

Master's thesis

Physics-Consistent Surrogate Models with Uncertainty Quantification for Model Predictive Control

Jan-David Ridder

Matr. 217599

First examiner: Prof. Dr.-Ing. Sergio Lucia
Second examiner: Prof. Dr.-Ing Hannsjörg Freund
Advisor: Niklas Kemmerling, M.Sc.
Year: 2025
Ident.: BCI-PAS-2025-M01

Technische Universität Dortmund
Faculty of Biochemical and Chemical Engineering
Laboratory for Process Automation Systems

Eidesstattliche Versicherung

(Affidavit)

Name, Vorname
(surname, first name)

Bachelorarbeit
(Bachelor's thesis)

Titel
(Title)

Matrikelnummer
(student ID number)

Masterarbeit
(Master's thesis)

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin“) zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:

Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification.*

Ort, Datum
(place, date)

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**

Kurzzusammenfassung

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Abstract

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Contents

Abstract	v
Notation	ix
List of Figures	xiii
List of Tables	xv
1 Introduction	1
2 Model Predictive Control under Uncertainty	3
2.1 Robust Optimization	3
2.2 Model Predictive Control	4
2.3 Multi-Stage Model Predictive Control	5
2.4 Neural Networks	7
2.4.1 Quantile Regression	8
2.5 NARX Models	8
2.5.1 Input Compression	9
2.5.2 Singular Value Decomposition	9
2.6 Uncertainty Quantification	11
2.7 Conservation of Atoms	12
3 Surrogate Modelling	13
3.1 Framework for general Uncertainty Quantification	13
3.2 NARX Framework for State Constraints	14
3.2.1 Quantile Conditioning	14
3.2.2 Weight Function	15
3.3 Physics-Constrained Neural Networks	16
4 Case Study	21
4.1 Reaction Kinetics	21
4.2 Packed-bed Tubular Reactor Model	22
4.2.1 PDE System Derivation	23
4.2.2 Discretization	24
4.3 Parametric Uncertainty	25
4.3.1 Radial bed heat conductivity	26
4.3.2 Activation Energies and Pre-exponential factors	26
4.4 Conservation of Atoms	26

4.5	Measurement Positions	28
4.6	Experimental Setup	29
4.6.1	Data Generation	29
4.6.2	Training	31
5	Results on Surrogate Model Performance	33
5.1	Training Effectivity	33
5.2	Physics Consistency	33
6	Conclusions	39
A	Datasheets	41
A.1	Parameter for	41
B	Proofs	43
B.1	Proof of Theorem X	43
	Bibliography	45

Notation

Numbers and Arrays

a	A scalar (integer or real)
\mathbf{a}	A vector
$\mathbf{a}_{[1:N]}$	A sequence of vectors from 1 to N , such that $\mathbf{a}_{[1:N]} = (\mathbf{a}_1, \dots, \mathbf{a}_N)$
\mathbf{A}	A matrix
\mathbf{A}	A tensor
I_n	Identity matrix with n rows and n columns
I	Identity matrix with dimensionality implied by context
$e^{(i)}$	Standard basis vector $[0, \dots, 0, 1, 0, \dots, 0]$ with a 1 at position i
$\text{diag}(\mathbf{a})$	A square, diagonal matrix with diagonal entries given by \mathbf{a}
\mathbf{a}	A scalar random variable
\mathbf{a}	A vector-valued random variable
\mathbf{A}	A matrix-valued random variable

Sets and Graphs

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

$\mathbb{A} \setminus \mathbb{B}$	Set subtraction, i.e., the set containing the elements of \mathbb{A} that are not in \mathbb{B}
\mathcal{G}	A graph
$Pa_{\mathcal{G}}(\mathbf{x}_i)$	The parents of \mathbf{x}_i in \mathcal{G}

Indexing

a_i	Element i of vector \mathbf{a} , with indexing starting at 1.
a_{-i}	All elements of vector \mathbf{a} except for element i
$A_{i,j}$	Element i,j of matrix \mathbf{A}
$\mathbf{A}_{i,:}$	Row i of matrix \mathbf{A}
$\mathbf{A}_{:,i}$	Column i of matrix \mathbf{A}
$A_{i,j,k}$	Element (i,j,k) of a 3-D tensor \mathbf{A}
$\mathbf{A}_{:,:,i}$	2-D slice of a 3-D tensor
\mathbf{a}_i	Element i of the random vector \mathbf{a}

Linear Algebra Operations

\mathbf{A}^\top	Transpose of matrix \mathbf{A}
\mathbf{A}^+	Moore-Penrose pseudoinverse of \mathbf{A}
$\mathbf{A} \odot \mathbf{B}$	Element-wise (Hadamard) product of \mathbf{A} and \mathbf{B}
$\det(\mathbf{A})$	Determinant of \mathbf{A}

Calculus

$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\mathbf{x}} y$	Gradient of y with respect to \mathbf{x}
$\nabla_{\mathbf{X}} y$	Matrix derivatives of y with respect to \mathbf{X}
$\nabla_{\mathbf{X}} y$	Tensor containing derivatives of y with respect to \mathbf{X}
$\frac{\partial f}{\partial \mathbf{x}}$	Jacobian matrix $\mathbf{J} \in \mathbb{R}^{m \times n}$ of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$
$\nabla_{\mathbf{x}}^2 f(\mathbf{x})$ or $\mathbf{H}(f)(\mathbf{x})$	The Hessian matrix of f at input point \mathbf{x}

$\int f(\mathbf{x}) d\mathbf{x}$	Definite integral over the entire domain of \mathbf{x}
$\int_{\mathbb{S}} f(\mathbf{x}) d\mathbf{x}$	Definite integral with respect to \mathbf{x} over the set \mathbb{S}

Probability and Information Theory

$a \perp b$	The random variables a and b are independent
$a \perp b c$	They are conditionally independent given c
$P(a)$	A probability distribution over a discrete variable
$p(a)$	A probability distribution over a continuous variable, or over a variable whose type has not been specified
$a \sim P$	Random variable a has distribution P
$\mathbb{E}_{x \sim P}[f(x)]$	Expectation of $f(x)$ with respect to $P(x)$
$\text{Var}(f(x))$	Variance of $f(x)$ under $P(x)$
$\text{Cov}(f(x), g(x))$	Covariance of $f(x)$ and $g(x)$ under $P(x)$
$H(x)$	Shannon entropy of the random variable x
$D_{\text{KL}}(P \ Q)$	Kullback-Leibler divergence of P and Q
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over \mathbf{x} with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f : \mathbb{A} \rightarrow \mathbb{B}$	The function f with domain \mathbb{A} and range \mathbb{B}
$f \circ g$	Composition of the functions f and g
$f(\mathbf{x}; \boldsymbol{\theta})$	A function of \mathbf{x} parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\mathbf{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation)
$\log x$	Natural logarithm of x
$\sigma(x)$	Logistic sigmoid, $\frac{1}{1 + \exp(-x)}$
$\zeta(x)$	Softplus, $\log(1 + \exp(x))$
$\ \mathbf{x}\ _p$	L^p norm of \mathbf{x}
$\ \mathbf{x}\ $	L^2 norm of \mathbf{x}
x^+	Positive part of x , i.e., $\max(0, x)$

Datasets and Distributions

p_{data}	The data generating distribution
\hat{p}_{data}	The empirical distribution defined by the training set
\mathbb{X}	A set of training examples
$\mathbf{x}^{(i)}$	The i -th example (input) from a dataset
$y^{(i)}$ or $\mathbf{y}^{(i)}$	The target associated with $\mathbf{x}^{(i)}$ for supervised learning
\mathbf{X}	The $m \times n$ matrix with input example $\mathbf{x}^{(i)}$ in row $\mathbf{X}_{i,:}$

List of Figures

2.1	Multi-stage MPC scenario tree	6
2.2	PCA encoder for feature reduction	11
3.1	Neural network with physics constrained activation function g_{pc}	18
4.1	Catalytic packed-bed flow reactor	23
4.2	Parameter distribution for the radial heat conductivity.	27
4.3	Parameter distribution for the activation energy and the collision factor. .	27
4.4	Measurement positions in the tubular reactor.	28
4.5	NARX framework for uncertainty quantification.	30
4.6	Excited input trajectories for the four wall temperatures.	31
5.1	Physics violation of the vanilla NARX	34
5.2	Physics violation of the naive physics-constrained NARX	35
5.3	Physics violation of the physics-constrained NARX	36

List of Tables

4.1 True kinetic parameters.	25
A.1 Mean and covariance of the parameters	41

Chapter 1

Introduction

In the chemical process industry, system models are a fundamental tool for design and operation. Due to its techno-scientific nature, chemical engineering is about modelling physical and chemical phenomena in an accurate enough manner to realize a save and profitable conversion process. These models vary in complexity and prediction potency from single equations such as the Bernoulli equation to detailed partial differential equation systems such as multi-phase Navier-Stokes. Physico-chemical models are at the core of optimal design or optimal control of most unit operations.

As these *first principle models* rely on intrinsic parameters such as activation energies, friction coefficients, etc. that are usually fitted by experiments, they exhibit some kind of prediction uncertainty. This uncertainty is heavily tied to the measurement uncertainty of the underlying experiments and the non-linearity in the propagation by the model. For example parametric uncertainties of $\pm 50\%$ can occur in multi-component reaction systems [1]. As most phenomena in chemical engineering exhibit strong non-linearities, such as reaction kinetics or activity coefficients, uncertainty quantification represents a great challenge.

