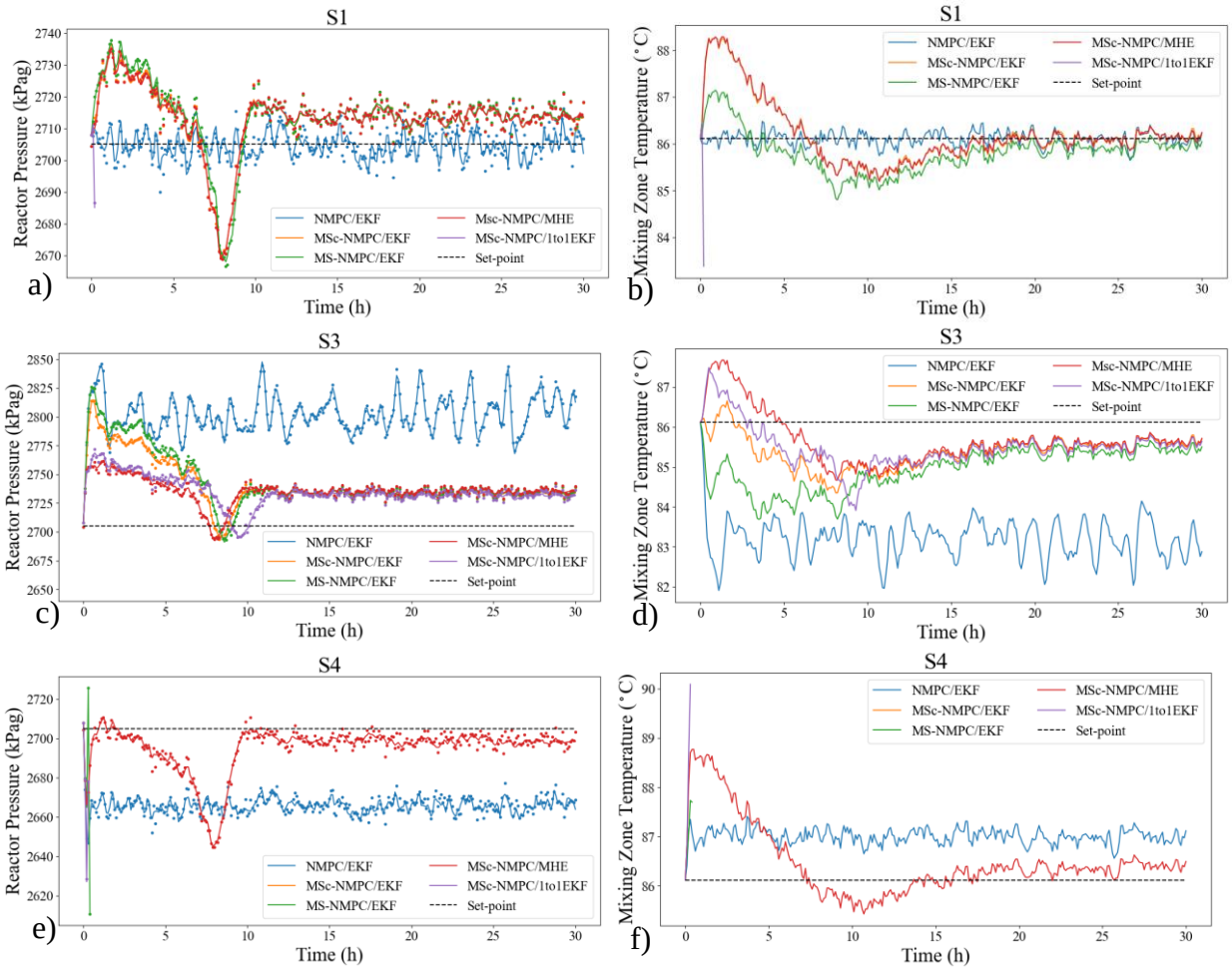


Figure 13 Response to different controller/state estimator combinations. Pressure and mixing zone temperature response in cases S1, a) and b); S3, c) and d); and S4, e) and f). The solid lines represent the plant output, and the markers are the output plus the measurement noise.

The only controller/state estimator combination attempted that resulted in a feasible closed-loop response under all cases was MSc-NMPC/MHE. MHE has the benefit of considering process constraints, which means the estimates from MHE will be within the feasible range of the process. As shown in Figure 14, the state estimates from MHE remain within the feasible bounds and the gap between the estimated and the true state is smaller compared to EKF. Figure 13 and the RMSE values in Table 7 show that MSc-NMPC/MHE displayed the best set-point tracking performance in most cases. Consequently, the closed-loop performance of MSc-NMPC/MHE is better than any robust controller/EKF combination. The main drawback of MHE compared to EKF is the increased computational time, which in this case was 17.5 times larger, however, considering that the sampling time of the process is 0.1 h, the MSc-NMPC/MHE combination can be suitable for this application.

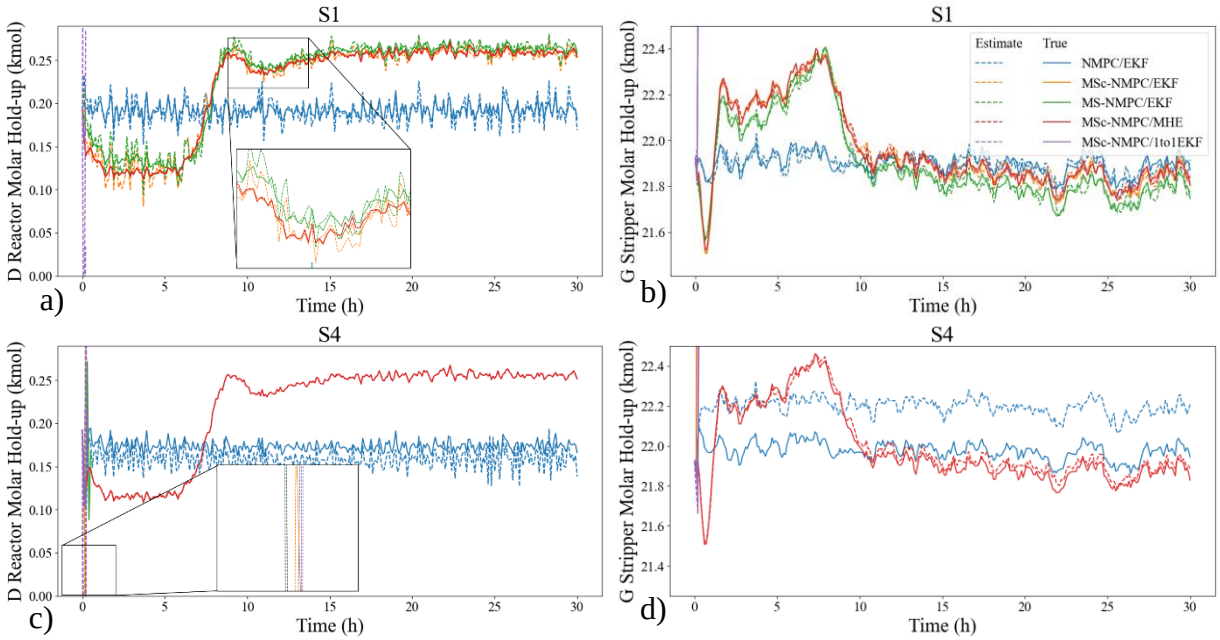


Figure 14 State estimation performance on the molar hold-up of D in the reactor in S1 a) and S4 c), and the hold-up of G in the stripper in S1 b) and S4 d). The true states (from the plant) are represented by solid lines and the estimated states are dashed lines.

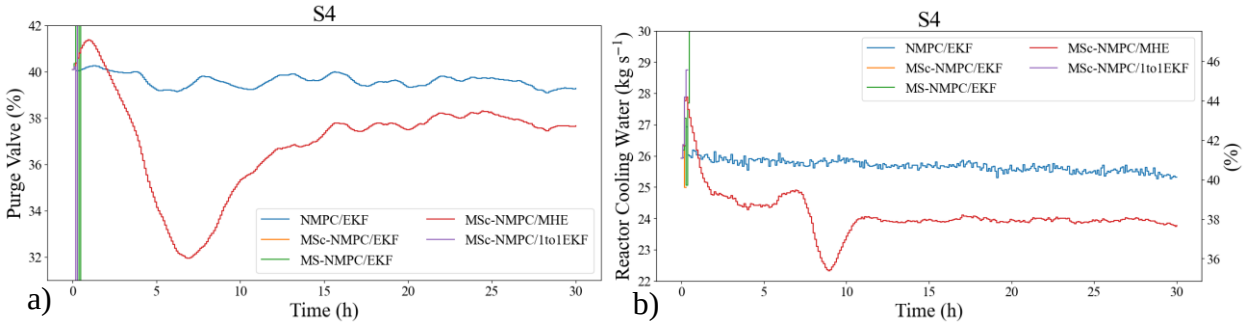


Figure 15 Control profiles of the purge valve a), and the reactor cooling water b) in S4.

3.4.3. Effect of the Robust Horizon

The size of the MS-NMPC problem can grow very quickly as the number of uncertain variables considered increases, as the number of discrete values of the uncertain variables increases, and as the RH increases (Lucia and Engell, 2015). To analyse the effect of the RH in this section, the number of uncertain variables was limited to one (the kinetics of reaction one), $\boldsymbol{e}^{NOM} = [\alpha_1^{NOM}]$, the number of discrete values to two, i.e. $\boldsymbol{e}_i^{r(j)} \in \{1.05\boldsymbol{e}^{NOM}, 0.95\boldsymbol{e}^{NOM}\}$, and varied the RH from one to three. This results in three MS-NMPCs with two, four and eight scenarios respectively for increasing RH. For comparison, MSc-NMPC was run with two, four and eight discrete values of the uncertain parameter that were uniformly distributed between $\pm$5% of the nominal value. Full access to the states was assumed, and the uncertain parameters in the plant were left at their nominal value. Figure 16 shows the effect of changing the RH on the reactor pressure and the mixing zone temperature, compared with MSc-NMPC. Table 8 summarizes the calculated RMSE and the average time that it takes to solve the control problem at each sampling step.

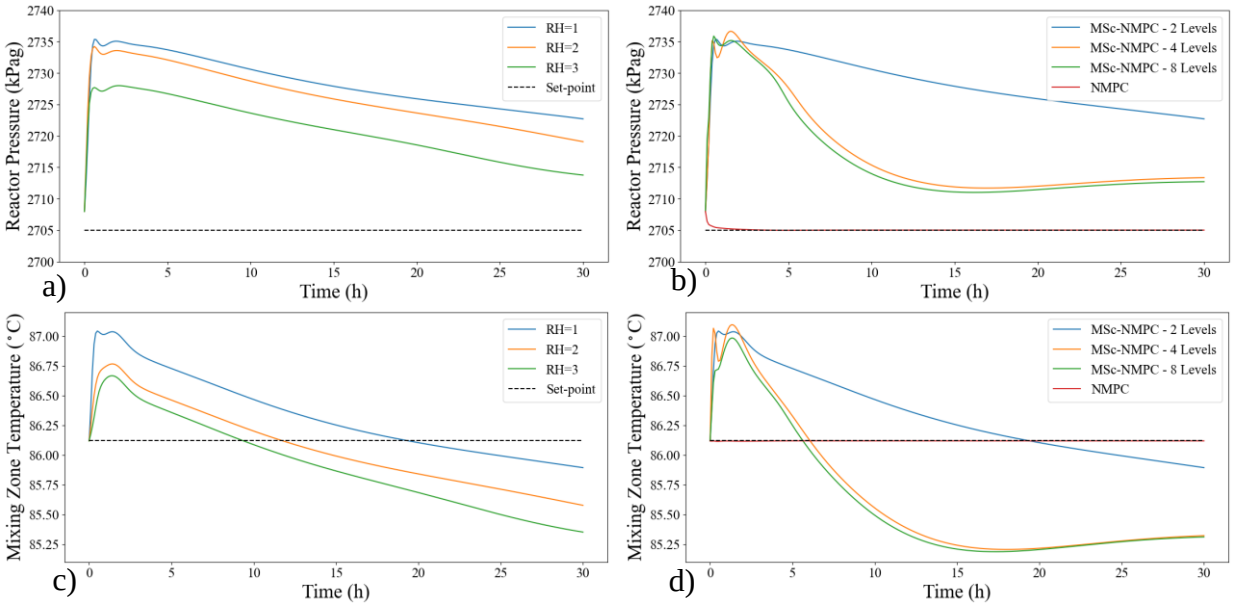


Figure 16 Set-point tracking performance of MS-NMPC at varying RHs on the reactor pressure a) and mixing zone temperature c), compared to MSc-NMPC, b) and d) respectively.

Table 8 Summary of the performance of MS-NMPC at varying RH compared with MSc-NMPC.

MS-NMPC	**RMSE P_r (kPag)**	**RMSE T_m (°C)**	**RMSE $x_{G,str}$**	**RMSE $x_{H,str}$**	**Solve Time (s)**
RH = 1	23.6	0.392	1.16E-3	2.77E-3	7.69
RH = 2	21.7	0.340	7.60E-4	1.23E-3	25.5
RH = 3	16.6	0.427	1.55E-3	2.91E-4	53.2
MSc-NMPC					
2-Levels	23.6	0.392	1.16E-3	2.77E-3	8.20
4-Levels	14.3	0.751	1.28E-3	2.13E-3	15.7
8-Levels	13.5	0.757	9.58E-4	1.83E-3	58.2
Traditional NMPC	0.215	0.00205	1.03E-5	7.21E-6	3.06

Robust NMPC is more conservative than standard NMPC, hence, larger offsets and longer settling times were observed for both MS-NMPC and MSc-NMPC. Figure A.1 in Appendix A shows the effect of decreasing the uncertainty from 5% to 1% when a MSc-NMPC with two levels is used. It was observed that the process settles much faster. Thus, the long settling time in Figure 16 may be due to the nonlinearity of the process. For non-linear systems, the worst-case scenario is not necessarily at the extreme value. Since the realization used in the plant is not one of the scenarios

considered in the MS-NMPC or MSc-NMPC formulations, a deteriorated performance is observed and the settling time is long. Increasing the RH led to less conservative responses because the manipulated variables can adjust to the past observations of the uncertain parameters in the prediction horizon; however, this led to an increased computational time due to the increase in the problem size. In the graphs in Figure 16 a) and c), it can be observed that as the RH increases, the controller approaches the set point faster. Note that the performance of an MS-NMPC with RH equal to one is identical to an MSc-NMPC with two discrete realizations of the uncertain parameter. Looking at the RMSE values in Table 8, increasing the number of discrete realizations in an MSc-NMPC from two to four led to improved performance on the reactor pressure and the mole fraction of H in the product stream; however, the opposite was observed with the mixing zone temperature and mole of G in the product. Running the same controllers with only the reactor pressure as a controlled variable in the MSc-NMPC objective function resulted in worse performance as the number of realizations increased (not shown for brevity). These results suggest that there is a trade-off between the controlled variables that the MSc-NMPC balances out as the number of discrete scenarios increases. A re-tuning of the weights in the MSc-NMPC may result in different results. Comparing MS-NMPC to MSc-NMPC with an equal number of discrete scenarios (i.e., RH of two to four levels, RH of three to eight levels), the set-point tracking performance may vary depending on the controlled variable that one focuses on; with MSc-NMPC tracking the pressure set point closer than MS-NMPC but performing poorer when tracking the mixing-zone temperature. Figure 17 shows the control profiles on the reactor cooling water and the purge valve. As shown in plots Figure 17 a) and Figure 17 c), changing the RH does not seem to affect the trajectory taken by these manipulated variables with most of the difference occurring near the end of the operation on the purge valve. Increasing the number of scenarios in MSc-

NMPC makes the control actions more aggressive near the beginning of the operation, but the controllers with four and eight scenarios eventually converge to somewhat similar values. As shown in Table 8, the computational time was very similar for both controllers and was directly related to the number of discrete scenarios.

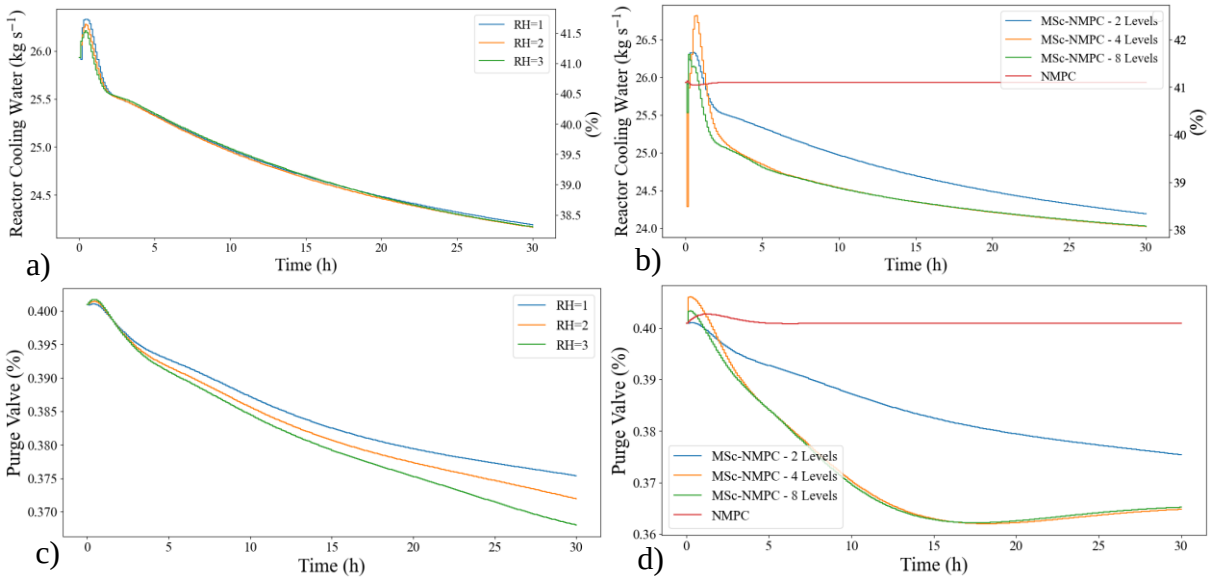


Figure 17 Effect of the RH on the control profiles of MS-NMPC on the reactor cooling water a) and the purge valve c), compared to MSc-NMPC, b) and d) respectively.

3.4.4. Set-point Changes and Disturbance Rejection

Based on the results presented in section 3.4.2, the MSc-NMPC/MHE combination is the only one that was able to generate a closed-loop response that did not violate constraints under all attempted cases. Thus, the robust performance of this closed-loop framework was tested under a series of changes in the modes of operation of the plant and unmeasured disturbances as recommended by Downs and Vogel (1993). Figure 18 shows the response to a series of set-point changes and disturbances. At 13 h, a set-point change of the reactor pressure from 2705 to 2645 kPag was implemented. At 25 h, a product mix set-point change is made from 50 G/50 H to 40 G/60 H. This is one of the recommended operating mode changes in Downs and Vogel (1993). At 41 h, the two

uncertain parameters in the plant are changed from their nominal value to both being 5% below in a step. Then, at 46 h, the kinetics of the first reaction is changed to 5% above the nominal value and the second parameter remains at 5% below. At 51 h, both parameters are changed to their nominal value. At 56 h, the inlet temperature of the reactor cooling water makes a step change from 35 to 40 ℃ and back to 35 ℃ at 61 h. Figure 19 shows the performance of the manipulated variables, the reactor cooling water and the purge valve, as the process is subjected to the various operational changes indicated above.

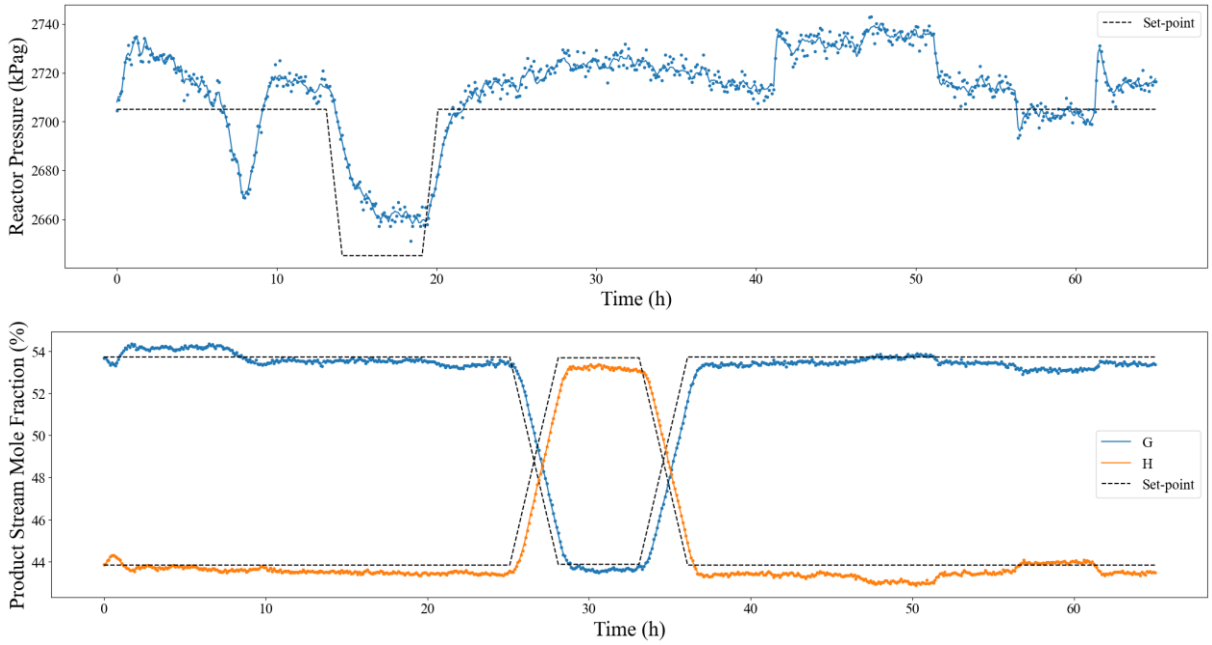


Figure 18 MSc-NMPC/MHE performance under set-point changes and disturbances.

Figure 18 and Figure 19 show that the RNMPC/MHE can accommodate the recommended set-point changes. As expected, the robust controller exhibited a steady-state offset between the set point and the controlled variable since it must remain feasible under all the scenarios considered. Additionally, it takes longer to reach steady-state compared to a traditional NMPC; however, as shown in previous sections, NMPC may cause a closed-loop unstable response in the presence of uncertainty. Moreover, MSc-NMPC/MHE was shown to remain in the feasible range even if the

uncertain parameters change over time. Furthermore, the cooling water disturbance showed that the MSc-NMPC/MHE can handle external sources of perturbations that may not be considered in the MSc-NMPC formulation. Since the cooling water temperature is not measured, its value cannot be updated within the NMPC model. When the same disturbance is used on the NMPC, even if there is not any uncertainty in the model parameters, the system exhibits very large offsets between the set-points and the controlled variables. This is shown in the reactor pressure and temperature response in Figure 20. The cooling water temperature directly affects the heat transfer rate for cooling the reactor, hence, the process is very sensitive to this disturbance.