Model Predictive Control (MPC) is a potent optimal control algorithm that allows the optimal operation of various systems. The algorithm optimizes a custom loss or reward function under a set of constraints and the dynamics of the system model. The optimal inputs of the system are determined at every time instant as the solution of the optimization. MPC is able to deal with uncertain parameter scenarios in the form of *multi-stage MPC*, where all possible scenarios are optimized over simultaneously. This leads to tremendous computational effort especially for detailed system models. The computational effort grows multiplicatively with the amount of parameters and exponentially with the robust horizon [1]. These optimization tasks are impossible to solve in real time with current hardware.

Neural Networks (NNs) are able to approximate any continuous function that maps from a finite dimensional space to another up to an arbitrary small error [1]. Thus, NNs are a key component of machine learning advancements of the 2010s enabling data approximation of almost any kind such as autoregressive language generation [2], protein folding [1] or reduction of partial differential equations (PDEs). Their universal approximation ability, differentiability and fast evaluation makes NNs greatly suitable to reduce first principle models in the context of optimal control.

For NNs to be applied successfully and safely in process engineering, another measure is of great importance – The alignment of predictions with physical conservation laws such

as mass or energy. By definition, NN predictions seek to minimize a loss function to the ground truth data. The alignment of the predictions with real balance equations is only achieved until test accuracy. Deviations accumulate especially in propagation scenarios where predictions are propagated through different networks or through a time horizon in autoregressive manner. This leads to bad predictions or even unstable behavior in late propagation stages and makes MPC implementations challenging.

This work investigates the usage of physics-consistent (PC) NNs in the context of robust model predictive control under parametric uncertainty. The NNs in focus shall significantly reduce the computational effort due to different parameter scenarios as well as guarantee physical correctness of the predictions over the whole prediction horizon. To account for uncertainties, quantile regression is utilized which captures a prediction confidence interval. PC is enforced by implementing a recent type of activation function that elegantly maps a prediction into the allowed output space without altering the inference time [3].

The developed surrogate models are later evaluated regarding their simulation accuracy using a fictional plug flow tubular reactor (PFTR) model of the ethene oxide synthesis.

Chapter 2

Model Predictive Control under Uncertainty

Before diving into the application of a special model predictive control (MPC) algorithm. Uncertain optimization is treated more generally. From economics, over self-driving cars to chemical plants, optimization problems are formulated in terms of uncertain variables in countless variations. These uncertainties arise in three different types – fluctuating disturbances such as wind or weather conditions, parametric offsets such as wrongly determined friction coefficients or model mismatches like unseen side reactions. As the straight maximization or minimization of an objective often puts a problem to its boundaries, unforeseen disturbances may lead to *constraint violation* or even *instability* [4]. Therefore, *robust* optimization formulations are needed, that are proof against mentioned challenges as well as retain the *performance* with respect to the objective function.

2.1 Robust Optimization

The following optimization task (2.1) under an uncertainty vector \mathbf{w} may be considered.

Definition 2.1. Semi-infinite program with the uncertainty \mathbf{w} .

$$\begin{aligned} & \min_{\mathbf{u}} \max_{\mathbf{w}} F_0(\mathbf{u}, \mathbf{w}) \\ \text{s.t. } & F_i(\mathbf{u}, \mathbf{w}) \leq 0 \quad \forall \mathbf{w} \in \mathbb{W} \end{aligned}$$

Here, the optimization of the objective is formulated in a bilevel form. The scalar objective function F_0 is to be minimized w.r.t the decision variables \mathbf{u} for the worst case scenario w.r.t the disturbance vector \mathbf{w} . It can be interpreted as a player-vs-player setup [4]. One player (the optimizer) seeks a minimum solution for the input \mathbf{u} , while the opposing player (nature) chooses a maximum solution for the disturbance variables \mathbf{w} . The inequality constraints F_i in addition, should be satisfied for all possible disturbances, that lie in a defined set \mathbb{W} . If this set is not finite, the number of possible constraints to be satisfied is infinitely big. Thus, this formulation is named as *semi-infinite-program* [5]. To reduce the number of constraints, the worst case estimate can be applied here (2.2).

Definition 2.2. Bi-level formulation of the semi-infinite program

$$\begin{aligned} & \min_{\mathbf{u}} \max_{\mathbf{w}} F_0(\mathbf{u}, \mathbf{w}) \\ \text{s.t. } & \varphi_i(\mathbf{u}) := \max_{\mathbf{w}} F_i(\mathbf{u}, \mathbf{w}) \leq 0 \quad \mathbf{w} \in \mathbb{W} \end{aligned}$$

Now, the inequality constraints are also formulated in a bi-level manner. The maximum of F_i is constant in \mathbf{w} and only a function of the decision variables \mathbf{u} . The function φ_i exactly describes the worst case for F_i parametrized by \mathbf{u} . In fact, if certain assumptions hold regarding the uncertainty set \mathbb{W} and the concavity of the maximization, φ_i can be exactly computed [6]. This would simplify the problem dramatically. In many real cases however, an analytical solution for φ_i is impossible to compute. But with the universal approximation capability of neural networks (NNs) in mind [7], a data-driven approach is promising, that delivers an approximated function $\tilde{\varphi}_i(\mathbf{u})$.

2.2 Model Predictive Control

Model predictive control (MPC) is a type of optimization, in which an optimal control problem is solved in a dynamic manner. The main goal of this control strategy is to find the best inputs to a system that maximize or minimize a defined objective. In a chemical process, for instance, the energy consumption for heating should be minimized. It should be minimized in such a way, that other variables remain within their bounds. For example, the product specification and, most importantly, the safety of the process needs to be ensured under all circumstances. In contrast to the *optimal design* of a process, that is done during the engineering phase, an MPC controller operates *online*. It acts on the process while it is running. Thus, it is coined *dynamic optimization*. To do so, the optimization problem needs to be solved recursively at given time instants. Chemical plants usually have large *time constants*, meaning, they react slowly to their inputs. Other physical systems with low time constants, like quadrocopters behave much faster. So, the computation time of an MPC algorithm is to be kept low [8] to be able to react to the system. To predict the future behavior of the system, MPC requires a system model.

In total, an MPC controller is based on three pillars – an objective function, a set of constraints and, crucially, the model of the system.

Definition 2.3. Continuous-time formulation of an MPC problem.

$$\begin{aligned} & \min_{\mathbf{u}} J(t_0, t_{\text{end}}, \mathbf{x}, \mathbf{u}) \\ \text{s.t. } & \dot{\mathbf{x}} = f(t, \mathbf{x}, \mathbf{u}) \\ & F_i(t, \mathbf{x}, \mathbf{u}) \leq 0 \quad \forall t \in [t_0, t_{\text{end}}] \\ & \mathbf{x}_0 = \mathbf{x}_{\text{init}} \end{aligned}$$

Optimal system inputs \mathbf{u} are to be determined, that minimize the objective function J . This objective varies with the inputs and the states of the system \mathbf{x} . It is to be minimized over a time horizon from t_0 to t_{end} , called the *prediction horizon*. The states of the system evolve along the dynamics of the system, which are described by the system model $f(t, \mathbf{x}, \mathbf{u})$. Thus, the quality of the controller heavily depends on the prediction accuracy of this model [9]. The system states however should remain in a set of

constraints at all times whereas the inputs are bounded, all summarized by the constraint functions F_i . In a physical sense, constraint violation can lead to instability and thus damage of the real system. For instance, a *temperature runaway* must be avoided at all costs in a chemical reactor. This happens, if the exothermic heat release exceeds the maximum possible cooling rate. Higher temperatures increase the reaction rate exponentially leading to a greater enthalpy release, which again causes a self-accelerating temperature rise [10]. That is why, stability and guaranteed mathematical constraint satisfaction are an important field of research in the discipline of optimal control.

2.3 Multi-Stage Model Predictive Control

Coming from this general perspective on robust optimization and basic MPC, *robust MPC* specializes *robustness* to a dynamic closed-loop control environment.

Definition 2.4. Continuous-time formulation of an MPC task under uncertainty.

$$\begin{aligned} & \min_{\boldsymbol{u}} \max_{\boldsymbol{w}} \quad J(t_0, t_{\text{end}}, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \\ \text{s.t. } & \dot{\boldsymbol{x}} = f(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \\ & F_i(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \leq 0 \quad \forall t \in [t_0, t_{\text{end}}], \forall \boldsymbol{w} \in \mathbb{W} \\ & \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} \end{aligned}$$

The only difference to the basic MPC formulation (2.3) is that the model, the objective and the constraints are functions of the uncertainty, denoted with \boldsymbol{w} . This MPC formulation (def. 2.4) is now infinite in time and in the uncertain parameters. To make computation feasible, the problem can be discretized in the time dimension either using single shooting, multiple shooting or orthogonal collocation on finite elements [11]. One possibility to roughly discretize the uncertain parameter set \mathbb{W} is done by *multi-stage-MPC* [12].

It generates a scenario tree (fig. 2.1) for all possible combinations of disturbances. In the algorithm design phase, scenario cases $w_k^{(\cdot)}$ must be defined a priori for each independent disturbance that may act on the system. As different disturbances occur in different combinations, each possible scenario is represented by a branch. At every time step in the *robust horizon*, branches for every possible uncertainty scenario are generated. This procedure is coined *scenario generation*. In practice, the worst cases for the disturbances are assumed plus an optional expected value. Mostly, this yields good results with respect to controller performance and robustness [9].

If the uncertainty is discrete and all cases are considered, the scenario tree formulation (fig. 2.1) is exact. In most applications however, when the uncertainties lie in a continuous space, the scenario tree is an extremely rough approximation. With a non-linear system, the scenarios are not monotonous regarding the disturbances. This means, even if the scenario branches cover the extreme values of the disturbances themselves, they can leave extreme scenarios caused by intermediate parameter combinations. Robust constraint satisfaction is not guaranteed and must be checked via a reachability analysis [12]. This can be an obstacle in safety-relevant applications. Another drawback

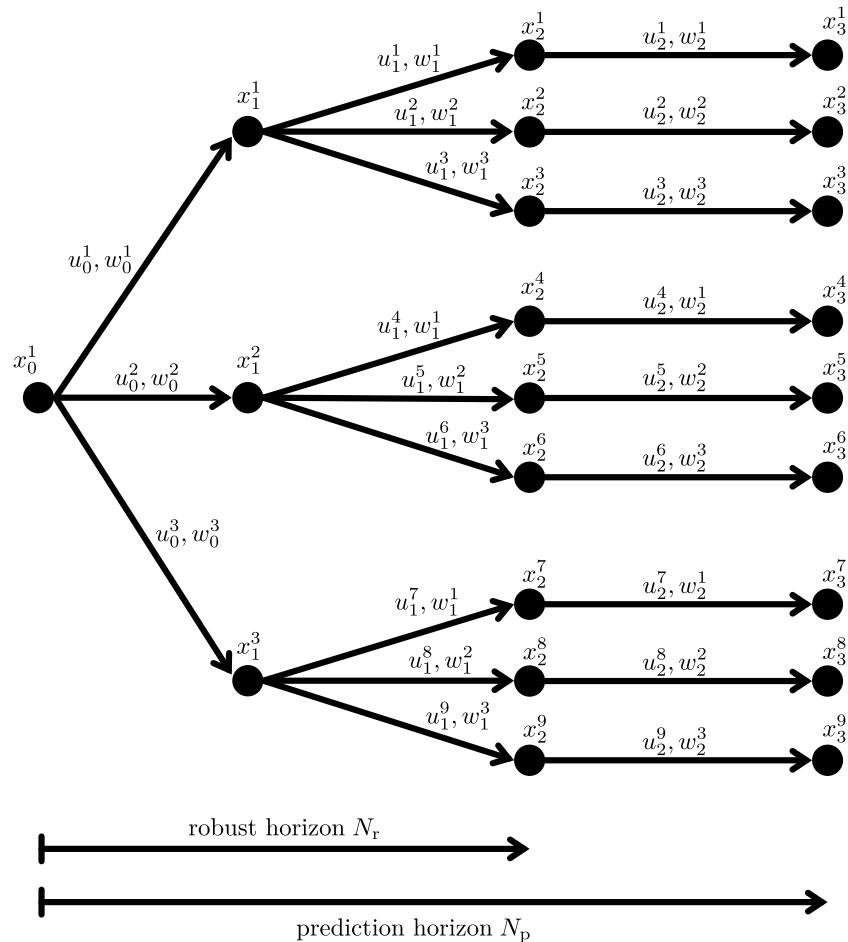


Figure 2.1: Multi-stage MPC discretizes the uncertainty set via user-defined possible cases $w_k^{(\cdot)}$. All possible case combinations are branched through the robust horizon N_r . To limit the complexity, the uncertainties are considered constant thereafter for the rest of the prediction horizon N_p .

of multi-stage MPC is, that it includes redundant scenarios not causing extreme cases due to the non-monotonicity. These scenarios still are part of the optimization leading to unnecessary complexity. On top of this, the number of optimization variables growths exponentially with the *robust horizon* N_r , the number of parameters and their respective values (eq. 2.1) [1].

$$n_s = \left(\prod_i^{n_w} \nu_i \right)^{N_r} \quad (2.1)$$

Here, n_w denotes the number of different disturbances and ν_i their respective discrete values. n_s describes the number of scenarios in the scenario tree (fig. 2.1). High order uncertainties lead to optimization problems that are simply impossible to solve in real time.

2.4 Neural Networks

Feed forward neural networks (NNs), sometimes referred to as *multi-layer-perceptrons* (MLPs), provide a stunning capability of approximating any multivariate function from any finite dimensional space to another. According to the *Universal Approximation Theorem*, any finite dimensional function can be reproduced if the number of hidden layers and respective neurons is high enough, up to an arbitrarily small error [7]. Thus, NNs play a major role in recent developments in machine learning. Large language models such as ChatGPT heavily rely on neural networks [2]. NNs are not only successfully applied in natural language processing but in the chemical process industry. Applications range from thermodynamic property prediction over unit operation design to corporate level decision making [13]. Their ability to approximate any non-linear function while being differentiable, makes NNs especially suitable for dynamic optimization such as MPC. In contrast to detailed first principle models, NNs have the advantage of making a prediction in a fraction of the simulation time of the first principle model. As MPC operates in real time parallel to the system to be controlled, the computation duration is a key measure for a successful MPC implementation. That is why, surrogate models in form of NNs are a crucial tool for dynamic optimization and heavily investigated in current research [14]. May a NN be given as an alternating series of function evaluations (def. 2.5).

Definition 2.5. Mathematical formulation of a neural network as a nested function evaluation.

$$\begin{aligned} \mathcal{NN}(\xi, \theta) &= g_{l+1} \circ h_{l+1} \circ \dots \circ g_1 \circ h_1(\xi) \\ h(\mathbf{a}_{l-1}) &= \theta_{Wl} \mathbf{a}_{l-1} + \theta_{bl} \\ \mathbf{a}_l &= g_l(h(\mathbf{a}_{l-1})) \end{aligned}$$

In fact, a NN is a series of function combinations denoted by the \circ -symbol alternating between a linear function $h(\mathbf{a})$ and a *non-linear* activation function $g(h(\mathbf{a}))$. The linear transformation matrices θ_{Wl} and offsets θ_{bl} , are called weights and biases respectively and are the trainable parameters of a NN. The activation functions g are important to introduce non-linearity into every linear projection. Otherwise, the complete function combination can be written as one singular linear transformation and the universal approximation capability is lost. Given a set of training data of m data points, the

optimal parameters for $\boldsymbol{\theta}$ are found by solving an optimization task that involves the minimization of a prediction loss function (eq. 2.2).

$$\begin{aligned}\mathbb{X}_{\text{train}} &:= \{\boldsymbol{\xi}^{(0)}, \dots, \boldsymbol{\xi}^{(m-1)}\} \\ \mathbb{Y}_{\text{train}} &:= \{\mathbf{y}^{(0)}, \dots, \mathbf{y}^{(m-1)}\} \\ \mathcal{L}(\boldsymbol{\theta}, \mathbb{X}_{\text{train}}, \mathbb{Y}_{\text{train}}) &= \frac{1}{m} \sum_{i=0}^{m-1} \left\| \mathbf{y}^{(i)} - \tilde{\mathbf{y}}(\boldsymbol{\theta}, \boldsymbol{\xi}^{(i)}) \right\|_2^2\end{aligned}\tag{2.2}$$

In *supervised learning*, the training data is separated in *features* $\mathbb{X}_{\text{train}}$, the inputs of the NN, and *labels* $\mathbb{Y}_{\text{train}}$. During training, the parameters $\boldsymbol{\theta}$ are determined such that the NN approximates the data in best possible manner. In this example the loss function \mathcal{L} portrays the mean squared error (MSE), that is used in most regression tasks. The difference between the model prediction, denoted as $\tilde{\mathbf{y}}(\boldsymbol{\xi}^{(i)})$, and the true data point $\mathbf{y}^{(i)}$ is penalized quadratically.

2.4.1 Quantile Regression

By utilizing a loss function in form of the MSE, a model is trained to achieve regression predictions in form of the squared mean. To tweak the prediction characteristics of a NN for the task at hand, the type of training loss function is decisive. In a probabilistic environment, the prediction of confidence intervals in the form of quantiles may be of major interest. To learn the quantiles of a distribution of data points, the *quantile loss* or *pinball loss* can be used (eq. 2.3).

$$\mathcal{L}_\tau = \begin{cases} \tau \cdot (y - \tilde{y}(\boldsymbol{\theta})), & y \geq \tilde{y}(\boldsymbol{\theta}) \\ (1 - \tau) \cdot (\tilde{y}(\boldsymbol{\theta}) - y), & y < \tilde{y}(\boldsymbol{\theta}) \end{cases}\tag{2.3}$$

The hyperparameter τ denotes the type and therefore position of the quantile. A small value for τ , e.g. 0.05 or 0.1, strongly penalizes data points that are below the threshold $y < \tilde{y}(\boldsymbol{\theta})$ via the $(1 - \tau)$ -term to keep the quantile small. A high value for τ , e.g. 0.95, focuses on the penalization of data points above the threshold $y \geq \tilde{y}(\boldsymbol{\theta})$ to cover a greater amount of points. For robust optimization, *quantile regression* can be applied to learn upper or lower bounds for the uncertainty-dependent constraints $F_i(\mathbf{u}, \mathbf{w})$ (eq. 2.2) in terms of confidence quantiles. This would reduce the semi-infinite program to an uncertainty-finite-program.