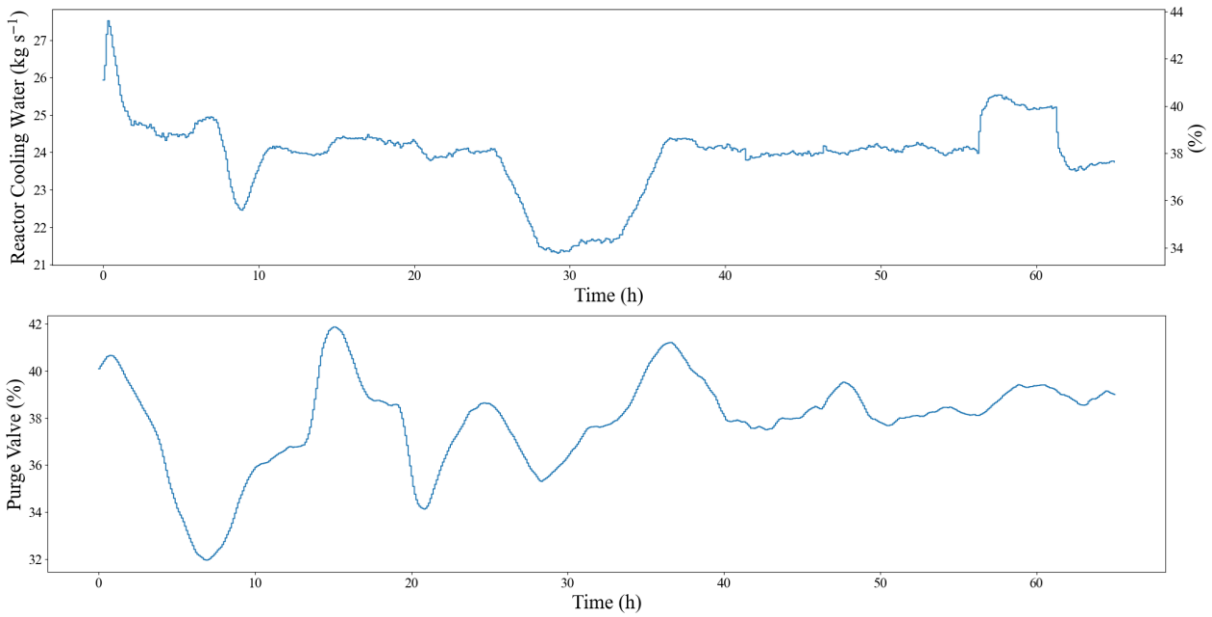


Figure 19 Control profile of reactor cooling water and purge valve of MSc-NMPC/MHE subjected to set-point changes and disturbances.

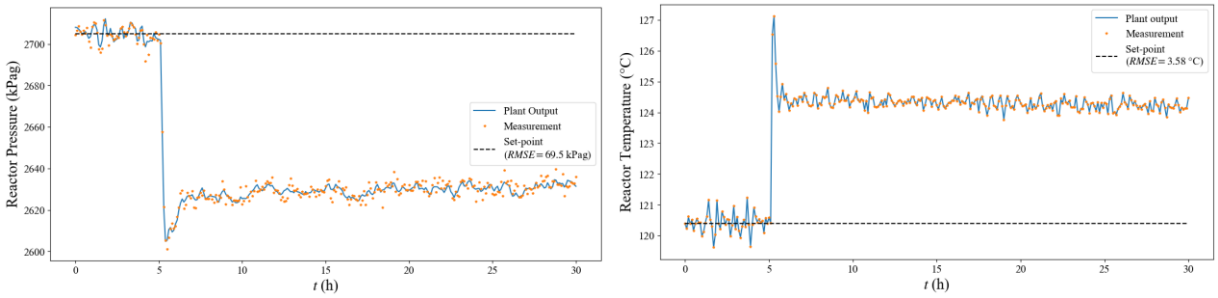


Figure 20 Reactor pressure and temperature response using NMPC/MHE subject to reactor cooling water inlet temperature disturbance.

3.5. Summary

This study presented challenges associated with the implementation of robust NMPC in the presence of uncertainty with EKF and MHE as state estimators, and showed the applicability of MS-NMPC and MSc-NMPC to a large-scale plant-wide problem like the TE. The two studied robust controllers were shown to prevent constraint violation in the process when model uncertainty may cause plant-model mismatch under different scenarios. However, when EKF was used as a state estimator, the mismatch between the EKF model and the plant caused the system to become destabilized because EKF made unrealistic estimates of the states. MHE was shown to resolve the problems of using EKF since it considers process constraints in its formulation, hence, even under uncertainty, MHE was able to find feasible state estimates. Robust NMPC and MHE led to an increase in the computation time required, compared to traditional NMPC and EKF; however, for the sampling time of the process, significant input delays are not expected for this plant.

4. Benchmarking Surrogate Embedding Strategies for NMPC

Complex non-linear systems, modelled by ODEs and PDEs, make NMPC computationally costly. As discussed in Chapter 2, NN surrogates have become a popular modelling approach for NMPC. In particular, various formulations of control-oriented PINNs have been proposed; a few have solved the optimal control problem by directly including the objective within the PINN loss function (Barry-Straume *et al.*, 2022; Mowlavi and Nabi, 2023). However, the solution is unique to initial conditions, and thus, cannot be used in feedback control because updating the initial condition requires retraining; thus, making this approach computationally expensive to perform in online feedback control. Other studies have proposed various NN structures, such as RNNs, LSTMs and CNNs, that map current states and control actions to states at the next sampling step (Chen, Shi and Zhang, 2019; Alhajeri *et al.*, 2022; Antonelo *et al.*, 2022; Gokhale, Claessens and Develder, 2022; Zhang, 2022; Zheng *et al.*, 2023). The NN model acts as a surrogate state-space model in NMPC. However, the representation of NN surrogates in a form that is amenable to NLP solvers is still an open research problem. Previous works have used sequential/shooting approaches to solve the surrogate NMPC problem (Antonelo *et al.*, 2022; Nicodemus *et al.*, 2022; Sanyal and Roy, 2023). Despite those efforts, optimization approaches where the modelling equations are solved simultaneously using direct transcription have become popular in NMPC (Wächter and Biegler, 2006; Biegler and Zavala, 2009; Biegler, 2021). There is a gap in the literature comparing different ways by which NN surrogates could be embedded in the NMPC framework using a simultaneous approach. To address this gap, two major embedding strategies are explored in this work: i) embedding the NN model as explicit algebraic constraints in the algebraic modelling language (AML), which uses the AD from the AML to provide the NLP solver with gradients, and ii) treatment of the surrogate model as an external function, which uses the AD from ML environment to evaluate the gradients. These NN surrogate embedding strategies are benchmarked

based on the computational performance and limitations of each approach. To generate the NN surrogates, a physics-informed approach has been considered in this work. Physics-informed NNs (PINNs) have become popular due to their ability to maintain fidelity to fundamental physics models while reducing the need for historical/process data (Raissi, Perdikaris and Karniadakis, 2019; Alhajeri *et al.*, 2022; Antonelo *et al.*, 2022; Hao *et al.*, 2023). Physics-Informed NN (PINNs) and physics-informed convolutional neural networks (PICNNs) were used as NN structures. Moreover, previous works have focused on ODE case studies involving a small number of system states (Antonelo *et al.*, 2022; Nicodemus *et al.*, 2022; Sanyal and Roy, 2023). In this work, the aim is to solve more complex PDE-constrained NMPC problems. Three non-linear benchmarking case studies of increasing complexity, based on PFRs modelled by PDEs, are used for illustration given that these are common problems in chemical engineering applications.

This study is structured as follows: Section 4.1 begins with a description of the class of problems considered in this study followed by the NN structure used to approximate the problem. Section 4.1.1 describes the structure of the PINNs and PICNNs used and their training. Section 4.1.2 describes the strategies to embed the NN surrogates within the NMPC framework. The benchmarking models are presented in section 4.2, and the computational times of each strategy and controller performance are compared in the results section, 4.3. Section 4.4 summarizes the major findings of this work.

4.1. Methods

The PDE-constrained NMPC problem, (74)-(80), poses a major computational challenge to NMPC (Biegler and Thierry, 2018; Christiansen and Jorgensen, 2018; Patrón and Ricardez-Sandoval, 2020a; Toffolo, Meunier and Ricardez-Sandoval, 2024). The state variables and control actions are denoted by $\boldsymbol{x}: \mathcal{T} \times \Omega \to \mathcal{X}$ ($\mathcal{X} \in \mathbb{R}^{N_{sv}}$) and $\boldsymbol{u}: \mathcal{T} \to \mathcal{U}$ ($\mathcal{U} \in \mathbb{R}^{N_u}$), respectively. $\mathcal{F}_{\boldsymbol{z}}$ denotes a

non-linear differential operator with respect to the spatial independent variables $\boldsymbol{z}$. The operator $\mathcal{B}$ denotes the boundary conditions whereas the operator $\mathcal{I}$ denotes the initial condition. The set $\mathcal{T}$ corresponds to the time domain, $\mathcal{T} = [t_0, t_f] \subset \mathbb{R}$; the set Ω specifies the spatial domain of the PDEs, $\Omega \subset \mathbb{R}^{N_z}$, where N_z is the number of spatial dimensions. The control actions $\boldsymbol{u}$ generally appear as boundary conditions. For example, the conditions at the inlet of a PFR or as forcing terms within the PDEs, $\mathcal{F}_{\boldsymbol{z}}$. The constraint (76) represents general, non-linear path constraints that may be imposed on the states and manipulated variables. The PDE constraints, equation (75), can be highly non-linear and due to the multidimensional/infinite dimensional nature (i.e., the PDEs can be functions of multiple continuous independent variables in the time and spatial dimensions), discretization methods can result in a large number of nonlinear constraints and variables. Consider a PFR system with five reactive components; it must be modelled by at least five PDEs if temperature effects are ignored. Assuming the problem is one-dimensional (axial direction), discretizing these PDEs with 50 spatial and 30 temporal finite elements results in at least 7500 (5×50×30) variables and constraints for the states alone. This excludes other supporting equations necessary to describe the system (e.g., reaction kinetics, equations of state, etc.), which would add 1500 (50×30) variables and constraints each. If the PFR also varies radially, the 7500 increases to 375000 (5×50×50×30), illustrating the complexity of PDE problems as the number of independent variables grows.

$$\min_{\boldsymbol{u}} \int_{\mathcal{T}\times\Omega} \left\| \boldsymbol{x_{sp}} - \boldsymbol{x} \right\|_{\boldsymbol{L}}^2 + \|\boldsymbol{u}\|_{\boldsymbol{W}}^2 \, d\boldsymbol{z}dt \tag{74}$$

$$\text{s.t. } \frac{\partial \boldsymbol{x}}{\partial t}(t, \boldsymbol{z}) = \mathcal{F}_{\boldsymbol{z}}(\boldsymbol{x}, \boldsymbol{u})(t, \boldsymbol{z}) \quad t \in \mathcal{T}, \boldsymbol{z} \in \Omega \tag{75}$$

$$g(\boldsymbol{x}, \boldsymbol{u})(t, \boldsymbol{z}) \leq 0 \tag{76}$$

$$\boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U \tag{77}$$

$$\boldsymbol{u}^L \leq \boldsymbol{u} \leq \boldsymbol{u}^U \tag{78}$$

$$\mathcal{B}(\boldsymbol{x}, \boldsymbol{u})(t, \boldsymbol{z}) = 0, \quad t \in \mathcal{T}, \boldsymbol{z} \in \partial\Omega \tag{79}$$

$$\mathcal{I}(\boldsymbol{x})(t, \boldsymbol{z}) = 0, \quad t \in \{t_0\}, \boldsymbol{z} \in \Omega \tag{80}$$

To make the problem (74)-(80) tractable, the direct transcription approach is to discretize the continuous domains of the PDEs modelling the system and approximate the differential terms in the PDEs with finite differences or collocation methods (Biegler, 2010). Thereby, the system of algebraic constraints that is generated is solved simultaneously by an NLP solver. The time domain can be discretized over the sampling interval $[0, \Delta t]$, such that $\mathcal{T}$ is approximated by the finite set $\hat{\mathcal{T}} = \{t_k : t_k = t_0 + k\Delta t, k \in \{0,1, \dots, P\}, t_f = P\Delta t\}$, where P is the prediction horizon. The spatial domain can also be discretized. For the sake of illustration, assume $\boldsymbol{z}$ is unidimensional, i.e., $\boldsymbol{z} \in \Omega \subset \mathbb{R}$, Ω can be approximated by the finite set $\hat{\Omega} = \left\{\boldsymbol{z}_v : \boldsymbol{z}_v \in \Omega, v \in \{1, \dots, N_{fe}\}\right\}$, where N_{fe} is the number of discretization points, i.e. number of finite elements considered in the spatial domain. Hence, the tensor $\boldsymbol{x}_{k,v} \in \mathbb{R}^{N_{sv}}$ represents the state variables of the PDE, equation (75), at time interval k and spatial location v. As depicted in equation (81), $\boldsymbol{x}_{k,v}$ approximates the solution of the PDE at the time t_k and spatial position $\boldsymbol{z}_v$. Similarly, $\boldsymbol{u}_k \in \mathbb{R}^{N_u}$ is the vector of control inputs at time t_k as shown in equation (82).

$$\boldsymbol{x}_{k,v} \approx \boldsymbol{x}(t_k, \boldsymbol{z}_v) \tag{81}$$

$$\boldsymbol{u}_k = \boldsymbol{u}(t_k) \tag{82}$$

The time and spatial derivatives in equation (75) can be approximated via finite difference or collocation (Cervantes and Biegler, 2008; Biegler, 2010, 2021; Pulsipher *et al.*, 2022). For example, the following shows a backward finite difference over the time domain:

$$\frac{\partial \boldsymbol{x}}{\partial t}(t_k, \boldsymbol{z}_v) \approx \frac{\partial \boldsymbol{x}_{k+1,v}}{\partial t} = \frac{\boldsymbol{x}_{k+1,v} - \boldsymbol{x}_{k,v}}{\Delta t} \tag{83}$$

Therefore, one arrives at the algebraic formulation (84)-(90) that approximates the PDE-constrained NMPC problem (74)-(80). The non-linear differential operator, $\mathcal{F}_{\boldsymbol{z}}$, is replaced by discretized approximation $\hat{\mathcal{F}}_{\boldsymbol{z}}$. Similarly, the boundary, $\mathcal{B}$, and initial condition, $\mathcal{I}$, operators are approximated by their discretized form, $\hat{\mathcal{B}}$ and $\hat{\mathcal{I}}$, respectively. The coefficients, $\eta_{k,v}$, are appropriately selected in a quadrature or sampling scheme to approximate the integral expression in the objective function shown in equation (74) (Pulsipher *et al.*, 2022).

$$\min_{\boldsymbol{u}_k} \sum_{k\in\{0,\dots,P\}} \sum_{v\in\{1,\dots N_{fe}\}} \eta_{k,v} \left(\left\| \boldsymbol{x}_{\boldsymbol{sp},k,v} - \boldsymbol{x}_{k,v} \right\|_{\boldsymbol{L}}^2 + \|\boldsymbol{u}_k\|_{\boldsymbol{W}}^2 \right) \tag{84}$$

$$\text{s.t. } \frac{\partial \boldsymbol{x}_{k+1,v}}{\partial t} = \hat{\mathcal{F}}_{\boldsymbol{z}}\left(\boldsymbol{x}_{k,v}, \boldsymbol{u}_k\right) \quad \forall k \in \{1,\dots,P\}, v \in \{1,\dots,N_{fe}\} \tag{85}$$

$$g\left(\boldsymbol{x}_{k,v}, \boldsymbol{u}_k\right) \leq 0 \quad \forall k \in \{1,\dots,P\}, v \in \{1,\dots,N_{fe}\} \tag{86}$$

$$\boldsymbol{x}^L \leq \boldsymbol{x}_{k,v} \leq \boldsymbol{x}^U \quad \forall k \in \{0,1,\dots,P\}, v \in \{1,\dots,N_{fe}\} \tag{87}$$

$$\boldsymbol{u}^L \leq \boldsymbol{u}_k \leq \boldsymbol{u}^U \quad \forall k \in \{0,1,\dots,P\} \tag{88}$$

$$\hat{\mathcal{B}}\left(\boldsymbol{x}_{k,v}, \boldsymbol{u}_k\right) = 0 \quad \forall k \in \{0,1,\dots,P\}, v \in \{0\} \tag{89}$$

$$\hat{\mathcal{I}}\left(\boldsymbol{x}_{k,v}\right) = 0, \quad \forall k \in \{0\}, v \in \{1,\dots,N_{fe}\} \tag{90}$$

The constraints represented by equation (85) are non-linear and the model discretization generates a problem with a large number of variables and constraints, which poses a major challenge for online feedback NMPC applications due to model inflation (i.e., the increase in problem size due to discretizing) and their corresponding computational costs. Alternatively, the PDE constraints can be replaced by an NN surrogate. PINNs are a particular approach that has gained popularity as a means to generate NN surrogates. However, PINNs, as originally formulated by Raissi et al (2017a, 2017b, 2019), are not amenable as surrogate models for NMPC. The solution to the PDE system found by the PINN is particular to a set of boundary and initial conditions, and to a control profile $\boldsymbol{u}(t)$ that is known *a priori*. Hence, to update the solution to new initial condition, like in closed-loop online feedback control, the PINN needs to be retrained which may become

computationally taxing for most chemical engineering systems of interest. Note that some previous works were able to solve a single instance of the optimal control problem by incorporating the objective function in the training loss (Barry-Straume *et al.*, 2022; Mowlavi and Nabi, 2023). However, their methods may also be computationally taxing for online feedback control because updating the initial condition requires retraining the PINN model. For this reason, the NN structure used in this work takes the current states, $\boldsymbol{x}_{k,v}$, and control actions, $\boldsymbol{u}_k$, as inputs and returns the states at the next sampling step, $\boldsymbol{x}_{k+1,v}$. This way, the model can accept changing initial conditions, $\boldsymbol{x}_{k,v}$, and control actions, $\boldsymbol{u}_k$, that can be propagated over the prediction horizon, P. Figure 21 depicts how the surrogate modelled by the neural network, NN_θ, propagates the initial state, $\boldsymbol{x}_{k,v}$, to future states in the prediction horizon, P, with changing control actions, $\boldsymbol{u}_k$, at each sampling step. This can be represented mathematically by equation (91). Since the PINN takes current states $\boldsymbol{x}_{k,v}$ and control actions $\boldsymbol{u}_k$ as inputs, instead of retraining the PINN every time the initial condition changes, the PINN is only trained once with several combinations of initial states $\boldsymbol{x}_{k,v}$ and control actions $\boldsymbol{u}_k$, which are randomly sampled from their feasible sets, $\boldsymbol{x}_{k,v} \in \mathcal{X}$ and $\boldsymbol{u}_k \in \mathcal{U}$, respectively. Accordingly, the same NN surrogate can be used to predict at every step of the prediction horizon as shown in Figure 21. It can be difficult to guarantee that the entire initial state space $\mathcal{X}$ and control action space $\mathcal{U}$ have been adequately captured in the PINN; unfortunately, there is no theoretical basis to guide the number of samples necessary (Van Waarde *et al.*, 2020; Daoutidis *et al.*, 2024). The *curse of dimensionality* is a well-known obstacle to general ML applications where the required training data grows exponentially with the dimensions of the inputs (Theodoridis and Koutroumbas, 2009). However, one can expect that as the data set sampled from $\mathcal{X}$ and $\mathcal{U}$ increases in size, the surrogate model will better capture the process dynamics (Antonelo *et al.*, 2022). Additionally, the use of test and validation data sets (i.e., data points sampled $\mathcal{X}$ and

$\mathcal{U}$ that were not used in the training) are important to verify that model does not overfit on the training data set.

$$\boldsymbol{x}_{k+1,v} = NN_\theta\left(\boldsymbol{x}_{k,v}, \boldsymbol{u}_k\right) \tag{91}$$

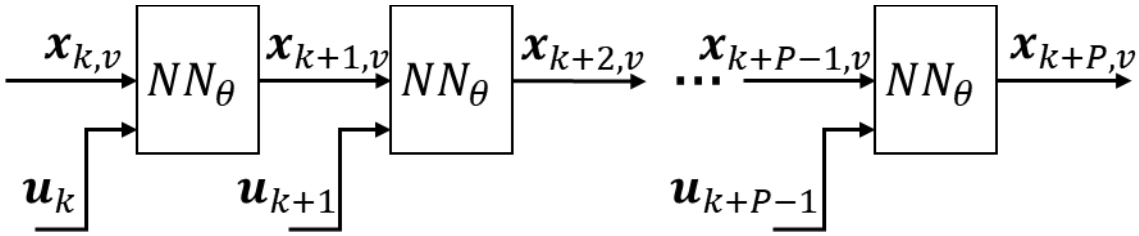


Figure 21 State propagation over the prediction horizon from the initial state and the control actions at each step in time.

Antonelo et al. (2022) have used a similar surrogate structure to represent ODE-constrained NMPC problems. They trained a PINN that accepts time, t, as a continuous input, the current states, $\boldsymbol{x}_k$, and the current control actions, $\boldsymbol{u}_k$. The PINN predicts the states at any time within the sampling interval $t \in [0, \Delta T]$ using a fully-connected feedforward deep NN (FNN). The output of the PINN can be provided as a new input to the NN model to make predictions over longer time horizons. However, they used a shooting/sequential approach to solve the NMPC problem. In a shooting method, the optimizer generates a sequence of control actions, $\{\boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \dots, \boldsymbol{u}_{k+M-1}\}$, which are passed to an integrator or model simulator to predict the future behaviour of the system. The prediction is used to evaluate the objective function and the constraints of the NMPC problem, which the optimizer then uses to update the control actions until it finds and optimal control sequence (Cervantes and Biegler, 2008; Biegler, 2021). Hence, the shooting method requires solving the system model at every iteration of the optimization algorithm. This is depicted in Figure 22a where it shows the interaction between the optimizer, the integrator and the NMPC objective and constraints. In the work by Antonelo et al. (2022), a Runge-Kuta (RK) integration scheme was

applied as the model simulator for the mechanistic model. In the case of the surrogate model, the NN was evaluated sequentially as in Figure 21 to represent the integrator or simulation model. The case studies were restricted to small ODE-constrained problems, involving the Van Der Pol oscillator with two states and a four-tank system with four states. Other works have expanded the work by Antonelo et al. (2022) to additional case studies, like a multilink manipulator with four states (Nicodemus *et al.*, 2022) and a quadrotor drone with 13 states (Sanyal and Roy, 2023). Their surrogate NMPC models were shown to have a slight improvement in the NMPC solution time compared to the mechanistic NMPC, using the shooting solution approach. However, this approach is yet to be tested on more complex problems like PDE-constrained NMPC. Moreover, direct transcription is a popular approach for solving NMPC problems (Lopez-Negrete *et al.*, 2013; Patrón and Ricardez-Sandoval, 2020b; Valipour and Ricardez-Sandoval, 2021a; Elorza Casas, Valipour and Ricardez Sandoval, 2023; Toffolo, Meunier and Ricardez-Sandoval, 2024). In direct transcription methods, the system modelling equations and the optimization tasks are performed simultaneously by fully discretizing the system of DAEs/PDEs over the independent variables, generating a system of algebraic constraints, like that shown in formulation (84)-(90) (Cervantes and Biegler, 2008; Biegler and Zavala, 2009; Biegler, 2021). This is depicted by Figure 22b which shows that the discretized system of PDEs/DAEs is treated as additional constraints in the optimization problem. Instead of treating the control actions as the only decision variables, like in the shooting methods, the state variables and the control actions are simultaneously considered as decision variables by the optimization solver, but the algebraic constraints from the discretized PDE/DAE system ensure the system dynamics are followed; for this reason, the optimization approach used to solve the direct transcription formulation is often referred to as simultaneous. Simultaneous approaches facilitate the efficient evaluation of first and second-order derivatives

through the AD from algebraic modelling languages (AMLs), the resulting problem is a large optimization program that can handle open-loop instability more reliably (Biegler, 2021). It is common to observe that each of the individual constraints that result from the discretization only includes a few of the optimization variables, so, the problem is often, but not always, sparse (i.e., the Jacobian and Hessian matrices are comprised of mostly zeros) which facilitates AD. Thus, this work aims to benchmark strategies that are amenable to simultaneous solution approaches to embed the NN surrogate model, equation (91), within the NMPC problem. Thereby, taking advantage of large-scale NLP solvers, like IPOPT, to solve the surrogate NMPC. As discussed, one potential approach is to embed the NN surrogate as algebraic constraints in the optimization problem, leveraging the AD in the AML to evaluate the gradients. Another approach is to treat the NN surrogate model as an external function, leveraging the efficient AD from the ML environment to evaluate the gradients. These two approaches are described in detail in section 4.1.2.