2.5 NARX Models

Because of their universal approximation capability [7] and their differentiability, NNs are well suited as *surrogate models* in MPC algorithms to approximate the system behavior represented by the function f (def. 2.3). This can significantly reduce the solution time in contrast to discretized systems of ordinary differential equations (ODEs) [8].

Definition 2.6. Time-discrete form of a NN to predict the future state, based on past information.

$$\begin{aligned}\mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k) \\ \tilde{\mathbf{x}}_{k+1} &= \mathcal{NN}(\mathbf{x}_k, \mathbf{x}_{k-1}, \dots, \mathbf{x}_{k-h}, \mathbf{u}_k, \mathbf{u}_{k-1}, \dots, \mathbf{u}_{k-h})\end{aligned}$$

In the discrete-time notation, the state at the next time instant \mathbf{x}_{k+1} is calculated by propagating the current state \mathbf{x}_k through a function $f(\mathbf{x}_k, \mathbf{u}_k)$ that describes the system dynamic (def. 2.6). In the original MPC formulation (def. 2.3) this is some numerical integration scheme for a system of differential equations [11] with the abbreviation *right-hand-side* (rhs). The state propagation by the function f is a classical example of a time series extrapolation. NNs have shown significant capabilities in time series forecasting. Deep learning based architectures exist for time series forecasting in various complexities. Convolutional NNs (CNNs), Long-Short-Term-Memory (LSTMs) cells and Transformers are the most prominent for time series prediction with varying complexity [2], [15]. These provide a great possibility to apply deep learning based forecasting within optimal control algorithms, where the next state $\tilde{\mathbf{x}}_{k+1}$ is the output of a neural network $\mathcal{N}\mathcal{N}$. In context of this work, shallow feedforward NNs or *single-layer perceptrons* (SLPs) are used for this purpose which technically do belong to *shallow learning* architectures.

In the realm of control, the non-linear system identification is summarized with the term *Non-linear AutoRegressive with eXogenous inputs* (NARX). Here, the term *autoregressive* means, that the future prediction is a function of a series of past system states $\{\mathbf{x}_k, \mathbf{x}_{k-1}, \dots, \mathbf{x}_{k-h}\}$, while *exogenous* implies, that a series of outside inputs $\{\mathbf{u}_k, \mathbf{u}_{k-1}, \dots, \mathbf{u}_{k-h}\}$ is passed to the function as well (def. 2.6). The hyperparameter h denotes the length of the time horizon of the past series. A high value for this parameter includes more past information, which possibly allows better predictions while increasing the network input. Between states and inputs, h can be chosen differently. In this work, h is set equal for states and inputs.

2.5.1 Input Compression

$$\phi_k := (\mathbf{x}_k, \mathbf{x}_{k-1}, \dots, \mathbf{x}_{k-h}, \mathbf{u}_k, \mathbf{u}_{k-1}, \dots, \mathbf{u}_{k-h}) \quad (2.4)$$

The concatenated input feature vector ϕ_k contains all past states and inputs within the time horizon h , leading to a vector length of $(n_x + n_u) \cdot (h + 1)$, where n_x and n_u represent the number of states and inputs respectively. For large systems, the input length increases the network size unacceptably quick, while the input time series often exhibit strong redundancies. To reduce the input size, the feature vector ϕ_k is passed to an *encoder* [16]. The encoder is based on feature compression via *principal component analysis* (PCA), that maps the feature space to a lower dimensional subspace. This subspace represents the space of the input data with the biggest variance.

2.5.2 Singular Value Decomposition

Singular value decomposition (SVD) is a fundamental building block in linear algebra. Its applications range from image compression, over exploratory data analysis to model reduction in fluid mechanics. The power of SVD is, that it maps a high dimensional space into a low dimensional subspace. The mapping is achieved by a simplistic matrix transformation. It not only identifies a subspace, it also identifies underlying patterns in the data encoded via the respective *singular vectors*. The compression and component identification is guaranteed to exist for every matrix $\mathbf{X} \in \mathbb{C}^{n \times m}$ in contrast to an eigendecomposition [17].

The SVD is extremely useful, when high dimensional data is presented, that embeds some kind of pattern. For example, simulations of partial differential equations (PDEs), images as collections of pixels and time serieses exhibit strong patterns within the data. This is especially true for data points in close temporal or spatial distance. Because the SVD is the basis for PCA to compress the feature vector ϕ_k and *model order reduction* (MOR) of ODEs or PDEs it is explained in further detail. According to the SVD, any matrix $\mathbf{X} \in \mathbb{C}^{n \times m}$ can be decomposed into three submatrices (th. 2.1).

Theorem 2.1. *Singular value decomposition of a data matrix \mathbf{X} .*

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^*$$

Here, the matrices $\mathbf{U} \in \mathbb{C}^{n \times n}$ and $\mathbf{V}^* \in \mathbb{C}^{m \times m}$ are quadratic, orthonormal matrices. The real, non-negative, diagonal matrix $\Sigma \in \mathbb{R}^{n \times m}$ contains the so called *singular values* of the matrix with the biggest as the top-left entry. The notation \mathbf{V}^* stands for the transposed complex-conjugate of \mathbf{V} . The computation in `python` can be done in `numpy` via the method `linalg.svd()` or directly in a training environment with `torch.linalg.svd()`. For large data matrices \mathbf{X} , that do not fit into memory, a batched or incremental SVD must be considered. A current `torch` implementation with GPU support exists under `torch_incremental_pca` [18]. The compression itself is done by truncating the yielded matrices by taking the first r columns of \mathbf{U} , the first r rows of \mathbf{V}^* and r diagonal elements of Σ (eq. 2.5).

$$\mathbf{X} \approx \Psi \tilde{\Sigma} \tilde{\mathbf{V}}^* \quad (2.5)$$

with $\Psi := \mathbf{U}_{:,r} \quad \tilde{\Sigma} = \Sigma_{:,r} \quad \tilde{\mathbf{V}}^* = \mathbf{V}_{:,r}^*$

This has the advantage, that an approximation of the data \mathbf{X} can be stored via the truncated matrices $\Psi, \tilde{\Sigma}, \tilde{\mathbf{V}}^*$ which take up less memory than the original matrix. It also allows to project the original matrix into a subspace of dimension r (eq. 2.6).

$$\tilde{\mathbf{X}} = \Psi^\top \mathbf{X} \iff \mathbf{X} \approx \Psi \tilde{\mathbf{X}} \quad (2.6)$$

The truncated matrix Ψ acts as the transformation matrix between the low dimensional space and the original space (eq. 2.6). It is important to note, that the back transformation into the original space only yields an approximation of the original data. It can be proven, that this approximation is the best existing one with respect to the Frobenius Norm as it solves following minimization [17] (eq. 2.7).

$$\begin{aligned} \Psi &= \underset{\Psi}{\operatorname{argmin}} \quad \|\mathbf{X} - \Psi \Psi^\top \mathbf{X}\|_F \\ &\text{s.t.} \quad \operatorname{rank}(\Psi) = r \end{aligned} \quad (2.7)$$

Reentering the concept of neural network NARX models, the large vector of input features ϕ_k (eq. 2.4) can be compressed via SVD into a low dimensional feature vector ξ_k that retains the most dominant basis vectors of the original input (fig. 2.2). This is done by an *PCA encoder* placed before the first layer of a NARX model to reduce the network size.

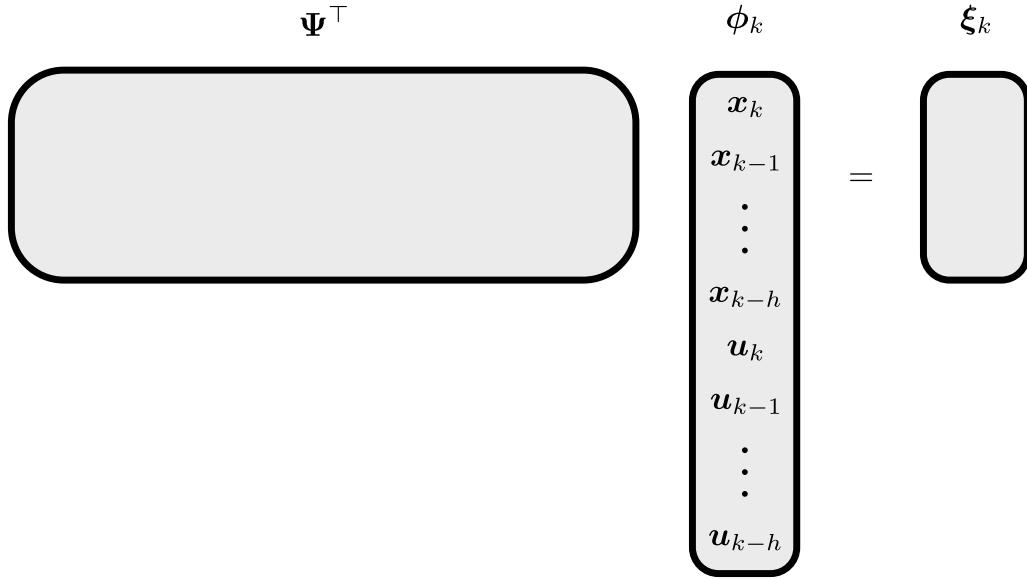


Figure 2.2: The PCA encoder uses SVD to derive a projection matrix Ψ that maps the high dimensional feature vector ϕ_k to a lower dimensional feature vector ξ_k which is fed into a NARX model. This reduces the network size significantly, while retaining the most important information about the input.