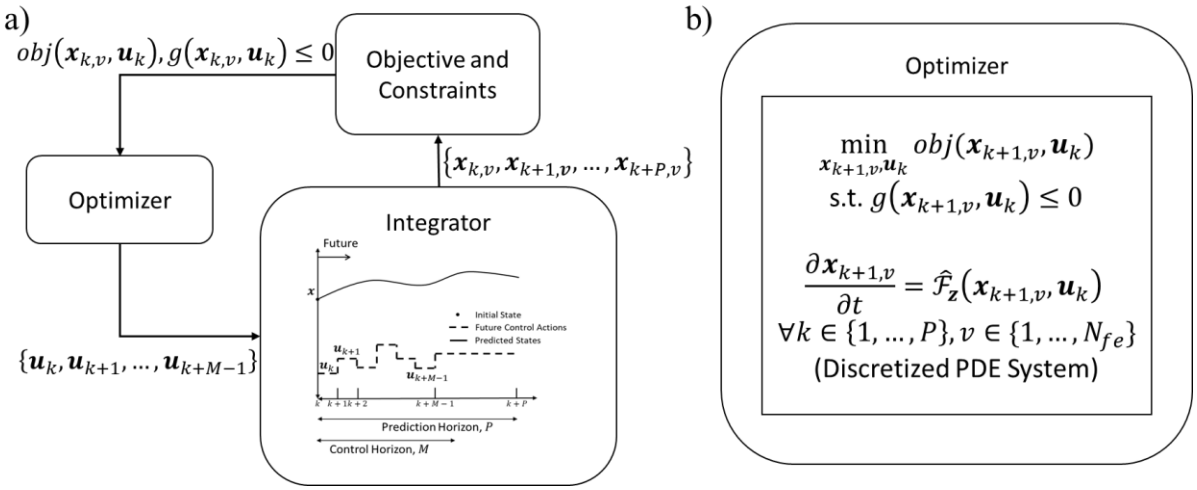


Figure 22 Dynamic optimization strategies; a) Sequential methods; b) Direct transcription.

As discussed, this work uses physics-informed methods to train NN surrogate models as a way to avoid relying on historical/process data (Raissi, Perdikaris and Karniadakis, 2019; Antonelo *et al.*, 2022; Hao *et al.*, 2023). FNNs and CNNs are used as neural network structures, referred to as

PINNs and PICNNs, respectively. PDEs are infinite-dimensional over the spatial and temporal domains (i.e., they are functions of continuous domains, time and space); thus, CNNs are a natural choice since they can capture the interaction of variables over spatial domains by providing a finite-dimensional approximation (i.e., they rely on discretizing the domain of the PDE) (Jiang and Zavala, 2021; Zhang, 2022). FNNs have been a typical structure of choice due to their simplicity (Raissi, Perdikaris and Karniadakis, 2019; Antonelo *et al.*, 2022; Gokhale, Claessens and Develder, 2022; Nicodemus *et al.*, 2022; Sanyal and Roy, 2023). A finite-dimensional approximation was also made with FNNs; however, FNNs can only handle unidimensional inputs. The input to the FNN must be flattened (i.e., made unidimensional) before evaluating the FNN model, eliminating the spatial relations between the variables. The next section discusses the method used to train the PINNs and PICNNs.

4.1.1. Physics-Informed Neural Networks

To incorporate the PDEs as physics-informed terms in the loss function of an FNN or a CNN, a discretization over the spatial and time domains is performed. Recall, the tensor $\boldsymbol{x}_{k,v} \in \mathbb{R}^{N_{sv}}$ represents the state variables of the PDE, equation (75), at time interval k and at spatial location v. Thus, $\boldsymbol{x}_{k,v}$ approximates the solution of the PDE at the time t_k and location $\boldsymbol{z}_v$ as seen in equation (81). Similarly, $\boldsymbol{u}_k \in \mathbb{R}^{N_u}$ is the vector of control inputs at time t_k as shown in equation (82). The time and spatial derivatives in equation (75) can be approximated by finite difference, like the example shown in equation (83) which demonstrated a backward finite difference over the time domain.

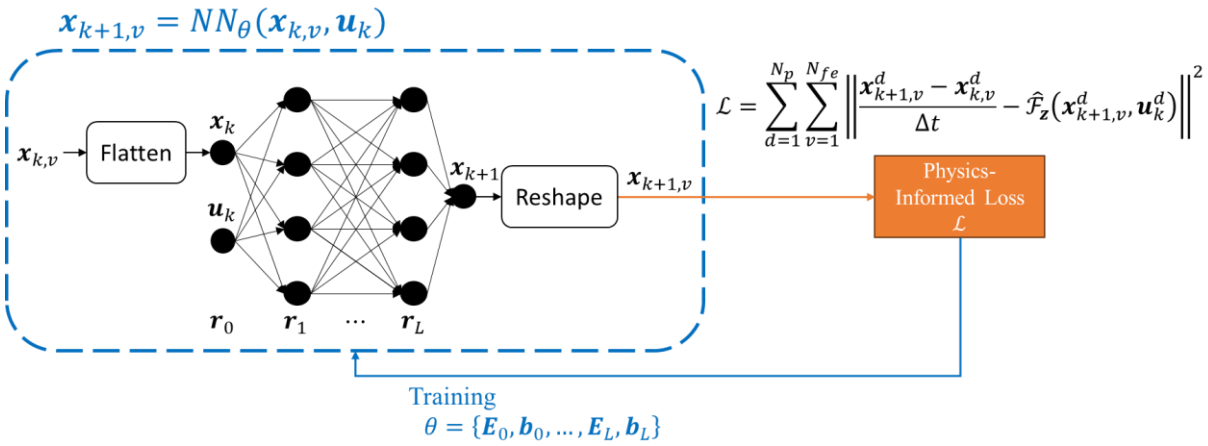


Figure 23 PINN model training framework.

To train the surrogate PINN or PICNN model, the initial conditions and control actions are randomly sampled from their feasible sets, $\boldsymbol{x}_{k,v} \in \mathcal{X}$ and $\boldsymbol{u}_k \in \mathcal{U}$ respectively, generating the data set $\left\{\boldsymbol{x}^d_{k,v}, \boldsymbol{u}^d_k\right\}_{d=1}^{N_p}$. The data is fed as inputs to the NN. The output, which is the prediction of the states at the next sampling step, $\boldsymbol{x}_{k+1,v}$, is used to evaluate the residuals of the discretized PDEs modelling the system. Then, the sum of the squared residuals becomes the loss function, $\mathcal{L}$, as described by equations (92) and (93), where $\boldsymbol{R}^d_v$ is the PDE residuals at data point d at spatial location v. This is depicted by Figure 23 where it shows that the NN_θ, parametrized by θ, is evaluated, and the outputs are used to evaluate the loss, $\mathcal{L}$. Furthermore, via backpropagation, the parameters of the model (the weights and biases in the case of PINNs or the convolution kernels in the case of PICNNs) are updated such that the loss function is minimized. Note that if process/historical data is available (i.e., $\left\{\left(\boldsymbol{x}^d_{k,v}\right)_{data}, \left(\boldsymbol{u}^d_k\right)_{data}, \left(\boldsymbol{x}^d_{k+1,v}\right)_{data}\right\}_{d=1}^{N_{data}}$), it could be added as an additional term in the loss function, as shown in equation (94).

$$\boldsymbol{R}^d_v = \frac{\boldsymbol{x}^d_{k+1,v} - \boldsymbol{x}^{\boldsymbol{d}}_{k,v}}{\Delta t} - \hat{\mathcal{F}}_{\boldsymbol{z}}\left(\boldsymbol{x}^d_{k+1,v}, \boldsymbol{u}^d_k\right) \quad \forall v \in \{1, \dots, N_{fe}\} \tag{92}$$

$$\mathcal{L} = \sum_{d=1}^{N_p} \sum_{v=1}^{N_{fe}} (\boldsymbol{R}_v^d)^2 \tag{93}$$

$$\mathcal{L}_{data} = \sum_{d=1}^{N_{data}} \sum_{v=1}^{N_{fe}} \left\| \left(\boldsymbol{x}_{k+1,v}^d\right)_{data} - NN_\theta \left(\left(\boldsymbol{x}_{k,v}^d\right)_{data}, \left(\boldsymbol{u}_k^d\right)_{data} \right) \right\|^2 \tag{94}$$

Note that FNNs can only accept one-dimensional inputs (i.e., vector inputs). Thus, before passing the input data to the PINN, the tensor $\boldsymbol{x}_{k,v}$ must be flattened along the state variable and spatial dimensions. Suppose $x_{c,k,v}$ is the c element of the state variables, $\boldsymbol{x}_{k,v}$ (c is the index for each dependent variable in the set of PDEs, i.e., $c \in \{1, \dots, N_{sv}\}$. For example, c can correspond to the concentration of a component, temperature, etc.), at time k and location v, then, the flattened input to the PINN would be the state vector, $\boldsymbol{x}_k$, defined in equation (95). The initial states and control actions can be passed to the NN as a single vector input as shown in equation (96). Each of the hidden FNN layers is evaluated according to equation (97), where $\boldsymbol{r}_l$ represents the output of the lth layer. From $\boldsymbol{r}_0$ to $\boldsymbol{r}_L$, where L is the number of hidden layers as displayed in the FNN depicted in Figure 23. Finally, the output of the last layer is the states at the next sampling step $\boldsymbol{x}_{k+1}$. This is shown in equation (98). The states at the next sampling step $\boldsymbol{x}_{k+1}$ can be reshaped back to $\boldsymbol{x}_{k+1,v}$. $\boldsymbol{E}_l$ and $\boldsymbol{b}_l$ are the weights matrix and bias vector at layer l, respectively. The term σ is the activation function. If σ is smooth, e.g., tanh or sigmoid functions, the trained NN can be formulated as continuous equality constraints in an NLP because the constraints shown in equation (97) could be differentiated to second-order derivatives with respect to $\boldsymbol{r}_l$ and $\boldsymbol{r}_{l+1}$.

$$\boldsymbol{x}_k = \left[x_{1,k,0}, \dots, x_{1,k,N_{fe}}, x_{2,k,0}, \dots, x_{2,k,N_{fe}}, \dots, x_{N_{sv},k,0}, \dots, x_{N_{sv},k,N_{fe}}\right]^T \tag{95}$$

$$\boldsymbol{r}_0 = [\boldsymbol{x}_k^T \quad \boldsymbol{u}_k^T]^T \tag{96}$$

$$\boldsymbol{r}_{l+1} = \sigma(\boldsymbol{E}_l \boldsymbol{r}_l + \boldsymbol{b}_l) \quad \forall l \in \{0, \dots, L-1\} \tag{97}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{E}_L \boldsymbol{r}_L + \boldsymbol{b}_L \tag{98}$$

In the case of PICNNs, it is not necessary to flatten the input, preserving the spatial relations between the input data. The state variable index, c, can correspond to the channel in a CNN, and the spatial index, v, to the CNN spatial dimension. The control inputs, $\boldsymbol{u}_k$, do not necessarily have a spatial dimension. Hence, the control input tensor can be made to have the same value along the spatial dimension as depicted in equation (99) and each channel, c, corresponds to a different manipulated variable. Then, the input to the first convolutional layer of a PICNN, $\boldsymbol{r}_{0,v} \in \mathbb{R}^{(N_{sv}+N_u)}$, is the concatenation of $\boldsymbol{x}_{k,v}$ and $\boldsymbol{u}_{k,v}$ over the channel dimension as shown in equations (100) and (101), where $r_{l,c,v}$ is the output of the l layer, channel c and spatial location v.

$$\boldsymbol{u}_{k,v} = \boldsymbol{u}_k \quad \forall v \in \{1, \dots, N_{fe}\} \tag{99}$$

$$r_{0,c,v} = x_{c,k,v} \quad \forall c \in \{1, \dots, N_{sv}\}\, \forall v \in \{1, \dots, N_{fe}\} \tag{100}$$

$$r_{0,c+N_{sv},v} = u_{c,k,v} \quad \forall c \in \{1, \dots, N_u\}\, \forall v \in \{1, \dots, N_{fe}\} \tag{101}$$

In this work, a multilayer CNN of the form displayed in Figure 24 was adopted, where, as described earlier, the state variable tensor $\boldsymbol{x}_{k,v}$ (green blocks) and manipulated variable tensors $\boldsymbol{u}_{k,v}$ (purple blocks) are concatenated to form the input to the first convolutional layer $\boldsymbol{r}_{0,v}$. The output of the 1-D convolution (Conv1d) is passed through the activation function σ, generating the output of the first convolutional block $\boldsymbol{r}_{1,v}$ (blue blocks). This structure repeats over the number of hidden layers, L, until $\boldsymbol{r}_{L,v}$ which is the output of the last convolutional block. Here, the output is flattened (i.e., $\boldsymbol{r}_{L,v}$ is made unidimensional), then passed through a dense/linear layer, of the form shown in equation (98), such that the final output, $\boldsymbol{x}_{k+1,v}$, has the same dimensions as $\boldsymbol{x}_{k,v}$. Figure 24 shows that the flatten block makes the output of the last convolutional later unidimensional, followed by

a dense/linear layer that reduces the size of the unidimensional output. Moreover, the reshape block returns the output $\boldsymbol{x}_{k+1,v}$ in the same dimensions as $\boldsymbol{x}_{k,v}$. A pooling layer, such as max pooling, was not included as this would lead to formulations that require binary variables when embedded in optimization programs. Additionally, pooling layers are generally used to extract features from the signal, resulting in signal size reduction (Jiang and Zavala, 2021). In this work, it is desired that the overall output of the PICNN conserves the signal size because $\boldsymbol{x}_{k+1,v}$ should have the same dimensions as $\boldsymbol{x}_{k,v}$. Hence, pooling layers that do not require binary variables, like average pooling, were also avoided. Like FNNs, smooth activations can easily be included in NLPs.

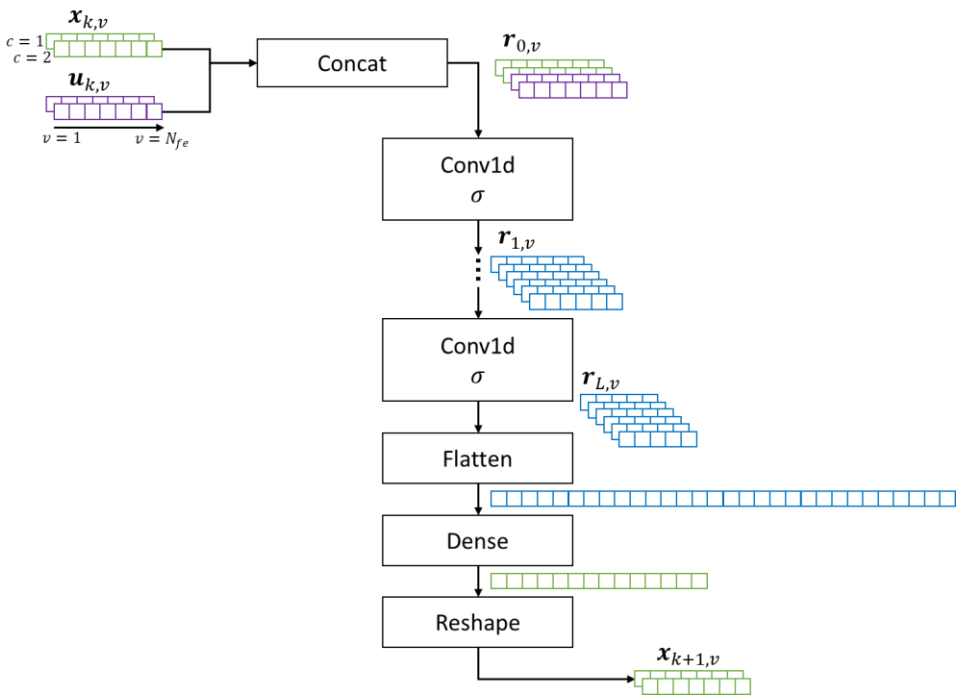


Figure 24 PICNN structure. The green blocks represent the state variable tensor; the purple block represents the control variable tensor; and the blue blocks represent the outputs of the internal/hidden layers of the CNN.

4.1.2. NMPC Surrogate Embedding

The PDE constraint NMPC formulation introduced in equations (74)-(80) must be modified to accommodate the surrogate process model. Either the PINN or the PICNN replaces the process

model, i.e., the PDE constraints shown in equation (75). This results in the formulation (102)-(109). The limits on the states and manipulated variables (constraints (105) and (106), respectively), must consider the domain over which the surrogate was trained. Recall that $\boldsymbol{x}_{k,v}^{d}$ and $\boldsymbol{u}_{k,v}^{d}$ were sampled from $\mathcal{X}$ and $\mathcal{U}$, respectively; hence, the trained model is unlikely to be accurate outside these sets.

$$\min_{\boldsymbol{u}_i^* \in \mathbb{R}^{N_u}\ \forall i \in \{k,k+1,\dots,k+P-1\}} \sum_{i=k+1}^{k+P} \left\| \boldsymbol{x_{sp}} - \boldsymbol{x}_i^* \right\|_{\boldsymbol{L}}^2 + \sum_{i=k}^{k+M-1} \|\Delta \boldsymbol{u}_i^*\|_{\boldsymbol{W}}^2 \tag{102}$$

$$\text{s.t. } \boldsymbol{x}_{i+1,v}^* = NN_\theta\left(\boldsymbol{x}_{i,v}^*, \boldsymbol{u}_i^*\right) \quad \forall i \in \{k, k+1, \dots, k+P-1\} \forall v \in \left\{1, \dots, N_{fe}\right\} \tag{103}$$

$$g\left(\boldsymbol{x}_{i,v}^*, \boldsymbol{u}_i^*\right) \quad \forall i \in \{k, k+1, \dots, k+P-1\} \forall v \in \left\{1, \dots, N_{fe}\right\} \tag{104}$$

$$\boldsymbol{x}^L \leq \boldsymbol{x}_{i,v}^* \leq \boldsymbol{x}^U \quad \forall i \in \{k, k+1, \dots, k+P\} \forall v \in \left\{1, \dots, N_{fe}\right\} \tag{105}$$

$$\boldsymbol{u}^L \leq \boldsymbol{u}_i^* \leq \boldsymbol{u}^U \quad \forall i \in \{k, k+1, \dots, k+P-1\} \tag{106}$$

$$\boldsymbol{u}_i^* = \boldsymbol{u}_{k+M-1}^* \quad \forall i \in \{k+M, k+M+1, \dots, k+P-1\} \tag{107}$$

$$\Delta \boldsymbol{u}_i^* = \boldsymbol{u}_i^* - \boldsymbol{u}_{i-1}^* \quad \forall i \in \{k, k+1, \dots, k+P-1\} \tag{108}$$

$$\boldsymbol{x}_{k,v}^* = \widehat{\boldsymbol{x}}_{k,v} \quad \forall v \in \left\{1, \dots, N_{fe}\right\} \tag{109}$$

The embedding of a surrogate model shown in equation (103) in a form that is amenable to simultaneous optimization approaches is not obvious. As discussed above, the simplest approach would be to directly transcribe the mechanistic modelling equations within the optimization problem by discretizing the differential equations over the independent variables. However, this results in a large-scale optimization problem that is non-linear with several states, which can be computationally expensive to solve online for large-scale systems such as those that involve PDEs. Two major strategies were considered in this work. Since NNs consist of a well-defined mathematical structure, they can be translated into mathematical expressions that can be included as explicit constraints in an NLP (only if he NN consists of smooth activation functions like tanh or sigmoid). The AD from the AML can then be used to evaluate the gradients, which the NLP solver uses to find the solution to the optimization problem. Alternatively, the NN model can be

evaluated as an external function, taking advantage of the efficient AD from ML packages to evaluate Jacobians and Hessians. Overall, three methods to solve the NMPC problem were implemented in this work, i.e., the two embedding strategies and the more traditional approach, which directly transcribes the complex non-linear mechanistic model. Figure 25 summarizes the different NN surrogate embedding strategies. Figure 25a shows that the NN surrogate can be converted into algebraic constraints and variables for an AML. Figure 25b shows that the surrogate NN model can be evaluated as an external function, directly from the ML environment, and provide the gradients to the NLP solver via the AD from the ML environment. Furthermore, Figure 25c shows that the mechanistic model, represented by DAEs, can be directly transcribed in the AML.