2.6 Uncertainty Quantification

Furthermore, *quantile regression* (2.4.1) is used to quantify the uncertainty of system states. Under model uncertainty, the evolvement of the system described by $f(\mathbf{x}, \mathbf{u}, \mathbf{w})$ heavily depends of the uncertainty vector $\mathbf{w} \in \mathbb{W}$ (def. 2.4). While multi-stage MPC roughly discretizes the parameter space for parameter extreme cases under tremendous computational effort, quantile regression can be used to directly discretize the space at given quantiles τ . This has the advantage, that the uncertainty quantification is done independently from the disturbances. As non-linear systems behave non-monotonous to uncertainties, extreme parameter cases do not necessarily account for system extreme cases. Explicit quantiles bound the system states with a given coverage lumping all possible disturbance effects together. Another advantage is, that the uncertainty quantification is done offline by training a quantile model beforehand instead of account for uncertainties during MPC runtime. This outsources computational effort out of the dynamic optimization loop.

Upper and lower NARX quantile models can be trained, to predict a confidence interval that includes predictions of future states with a given coverage. In total, three predictions must be made to simulate the system forward in time while quantifying the uncertainty. These consist of the expected value of the states at the next instant, which is denoted as $\mathbb{E}(\mathbf{x}_{k+1}|\phi_k)$, and two quantiles, $Q_\tau(\mathbf{x}_{k+1}|\phi_k)$, with pre-defined probability coverages τ . All are predicted given the feature vector ϕ_k of past states and inputs. In this work, models that predict the expected value \mathbb{E} are trained using the *mean squared*

error as loss function (eq. 2.2). Quantile models Q_τ are trained using the *quantile loss* function (eq. 2.3)

2.7 Conservation of Atoms

The derived methodology will later be applied to control a *packed-bed tubular reactor* under parametric uncertainty which is a common realization for many chemical processes. For this sake, the *conservation of atoms* during chemical reactions is introduced. Similar to the conservation of mass or energy, the conservation of atoms demands, that all atoms that enter a chemical reaction must also leave this reaction. While they change the species they form, the atoms themselves are conserved. This can be expressed as follows.

Theorem 2.2. *Conservation of atoms during a chemical reaction.*

$$\mathbf{B}\Delta\mathbf{n} = \mathbf{0}$$

Here, \mathbf{B} denotes the *element-species-matrix* and $\Delta\mathbf{n}$ the changes in the molar amounts of all participating specieses in a reaction network. The sum of all atoms in all species must be equal at all times regardless of how the underlying reactions take place. This builds the foundation of reaction network analysis and the determination of key and non-key-components [10].

$$\mathbf{B} = \begin{pmatrix} \beta_{1,1} & \dots & \beta_{1,\alpha} \\ \vdots & \ddots & \vdots \\ \beta_{h,1} & \dots & \beta_{h,\alpha} \end{pmatrix} \quad (2.8)$$

The element-species-matrix contains the elements h as rows and the species α as columns. $\beta_{h,\alpha}$ describes the number of atoms of element h in the species α . This implies, that when the initial molar amounts of all species are known, all possible combinations of molar amounts must lie in a pre-defined space.

Chapter 3

Surrogate Modelling

In this work, several methodologies are combined and assessed together to implement a robust MPC algorithm for systems that exhibit uncertainty. First, NARX models are implemented to simulate a system forward in time. Second, *quantile*-NARX models are trained to simulate confidence intervals forward in time. Third, a new type of activation function is added to all mentioned NARX architectures to enforce their alignment with physical balance equations. Later, these surrogate models are compared w.r.t training efficiency, physics consistency and uncertainty quantification capability. This is done with the intention to use these surrogate models within an MPC optimization loop (def. 2.4).

3.1 Framework for general Uncertainty Quantification

To be able, to simulate the surrogate models forward in time, following iterative formulation is set up as the RHS.

Definition 3.1. General RHS equation for uncertainty quantification.

$$\begin{pmatrix} \mathbf{x}^\uparrow \\ \mathbf{x}_{\text{nom}} \\ \mathbf{x}^\downarrow \end{pmatrix}_{k+1} = \begin{pmatrix} \hat{Q}_{\tau_1}(\mathbf{x}_{k+1} | \boldsymbol{\phi}_k^\uparrow) \\ \mathbb{E}(\mathbf{x}_{k+1} | \boldsymbol{\phi}_{\text{nom},k}) \\ \hat{Q}_{\tau_2}(\mathbf{x}_{k+1} | \boldsymbol{\phi}_k^\downarrow) \end{pmatrix} =: f_{\text{UQ}} \begin{pmatrix} \boldsymbol{\phi}_k^\uparrow \\ \boldsymbol{\phi}_{\text{nom},k} \\ \boldsymbol{\phi}_k^\downarrow \end{pmatrix}$$

with $\boldsymbol{\phi}_0^\uparrow = \boldsymbol{\phi}_{\text{nom},0} = \boldsymbol{\phi}_0^\downarrow = \boldsymbol{\phi}_{\text{init}}$

Here, \mathbb{E} , \hat{Q}_{τ_1} and \hat{Q}_{τ_2} are NARX models to predict the expected value and two quantiles for all states at the next time instant $k + 1$ with the given feature vector for past states and inputs $\boldsymbol{\phi}_k$. The states pertaining to the upper quantile function, are indicated with " \uparrow ". Similarly, the lower quantile states are denoted with a " \downarrow ", while "nom" refers to the nominal scenario. Technically, three RHS functions are propagated independently through time except for the initial state, which is the same for all three. Therefore, the feature vector (eq. 2.4) is shifted after each iteration using the resulting state vector of the according NARX model. The function notation is summarized as f_{UQ} . It is important to note, that the initial feature vector $\boldsymbol{\phi}_{\text{init}}$ contains $h + 1$ past states and measurements. Which must be known to initiate the RHS based a NARX

framework.

3.2 NARX Framework for State Constraints

When recalling the bi-level formulation of a semi-infinite program (def. 2.2), worst case functions $\varphi_i(\mathbf{u})$ are assumed to ensure robust constraint satisfaction. For simplicity, only state constraints in the form of $\mathbf{x}_k \leq 0 \forall k$ are considered. This resembles a great amount of cases, as constraints often appear as pure state constraints. So, the uncertainty quantification in form of a quantile NARX model only needs to cover the states, that are part of state constraints. Furthermore, quantile regression for states with counteracting dynamics lead to wrong predictions. Therefore it is reasonable to separate the prediction of quantile states from the prediction of all other states.

Definition 3.2. Modified function \hat{Q}_τ to predict quantiles of critical states and according expected values of all other states.

$$\hat{Q}_\tau(\mathbf{x}_{k+1}|\boldsymbol{\phi}_k) = \begin{pmatrix} Q_\tau(\mathbf{x}_{k+1,:n_{\text{crit}}}|\boldsymbol{\phi}_k) \\ \mathbb{E}_\tau(\mathbf{x}_{k+1,n_{\text{crit}}:}|\boldsymbol{\phi}_k) \end{pmatrix}$$

Only constraint critical states $\mathbf{x}_{:,n_{\text{crit}}}$ undergo real quantile regression by the function Q_τ . The quantity n_{crit} stands for the ending index of constraint critical states in the state vector \mathbf{x} . All other states are to be inferred such that they match the quantile prediction of the critical states. This is marked by an expected value \mathbb{E}_τ , that is conditioned by the quantile of the critical states. Concatenated together, these two NARX architectures allow a state propagation. Whether the predictions of the conditioned model are physical meaningful is questionable, because it is trained using some correlation between the quantile model's predictions and ground truth.

3.2.1 Quantile Conditioning

The quality of predictions made by \hat{Q}_τ heavily depends on the reliability of the expected values of those states, that are not inferred by quantile regression. To train a NARX model \mathbb{E}_τ , that learns the expected values for the other states that belong to the quantile predictions of the critical states, several different weighted loss functions can be used. These loss functions all have in common that they emphasize data points in close proximity to the quantile prediction and penalize data points with growing distance.

$$d = \|Q_\tau(\boldsymbol{\phi}_k) - \mathbf{x}_{k+1,:n_{\text{crit}}}\|_2 \quad (3.1)$$

The distances d of quantile predictions to the ground truth values at the next time instant is calculated as the l_2 -norm (eq. 3.1).

$$\mathcal{L}_w(\boldsymbol{\theta}) = \frac{1}{m} \sum_i^m w(d^{(i)}) \cdot \|\mathbf{x}_{k+1}^{(i)} - \mathcal{NN}(\boldsymbol{\theta}, \boldsymbol{\phi}_k^{(i)})\|_2^2 \quad (3.2)$$

These distances are used to calculate certain weights w for a weighted MSE to serve as the loss function for \mathbb{E}_τ (eq. 3.2). To train such a conditioned NARX model following steps must be executed (alg. 1).

Algorithm 1 Calculation of the quantile based weights for a weighted MSE.

Require: Ξ, \mathbb{X}, Q_τ

```

 $\mathbb{X}_{\text{crit}} \leftarrow \mathbb{X}_{:n_{\text{crit}}}$ 
 $\tilde{\mathbb{X}}_{\text{crit}} \leftarrow Q_\tau(\Xi)$ 
 $\mathbf{d} \leftarrow \|\mathbb{X}_{\text{crit}} - \tilde{\mathbb{X}}_{\text{crit}}\|_{(.)}$ 
 $\mathbf{w} \leftarrow w(\mathbf{d})$ 
return  $\mathbf{w}$ 

```

First, the quantile model Q_τ must have been trained. Ξ denotes the data matrix of encoded feature vectors and \mathbb{X} the data matrix of states at the next time instant. The vector \mathbf{d} contains the distances of the inferred quantile predictions to the ground truth in the critical states. These are passed to a weight function w to generate a weight vector \mathbf{w} , that collects all weights for all data points.

3.2.2 Weight Function

To weight the data points in the MSE using the distance measure d , a lot of possible functions exist. This is a straight tuning question. Inspired by the distributive character of uncertain states, that result from Gaussian distributed model parameters, a modified Gaussian is used to weight the MSE.

Definition 3.3. Gaussian-tube as weight function.

$$w(d) = \frac{2}{\sqrt{2\pi}\sigma_w} \cdot e^{-\frac{1}{2}\frac{d^2}{\sigma_w^2}} \quad d \in \mathbb{R}_+$$

The Gaussian strongly weights points with closest distance to the quantile predictions. As the distance increases, the weight first decays softly, than swiftly and finally tends to zero. It almost acts like an ϵ -tube with a soft boundary. The standard deviation σ_w acts as a hyperparameter to tune the width of the Gaussian-tube similarly to a radial basis function kernel with the length scale as the according hyperparameter. Small values for this parameter set greater emphasis on points in close proximity of the quantile prediction. Weight functions should have an integral value of one when integrated over the whole domain. As the domain of the distance measure is $d \in \mathbb{R}_+$, the Gaussian is simply scaled by two.

Using such an approach of weighting quantile predictions to condition another model to deliver expected values for all non-quantile states, can be a source of error. The training of the conditioned model \mathbb{E}_τ does not only rely on ground truth data, but on a correlation of a subjective distance measure d in some vector space and a weight function. This may lead to bad predictions in the sense of system tracking and physical consistency. To guarantee physical consistency, a new type of activation function is presented, that forces NARX predictions into the physically allowed space. This is done primarily to enhance the prediction quality of these conditioned models.

3.3 Physics-Constrained Neural Networks

By definition, neural networks (NNs) are designed to reproduce their training data in a best possible manner, defined by the characteristics of the loss function used during the training process. NN predictions minimize the loss towards the training data seen beforehand. Circumstances exist, in which the output of a NN should strictly satisfy a certain set of equations. In case of physical modeling, the prediction of a surrogate should follow the same conservation laws as a first principle model. Especially, when predictions are propagated through several NNs or simulated in an autoregressive manner, errors accumulate. Final predictions are untrustworthy or simulations exhibit instability [3]. As a MPC algorithm needs a reasonable approximation of the system behavior over the complete prediction horizon, the accumulation of conservation violations become an issue.

To realize physics consistent NN outputs, several strategies exists. Physics-informed neural networks (*PINNs*) for instance, embed physical balance equations into the loss function similar to a regularization term [19]. This serves as a soft constraint and comes with two downsides. The NN output does not strictly follow the constraint equation, because the physical consistency is fulfilled up to the trade off between data fit and constraint satisfaction. Every new balance equation to be included introduces a new hyperparameter worsening the ability to include network parameter regularization. To guarantee strict constraint satisfaction and thus perfect physics consistency up to machine precision, a new type of activation function is presented equivalently to the work by Chen et al. [3]. This activation function serves as a linear projection from the NN output space into the physical allowed subspace without adding neither learnable parameters nor hyperparameters. Let the output of a NN be defined as well as a set of linear equality constraints that summarize some physical conservation quantity (def. 3.4).

Definition 3.4. Crude NN output with a given set of linear equality constraints.

$$\begin{aligned} \tilde{\mathbf{y}}(\boldsymbol{\theta}, \boldsymbol{\xi}) &:= \mathcal{N}\mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\xi}) \\ \mathbf{A}(\mathbf{y} - \mathbf{z}_0) &= \mathbf{b} \\ \text{with } \tilde{\mathbf{y}}(\boldsymbol{\theta}, \boldsymbol{\xi}), \mathbf{z}_0 &\in \mathbb{R}^{n_{\text{out}}} \quad \mathbf{A} \in \mathbb{R}^{n_c \times n_{\text{out}}} \quad \mathbf{b} \in \mathbb{R}^{n_c} \end{aligned}$$

$\tilde{\mathbf{y}}$ denotes the NN output, $\boldsymbol{\xi}$ the input and the matrices \mathbf{A} and \mathbf{b} portray the feasible space of points. The variable \mathbf{z}_0 illustrates an external input into the constraint. This may be a NN input, a constant, or another batched reference point. In a physical sense, \mathbf{z}_0 is often introduced as a boundary or initial condition. A feasible point \mathbf{y} now needs to be determined that fulfills the set of linear equality constraints. For a straight forward implementation, the formulation appears in such a way that it supports batching of a certain batch size n_B . n_{out} and n_c describe the number of output features and numbers of linear equality constraints respectively. The main idea is, to find a point in the feasible subspace that is closest to the NN prediction. This is forced by following quadratic programm (QP) (def. 3.5).

Definition 3.5. Minimization of the euclidian distance towards the NN output as a linear equality constrained optimization that supports batching.

$$\begin{aligned} \mathbf{y}^* &= \underset{\mathbf{y}}{\operatorname{argmin}} \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 \\ \text{s.t. } (\mathbf{y} - \mathbf{z}_0) \mathbf{A}^\top - \mathbf{b} &= \mathbf{0} \end{aligned}$$

This means, the output of the network $\tilde{\mathbf{y}}$, the optimal solution \mathbf{y}^* and \mathbf{z}_0 are matrices of dimension $n_B \times n_{\text{out}}$. Computing frameworks that support broadcasting such as `numpy` or `pytorch` allow the implementation of \mathbf{z}_0 in the shape of $1 \times n_{\text{out}}$ to reduce memory usage as long as it remains constant [1]. The transposed constraint matrix \mathbf{A}^\top must be of shape $n_c \times n_B$. The vector \mathbf{b} represents the offset of the linear equality constraints. This optimization task can be solved analytically by setting up the *Karush-Kuhn-Tucker* (KKT) conditions (th. 3.1).

Theorem 3.1. *Lagrangian and KKT conditions of the QP*

$$\begin{aligned} \mathcal{L}(\mathbf{y}, \boldsymbol{\nu}) &= \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 + \boldsymbol{\nu}^\top (\mathbf{A}(\mathbf{y} - \mathbf{z}_0)^\top - \mathbf{b}) \\ \nabla_{\mathbf{y}} \mathcal{L}(\mathbf{y}, \boldsymbol{\nu}) &= 2(\mathbf{y}^* - \tilde{\mathbf{y}}) + \boldsymbol{\nu}^\top \mathbf{A} \stackrel{!}{=} \mathbf{0} \\ \nabla_{\boldsymbol{\nu}} \mathcal{L}(\mathbf{y}, \boldsymbol{\nu}) &= (\mathbf{y}^* - \mathbf{z}_0) \mathbf{A}^\top - \mathbf{b} \stackrel{!}{=} \mathbf{0} \end{aligned}$$

The Lagrangian \mathcal{L} arises as the sum of the objective function and all linear equality constraints. The vector $\boldsymbol{\nu}$ contains the dual variables of the linear equality constraints. The optimality conditions lead to a linear system of equations (cor. 3.1).

Corollary 3.1. *The optimal solution in the feasible space as projection of an activation function*

$$\begin{aligned} \begin{pmatrix} \mathbf{y}^* & (\boldsymbol{\nu}^*)^\top \end{pmatrix} \begin{pmatrix} 2\mathbf{I} & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{0} \end{pmatrix} &= \begin{pmatrix} 2\tilde{\mathbf{y}} & \mathbf{z}_0 \mathbf{A}^\top + \mathbf{b} \end{pmatrix} \\ g_{pc}(\tilde{\mathbf{y}}, \mathbf{z}_0) &\coloneqq \begin{pmatrix} 2\tilde{\mathbf{y}} & \mathbf{z}_0 \mathbf{A}^\top + \mathbf{b} \end{pmatrix} \begin{pmatrix} 2\mathbf{I} & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{0} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{y}^* & (\boldsymbol{\nu}^*)^\top \end{pmatrix} \end{aligned}$$

The optimal solution denoted with "/*" is directly yielded by projecting the NN output $\tilde{\mathbf{y}}$ using the additional input \mathbf{z}_0 and a projection matrix. This projection matrix only contains the constraint matrix \mathbf{A} can be computed before runtime which drastically speeds up forward and backward passes in contrast to implicit methods [20]. The solution elegantly serves as an activation function $g_{pc}(\tilde{\mathbf{y}}, \mathbf{z}_0)$, that maps the crude NN output into a subspace defined by \mathbf{A} and \mathbf{b} . The final output for \mathbf{y}^* satisfies the linear equality constraints by definition, while it is located as close to the crude output $\tilde{\mathbf{y}}$ as possible. This is not only applicable in the field of chemical engineering but a general formulation that allows to integrate linear equality constraints into NNs. Additionally, the type of network architecture is independent of the mapping. It can be appended to any network type such as recurrent NNs, convolutional NNs or transformer-based architectures.