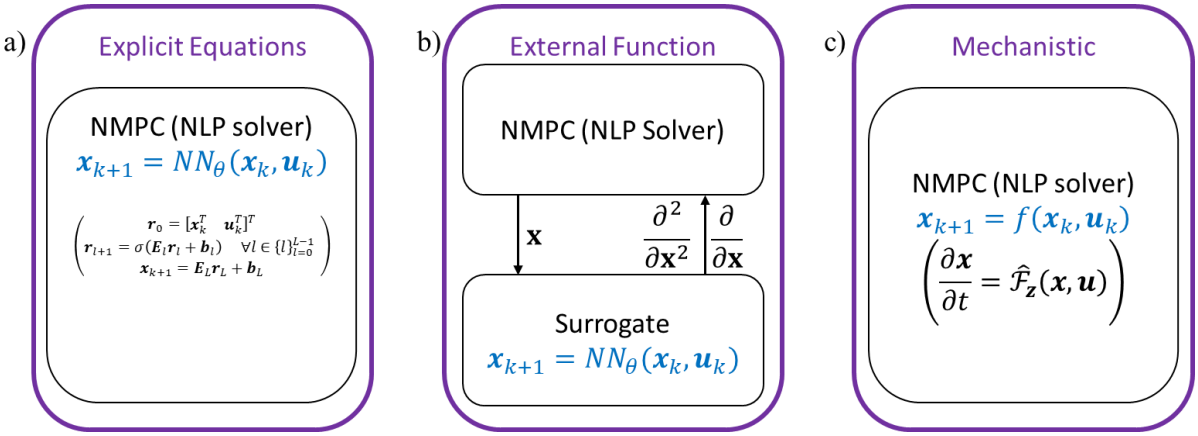


Figure 25 NN surrogate embedding strategies

The discussed embedding strategies fundamentally differ in the ways they perform AD. In general terms, AD refers to the evaluation of exact derivatives by following the symbolic derivative rules on function composition of elementary operations, i.e., product, summation, transcendental functions like exp and sin, etc. (Griewank and Walther, 2003; Baydin *et al.*, 2018). Note that AD is not symbolic differentiation, where the rules of differentiation are used to take a symbolic

expression to generate a new symbolic expression of the exact derivatives. AD instead keeps track of intermediate variables and their derivatives when a sequence of computations is performed to return an evaluation of the exact derivatives. AD can be performed using forward and reverse modes. In the forward mode, the gradients are accumulated forward from the input variables to the outputs by applying the chain rule. In the reverse mode, the derivatives are propagated backwards from a given output to the inputs. Backpropagation is a special case of reverse mode AD applied to neural networks, though the term is often used interchangeably with AD in ML (Baydin *et al.*, 2018). Both AMLs and ML environments usually have integrated AD for evaluating gradients. AMLs have generally been developed for the purpose of solving optimization problems, while ML environments have been developed for training ML models. In the case where the internal model of NMPC has been replaced with a NN surrogate model, it is not obvious which approach is more suitable. Therefore, this work compares two embedding strategies, one that uses the AD from an AML (i.e., embedding the surrogate model as algebraic constraints) and one the uses the AD from the ML environment (i.e., embedding the surrogate model as an external function). Section 4.1.2.1 describes the methods by which NN surrogates can be embedded as algebraic constraints in NMPC. Section 4.1.2.2 describes the approach where NN surrogates can be embedded as external functions.

A particular approach for directly transcribing the mechanistic model within NMPC is Pyomo DAE which supports model transformations that discretize the system of DAEs, using finite elements or collocation, to generate algebraic expressions that are included as constraints in an NLP (Nicholson *et al.*, 2018). This approach is used to create mechanistic NMPC models in this work. The implementation of the discussed embedding strategies is available on GitHub (https://git.uwaterloo.ca/ricardez_group).

4.1.2.1. Surrogate Embedding as Algebraic Constraints

Multiple optimization formulations have been proposed in the literature to embed FNNs as algebraic constraints in an AML. For FNNs with smooth activations, the full-space smooth (FS) and reduced-space smooth (RS) are the foremost approaches (Schweidtmann and Mitsos, 2019; Ceccon *et al.*, 2022) which are amenable to NLP solvers. Non-smooth activations, such as rectified-linear units (ReLU), can be expressed with BigM formulations (Anderson *et al.*, 2019), or partition-based formulations (Tsay *et al.*, 2021), that result in the generation of linear mixed integer constraints. Complementary formulations for ReLU networks can be included in NLPs (Yang, Balaprakash and Leyffer, 2022); however, NLP solvers struggle with complementarity constraints due to degeneracy. Thus, neither of these approaches is suitable for online feedback NMPC applications since it would involve solving mixed-integer programs or programs with complementarity constraints. The FS formulation of an FNN dense layer with activation, such as that shown in equation (97), requires the representation of every node in every layer by the following equations:

$$r'_{l+1,p} = \sum_{p'=1}^{N_{r,l}} E_{l,p,p'} r_{l,p'} + b_{l,p} \quad \forall l \in \{0, \dots, L-1\} \, \forall p \in \{1, \dots, N_{r,l+1}\} \tag{110}$$

$$r_{l+1,p} = \sigma(r'_{l+1,p}) \quad \forall l \in \{0, \dots, L-1\} \, \forall p \in \{1, \dots, N_{r,l+1}\} \tag{111}$$

where $r_{l,p}$ is the p element of the l layer output, $\boldsymbol{r}_l$.Then, $r'_{l,p}$ is $r_{l,p}$ before activation. $E_{l,p,p'}$ is the (p, p') entry of the weight matrix $\boldsymbol{E}_l \in \mathbb{R}^{N_{r,l+1} \times N_{r,l}}$ at layer l. $b_{l,p}$ is the p entry of the bias $\boldsymbol{b}_l \in \mathbb{R}^{N_{r,l+1}}$ of the l layer. $N_{r,l}$ is the number of nodes/neurons in layer l. For every neuron in every layer, two constraints and two variables are generated within the NLP. Thus, each layer generates $N_{r_l} \times 2$ constraints and auxiliary variables. The RS formulation reduces the total number of

variables and constraints. This is accomplished by directly substituting the outputs of a given layer to the next layer, from the input layer to the output layer. Even though this reduces the number of variables and constraints, it also results in very large expressions.

Convolutional layers use the discrete convolution operation to handle spatially dependent data. A 1-D convolutional layer with activation can be represented with a FS formulation via the following equations (Jiang and Zavala, 2021):

$$r'_{l+1,c,v} = \sum_{c'=1}^{N_{c,l}} \sum_{v'=1}^{N_{\kappa,l}} \kappa_{l,c,c',v'} r_{l,c',v+v'-1} + b_{l,c} \quad \forall l \in \{0,\dots,L-1\}\, \forall c \in \{1,\dots,N_{c,l+1}\}\, \forall v \in \{1,\dots,N_{v,l} - N_{\kappa,l} + 1\} \tag{112}$$

$$r_{l+1,c,v} = \sigma\left(r'_{l+1,c,v}\right) \quad \forall l \in \{0,\dots,L-1\}\, \forall c \in \{1,\dots,N_{c,l+1}\}\, \forall v \in \{1,\dots,N_{v,l+1}\} \tag{113}$$

The convolutional kernel for the input channel c' to output channel c in layer l is represented by $\boldsymbol{\kappa}_{l,c,c'} \in \mathbb{R}^{N_{\kappa,l}}$; where $N_{\kappa,l}$ is the kernel size. The output signal of layer l is represented by $\boldsymbol{r}_{l,v} \in \mathbb{R}^{N_{c,l}}$, where $N_{c,l}$ is the number of channels and $N_{v,l}$ is the signal size. Figure 26a shows a depiction of an illustrative example of a 1-D convolutional layer. As shown in this Figure, three stacked input signals (i.e., $c' \in \{1,2,3\}$, green blocks) with seven output signals (i.e., $c \in \{1,\dots,7\}$, blue blocks) are considered. Figure 26b shows the operation that occurs between the input channel $c' = 3$ and the output channel $c = 7$ (the orange connection in Figure 26a). This is known as the discrete convolution operation (Jiang and Zavala, 2021). This example uses a kernel $\boldsymbol{\kappa}_{l,c,c'}$ of size 3. The kernel moves forward one step at a time over the input signal to obtain the output signal. If the kernel moves multiple steps at a time, the number of steps is known as the stride. Note that the size of the output signal is smaller than the input. This is because convolution operation is not well-defined at the edges. Adding what is known as padding on both ends of the input signal can

prevent this signal reduction. Typically, this involves adding zeros to both ends of the input. Also, note that the operation shown in Figure 26 is only the summation over the spatial dimension v' shown in equation (112). There is one convolutional kernel for every input channel, $c' \in \{1,2,3\}$, going into output channel $c = 7$. Thus, to obtain the output channel $c = 7$, summation must be performed over the input channels. For a stride of one and no padding, the output signal size is related to the input signal size by $N_{v,l+1} = N_{v,l} - N_{\kappa,l} + 1$. Note that each layer generates $N_{c,l+1} \times N_{v,l+1} \times 2$ constraints and auxiliary variables.

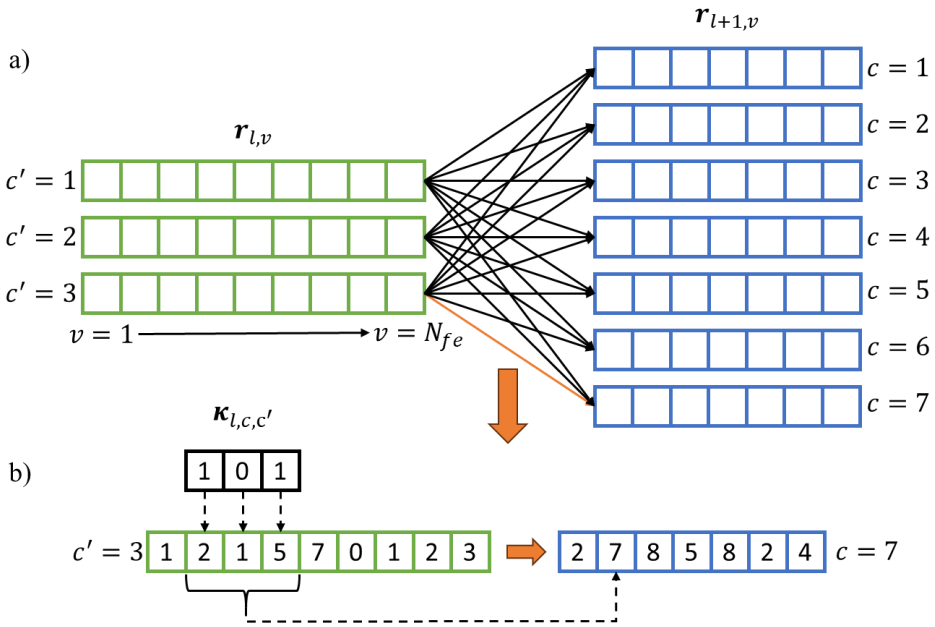


Figure 26 Depiction of 1-D convolutional layer, a), and discrete convolution operation, b).

The Optimization and Machine Learning Toolkit (OMLT) is a particular platform that supports several NN reformulations as constraints in the Pyomo optimization environments (Ceccon *et al.*, 2022). As shown in Figure 27, OMLT can read the trained model from an ML Package such as Pytorch (Ansel *et al.*, 2024). As discussed, the NN model follows a well-known mathematical structure. Therefore, a set of equality constraints and auxiliary variables express each node/neuron of the NN. OMLT creates Pyomo expressions where the inputs, outputs, and each neuron within

the hidden layers become variables within a larger optimization program. Therefore, OMLT is leveraged in this work to pursue this embedding method. OMLT supports the FS and RS formulation for FNNs; however, it only supports the FS formulation for CNNs.

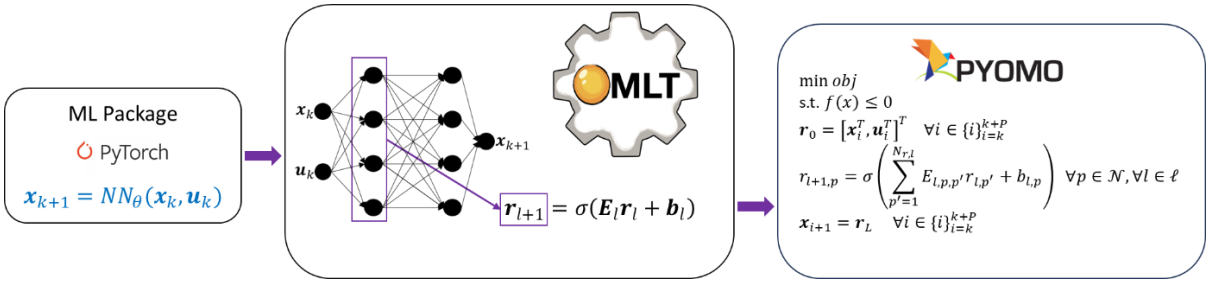


Figure 27 OMLT NN model transcription to Pyomo expressions.

4.1.2.2. External Function Surrogate Embedding

As an alternative to embedding the surrogate model as algebraic constraints, the surrogate model, and its gradients, can be evaluated as an external function. An external function can generally be represented in this form,

$$G(\mathbf{x}) = 0 \tag{114}$$

where $\mathbf{x} \in \mathbb{R}^{N_\mathrm{x}}$ are the decision variables of the optimization program and $G: \mathbb{R}^{N_\mathrm{x}} \to \mathbb{R}^{N_G}$ represents the external function that must return zeros at a feasible solution. The external function must then provide the evaluation of G, the Jacobian of G, i.e., $\frac{\partial}{\partial \mathbf{x}} G(\mathbf{x})$, and the Hessian of the dot product between the Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{N_G}$ and the corresponding constraints, i.e., $\frac{\partial^2}{\partial \mathbf{x}^2} \boldsymbol{\lambda}^T G(\mathbf{x})$. There is one Lagrange multiplier for every constraint in G, so the product $\boldsymbol{\lambda}^T G(\mathbf{x})$ is a scalar. The Lagrange multipliers assign the contribution of every constraint to the gradient of the optimization problem at optimality conditions. To embed a surrogate model using this approach, the constraints in equation (103) must be represented in the form of equation (114). One may place

the predicted states and future control actions within the decision variables, to represent the decision variables of the NLP, as depicted in equation (115). Then, the external function can be defined as in equation (116).

$$\mathbf{x} = \left[\boldsymbol{x}_k^{*T}, \dots, \boldsymbol{x}_{k+P}^{*T}, \boldsymbol{u}_k^{*T}, \dots, \boldsymbol{u}_{k+M-1}^{*T}\right]^T \tag{115}$$

$$G(\mathbf{x}) = \begin{bmatrix} \boldsymbol{x}_{k+1}^* - NN_\theta(\boldsymbol{x}_k^*, \boldsymbol{u}_k^*) \\ \vdots \\ \boldsymbol{x}_{k+P}^* - NN_\theta(\boldsymbol{x}_{k+P-1}^*, \boldsymbol{u}_{k+M-1}^*) \end{bmatrix} \tag{116}$$

PyNumero is a particular platform that facilitates the evaluation of external functions from the Pyomo optimization environment, which they refer to as Greybox Interface (Rodriguez *et al.*, 2023). The idea is to use the AD capabilities of ML environments, like Pytorch (Ansel *et al.*, 2024), to evaluate first and second-order derivatives. Figure 28 depicts the interaction between the PyNumero interface and the external model. PyNumero obtains the current guess of the decision variables from the NLP solver and passes it to the external model, e.g., Pytorch. The external model evaluates the model residuals, G, the Jacobian, $\frac{\partial}{\partial \mathbf{x}} G(\mathbf{x})$, and Hessian, $\frac{\partial^2}{\partial \mathbf{x}^2} \boldsymbol{\lambda}^T G(\mathbf{x})$, matrices. The model evaluations are passed back to PyNumero, which the NLP solver uses to update the decision variables. The main drawback of this approach is that the overhead required for the communication between the different Python packages and the conversion between different data formats is computationally taxing and is likely to increase the solution time compared to the other strategies.

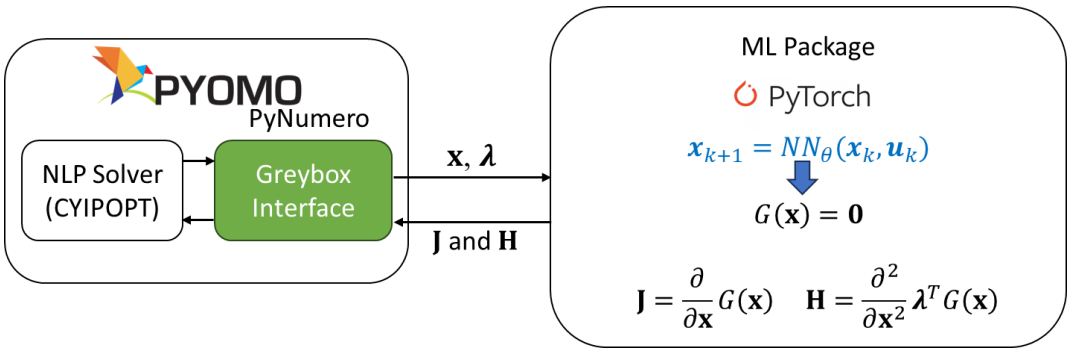


Figure 28 PyNumero interface interaction with external functions and Pyomo solvers.

Pytorch can make multiple evaluations of an NN or CNN model if the inputs are provided in a batch. Therefore, the constraints in (116) can be efficiently evaluated batch-wise. Pytorch provides two methods by which Jacobians and Hessians can be evaluated. The method *torch.autograd.functional* provides two functions, i.e., *jacobian* and *hessian*. These two functions evaluate the entries of the Jacobian and Hessian matrices element by element. Alternatively, the method *torch.func* provides the functions *jacrev* and *jacfwd* to evaluate Jacobians and the function *hessian*. This method uses a vectorized approach, hence, *jacrev* evaluates the Jacobian row by row and *jacfwd* column by column. The *hessian* function applies *jacrev* followed by *jacfwd*. Therefore, one can reduce the evaluation time by choosing a function that requires the least computations. In an optimization problem, one would expect to have some degrees of freedom, i.e. $N_{\mathrm{x}} > N_G$. In this case, the Jacobian has fewer rows than columns, hence, *jacrev* is expected to be faster than *jacfwd* for this particular problem.

4.2. Benchmarking Models

Three benchmarking models are used to demonstrate the challenges and limitations of the different surrogate embedding strategies described in the previous section. Benchmark 1 is a 1-D isothermal PFR; benchmark 2 is a non-isothermal 1-D PFR; benchmark 3 is a highly non-linear methane steam reforming PFR. The models of each of the case studies are described in the following

sections. The tests are performed on a Windows 10 computer equipped with 32 GB RAM, Intel® Core™ i9-10980HK CPU @ 2.40 GHz 3.10 GHz, NVIDIA RTX 2070s GPU 8GB VRAM. All models were trained with Pytorch 2.0.1 on the GPU using ADAM optimizer with an initial learning rate of 1.e−2 and exponential decay factor of 0.7 every 100 epochs.

4.2.1. Benchmark 1: Isothermal Plug-Flow Reactor

This model is meant to be a simple illustrative example. The PDE for benchmark 1 is shown in equation (117); where C is the concentration, V the reactor volume and k_{rxn} the reaction rate constant. For this problem, the outlet concentration, C_{out}, was selected as the controlled variable. The inlet concentration, C_{in}, and the volumetric flow rate, F, were the manipulated variables. The boundary condition is given by the inlet concentration as seen in equation (118). The initial condition is given by the equation (119), where C_0 represents a concentration profile as a function of the spatial dimension V. The sampling period was selected to be 0.1 s. Table 9 summarizes the nominal values and bounds around the model variables and parameters.

$$\frac{\partial C}{\partial t} = -F\frac{\partial C}{\partial V} - k_{rxn}C^2 \tag{117}$$

$$C(t,V) = C_{in}(t) \quad V = 0 \tag{118}$$

$$C(t,V) = C_0(V) \quad t = t_0 \tag{119}$$

For benchmark 1, the PINN consisted of a structure involving six hidden layers with 24 neurons at each hidden layer. The PDE is discretized into 10 finite elements over the spatial domain, therefore, the PINN has 12 inputs (10 states and 2 manipulated variables), and 10 outputs. The PICNN has a structure with 32 hidden channels, two hidden layers and a kernel size of 4, with 3 input channels (1 state and 2 manipulated variables, i.e., $N_{sv} = 1$, $N_u = 2$) and 1 output channel.

The input and output channels have a size of 10 spatial elements (i.e., $N_{fe} = 10$), so the spatial location v goes from 1 to 10. Figure 29 shows the training curves of PINN and PICNN for benchmark 1. As shown in equation (93), the loss function measures model error from the mechanistic process model; and the NN parameters are optimized such that this function is minimized. The PINN model (Figure 29a) and PICNN model (Figure 29b) both converge to low loss values. Recall that it can be difficult to guarantee that the sampled training data fully captures all possible combinations of initial states and control action. when testing the size of the random sample, it was observed that increasing the size of the sampled data does not further improve accuracy. In addition, the training curve and the test curve follow similar trajectories in the graphs shown in Figure 29. This suggests that the surrogate NN models are generalizing to data points that it does not observe during training and is not overfitting on the training data set.

Table 9 Parameters and variable bounds for benchmark model 1.

Parameter or Variable	Nominal Value	Lower Bound	Upper Bound
k_{rxn} (L mol^{-1} s^{-1})	1.0	--	--
V (L)	--	0.0	1.0
F (L s^{-1})	1.0	0.0	1.0
C_{in} (mol L^{-1})	0.5	0.1	1.0
C_0 (mol L^{-1})	0.5	0.1	1.0

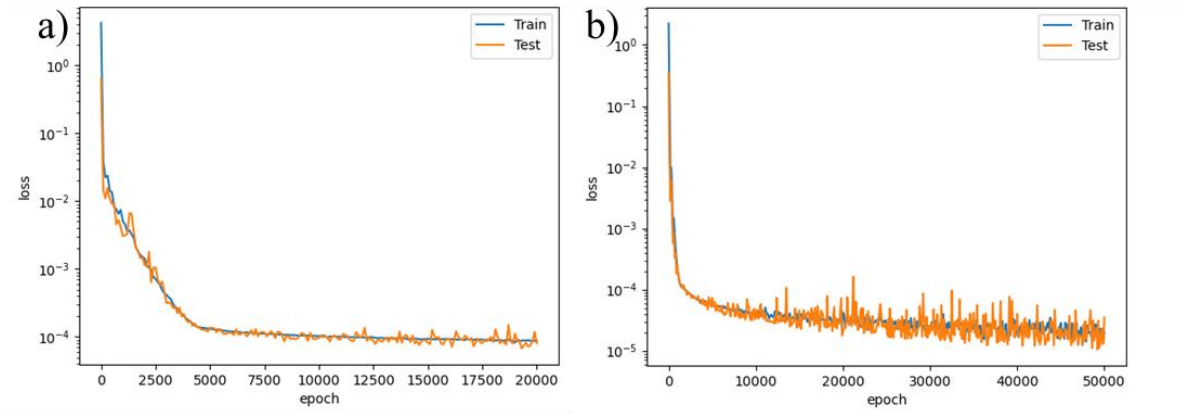


Figure 29 PINN, a), and PICNN, b), training curves for benchmark 1.

4.2.2. Benchmark 2: Non-isothermal Plug Flow Reactor with Heat Exchange

Benchmark 2 is modelled by a set of two PDEs: a mole balance, equation (120), and an energy balance, equation (121). The reaction rate constant, k_{rxn}, is sensitive to temperature and it is modelled by the Arrhenius equation, (122). The controlled variables for this problem are the outlet concentration, C_{out}, and outlet temperature, T_{out}. The manipulated variables are the inlet concentration, C_{in}, the inlet temperature, T_{in}, the exchanger heating temperature, T_a, and the volumetric flow rate, F. The boundary conditions are given by the inlet concentration and temperature as shown in equation (123). The initial conditions are given by equation (124), where C_0 and T_0 represent concentration and temperature profiles, respectively, as a function of the spatial dimension V. The sampling period was selected to be 100 s. Table 10 summarizes the model parameters, the variables, and the bounds of each variable which are adapted from a textbook problem (Fogler, 2016).

$$\frac{\partial C}{\partial t} = -F\frac{\partial C}{\partial V} - k_{rxn}C^2 \tag{120}$$

$$\rho C_p \frac{\partial T}{\partial t} = -\rho C_p F\frac{\partial T}{\partial V} + Ua(T_a - T) + k_{rxn}C^2(-\Delta H) \tag{121}$$

$$k_{rxn} = k_0 \exp\left(-\frac{E_A}{RT}\right) \tag{122}$$

$$C(t,V) = C_{in}(t), T(t,V) = T_{in}(t) \quad V = 0 \tag{123}$$

$$C(t,V) = C_0(V), T(t,V) = T_0(V) \quad t = t_0 \tag{124}$$

Table 10 Parameters and variable bounds for benchmark model 2.

Parameter or Variable	Nominal Value	Lower Bound	Upper Bound
k_0 (L mol^{-1} s^{-1})	47.9E6	--	--
E_A (J mol^{-1})	65730	--	--
ΔH (J mol^{-1})	-34.5	--	--
ρC_p (J L^{-1} K^{-1})	295.7	--	--
Ua (J L^{-1} K^{-1})	1.389	--	--
V (L)	--	0.0	5.0E3
F (L h^{-1})	2.125	1.062	4.249
T_a (K)	315	300	320
C_{in} (mol L^{-1})	1.86	0.2	1.9
C_0 (mol L^{-1})	1.86	0.2	1.9
T_{in} (K)	305	300	320
T_0 (K)	305	300	320

For benchmark 2, a PINN that could achieve a low enough loss could not be trained such that either the explicit algebraic formulation or the external function formulation could find feasible solutions when the model was embedded as the NMPC surrogate. The resulting loss function value was around 1e−2 at the lowest. Nevertheless, a PICNN model was still trained for benchmark 2. Recall that this work aims to benchmark the surrogate embedding methods in NMPC. It may be possible to find a PINN structure that improves the model predictions by varying the FNN hyperparameters (i.e., number of layers, neurons per layer, training algorithm, etc.); this is beyond the scope of this work. In this work, a trial-and-error approach is employed to determine the

hyperparameters until a structure that results in a satisfactory loss function value is found. The PICNN has 32 hidden channels, two hidden layers and a kernel size of 4. Benchmark 2 is also discretized into 10 spatial finite elements; thus, the input is a tensor of dimensions 6×10, i.e., 6 input channels (6 corresponds to 2 state variables, i.e., the concentration and temperature of the PFR, and 4 manipulated variables), 2 output channels and 10 spatial elements. Thus, in this case, $N_{sv} = 2$, $N_u = 4$ and the spatial location v goes from 1 to 10. Figure 30 shows the PICNN training curve for benchmark model 2, which indicates that an acceptable loss value is achieved at the end of training.

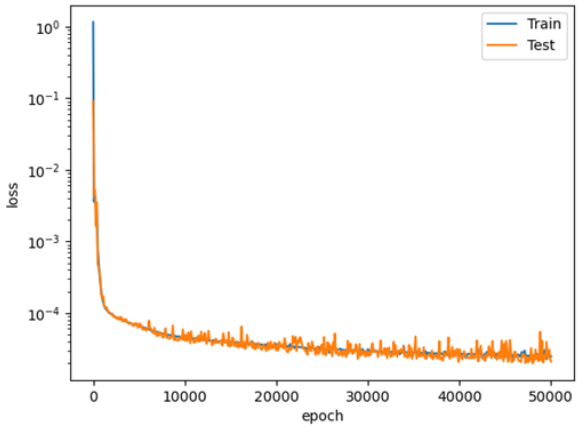


Figure 30 PICNN training curve for benchmark 2.

4.2.3. Benchmark 3: Steam Reformer PFR

The steam reforming process consists of the following three gas-phase reactions:

$$CH_4 + H_2O \rightleftharpoons CO + 3H_2 \quad (125)$$
$$H_2O + CO \rightleftharpoons H_2 + CO_2 \quad (126)$$
$$CH_4 + 2H_2O \rightleftharpoons CO_2 + 4H_2 \quad (127)$$

This process requires five-mole balances depicted in equation (128). The mole fraction of each component, y_s, is related to the molar flow rate, F_s, via equation (129), from which one can

calculate the partial pressure of each component P_s via equation (130). It was assumed that the pressure remained constant throughout the length of the reactor. Additionally, all components were assumed to behave as ideal gases. Hence, the molar concentration of each component C_s follows from equation (131). The equations modelling the reaction kinetics and thermodynamics are shown in Appendix B. The kinetic parameters are obtained from Xu and Froment (1989). The thermodynamic parameters are obtained from Yaws et al. (1999). The reaction rates RR_j' are highly non-linear functions of the partial pressures of the reactants and the temperature. The manipulated variables for this problem are the inlet flow rates of hydrogen and methane, $F_{\mathrm{H}_2,in}$ and $F_{\mathrm{CH}_4,in}$ respectively, and the reactor temperature, T. The controlled variables are the outlet flow rates of hydrogen, methane, and carbon monoxide, $F_{\mathrm{H}_2,out}$, $F_{\mathrm{CH}_4,out}$ and $F_{\mathrm{CO},out}$, respectively. The boundary conditions are given by the inlet flow rate of each component as in equation (132). The initial conditions are given by equation (133), where $F_{s,0}$ represents the flow rate profile of each component, s, as a function of the reactor length z. The sampling period was selected to be 0.004 s. Table 11 summarizes the reactor specifications and variable bounds. The initial states were restricted to an upper and lower limit on the flow rate on each component, i.e., $F_{s,0}^{L}(z) \leq F_{s,0}(z) \leq F_{s,0}^{U}(z)$. The bounds are shown in Figure B.1 of Appendix B.

$$\frac{\partial C_s}{\partial t} = -\frac{1}{A}\frac{\partial F_s}{\partial z} + \rho_c \sum_{j=1}^{3} \nu_{js} RR_j' \quad \forall s \in \{\mathrm{CH_4}, \mathrm{H_2O}, \mathrm{H_2}, \mathrm{CO_2}, \mathrm{CO}\} \tag{128}$$

$$y_s = \frac{F_s}{\sum_s F_s} \quad \forall s \in \{\mathrm{CH_4}, \mathrm{H_2O}, \mathrm{H_2}, \mathrm{CO_2}, \mathrm{CO}\} \tag{129}$$

$$P_s = y_s P_{tot} \quad \forall s \in \{\mathrm{CH_4}, \mathrm{H_2O}, \mathrm{H_2}, \mathrm{CO_2}, \mathrm{CO}\} \tag{130}$$

$$C_s = \frac{P_s}{RT} \quad \forall s \in \{\mathrm{CH_4}, \mathrm{H_2O}, \mathrm{H_2}, \mathrm{CO_2}, \mathrm{CO}\} \tag{131}$$

$$F_s(t, z) = F_{s,in}(t) \quad z = 0, \forall s \in \{\mathrm{CH_4}, \mathrm{H_2O}, \mathrm{H_2}, \mathrm{CO_2}, \mathrm{CO}\} \tag{132}$$

$$F_s(t, z) = F_{s,0}(z) \quad t = t_0, \forall s \in \{\mathrm{CH_4}, \mathrm{H_2O}, \mathrm{H_2}, \mathrm{CO_2}, \mathrm{CO}\} \tag{133}$$

Table 11 Variable bounds and reactor specifications for benchmark model 3.

Parameter or Variable	Nominal Value	Lower Bound	Upper Bound
$F_{CH_4,in}$ (mmol s^{-1})	3.11	1.56	6.22
$F_{H_2,in}$ (mmol s^{-1})	9.33	4.67	18.67
T (K)	848	838	858
z (cm)	--	0.0	2.2
A (cm^2)	0.785	--	--
ρ_c (g cm^{-3})	2.835	--	--
P_{tot} (atm)	10.0	--	--

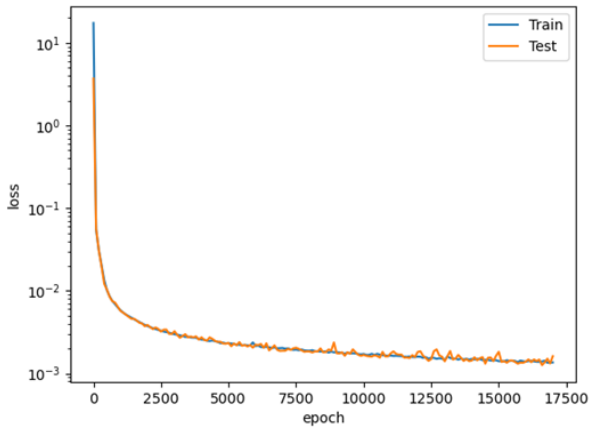


Figure 31 PICNN training curve for benchmark 3.

Similar to benchmark 2, a PINN model that achieved a low enough loss function could not be trained such that the surrogate NMPC model could find a feasible solution. However, a PICNN was still trained. The PICNN for benchmark 3 has 32 hidden channels, 3 hidden layers and a kernel size of 4. Benchmark 3 is discretized into 50 spatial elements; thus, the input tensor has dimensions of 8×50, i.e., 8 input channels (8 corresponds to 5 state variables and 3 manipulated variables), 5 output channels and 50 spatial elements. Thus, in this case, $N_{sv} = 5$, $N_u = 3$ and the spatial location v goes from 1 to 50. Figure 31 shows the training curve of PICNN for benchmark 3. The

loss converges to a value that was still relatively high at the end of the training, thus, some plant/model mismatch is expected between the PICNN and the mechanistic plant model.

4.3. Results

This section presents the results of embedding the physics-informed surrogate models within the NMPC framework. The different strategies are compared in terms of the computational time required to solve the NMPC problem, the accuracy of the methods compared to the mechanistic models, and the challenges in implementation, e.g., overhead involved in the interaction between different packages, the size of the optimization problem, etc. The surrogate NMPC model is connected in closed-loop with the plant (simulated by the mechanistic model) to measure the NMPC computational time at every sampling step. Figure 32 depicts this by showing that NMPC uses the surrogate as its internal plant model. First, the computational time to solve the NMPC problem using different embedding methods is compared for each of the benchmarking models. The formulations compared are the full-space (FS), reduced-space (RS), external function method 1 (EF1), external function method 2 (EF2) and the mechanistic model. EF1 uses *torch.func* to evaluate the Jacobian and Hessian, which uses the vectorized approach. EF2 uses *torch.autograd.functional*, which evaluates the Jacobian and Hessian element by element. Additionally, the shooting optimization approach was also implemented for benchmark model 1 to compare against the direct transcription method. The NMPC solution times are compared in section 4.3.1. The accuracy of the surrogate models is evaluated by comparing the closed-loop solution found using NMPC with a surrogate as its internal model against NMPC with the mechanistic model (ground truth). In each closed-loop simulation, the process is allowed to run for a few sampling steps, until a step set-point change is introduced for each benchmark. Table 12 summarizes the set-point changes implemented on each benchmark model and the sampling step

at which the set-point change is introduced. The focus of this study is to compare the various surrogate embedding strategies and not model training; however, this serves to ensure that the solution found by the surrogate NMPC is independent of the embedding strategy. The model accuracy is compared in section 4.3.2.

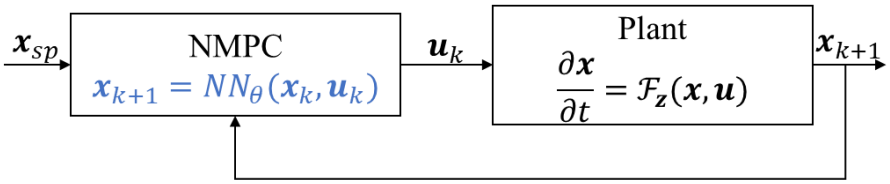


Figure 32 Closed-loop framework where NMPC uses a surrogate as the internal model for prediction.

Table 12 Set-point changes implemented in closed-loop simulations and sampling step of change.

Benchmark	Controlled-Variable	Set-point Change Time Step (k)	Initial Set-point	Set-point Change
1	C_{out} (mol L^{-1})	50	0.4	0.3
2	C_{out} (mol L^{-1})	100	570	760
	T_{out} (°C)	50	315	312
3	$F_{H_2,out}$ (mmol s^{-1})	15	5.69	6.69

NMPC problems formulated using OMLT were solved on Pyomo 6.5.0 with Python 3.9 using IPOPT 3.11.1. OMLT was used to implement both the FS and RS formulations. For the sake of illustration, NMPC problems formulated using the mechanistic model of the process were also solved on the same versions of Pyomo, Python and IPOPT, using Pyomo DAE. Due to compatibility issues, NMPC problems formulated using the external function approach were solved on Pyomo 6.4.2 with Python 3.8 using the IPOPT Cython wrapper, CYIPOPT 1.1.0, which uses IPOPT 3.14.9. The PyNumero package version was 1.3. The Pytorch version used to evaluate the external model was 2.2.1. Since IPOPT runs on the CPU, the external function method does

not benefit from the parallel computing capabilities of GPUs. Hence, the external models were evaluated on the CPU. The shooting method is implemented using the Scipy optimizer *fmin_slsqp*, which utilizes a sequential least-squares quadratic programming approach (Virtanen *et al.*, 2020). The integrator is implemented via the method of lines, where the PDE is discretized over the spatial domain via finite differences to generate a set of ODEs with respect to time. Then, integration was performed with respect to time via the Runge-Kutta method of order 5(4) (RK45) (Dormand and Prince, 1980; Virtanen *et al.*, 2020). Note that the sequential method implemented is only meant for illustration. Its performance may vary depending on the optimization algorithm, the precision of the integrator and integrator algorithm, and the used of more sophisticated shooting approaches, e.g., multiple-shooting (Biegler, 2010).

4.3.1. Computational Costs

The first sampling step at which NMPC is solved often takes longer as the initial guess provided to the solver may not be close to the optimal solution. At subsequent sampling steps, the solver can use the solution found at the previous sampling step to update the initial guess and reduce the computational time. Hence, both the time required to solve the first iteration and the average over all simulated sampling steps were reported. Figure 33 shows the time required to solve the NMPC problem at every sampling step for FS (blue) and EF1 (orange) using PICNN, and the mechanistic Pyomo model (green) for benchmark 1. As shown in this figure, in all embedding methods, the first iteration takes the longest to solve and the following iterations take less time. Whenever a change is implemented in the system, e.g., a set-point change, there is a jump increase in the computational time as shown at sampling step 50 in Figure 33 (vertical, red-dashed line).

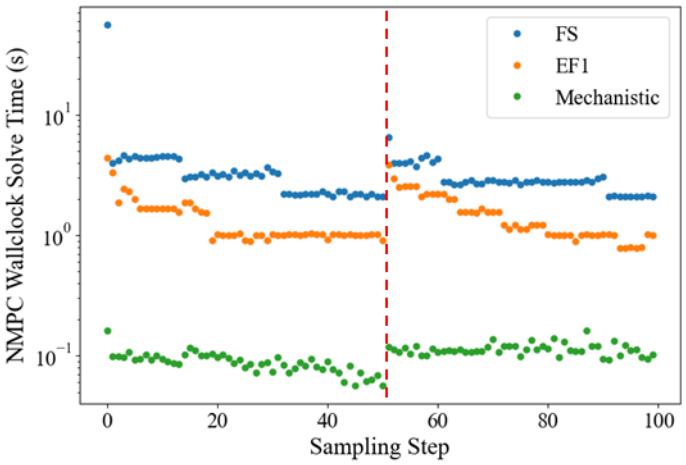


Figure 33 NMPC wallclock solve time at each sampling step for FS, EF1 using PICNN, and the mechanistic Pyomo model in benchmark 1.

Figure 33 highlights the general pattern observed in all case studies. Using a surrogate did not lead to improved solution time compared to using the mechanistic process model, regardless of the embedding method. NN surrogates generally lead to larger models due to the use of hidden layers, which is a possible reason why the use of a surrogate model does not lead to improvements. It may also be the case that more complex models, e.g., those involving high-order derivatives, could be more suitable for this modelling approach. The PFR models that were used as benchmarks only involved advection terms; dispersion/diffusion terms introduce higher-order derivatives that involve more complex boundary conditions. Table 13 shows the average wallclock time to solve the NMPC problem over all sampling steps, the wallclock time to solve the first iteration and the number of variables in the NLP formulation for all benchmarks for all embedding strategies. As depicted in this table, the pattern, where the mechanistic Pyomo model is the fastest, holds for benchmarks 2 and 3. Antonelo et al. (2022) used a shooting approach to solve the NMPC problem. They reported slight advantages when a surrogate NMPC model was used over the mechanistic model. However, in this work, it was observed that the simultaneous approach (i.e., direct transcription) generally led to significantly shorter solution times than the shooting approach. The

simultaneous approach usually led to an order of magnitude improvement in NMPC solution time. Looking at benchmark 1, the mechanistic model implemented in Pyomo DAE had an average solution time of 0.101 s, while the mechanistic model implemented using the shooting method had an average solution time of 1.284 s. Similarly, the EF1 using the PINN surrogate had an average solution time of 0.402 s, meanwhile, the shooting method solved in 4.09 s. Hence, for the algorithms and methods implemented, the simultaneous optimization approaches are shown to be more computationally efficient. However, note that more sophisticate shooting methods have been developed (Biegler, 2010; Soledad Aronna, Frédéric Bonnans and Martinon, 2013; Assassa and Marquardt, 2014; Aydin, Bonvin and Sundmacher, 2017; Andrés-Martínez, Biegler and Flores-Tlacuahuac, 2020); their implementation is beyond the scope of this work. The observed advantage of simultaneous methods is likely because the shooting method requires solving the modelling equations at every iteration of the optimization algorithm, which increases the number of computations. Additionally, the optimization algorithm used in the shooting method must estimate the Jacobian of the optimization problem via numerical methods since it does not have direct access to the model.

Moreover, if one is to use a surrogate, EF1 generally leads to shorter NMPC solution times compared to FS or RS formulations. This is likely because explicit equation formulations require auxiliary variables and constraints to represent the internal layers of the NN. Hence, it is difficult to provide a good initialization for the auxiliary variables. The external function method, on the other hand, only needs to consider the inputs and outputs of the NN as decision variables. In benchmark 1, for example, the FS formulation using a PINN model has 15120 variables, while EF1 only has 430 variables. The only exception is benchmark 2 where EF1 took slightly longer to solve than FS on average (9.586 s and 7.160 s, respectively). However, the first iteration of EF1

was still significantly shorter than FS, likely because the auxiliary variables are farthest from a feasible solution at this iteration. This was observed even though the external function method is impacted by the overhead necessary to communicate the different Python packages and the data format conversions. For example, PyNumero provides the decision variables to Pytorch as a Python list, which must be converted to a Pytorch tensor object to evaluate the model. The Jacobian and Hessian must be converted from Pytorch tensors to Scipy sparse arrays when returned to PyNumero. Furthermore, EF2 always took longer than EF1; EF2 evaluates the Jacobian and Hessian element by element, which significantly increases the number of computations, and as a result, the evaluation time. EF1 utilizes a vectorized approach, evaluating the matrices row by row or column by column, hence, a reduction in the computational time was observed.

Table 13 Summary of computational time required to solve the NMPC problem for different embedding strategies in all benchmark models, and the number of variables in the NLP formulation.

Benchmark	Embedding Method	Model Type	Ave. Wallclock Time (s)	First Iteration Wallclock Time (s)	Number of Variables
1	FS	PINN	0.692	5.832	15120
	RS	PINN	0.666	1.108	2320
	EF1	PINN	0.402	1.89	430
	EF2	PINN	2.219	10.403	430
	Shooting	PINN	4.09	13.704	20
	FS	PICNN	3.585	56.195	33920
	EF1	PICNN	1.427	4.377	430
	EF2	PICNN	5.40	17.527	430
	Shooting	PICNN	5.04	13.26	20
	Shooting	Mechanistic	1.284	9.342	20
	Pyomo DAE	Mechanistic	0.101	0.161	1461
2	FS	PICNN	7.16	105.367	39680
	EF1	PICNN	9.586	35.356	860
	EF2	PICNN	35.63	140.76	860
	Pyomo DAE	Mechanistic	0.209	1.224	2922
3	FS	PICNN	8048.77	92608.582	327120
	EF1	PICNN	2149.671	1761.363	7780
	Pyomo DAE	Mechanistic	13.329	53.265	75527

Furthermore, the RS formulation (0.666 s) showed very similar performance to the FS formulation (0.692 s) on average in benchmark 1. However, the solve time for FS on the first iteration was almost 6 times higher than RS. Again, this is likely because the FS formulation requires more auxiliary variables, which cannot be easily initialized. Once the solver has a good warm-start initialization, obtained from solving the first iteration, the benefit of eliminating auxiliary variables that RS formulation provides disappears for this problem.

4.3.2. Model Accuracy in Closed-Loop Simulations

The focus of this work was on the surrogate embedding methods and not the surrogate training. However, to demonstrate consistency between the different embedding methods, the closed-loop performance of the surrogate NMPC was evaluated with respect to its error from the ground truth (using the mechanistic model as the internal NMPC model). The difference between the NMPC solutions found by the different embedding methods was also analysed. One would expect the solution found by the explicit equation approach to be the same as the solution found by the external function approach since they have the same surrogate model. Figure 34 shows the set-point tracking performance for each embedding method in benchmark models 1 and 3. Note that the curves for FS and EF1 using a PINN model overlap in benchmark 1; similarly, for FS and EF1 using PICNN in both benchmarks 1 and 3. The curve for RS using PINN in benchmark 1 also overlaps FS, hence, it was not shown for brevity. Similarly, the curves for EF2 overlap EF1 for all benchmarks, hence, these were also not shown for brevity. The same controller tuning parameters were used for all embedding strategies (i.e., objective weights, control horizon, prediction horizon etc.). For benchmark 1, the controllers closely track the set-point for all embedding methods. The error between the solution found by the surrogate models and mechanistic Pyomo was small. For example, the curve for FS using PINN (orange) lies closely to the mechanistic curve (blue). Thus,

the mismatch between the surrogate model and the plant model was deemed acceptable. This is expected because the training loss had a very low value at the end of training for benchmark 1.