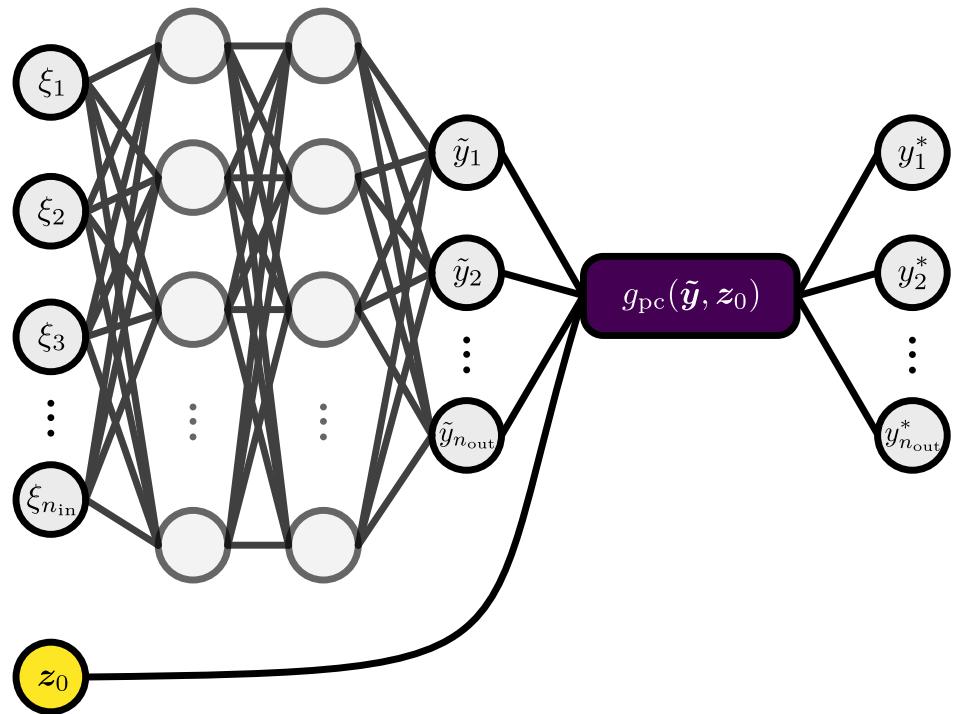


Figure 3.1: The physics constrained NN turns the input vector ξ into the crude output $\tilde{\mathbf{y}}$ via a feedforward NN, which is the projected by the physics constrained activation function g_{pc} into the feasible output \mathbf{y}^* . It allows to include data points such as boundary conditions as needed in the form of z_0

Due to its general applicability, the activation function could be used within a network to correct intermediate values. In most use cases however, it is applied as a final correction function at the output (fig. 3.1). The input ξ is propagated through a feedforward NN with the physics constrained activation function behind the last layer (purple). It takes the reference value z_0 (yellow) without being forwarded through the network. In recent literature, NNs that utilize such a type of activation function can be found as *KKT-hPINN* [3], which is short for *Karush-Kuhn-Tucker-hard-physics-informed-neural-network*. For later reference, this activation function g_{pc} will be referred to as *physics constrained activation* (cor. 3.1) to have a short name in the context of chemical engineering. Furthermore, this procedure could be extended to support arbitrary non-linear equality constraints by splitting the network output in frozen and unfrozen variables [21]. It enables the integration of enthalpy balances that are of non-linear nature. As this work combines the implementation of physics consistent NNs with parametric uncertainty for MPC, the focus remains on linear equality constraints.

Algorithm 2 Forward pass of the physics constrained activation function.

Require: $\mathbf{K} = \begin{pmatrix} 2\mathbf{I} & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{0} \end{pmatrix}^{-1}$, $\tilde{\mathbf{y}}, z_0, \mathbf{A}, \mathbf{b}$

$$\mathbf{h} \leftarrow \begin{pmatrix} 2\tilde{\mathbf{y}} & z_0\mathbf{A}^\top + \mathbf{b} \end{pmatrix}$$

$$\mathbf{y}^* \leftarrow \mathbf{h}\mathbf{K}$$

$$\mathbf{y}^* \leftarrow \mathbf{y}_{:,n_{out}}^*$$
return \mathbf{y}^*

The forward pass through the physics constrained activation involves the setup of a temporary vector \mathbf{h} (alg. 2) and the truncation of the dual variables in the solution. The matrix inversion to yield \mathbf{K} should be done offline and then saved as untrainable model parameter. In `pytorch` this can be done using the `register_buffer()` method. An implementation example of the physics consistent activation can be found in the appendix (XX).

Chapter 4

Case Study

The goal of this work is to derive a physics-consistent surrogate modelling technique to be applied within a robust MPC algorithm. Physics consistency, prediction accuracy and uncertainty quantification will be assessed using a case study of a catalytic *packed-bed tubular reactor*. This reactor type is common in the chemical industry in many processes such as ethylene oxidation, carbon dioxide methanization or steam reforming. Especially with regard to *load flexible* processes using renewable energy sources, dynamic control such as MPC can enable optimal operation [24]. For example, the energy supply can be closely tied to weather conditions or the feed composition may fluctuate due to biological resources.

For exothermal reactions, *transient behavior* is a major safety concern of packed-bed tubular reactors [10]. In a dynamic operating environment, the thermal inertia of the catalytic packed bed can abruptly heat up unreacted feed mixture. This can cause a self-accelerating temperature runaway leading to thermal damage or even explosion. Furthermore, high temperatures can cause several catalyst deactivation effects such as sintering. Therefore, the temperature is considered a safety-crucial state that needs to obey to a maximum constraint in the sense of $F_i \leq 0$ in the original optimization (def. 2.2) for all possible disturbances \mathbf{w} . The maximum of F_i should be approximated by $\tilde{\varphi}_i(\mathbf{u})$. This will be achieved by training a NARX model that predicts a tuned upper and lower bound for $\tilde{\varphi}_i(\mathbf{u})$ independent from the uncertainty \mathbf{w} using quantile regression. Apart from safety, the maximization of the selectivity towards the value product is a reasonable control objective, as the selectivity is most correlated to process profitability in many cases. To achieve a reasonable turnover from educts to products, a minimum conversion is set to be required via a constraint. In total, following general optimization structure is proposed.

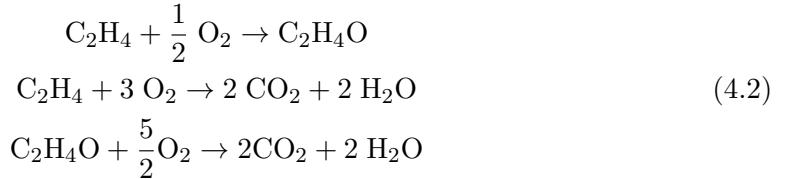
$$\begin{aligned} & \max S \\ \text{s.t. } & T \leq T_{\max} \\ & X \geq X_{\min} \end{aligned} \tag{4.1}$$

4.1 Reaction Kinetics

To focus on the surrogate modelling methodology, the example system is kept rather simplistic. The *ethylene oxidation* process consists of only two parallel reactions. Parametric

data is readily available due to extensive literature about this reaction system. Further, the reaction and reactor configuration is taken from an optimal reactor design study by Pietschak et al. [22] and adjusted for the needs at hand. Thus, some proposed operating conditions may be unrealistic due to safety concerns or over-simplifications.

Ethene is partially oxidized to ethylene oxide, while the total oxidation (combustion) of ethene takes place as a parallel reaction to carbon dioxide and water (eq. 4.2). According to the kinetics of Al-Saleh et al. [23], the consecutive reaction from ethylene oxide to water and carbon dioxide can be neglected.



The reaction rates can be described using following rate expression (eq. 4.3).

$$r_j = \frac{k_j p_{\text{E}}^{n_j^{(\text{E})}} p_{\text{O}_2}^{n_j^{(\text{O}_2)}}}{1 + K_j p_{\text{CO}_2}} \quad (4.3)$$

Here, r_j denotes the reaction rate of reaction j , k_j the collision factor, p_α the partial pressures of component α and K_j the adsorption coefficient for carbon dioxide. The reaction orders w.r.t ethene and oxygen are expressed by $n_j^{(\text{E})}$ and $n_j^{(\text{O}_2)}$ respectively. The collision factor and the adsorption coefficient follow simple Arrhenius correlations (eq. 4.4, 4.5).

$$k_j = k_{0,j} \exp\left(\frac{-E_{\text{A},j}}{RT}\right) \quad (4.4)$$

$$K_j = K_{0,j} \exp\left(\frac{T_{\text{ads},j}}{T}\right) \quad (4.5)$$

These introduce a strong non-linearity in the temperature T and the kinetic parameters. The upright letter R represents the universal gas constant, E_{A} the activation energy, $k_{0,j}$ and $K_{0,j}$ the respective pre-exponential factors and $T_{\text{ads},j}$ the adsorption temperature. To simulate these kinetics, the partial pressures of the rate determining species ethene, oxygen and carbon dioxide must be known. The kinetic parameters are listed below (tab. 4.1) and will further underly probabilistic uncertainty.

4.2 Packed-bed Tubular Reactor Model

The reactor as a packed-bed tubular reactor (TR) is assumed to be already existend with a fixed design to clearly separate the model derivation from an optimal design task (fig. 4.1). To ensure thermal stability, the additional enthalpy balance is mandatory.