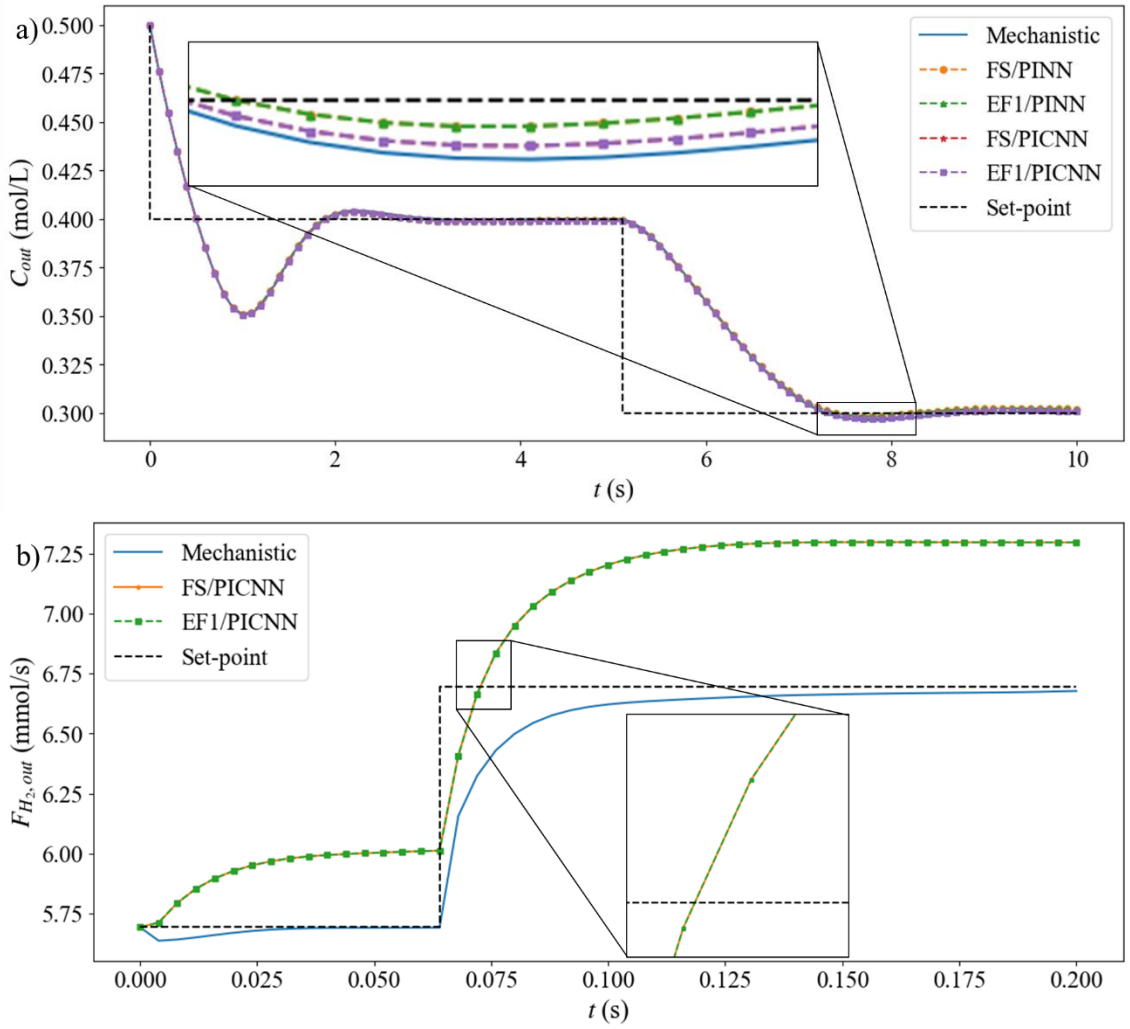


Figure 34 NMPC set-point tracking for benchmarks 1 and 3, a) and b), respectively, for selected embedding methods. Benchmark 1 shows the outlet concentration from the PFR. Benchmark 3 shows the outlet flow rate of hydrogen from the steam reformer.

On the other hand, in benchmark 3, the solutions from both FS and EF1 using PICNN have a significant error from the mechanistic solution. Thus, the mismatch between the surrogate model and the plant model was very significant which also resulted in a large set-point tracking error. This is expected as the loss for the PICNN trained on benchmark 3 is relatively high at the end of

training as seen in Figure 31. In general, the NMPC solutions found by the external function methods are the same solutions found by explicit algebraic constraint formulations. For example, as mentioned, the curves for FS and EF1 using a PINN in benchmark 1 overlap (same for EF1 and FS using PICNN in benchmark 3), thus, the explicit equations and external model approaches are equivalent. However, the plots in Figure 34 only show the time trajectory followed by only one state. To further confirm this observation, the error between the state trajectory followed by the system controlled by the surrogate NMPC and the mechanistic NMPC was evaluated. To measure the error, the squared error of the trajectory followed the states was used as defined in equation (134); where $\boldsymbol{x}_k^{true}$ is the state at sampling step *k* of the plant controlled by the mechanistic NMPC, $\boldsymbol{x}_k^{sur}$ is the state of the plant that is controlled by the surrogate NMPC, and $\boldsymbol{x}^{factor}$ is the normalization/scaling factor used at the input of the NN models to ensure the states have similar scales. Hence, this measures the deviation from the closed-loop response of a plant controlled by a mechanistic NMPC and a plant controlled by a surrogate NMPC. Similarly, one may evaluate the error for the control actions found by the mechanistic NMPC, $\boldsymbol{u}_k^{true}$, and the control actions found by the surrogate NMPC, $\boldsymbol{u}_k^{sur}$, as shown in equation (135). Hence, this measures the deviation of the control actions implemented on the plant by a mechanistic NMPC from the control actions implemented by a surrogate NMPC.

$$\|\boldsymbol{x}_k^{true} - \boldsymbol{x}_k^{sur}\|^2_{\text{diag}(\boldsymbol{x}^{factor})^{-1}} = \sum_{i=1}^{N_x} \left(\frac{x_k^{true}{}_i - x_k^{sur}{}_i}{x_i^{factor}} \right)^2 \tag{134}$$

$$\|\boldsymbol{u}_k^{true} - \boldsymbol{u}_k^{sur}\|^2_{\text{diag}(\boldsymbol{u}^{factor})^{-1}} = \sum_{i=1}^{N_u} \left(\frac{u_k^{true}{}_i - u_k^{sur}{}_i}{u_i^{factor}} \right)^2 \tag{135}$$

Figure 35a shows the state trajectory error, and Figure 35b the control action trajectory error, over time for benchmark 1. As discussed, the plant/model mismatch for PINN NMPC and PICNN

NMPC was small in benchmark 1. Hence, regardless of the embedding method and type of surrogate model, the error is small in both Figure 35a and Figure 35b. Furthermore, the errors for FS and EF1 using PINN overlap. This means that both the explicit algebraic constraints and the external function methods find the same control actions, which results in the same closed-loop responses; confirming that both embedding methods are equivalent. The same observation can be made for FS and EF1 using PICNN.

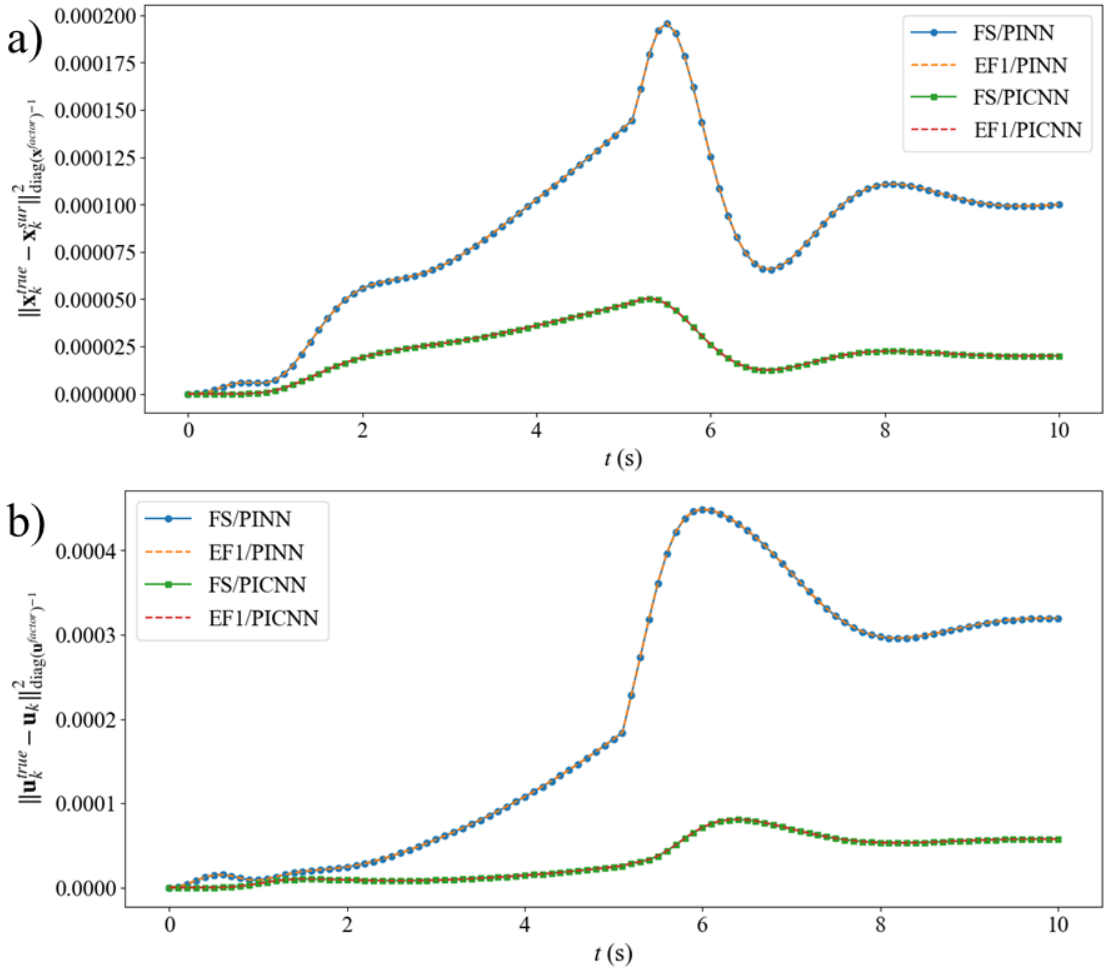


Figure 35 State trajectory error, a), and control action trajectory error, b), plotted against time for benchmark 1.

Figure 36 shows the set-point tracking performance of the shooting method with the mechanistic model (orange), the PINN model (green) and the PICNN model (red) as the internal plant model for benchmark 1. The blue line is the mechanistic model implemented in Pyomo DAE. It can be observed that the shooting method is less consistent in the solutions found for the NMPC problem depending on the model used. For example, in the case of a PICNN, the simultaneous methods found solutions that were close to the solution found by the mechanistic Pyomo model. However, in Figure 36, there is a large error between the mechanistic Pyomo model and the shooting method using PICNN. There also is a larger error between the shooting method using the mechanistic model and the shooting method using PICNN. The reason may be because the shooting method must estimate the Jacobian of the model using numerical methods, instead of analytical methods like in the simultaneous approaches. This may be leading the shooting method to find different local solutions or suboptimal solutions.

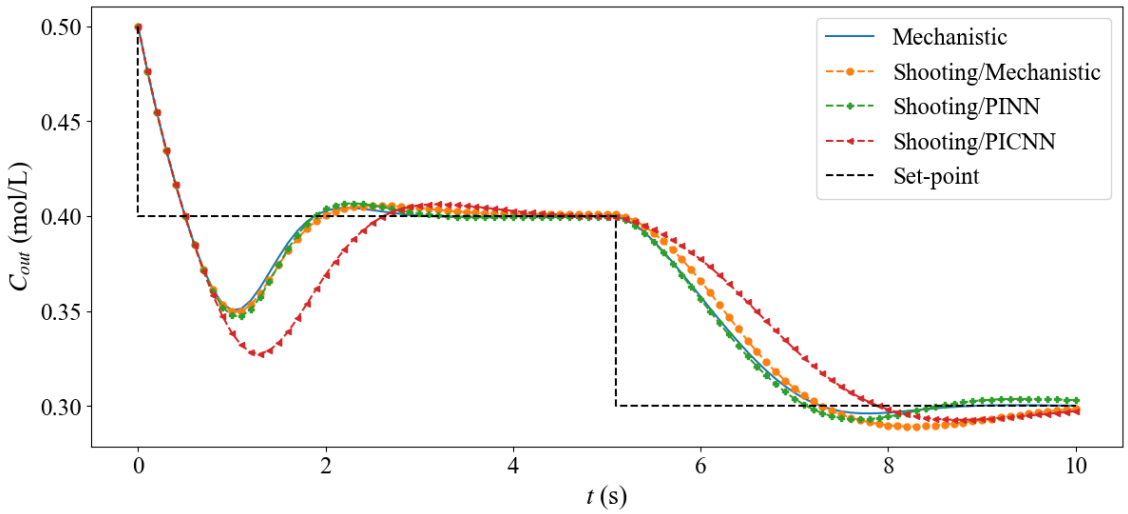


Figure 36 Set-point tracking performance of shooting method using different models for benchmark 1.

4.4. Summary

This study presented the benchmarking of different NN surrogate embedding strategies in NMPC. The direct transcription of the surrogate to explicit constraint equations was compared against the external function evaluation of the surrogate model. In general, it was observed that the use of surrogates did not accelerate the computational performance of the NMPC compared to the traditional approach of directly including the non-linear mechanistic modelling equations of the process in the NMPC formulation. This is likely because the use of hidden layers within the PINN and PICNN surrogates results in an increase in problem size. Simultaneous solution approaches generally lead to a decrease in solution time of an order of magnitude compared to shooting/sequential approaches. The shooting approach requires solving the model of the system at every iteration of the optimization algorithm, which increases the number of computations. Moreover, it was observed that the external model evaluation generally resulted in better computational performance than explicitly representing the NN surrogate equations within the optimization program. This is because explicitly representing the NN surrogate equations results in the generation of auxiliary variables and constraints that increase the size of the problem and cannot be easily initialized. Both embedding methods were shown to be consistent, i.e., they find the same solution to the surrogate NMPC problem, and in addition, the solutions found are close to the solution of the mechanistic NMPC when the mismatch between the surrogate and the mechanistic model is small.

5. Conclusions and Recommendations

This thesis aimed to address challenges in the implementation of NMPC on large-scale processes. Specifically, this work studied the combined effect of model uncertainty and state estimation on the NMPC performance. Two robust NMPC algorithms, MS-NMPC and MSc-NMPC, were applied to the TE process with EKF and MHE as state estimators. Thereby, the applicability of robust NMPC was demonstrated on a large-scale process. The following conclusions were drawn from this implementation:

- Both MS-NMPC and MSc-NMPC could prevent constraint violation and closely track the set-point, even in scenarios where the process response to traditional NMPC became unstable due to the plant/model mismatch introduced by the parametric uncertainty.
- When an unconstrained estimator like EKF was used, it made unrealistic estimates of the states due to the mismatch between the EKF model and the plant. This caused the plant to become destabilized because the initial states provided to NMPC were either infeasible or the solution found by NMPC was suboptimal.
- MHE resolved the challenges involved with EKF. Since MHE considers process constraints in its formulation, it was able to find feasible state estimates even when there was mismatch between the MHE and plant models.
- The combination of robust NMPC and MHE led to an increase in computational costs over simpler approaches like traditional NMPC and EKF; however, for the process sampling time, input delays are expected to be significant.

The second aim of this thesis was to benchmark various strategies to embed NN surrogate models in NMPC. PINNs and PICNNs were employed as NN surrogates. One strategy was to directly transcribe the NN equations as explicit algebraic constraints. The other involved evaluating the

surrogate model as an external function from the optimization framework. The following conclusions were drawn from this study:

- Replacing the mechanistic process plant model with an NN surrogate does not always offer computational advantages, even when the process is highly non-linear. Although smooth activation functions are amenable to NLP solvers, they provide no advantage over mechanistic equations. The evaluation of the internal layers of the NN surrogates increases the model size. Hence, the use of a surrogate usually does not result in a model reduction that may result in shorter computational times compared to the use of the full mechanistic process model.
- Simultaneous solution methods outperformed shooting methods by an order of magnitude for the implementation used in this work. This is likely attributed to having to solve the system model at every iteration of the shooting method optimization algorithm, which increases the number of computations. However, note that more sophisticate shooting methods were not implemented.
- When an NN surrogate was used, the external model evaluation strategy generally outperformed the direct transcription of the NN as algebraic equations. This is likely because initializing the auxiliary variables introduced by the explicit algebraic reformulation is difficult.
- Both embedding strategies were shown to find the same solutions to the NMPC problem, despite using different approaches to evaluate the NN surrogate model.

5.1. Future Work

Despite the advances presented in this thesis, there are still gaps in these research topics that can be further investigated. Regarding the work on robust NMPC and state estimation on large-scale systems that was studied in Chapter 3, the following are avenues of potential future investigation:

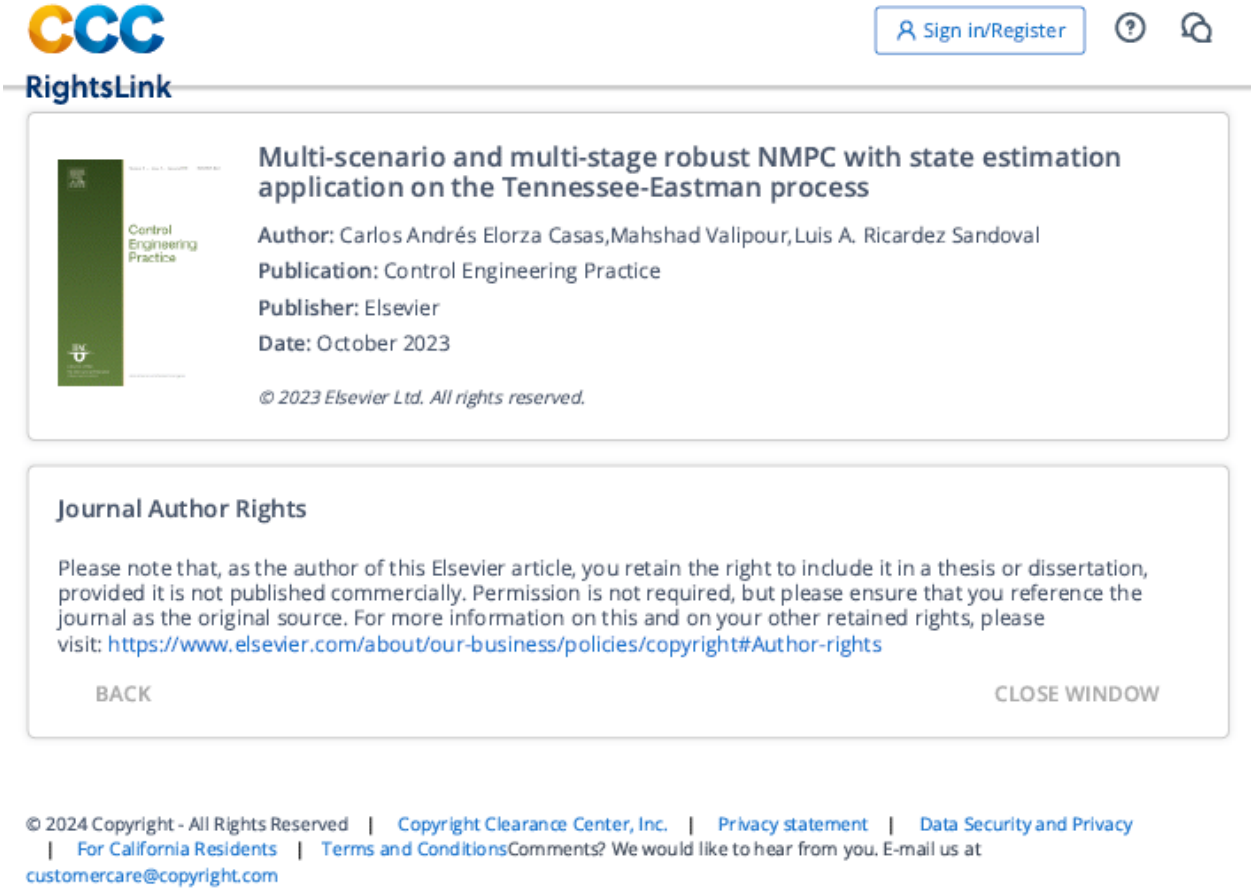
- The full control framework (i.e., robust NMPC plus state estimation) should be applied to other large-scale problems. The TE problem presented particular challenges, like being open-loop unstable. However, there are aspects of more complex systems that should be considered, e.g., processes modelled by PDEs.
- The impact of the arrival cost on the MHE performance should be considered in the future. The MHE computational costs could be reduced by decreasing the size of the moving window. However, the arrival cost becomes more significant as the window shortens. Hence, adequate approximation is necessary.
- Other state estimation schemes, such as constrained EKF, should be considered. Even though EKF failed to provide feasible state estimates, it was significantly faster to evaluate than MHE. Hence, EKF formulations that consider constraints are promising.

Regarding the work on PINNs as NMPC surrogate models studied in Chapter 4, it was observed that replacing the mechanistic plant with a NN does not always offer computational advantages. Despite the unexpected results, there are potential routes for future developments in the application of ML in process control:

- The proposed surrogate embedding strategies could still be useful in cases where only surrogates are available. For example, in a process where a mechanistic model is not available, a data-driven NN model could be used.

- NNs could be used to model the error between the mechanistic model and the actual process plant and be embedded into NMPC using the proposed strategies. Although rigorous mechanistic models could be developed, a mismatch between the model and the actual process may still be present, e.g., due to poorly understood physical phenomena. The NN could then be used to correct the mismatch and improve the accuracy of the predictions.
- The external function evaluation strategy could be improved. The Jacobian and Hessian matrices for the surrogate NMPC problems are highly sparse; hence, these could be evaluated using more efficient methods.
- As a means to decrease online computational costs, ML models could be trained to learn the solution of the NMPC problem to generate an explicit control law, i.e., develop an explicit model predictive controller (EMPC).

Letter of Copyright Permission



RightsLink

Sign in/Register

Multi-scenario and multi-stage robust NMPC with state estimation application on the Tennessee-Eastman process

Author: Carlos Andrés Elorza Casas,Mahshad Valipour,Luis A. Ricardez Sandoval

Publication: Control Engineering Practice

Publisher: Elsevier

Date: October 2023

© 2023 Elsevier Ltd. All rights reserved.

Journal Author Rights

Please note that, as the author of this Elsevier article, you retain the right to include it in a thesis or dissertation, provided it is not published commercially. Permission is not required, but please ensure that you reference the journal as the original source. For more information on this and on your other retained rights, please visit: https://www.elsevier.com/about/our-business/policies/copyright#Author-rights

BACK CLOSE WINDOW

References

Alamir, M. and Balloul, I. (1999) 'Robust constrained control algorithm for general batch processes', *International Journal of Control*, 72(14), pp. 1271–1287. Available at: https://doi.org/10.1080/002071799220254.

Alhajeri, M.S. *et al.* (2022) 'Physics-informed machine learning modeling for predictive control using noisy data', *Chemical Engineering Research and Design*, 186, pp. 34–49. Available at: https://doi.org/10.1016/j.cherd.2022.07.035.

Amari, S. (1993) 'Backpropagation and stochastic gradient descent method', *Neurocomputing*, 5(4–5), pp. 185–196. Available at: https://doi.org/10.1016/0925-2312(93)90006-O.

Anderson, R. *et al.* (2019) 'Strong mixed-integer programming formulations for trained neural networks'. arXiv. Available at: http://arxiv.org/abs/1811.08359 (Accessed: 8 May 2024).

Anderson, T.L., Ellis, M. and Christofides, P.D. (2015) 'Distributed Economic Model Predictive Control of a Catalytic Reactor: Evaluation of Sequential and Iterative Architectures', *IFAC-PapersOnLine*, 48(8), pp. 26–31. Available at: https://doi.org/10.1016/j.ifacol.2015.08.152.

Andrés-Martínez, O., Biegler, L.T. and Flores-Tlacuahuac, A. (2020) 'An indirect approach for singular optimal control problems', *Computers & Chemical Engineering*, 139, p. 106923. Available at: https://doi.org/10.1016/j.compchemeng.2020.106923.

Ansel, J. *et al.* (2024) 'PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation', in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. *ASPLOS '24: 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, La Jolla CA USA: ACM, pp. 929–947. Available at: https://doi.org/10.1145/3620665.3640366.

Antonelo, E.A. *et al.* (2022) 'Physics-Informed Neural Nets for Control of Dynamical Systems'. arXiv. Available at: http://arxiv.org/abs/2104.02556 (Accessed: 16 December 2022).

Arulampalam, M.S. *et al.* (2002) 'A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking', *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 50(2).

Assassa, F. and Marquardt, W. (2014) 'Dynamic optimization using adaptive direct multiple shooting', *Computers & Chemical Engineering*, 60, pp. 242–259. Available at: https://doi.org/10.1016/j.compchemeng.2013.09.017.

Aydin, E., Bonvin, D. and Sundmacher, K. (2017) 'Dynamic optimization of constrained semi-batch processes using Pontryagin's minimum principle—An effective quasi-Newton approach', *Computers & Chemical Engineering*, 99, pp. 135–144. Available at: https://doi.org/10.1016/j.compchemeng.2017.01.019.

Barry-Straume, J. *et al.* (2022) 'Physics-informed neural networks for PDE-constrained optimization and control'. arXiv. Available at: http://arxiv.org/abs/2205.03377 (Accessed: 22 March 2024).

Baydin, A.G. *et al.* (2018) 'Automatic Differentiation in Machine Learning: a Survey', *Journal of Machine Learning Research*, 18. Available at: https://www.jmlr.org/papers/v18/17-468.html.

Bayer, F.A., Muller, M.A. and Allgower, F. (2016) 'Min-max economic model predictive control approaches with guaranteed performance', in *2016 IEEE 55th Conference on Decision and Control (CDC)*. *2016 IEEE 55th Conference on Decision and Control (CDC)*, Las Vegas, NV, USA: IEEE, pp. 3210–3215. Available at: https://doi.org/10.1109/CDC.2016.7798751.

Bemporad, A. and Morari, M. (1999) 'Robust model predictive control: A survey', in A. Garulli and A. Tesi (eds) *Robustness in identification and control*. London: Springer London (Lecture Notes in Control and Information Sciences), pp. 207–226. Available at: https://doi.org/10.1007/BFb0109870.

Biegler, L.T. (2010) *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Society for Industrial and Applied Mathematics. Available at: https://doi.org/10.1137/1.9780898719383.

Biegler, L.T. (2021) 'A perspective on nonlinear model predictive control', *Korean Journal of Chemical Engineering*, 38(7), pp. 1317–1332. Available at: https://doi.org/10.1007/s11814-021-0791-7.

Biegler, L.T. and Thierry, D.M. (2018) 'Large-scale Optimization Formulations and Strategies for Nonlinear Model Predictive Control', *IFAC-PapersOnLine*, 51(20), pp. 1–15. Available at: https://doi.org/10.1016/j.ifacol.2018.10.167.

Biegler, L.T. and Zavala, V.M. (2009) 'Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization', *Computers & Chemical Engineering*, 33(3), pp. 575–582. Available at: https://doi.org/10.1016/j.compchemeng.2008.08.006.

Cai, S. *et al.* (2021) 'Physics-Informed Neural Networks for Heat Transfer Problems', *Journal of Heat Transfer*, 143(6), p. 060801. Available at: https://doi.org/10.1115/1.4050542.

Ceccon, F. *et al.* (2022) 'OMLT: Optimization & Machine Learning Toolkit'. arXiv. Available at: http://arxiv.org/abs/2202.02414 (Accessed: 22 February 2024).

Cervantes, A. and Biegler, L.T. (2008) 'Optimization Strategies for Dynamic Systems', in C.A. Floudas and P.M. Pardalos (eds) *Encyclopedia of Optimization*. Boston, MA: Springer US, pp. 2847–2858. Available at: https://doi.org/10.1007/978-0-387-74759-0_488.

Chaffart, D., Yuan, Y. and Ricardez-Sandoval, L.A. (2024) 'Multiscale Physics-Informed Neural Network Framework to Capture Stochastic Thin-Film Deposition', *The Journal of Physical Chemistry C*, 128(9), pp. 3733–3750. Available at: https://doi.org/10.1021/acs.jpcc.3c07168.

Chen, Y., Shi, Y. and Zhang, B. (2019) 'Optimal Control Via Neural Networks: A Convex Approach'. arXiv. Available at: http://arxiv.org/abs/1805.11835 (Accessed: 22 March 2024).

Christiansen, L.H. and Jorgensen, J.B. (2018) 'A fast PDE-constrained optimization solver for nonlinear diffusion-reaction processes', in *2018 IEEE Conference on Decision and Control (CDC)*. *2018 IEEE Conference on Decision and Control (CDC)*, Miami Beach, FL: IEEE, pp. 2635–2640. Available at: https://doi.org/10.1109/CDC.2018.8619280.

Christofides, P.D., Liu, J. and Muñoz De La Peña, D. (2011) 'Lyapunov-Based Model Predictive Control', in Christofides, P. D., Liu, J., and Muñoz De La Peña, D., *Networked and Distributed Predictive Control*. London: Springer London (Advances in Industrial Control), pp. 13–45. Available at: https://doi.org/10.1007/978-0-85729-582-8_2.

Daoutidis, P. *et al.* (2024) 'Machine learning in process systems engineering: Challenges and opportunities', *Computers & Chemical Engineering*, 181, p. 108523. Available at: https://doi.org/10.1016/j.compchemeng.2023.108523.

Das, B. and Mhaskar, P. (2014) 'Lyapunov-based offset-free model predictive control of nonlinear systems', in *2014 American Control Conference*. *2014 American Control Conference - ACC 2014*, Portland, OR, USA: IEEE, pp. 2839–2844. Available at: https://doi.org/10.1109/ACC.2014.6859472.

Dormand, J.R. and Prince, P.J. (1980) 'A family of embedded Runge-Kutta formulae', *Journal of Computational and Applied Mathematics*, 6(1), pp. 19–26. Available at: https://doi.org/10.1016/0771-050X(80)90013-3.

Downs, J.J. and Vogel, E.F. (1993) 'A Plant-wide Industrial Process Control Problem', *Computers & Chemical Engineering*, 17(3), pp. 245–255. Available at: https://doi.org/10.1016/0098-1354(93)80018-I.

Ellis, M., Liu, J. and Christofides, P.D. (2017) *Economic Model Predictive Control*. Cham: Springer International Publishing (Advances in Industrial Control). Available at: https://doi.org/10.1007/978-3-319-41108-8.

Elorza Casas, C.A., Valipour, M. and Ricardez Sandoval, L.A. (2023) 'Multi-scenario and multi-stage robust NMPC with state estimation application on the Tennessee-Eastman process', *Control Engineering Practice* [Preprint]. Available at: https://doi.org/10.1016/j.conengprac.2023.105635.

Esche, E. *et al.* (2022) 'Architectures for neural networks as surrogates for dynamic systems in chemical engineering', *Chemical Engineering Research and Design*, 177, pp. 184–199. Available at: https://doi.org/10.1016/j.cherd.2021.10.042.

Eydenberg, M.S. *et al.* (2022) 'Physics-Informed Machine Learning with Optimization-Based Guarantees: Applications to Ac Power Flow', *SSRN Electronic Journal* [Preprint]. Available at: https://doi.org/10.2139/ssrn.4245628.

Fioretto, F. *et al.* (2020) 'Lagrangian Duality for Constrained Deep Learning'. arXiv. Available at: http://arxiv.org/abs/2001.09394 (Accessed: 11 August 2023).

Fogler, H.S. (2016) *Elements of chemical reaction engineering*. Fifth edition. Boston Columbus Indianapolis: Prentice-Hall, Pearson (Prentice-Hall international series in the physical and chemical engineering sciences).

Fuchigami, S., Niina, T. and Takada, S. (2020) 'Particle Filter Method to Integrate High-Speed Atomic Force Microscopy Measurements with Biomolecular Simulations', *Journal of Chemical Theory and Computation*, 16(10), pp. 6609–6619. Available at: https://doi.org/10.1021/acs.jctc.0c00234.

Gao, H., Sun, L. and Wang, J.-X. (2021) 'PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain', *Journal of Computational Physics*, 428, p. 110079. Available at: https://doi.org/10.1016/j.jcp.2020.110079.

Gokhale, G., Claessens, B. and Develder, C. (2022) 'Physics Informed Neural Networks for Control Oriented Thermal Modeling of Buildings', *Applied Energy*, 314, p. 118852. Available at: https://doi.org/10.1016/j.apenergy.2022.118852.

Golshan, M., Boozarjomehry, R.B. and Pishvaie, M.R. (2005) 'A new approach to real time optimization of the Tennessee Eastman challenge problem', *Chemical Engineering Journal*, 112(1–3), pp. 33–44. Available at: https://doi.org/10.1016/j.cej.2005.06.005.

Golshan, M., Pishvaie, M.R. and Bozorgmehry Boozarjomehry, R. (2008) 'Stochastic and global real time optimization of Tennessee Eastman challenge problem', *Engineering Applications of Artificial Intelligence*, 21(2), pp. 215–228. Available at: https://doi.org/10.1016/j.engappai.2007.04.004.

Grancharova, A. and Johansen, T.A. (2005) 'EXPLICIT MIN-MAX MODEL PREDICTIVE CONTROL OF CONSTRAINED NONLINEAR SYSTEMS WITH MODEL UNCERTAINTY', *IFAC Proceedings Volumes*, 38(1), pp. 1019–1024. Available at: https://doi.org/10.3182/20050703-6-CZ-1902.00826.

Grancharova, A. and Johansen, T.A. (2009) 'Computation, approximation and stability of explicit feedback min–max nonlinear model predictive control', *Automatica*, 45(5), pp. 1134–1143. Available at: https://doi.org/10.1016/j.automatica.2008.12.023.

Grancharova, A. and Johansen, T.A. (2012) 'Explicit Min-Max MPC of Constrained Nonlinear Systems with Bounded Uncertainties', in *Explicit Nonlinear Model Predictive Control: Theory and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Control and Information Sciences). Available at: https://doi.org/10.1007/978-3-642-28780-0.

Griewank, A. and Walther, A. (2003) 'Introduction to Automatic Differentiation', *PAMM*, 2(1), pp. 45–49. Available at: https://doi.org/10.1002/pamm.200310012.

Hammerstrom, D. (1993) 'Working with neural networks', *IEEE Spectrum*, 30(7), pp. 46–53. Available at: https://doi.org/10.1109/6.222230.

Hao, Z. *et al.* (2023) ‘Physics-Informed Machine Learning: A Survey on Problems, Methods and Applications’. arXiv. Available at: http://arxiv.org/abs/2211.08064 (Accessed: 7 March 2024).

Hillier, F.S. and Lieberman, G.J. (2015) *Introduction to operations research*. Tenth edition. New York, NY: McGraw-Hill Education.

Holtorf, F., Mitsos, A. and Biegler, L.T. (2019) ‘Multistage NMPC with on-line generated scenario trees: Application to a semi-batch polymerization process’, *Journal of Process Control*, 80, pp. 167–179. Available at: https://doi.org/10.1016/j.jprocont.2019.05.007.

Huang, R. and Biegler, L.T. (2009) ‘Robust nonlinear model predictive controller design based on multi-scenario formulation’, in *2009 American Control Conference*. *2009 American Control Conference*, pp. 2341–2342. Available at: https://doi.org/10.1109/ACC.2009.5160486.

Huang, R., Biegler, L.T. and Patwardhan, S.C. (2010) ‘Fast Offset-Free Nonlinear Model Predictive Control Based on Moving Horizon Estimation’, *Industrial & Engineering Chemistry Research*, 49(17), pp. 7882–7890. Available at: https://doi.org/10.1021/ie901945y.

Huang, R., Zavala, V.M. and Biegler, L.T. (2009) ‘Advanced step nonlinear model predictive control for air separation units’, *Journal of Process Control*, 19(4), pp. 678–685. Available at: https://doi.org/10.1016/j.jprocont.2008.07.006.

Jalanko, M. *et al.* (2021) ‘Flooding and offset-free nonlinear model predictive control of a high-purity industrial ethylene splitter using a hybrid model’, *Computers & Chemical Engineering*, 155, p. 107514. Available at: https://doi.org/10.1016/j.compchemeng.2021.107514.

Jang, H., Lee, J.H. and Biegler, L.T. (2016) ‘A robust NMPC scheme for semi-batch polymerization reactors’, *IFAC-PapersOnLine*, 49(7), pp. 37–42. Available at: https://doi.org/10.1016/j.ifacol.2016.07.213.

Jiang, S. and Zavala, V.M. (2021) ‘Convolutional Neural Nets in Chemical Engineering: Foundations, Computations, and Applications’, *AIChE Journal*, 67(9), p. e17282. Available at: https://doi.org/10.1002/aic.17282.

Jockenhövel, T., Biegler, L.T. and Wächter, A. (2003) ‘Dynamic optimization of the Tennessee Eastman process using the OptControlCentre’, *Computers & Chemical Engineering*, 27(11), pp. 1513–1531. Available at: https://doi.org/10.1016/S0098-1354(03)00113-3.

Kingma, D.P. and Ba, J. (2017) ‘Adam: A Method for Stochastic Optimization’. arXiv. Available at: http://arxiv.org/abs/1412.6980 (Accessed: 2 May 2024).

Kraus, T. *et al.* (2006) ‘A Moving Horizon State Estimation algorithm applied to the Tennessee Eastman Benchmark Process’, in *2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. *2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Heidelberg, Germany: IEEE, pp. 377–382. Available at: https://doi.org/10.1109/MFI.2006.265620.

Kühl, P. *et al.* (2011) ‘A real-time algorithm for moving horizon state and parameter estimation’, *Computers & Chemical Engineering*, 35(1), pp. 71–83. Available at: https://doi.org/10.1016/j.compchemeng.2010.07.012.

Kumar, K., Patwardhan, S.C. and Noronha, S. (2020) ‘Grade Transition Control of Tennessee Eastman Process Using Adaptive Dual MPC’, p. 4.

Kumar Kottakki, K., Bhushan, M. and Bhartiya, S. (2017) ‘Unconstrained Nonlinear State Estimation for Tennessee Eastman Challenge Process’, *IFAC-PapersOnLine*, 50(1), pp. 12919–12924. Available at: https://doi.org/10.1016/j.ifacol.2017.08.1788.

Kummer, A., Nagy, L. and Varga, T. (2020) ‘NMPC-based control scheme for a semi-batch reactor under parameter uncertainty’, *Computers & Chemical Engineering*, 141, p. 106998. Available at: https://doi.org/10.1016/j.compchemeng.2020.106998.

Lee, J.H. (2011) ‘Model predictive control: Review of the three decades of development’, *International Journal of Control, Automation and Systems*, 9(3), pp. 415–424. Available at: https://doi.org/10.1007/s12555-011-0300-6.

Lee, J.H. and Yu, Z. (1997) ‘Worst-case formulations of model predictive control for systems with bounded parameters’, *Automatica*, 33(5), pp. 763–781. Available at: https://doi.org/10.1016/S0005-1098(96)00255-5.

Leyffer, S. *et al.* (2018) ‘Nonlinear Robust Optimization’.

Limon, D. *et al.* (2009) ‘Input-to-State Stability: A Unifying Framework for Robust Model Predictive Control’, in L. Magni, Davide Martino Raimondo, and F. Allgöwer (eds) *Nonlinear Model Predictive Control*. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Control and Information Sciences), pp. 1–26. Available at: https://doi.org/10.1007/978-3-642-01094-1_1.

Lopez-Negrete, R. *et al.* (2013) ‘Fast nonlinear model predictive control: Formulation and industrial process applications’, *Computers & Chemical Engineering*, 51, pp. 55–64. Available at: https://doi.org/10.1016/j.compchemeng.2012.06.011.

Lucia, S. *et al.* (2014) ‘Handling uncertainty in economic nonlinear model predictive control: A comparative case study’, *Journal of Process Control*, 24(8), pp. 1247–1259. Available at: https://doi.org/10.1016/j.jprocont.2014.05.008.

Lucia, S. *et al.* (2017) ‘Rapid development of modular and sustainable nonlinear model predictive control solutions’, *Control Engineering Practice*, 60, pp. 51–62. Available at: https://doi.org/10.1016/j.conengprac.2016.12.009.

Lucia, S. and Engell, S. (2012) ‘Multi-stage and Two-stage Robust Nonlinear Model Predictive Control’, *IFAC Proceedings Volumes*, 45(17), pp. 181–186. Available at: https://doi.org/10.3182/20120823-5-NL-3013.00015.

Lucia, S. and Engell, S. (2015) 'Potential and Limitations of Multi-stage Nonlinear Model Predictive Control', *IFAC-PapersOnLine*, 48(8), pp. 1015–1020. Available at: https://doi.org/10.1016/j.ifacol.2015.09.101.

Luenberger, D.G. and Ye, Y. (2016) *Linear and nonlinear programming*. Fourth edition. Cham Heidelberg New York Dordrecht London: Springer (International series in operations research & management science, volume 228).

Mao, Z., Jagtap, A.D. and Karniadakis, G.E. (2020) 'Physics-informed neural networks for high-speed flows', *Computer Methods in Applied Mechanics and Engineering*, 360, p. 112789. Available at: https://doi.org/10.1016/j.cma.2019.112789.

Mayne, D.Q. *et al.* (2011) 'Tube-based robust nonlinear model predictive control', *International Journal of Robust and Nonlinear Control*, 21(11), pp. 1341–1353. Available at: https://doi.org/10.1002/rnc.1758.

Mayne, D.Q. (2014) 'Model predictive control: Recent developments and future promise', *Automatica*, 50(12), pp. 2967–2986. Available at: https://doi.org/10.1016/j.automatica.2014.10.128.

Mesbah, A. (2016) 'Stochastic Model Predictive Control: An Overview and Perspectives for Future Research', *IEEE Control Systems*, 36(6), pp. 30–44. Available at: https://doi.org/10.1109/MCS.2016.2602087.

Mesbah, A. *et al.* (2022) 'Fusion of Machine Learning and MPC under Uncertainty: What Advances Are on the Horizon?', in *2022 American Control Conference (ACC)*. *2022 American Control Conference (ACC)*, Atlanta, GA, USA: IEEE, pp. 342–357. Available at: https://doi.org/10.23919/ACC53348.2022.9867643.

Ming Yan and Ricker, N.L. (1997) 'On-line optimization of the Tennessee Eastman challenge process', in *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)*. *Proceedings of 16th American CONTROL Conference*, Albuquerque, NM, USA: IEEE, pp. 2960–2965 vol.5. Available at: https://doi.org/10.1109/ACC.1997.612000.

Misener, R. and Biegler, L. (2023) 'Formulating data-driven surrogate models for process optimization', *Computers & Chemical Engineering*, 179, p. 108411. Available at: https://doi.org/10.1016/j.compchemeng.2023.108411.

Mowlavi, S. and Nabi, S. (2023) 'Optimal control of PDEs using physics-informed neural networks', *Journal of Computational Physics*, 473, p. 111731. Available at: https://doi.org/10.1016/j.jcp.2022.111731.

Mukherjee, T. *et al.* (2021) 'Broyden's update based extended Kalman Filter for nonlinear state estimation', *Journal of Process Control*, 105, pp. 267–282. Available at: https://doi.org/10.1016/j.jprocont.2021.08.007.

Nabian, M.A., Gladstone, R.J. and Meidani, H. (2021) 'Efficient training of physics-informed neural networks via importance sampling', *Computer-Aided Civil and Infrastructure Engineering*, 36(8), pp. 962–977. Available at: https://doi.org/10.1111/mice.12685.

Nicholson, B. *et al.* (2018) 'pyomo.dae: a modeling and automatic discretization framework for optimization with differential and algebraic equations', *Mathematical Programming Computation*, 10(2), pp. 187–223. Available at: https://doi.org/10.1007/s12532-017-0127-0.

Nicodemus, J. *et al.* (2022) 'Physics-informed Neural Networks-based Model Predictive Control for Multi-link Manipulators', *IFAC-PapersOnLine*, 55(20), pp. 331–336. Available at: https://doi.org/10.1016/j.ifacol.2022.09.117.

Pannocchia, G., Gabiccini, M. and Artoni, A. (2015) 'Offset-free MPC explained: novelties, subtleties, and applications', *IFAC-PapersOnLine*, 48(23), pp. 342–351. Available at: https://doi.org/10.1016/j.ifacol.2015.11.304.

Patrón, G.D. and Ricardez-Sandoval, L. (2020a) 'A robust nonlinear model predictive controller for a post-combustion CO2 capture absorber unit', *Fuel*, 265, p. 116932. Available at: https://doi.org/10.1016/j.fuel.2019.116932.

Patrón, G.D. and Ricardez-Sandoval, L. (2020b) 'Real-Time Optimization and Nonlinear Model Predictive Control for a Post-Combustion Carbon Capture Absorber', *IFAC-PapersOnLine*, 53(2), pp. 11595–11600. Available at: https://doi.org/10.1016/j.ifacol.2020.12.639.

Patrón, G.D. and Ricardez-Sandoval, L. (2022) 'An integrated real-time optimization, control, and estimation scheme for post-combustion CO2 capture', *Applied Energy*, 308, p. 118302. Available at: https://doi.org/10.1016/j.apenergy.2021.118302.

Patrón, G.D. and Ricardez-Sandoval, L. (2024) 'Economically optimal operation of recirculating aquaculture systems under uncertainty', *Computers and Electronics in Agriculture*, 220, p. 108856. Available at: https://doi.org/10.1016/j.compag.2024.108856.

Patrón, G.D., Toffolo, K. and Ricardez-Sandoval, L. (2024) 'Economic model predictive control for packed bed chemical looping combustion', *Chemical Engineering and Processing - Process Intensification*, 198, p. 109731. Available at: https://doi.org/10.1016/j.cep.2024.109731.

Piceno-Díaz, E.R. *et al.* (2020) 'Robust Nonlinear Model Predictive Control for Two-Stage Anaerobic Digesters', *Industrial & Engineering Chemistry Research*, 59(52), pp. 22559–22572. Available at: https://doi.org/10.1021/acs.iecr.0c03809.

Prasad, G. *et al.* (1999) 'Plant-wide physical model-based control for a thermal power plant', in *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304). 1999 Conference on Decision and Control*, Phoenix, AZ, USA: IEEE, pp. 4631–4636. Available at: https://doi.org/10.1109/CDC.1999.833273.

Pulsipher, J.L. *et al.* (2022) 'A unifying modeling abstraction for infinite-dimensional optimization', *Computers & Chemical Engineering*, 156, p. 107567. Available at: https://doi.org/10.1016/j.compchemeng.2021.107567.

Puschke, J. and Mitsos, A. (2018) ‘Robust feasible control based on multi-stage eNMPC considering worst-case scenarios’, *Journal of Process Control*, 69, pp. 8–15. Available at: https://doi.org/10.1016/j.jprocont.2018.07.004.

Raimondo, D.M. *et al.* (2007) ‘Towards the practical implementation of min-max nonlinear Model Predictive Control’, in *2007 46th IEEE Conference on Decision and Control*. *2007 46th IEEE Conference on Decision and Control*, New Orleans, LA, USA: IEEE, pp. 1257–1262. Available at: https://doi.org/10.1109/CDC.2007.4434980.

Raissi, M., Perdikaris, P. and Karniadakis, G.E. (2017a) ‘Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations’. arXiv. Available at: http://arxiv.org/abs/1711.10561 (Accessed: 30 November 2022).

Raissi, M., Perdikaris, P. and Karniadakis, G.E. (2017b) ‘Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations’. arXiv. Available at: http://arxiv.org/abs/1711.10566 (Accessed: 30 November 2022).

Raissi, M., Perdikaris, P. and Karniadakis, G.E. (2019) ‘Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations’, *Journal of Computational Physics*, 378, pp. 686–707. Available at: https://doi.org/10.1016/j.jcp.2018.10.045.

Rasoulian, S. and Ricardez-Sandoval, L.A. (2015) ‘A robust nonlinear model predictive controller for a multiscale thin film deposition process’, *Chemical Engineering Science*, 136, pp. 38–49. Available at: https://doi.org/10.1016/j.ces.2015.02.002.

Rawlings, J.B., Mayne, D.Q. and Diehl, M. (2017) *Model predictive control: theory, computation, and design*. 2nd edition. Madison, Wisconsin: Nob Hill Publishing.

Ricker, N.L. and Lee, J.H. (1995a) ‘Nonlinear model predictive control of the Tennessee Eastman challenge process’, *Computers & Chemical Engineering*, 19(9), pp. 961–981. Available at: https://doi.org/10.1016/0098-1354(94)00105-W.

Ricker, N.L. and Lee, J.H. (1995b) ‘Nonlinear modeling and state estimation for the Tennessee Eastman challenge process’, *Computers & Chemical Engineering*, 19(9), pp. 983–1005. Available at: https://doi.org/10.1016/0098-1354(94)00113-3.

Rodriguez, J.S. *et al.* (2023) ‘Scalable Parallel Nonlinear Optimization with PyNumero and Parapint’, *INFORMS Journal on Computing*, 35(2), pp. 509–517. Available at: https://doi.org/10.1287/ijoc.2023.1272.

Santander, O., Elkamel, A. and Budman, H. (2019) ‘Robust economic model predictive control: disturbance rejection, robustness and periodic operation in chemical reactors’, *Engineering Optimization*, 51(5), pp. 896–914. Available at: https://doi.org/10.1080/0305215X.2018.1497617.

Sanyal, S. and Roy, K. (2023) ‘RAMP-Net: A Robust Adaptive MPC for Quadrotors via Physics-informed Neural Network’, in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London,

United Kingdom: IEEE, pp. 1019–1025. Available at: https://doi.org/10.1109/ICRA48891.2023.10161410.

Schweidtmann, A.M. and Mitsos, A. (2019) 'Deterministic Global Optimization with Artificial Neural Networks Embedded', *Journal of Optimization Theory and Applications*, 180(3), pp. 925–948. Available at: https://doi.org/10.1007/s10957-018-1396-0.

Scokaert, P.O.M. and Mayne, D.Q. (1998) 'Min-max feedback model predictive control for constrained linear systems', *IEEE Transactions on Automatic Control*, 43(8), pp. 1136–1142. Available at: https://doi.org/10.1109/9.704989.

Segovia, P. *et al.* (2019) 'Model predictive control and moving horizon estimation for water level regulation in inland waterways', *Journal of Process Control*, 76, pp. 1–14. Available at: https://doi.org/10.1016/j.jprocont.2018.12.017.

Siemers, F.M., Feldmann, C. and Bajorath, J. (2022) 'Minimal data requirements for accurate compound activity prediction using machine learning methods of different complexity', *Cell Reports Physical Science*, 3(11), p. 101113. Available at: https://doi.org/10.1016/j.xcrp.2022.101113.

Sirignano, J. and Spiliopoulos, K. (2018) 'DGM: A deep learning algorithm for solving partial differential equations', *Journal of Computational Physics*, 375, pp. 1339–1364. Available at: https://doi.org/10.1016/j.jcp.2018.08.029.

Skupin, P. *et al.* (2022) 'Robust nonlinear model predictive control of cascade of fermenters with recycle for efficient bioethanol production', *Computers & Chemical Engineering*, 160, p. 107735. Available at: https://doi.org/10.1016/j.compchemeng.2022.107735.

Soledad Aronna, M., Frédéric Bonnans, J. and Martinon, P. (2013) 'A Shooting Algorithm for Optimal Control Problems with Singular Arcs', *Journal of Optimization Theory and Applications*, 158(2), pp. 419–459. Available at: https://doi.org/10.1007/s10957-012-0254-8.

Subramanian, S., Lucia, S. and Engel, S. (2015) 'Adaptive Multi-stage Output Feedback NMPC using the Extended Kalman Filter for time varying uncertainties applied to a CSTR', *IFAC-PapersOnLine*, 48(23), pp. 242–247. Available at: https://doi.org/10.1016/j.ifacol.2015.11.290.

Tatjewski, P. and Ławryńczuk, M. (2020) 'Algorithms with state estimation in linear and nonlinear model predictive control', *Computers & Chemical Engineering*, 143, p. 107065. Available at: https://doi.org/10.1016/j.compchemeng.2020.107065.

Tătulea-Codrean, A., Fischer, J. and Engell, S. (2020) 'A Multi-stage Economic NMPC for the Tennessee Eastman Challenge Process', *IFAC-PapersOnLine*, 53(2), pp. 6069–6075. Available at: https://doi.org/10.1016/j.ifacol.2020.12.1678.

Thangavel, S. *et al.* (2018) 'Dual robust nonlinear model predictive control: A multi-stage approach', *Journal of Process Control*, 72, pp. 39–51. Available at: https://doi.org/10.1016/j.jprocont.2018.10.003.

Thangavel, S., Paulen, R. and Engell, S. (2020) 'Robust Multi-Stage Nonlinear Model Predictive Control Using Sigma Points', *Processes*, 8(7), p. 851. Available at: https://doi.org/10.3390/pr8070851.

Theodoridis, S. and Koutroumbas, K. (2009) *Pattern recognition*. 4th ed. Burlington, MA London: Academic Press.

Thierry, D. and Biegler, L. (2020) 'The ℓ1—Exact Penalty-Barrier Phase for Degenerate Nonlinear Programming Problems in Ipopt', *IFAC-PapersOnLine*, 53(2), pp. 6496–6501. Available at: https://doi.org/10.1016/j.ifacol.2020.12.1798.

Thombre, M. *et al.* (2021) 'Sensitivity-Assisted multistage nonlinear model predictive control: Robustness, stability and computational efficiency', *Computers & Chemical Engineering*, 148, p. 107269. Available at: https://doi.org/10.1016/j.compchemeng.2021.107269.

Toffolo, K., Meunier, S. and Ricardez-Sandoval, L. (2024) 'Optimal operation of a large-scale packed bed chemical-looping combustion process using nonlinear model predictive control', *Fuel*, 357, p. 129876. Available at: https://doi.org/10.1016/j.fuel.2023.129876.

Tsay, C. *et al.* (2021) 'Partition-based formulations for mixed-integer optimization of trained ReLU neural networks'. arXiv. Available at: http://arxiv.org/abs/2102.04373 (Accessed: 9 May 2024).

Valipour, M. and Ricardez-Sandoval, L.A. (2021a) 'Assessing the Impact of EKF as the Arrival Cost in the Moving Horizon Estimation under Nonlinear Model Predictive Control', *Industrial & Engineering Chemistry Research*, 60(7), pp. 2994–3012. Available at: https://doi.org/10.1021/acs.iecr.0c06095.

Valipour, M. and Ricardez-Sandoval, L.A. (2021b) 'Constrained Abridged Gaussian Sum Extended Kalman Filter: Constrained Nonlinear Systems with Non-Gaussian Noises and Uncertainties', *Industrial & Engineering Chemistry Research*, 60(47), pp. 17110–17127. Available at: https://doi.org/10.1021/acs.iecr.1c02804.

Valipour, M., Toffolo, K.M. and Ricardez-Sandoval, L.A. (2021) 'State estimation and sensor location for Entrained-Flow Gasification Systems using Kalman Filter', *Control Engineering Practice*, 108, p. 104702. Available at: https://doi.org/10.1016/j.conengprac.2020.104702.

Van Waarde, H.J. *et al.* (2020) 'Data Informativity: A New Perspective on Data-Driven Analysis and Control', *IEEE Transactions on Automatic Control*, 65(11), pp. 4753–4768. Available at: https://doi.org/10.1109/TAC.2020.2966717.

Vinoth Upendra, J. and Prakash, J. (2013) 'Comparison of State Estimation Algorithms on the Tennessee Eastman Process', in R. Malathi and J. Krishnan (eds) *Recent Advancements in System Modelling Applications*. India: Springer India (Lecture Notes in Electrical Engineering), pp. 357–368. Available at: https://doi.org/10.1007/978-81-322-1035-1_31.

Virtanen, P. *et al.* (2020) 'SciPy 1.0: fundamental algorithms for scientific computing in Python', *Nature Methods*, 17(3), pp. 261–272. Available at: https://doi.org/10.1038/s41592-019-0686-2.

Wächter, A. and Biegler, L.T. (2006) ‘On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming’, *Mathematical Programming*, 106(1), pp. 25–57. Available at: https://doi.org/10.1007/s10107-004-0559-y.

Wang, S., Teng, Y. and Perdikaris, P. (2020) ‘Understanding and mitigating gradient pathologies in physics-informed neural networks’. arXiv. Available at: http://arxiv.org/abs/2001.04536 (Accessed: 24 May 2024).

Welch, G. and Bishop, G. (2006) ‘An Introduction to the Kalman Filter’.

Xu, J. and Froment, G.F. (1989) ‘Methane steam reforming, methanation and water-gas shift: I. Intrinsic kinetics’, *AIChE Journal*, 35(1), pp. 88–96. Available at: https://doi.org/10.1002/aic.690350109.

Yan, M. and Ricker, N.L. (1995) ‘Multi-objective control of the Tennessee Eastman challenge process’, in *Proceedings of 1995 American Control Conference - ACC’95*. *Proceedings of 1995 American Control Conference - ACC’95*, pp. 245–249 vol.1. Available at: https://doi.org/10.1109/ACC.1995.529246.

Yang, D., Balaprakash, P. and Leyffer, S. (2022) ‘Modeling design and control problems involving neural network surrogates’, *Computational Optimization and Applications*, 83(3), pp. 759–800. Available at: https://doi.org/10.1007/s10589-022-00404-9.

Yang, X. and Scott, J.K. (2020) ‘Accurate Uncertainty Propagation for Discrete-Time Nonlinear Systems Using Differential Inequalities With Model Redundancy’, *IEEE Transactions on Automatic Control*, 65(12), pp. 5043–5057. Available at: https://doi.org/10.1109/TAC.2020.2968241.

Yanıkoğlu, İ., Gorissen, B.L. and den Hertog, D. (2019) ‘A survey of adjustable robust optimization’, *European Journal of Operational Research*, 277(3), pp. 799–813. Available at: https://doi.org/10.1016/j.ejor.2018.08.031.

Yaws, C.L. (ed.) (1999) *Chemical Properties Handbook*. 1st Edition. McGraw-Hill Education. Available at: https://www.accessengineeringlibrary.com/content/book/9780070734012 (Accessed: 17 April 2024).

Zavala, V.M. and Biegler, L.T. (2009) ‘Optimization-based strategies for the operation of low-density polyethylene tubular reactors: Moving horizon estimation’, *Computers & Chemical Engineering*, 33(1), pp. 379–390. Available at: https://doi.org/10.1016/j.compchemeng.2008.10.008.

Zhang, J. and Liu, J. (2013) ‘Lyapunov-based MPC with robust moving horizon estimation and its triggered implementation’, *AIChE Journal*, 59(11), pp. 4273–4286. Available at: https://doi.org/10.1002/aic.14187.

Zhang, Z. (2022) ‘A physics-informed deep convolutional neural network for simulating and predicting transient Darcy flows in heterogeneous reservoirs without labeled data’, *Journal of*

Petroleum Science and Engineering, 211, p. 110179. Available at: https://doi.org/10.1016/j.petrol.2022.110179.

Zheng, A. (1998) 'Nonlinear model predictive control of the Tennessee Eastman process', in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*. *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, pp. 1700–1704 vol.3. Available at: https://doi.org/10.1109/ACC.1998.707296.

Zheng, H., Ricardez-Sandoval, L. and Budman, H. (2020) 'Robust estimation and economic predictive control for dynamic metabolic flux systems under probabilistic uncertainty', *Computers & Chemical Engineering*, 140, p. 106918. Available at: https://doi.org/10.1016/j.compchemeng.2020.106918.

Zheng, Y. *et al.* (2023) 'Physics-informed recurrent neural network modeling for predictive control of nonlinear processes', *Journal of Process Control*, 128, p. 103005. Available at: https://doi.org/10.1016/j.jprocont.2023.103005.

Appendices

Appendix A — Supplementary Material for Chapter 3

This section presents the results of replicating the test from section 3.4.3 for a MSc-NMPC with two levels of the uncertain parameter with the uncertainty reduced from 5% to 1%. The uncertain parameter was the kinetics of the first reaction, $\boldsymbol{e}^{NOM} = [\alpha_1^{NOM}]$, thus, the two scenarios considered in MSc-NMPC are the kinetics 1% above the nominal value and 1% below. The uncertain parameter in the plant was the nominal value. It was observed that reducing the uncertainty causes the process to settle much faster than in Figure 16. This is shown in Figure A.1, which shows the set-point tracking performance of the reactor pressure and the mixing-zone temperature. This suggests that the long settling times observed in Figure 16 may be due to the nonlinearity of the process. For non-linear systems, the worst-case scenario does not necessarily lie at the extreme values of the uncertain parameters. Since the plant uses the nominal value of the parameter, this is not one of the scenarios considered within the MSc-NMPC formulation, this results in the observed deteriorated performance with slower settling time.

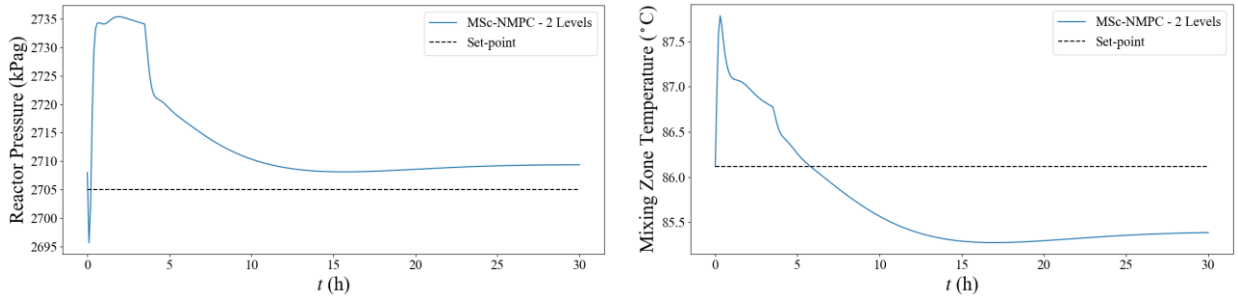


Figure A.1 Set-point tracking performance of MSc-NMPC with 1% uncertainty in the kinetics of the first reaction. The plant uses the nominal values of the parameters.

Appendix B — Steam Reformer Model Kinetics and Thermodynamics

The kinetics of the reactions occurring in the steam reformer are modelled by equations (B.1)-(B.3). Here, RR_j' is the reaction rate of reaction j based on catalyst weight. The rate constant of reaction j is given by $k_{rxn,j}$.

$$RR_1' = \frac{k_{rxn,1}}{P_{\mathrm{H_2}}^{2.5}}\left(P_{\mathrm{CH_4}}P_{\mathrm{H_2O}} - \frac{P_{\mathrm{H_2}}^3 P_{\mathrm{CO}}}{K_1}\right)/(DEN)^2 \quad \text{(B.1)}$$

$$RR_2' = \frac{k_{rxn,2}}{P_{\mathrm{H_2}}}\left(P_{\mathrm{CO}}P_{\mathrm{H_2O}} - \frac{P_{\mathrm{H_2}}P_{\mathrm{CO_2}}}{K_2}\right)/(DEN)^2 \quad \text{(B.2)}$$

$$RR_3' = \frac{k_{rxn,3}}{P_{\mathrm{H_2}}^{3.5}}\left(P_{\mathrm{CH_4}}P_{\mathrm{H_2O}}^2 - \frac{P_{\mathrm{H_2}}^4 P_{\mathrm{CO_2}}}{K_3}\right)/(DEN)^2 \quad \text{(B.3)}$$

The DEN term is defined in equation (B.4) which depends on the absorption constants $K_{a,s}$. The equilibrium constants of reaction j are given by K_j. The dependence on temperature of the rate constants and adsorption constants is modelled by the Arrhenius equation, (B.5) and (B.6), respectively.

$$DEN = 1 + K_{a,\mathrm{CH_4}}P_{\mathrm{CH_4}} + K_{a,\mathrm{H_2}}P_{\mathrm{H_2}} + K_{a,\mathrm{CO}}P_{\mathrm{CO}} + K_{a,\mathrm{H_2O}}P_{\mathrm{H_2O}}/P_{\mathrm{H_2}} \quad \text{(B.4)}$$

$$k_{rxn,j} = k_{rxn,j}^0 \exp\left(-\frac{E_{A_j}}{RT}\right) \quad \forall j \in \{1,2,3\} \quad \text{(B.5)}$$

$$K_{a,s} = B_{a,s} \exp\left(-\frac{\Delta H_{a,s}}{RT}\right) \quad \forall s \in \{\mathrm{CH_4}, \mathrm{H_2O}, \mathrm{H_2}, \mathrm{CO}\} \quad \text{(B.6)}$$

The temperature dependence of the equilibrium constant is modelled by the Van't Hoff equation, (B.7). The constant $K_{0,j}$ is the equilibrium constant at the reference temperature, T_{ref}, which can be calculated from the standard Gibb's free energy of reaction ΔG_j^0 via equation (B.8). The constant ΔG_j^0 can be calculated from the Gibb's free energy of formation, ΔG_s^f, via equation (B.9). The enthalpy of reaction ΔH_j is calculated from the standard enthalpy of reaction ΔH_j^0 and the heat

capacities of each component $C_{p,s}$ via equation (B.10). The constant ΔH_j^0 is calculated from the enthalpies of formation ΔH_s^f via equation (B.11). The dependence on temperature of the heat capacities is given by equation (B.12).

$$K_j = K_{0,j} \exp\left(\frac{1}{R}\int_{T_{ref}}^{T} \Delta H_j dT\right) \quad \forall j \in \{1,2,3\} \tag{B.7}$$

$$K_{0,j} = \exp\left(-\frac{\Delta G_j^0}{RT_{ref}}\right) \quad \forall j \in \{1,2,3\} \tag{B.8}$$

$$\Delta G_j^0 = \sum_s \nu_{js} \Delta G_s^f \quad \forall j \in \{1,2,3\} \tag{B.9}$$

$$\Delta H_j = \Delta H_j^0 + \int_{T_{ref}}^{T} \sum_s \nu_{js} C_{p,s}\, dT \quad \forall j \in \{1,2,3\} \tag{B.10}$$

$$\Delta H_j^0 = \sum_s \nu_{js} \Delta H_s^f \quad \forall j \in \{1,2,3\} \tag{B.11}$$

$$C_{p,s} = \alpha_s + \beta_s T + \gamma_s T^2 + \delta_s T^3 + \varepsilon_s T^4 \quad \forall s \in \{\mathrm{CH_4}, \mathrm{H_2O}, \mathrm{H_2}, \mathrm{CO_2}, \mathrm{CO}\} \tag{B.12}$$

To reduce the space of feasible initial conditions for the steam reformer model, the steady-state flow rate profile at the nominal operation was treated as the nominal initial condition. Then, the initial state space of each component was restricted to be between a lower and an upper profile, i.e., $F_{s,0}^L(z) \leq F_{s,0}(z) \leq F_{s,0}^U(z)$, which were determined by varying the manipulated variables between their lower and upper bounds. Figure B.1 shows the bounds on the initial states of the steam reforming model.

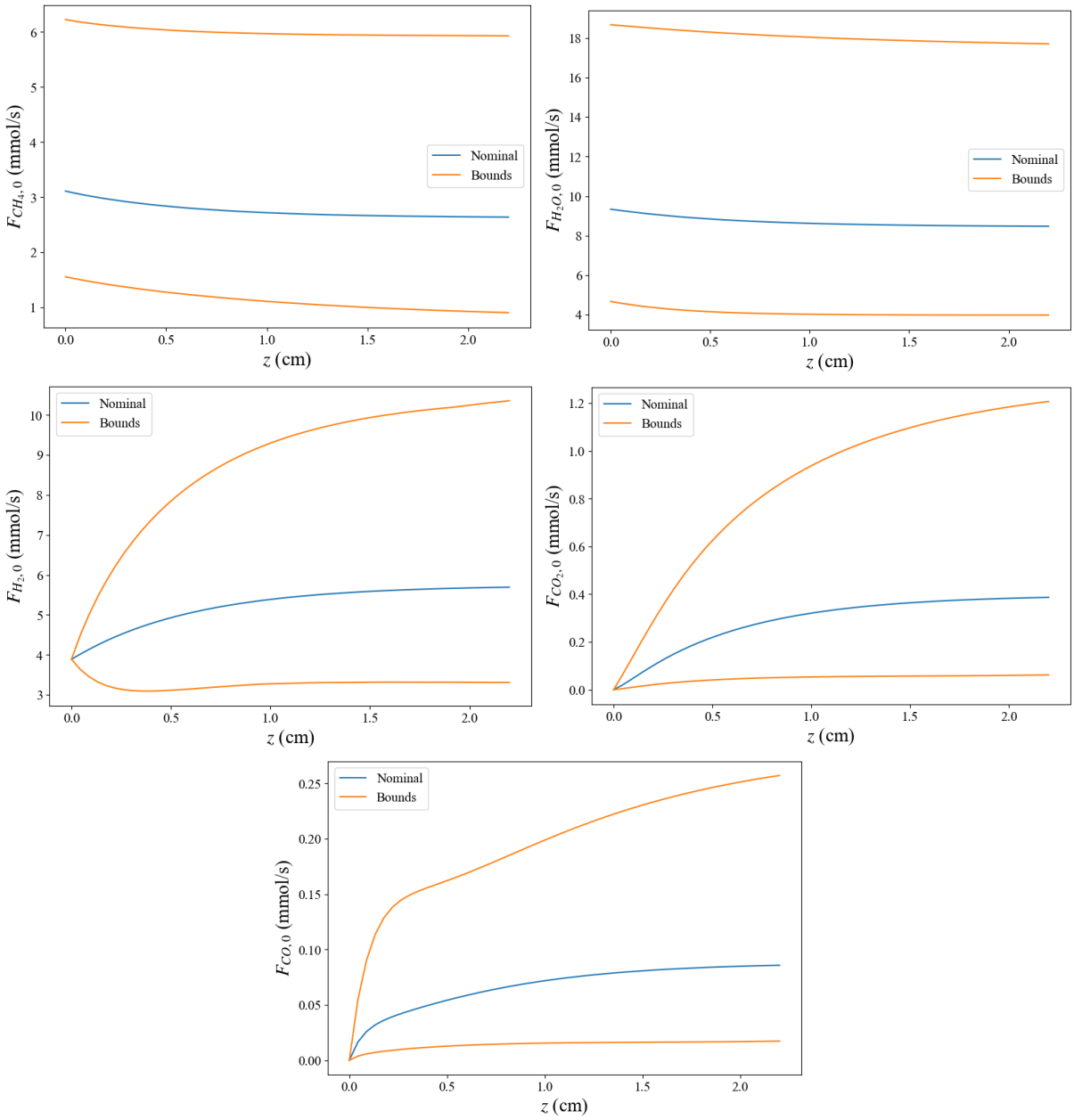


Figure B.1 Initial state bounds for steam reforming model.