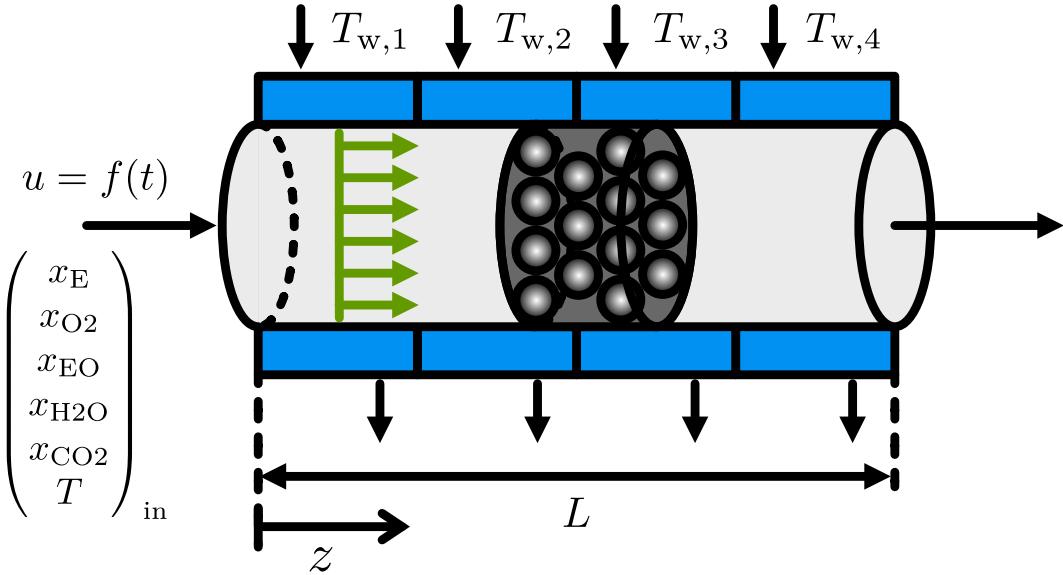


Figure 4.1: The packed-bed reactor is assumed to exhibit plug flow behavior (green). Catalyst and gaseous phase are lumped into one pseudo-homogenous reaction phase. The inlet conditions remain constant for the temperature T and the molar fractions of each component $x_{(\cdot)}$, while the inlet flow speed u can fluctuate due to flexible feed availability. Four cooling sections (blue) are integrated.

4.2.1 PDE System Derivation

The technical reactor itself is equipped with four wall cooling sections ($T_{w,1}$ to $T_{w,4}$) to manipulate the temperature profile (blue). The input flow velocity u can fluctuate due to flexible feed availability such as in carbon dioxide methanization, where the throughput heavily relies on the hydrogen supply [24]. Because isobaric conditions and a non-compressible phase are assumed, the simulated states narrow down to the molar fractions of the rate-determining species x_α and the temperature T . These are considered to be constant at the inlet, which is important for the later application of linear equality constraints in a NN. Using the component balance in local form (eq. 4.6), a PDE can be derived that describes the dynamic concentration gradients of the components α . Furthermore, the enthalpy balance shall be covered to calculate the temperature field regarding the exothermic reactions.

$$\frac{\partial \rho_\alpha}{\partial t} = -\frac{\partial}{\partial z_k} (\rho_\alpha u_k + j_{\alpha,k}) + \sigma_\alpha \quad (4.6)$$

The general component density ρ_α is expressed in mol/m³, while ∂z_k indicates the spatial derivative in direction k . u_k and $j_{\alpha,k}$ denote the convective velocity and the diffusive mass transport in direction k . The source or sink term σ_α allows to consider chemical reactions. Ideal gas behavior is considered (eq. 4.7).

$$p_\alpha = \rho_\alpha R T \quad (4.7)$$

Moreover, *plug flow* and constant pressure are assumed. Under the consideration of a 1D-pseudo-homogenous model with gradients only in axial direction and no diffusive mass transfer, this equation reduces to an ideal *plug flow tubular reactor* (PFTR) (eq. 4.8). A detailed derivation of the model equations can be found in the appendix (XX).

$$\frac{\partial \chi_\alpha}{\partial t} = -u \frac{\partial \chi_\alpha}{\partial z} + (1 - \epsilon) \frac{RT_*}{px_*} \sum_j \nu_{\alpha,j} r_j(\chi_\alpha, T') \quad (4.8)$$

$$\text{with } \chi_\alpha = \frac{x'_\alpha}{T'} \quad x'_\alpha = \frac{x_\alpha}{x_*} \quad T' = \frac{T}{T_*}$$

To improve numerical stability, the mole fractions x_α and the temperature occur in a dimensionless form indicated by a prime (' symbol). This is done by diving by a reference value, denoted with a star (*) (eq. 4.8). It not only improves numerical simulation stability. For the later training of surrogate models, it also eliminates the need for an *output scalar* which simplifies implementation and training. These reference values are defined as follows (eq. 4.9).

$$x_* = x_{E,\text{in}} \quad T_* = T_{\text{in}} \quad (4.9)$$

Due to the ideal gas behavior (eq. 4.7), the molar density ρ_α varies not only with the molar fraction but also with the temperature. Therefore, an artificial state χ_α is introduced to enable an easy and explicit formulation of the PDE via substitution (eq. 4.8).

The general enthalpy balance in local form (eq. 4.10) can be significantly reduced under isobaric conditions with neither volume forces, nor friction, nor dispersive, nor conductive enthalpy transfer.

$$\frac{\partial(\rho h)}{\partial t} - \frac{\partial p}{\partial t} = -\frac{\partial}{\partial z_k} \left(\rho h u_k + q'_k \right) + u_k \frac{\partial p}{\partial z_k} + \sum_i j_{k,i} f_{k,i} - \Pi_{j,k} \frac{\partial u_j}{\partial z_k} + \sigma_h \quad (4.10)$$

The final PDE to describe the temperature field in the reactor is yielded (eq. 4.11).

$$\frac{\partial T'}{\partial t} = -u \frac{\partial T'}{\partial z} + \frac{1}{\rho c_p} \left(\frac{1 - \epsilon}{T_*} \sum_j (-\Delta h_{R,j}) r_j + \frac{4}{d_t} \alpha \left(\frac{T_w}{T_*} - T' \right) \right) \quad (4.11)$$

Important parameters are the bed void fraction ϵ , the tube diameter d_t , the enthalpy of reaction $\Delta h_{R,j}$, the total density ρ and the total heat capacity c_p . The wall heat transfer is assumed to only be governed by the fluid-wall heat transfer coefficient α , which is a function of the fluid velocity and the radial heat conductivity of the catalytic bed (eq. 4.12). In fact, this parameter abbreviated with λ_{bed} is crucial for the temperature behavior of the TR. Underlying dependecies like Reynolds and Prandl expressions are part of the appendix to keep the derivation tidy.

$$\alpha = f(\text{Re}, \text{Pr}, \lambda_{\text{bed}}) \quad (4.12)$$

4.2.2 Discretization

The system of non-linear PDEs is discretized in z -direction via *backward finite differences*.

Table 4.1: True kinetic parameters for the ethene oxidation kinetics [23].

parameter	value	
	main reaction	side reaction
$k_{0,j}$	$6.275 \times 10^6 \text{ mol kg}_{\text{cat}}^{-1} \text{ Pa}^{1.1} \text{ s}^{-1}$	$1.206 \times 10^7 \text{ mol kg}_{\text{cat}}^{-1} \text{ Pa s}^{-1}$
$E_{\text{A},j}$	$74\,900 \text{ J mol}^{-1}$	$89\,900 \text{ J mol}^{-1}$
$K_{0,j}$	$1.985 \times 10^2 \text{ Pa}^{-1}$	$1.08 \times 10^2 \text{ Pa}^{-1}$
$T_{\text{ads},j}$	2400 K	1530 K
$\Delta h_{\text{R},j}$	$-1.07 \times 10^5 \text{ J mol}^{-1}$	$-1.323 \times 10^6 \text{ J mol}^{-1}$
$n_j^{(\text{E})}$	0.6	0.5
$n_j^{(\text{O}_2)}$	0.5	0.5

Definition 4.1. Discretized system of PDEs using backward finite differences to yield an ODE system

$$\begin{aligned} \frac{d\chi_{\alpha,i}}{dt} &= -u \frac{\chi_{\alpha,i} - \chi_{\alpha,i-1}}{\Delta z} + (1-\epsilon) \frac{RT_*}{px_*} \sum_j \nu_{\alpha,j} r_j(\chi_{\alpha,i}, T'_i) \\ \frac{dT'_i}{dt} &= -u \frac{T'_i - T'_{i-1}}{\Delta z} + \frac{1}{\rho c_p} \left(\frac{1-\epsilon}{T_*} \sum_j (-\Delta h_{\text{R},j}) r_j(\chi_{\alpha,i}, T'_i) + \frac{4}{d_t} \alpha \left(\frac{T_{\text{w},i}}{T_*} - T'_i \right) \right) \end{aligned}$$

The nodes of the discretization are chosen to be equidistant. For a reactor length of $L = 10 \text{ m}$, 128 points are used. This results in a length of $\Delta z = 0.0781 \text{ m}$ for each element. As the finite difference method exhibits a significant truncation error of $\mathcal{O}(\Delta z)$, more accurate discretization methods such as finite volumes should be used for a real application.

4.3 Parametric Uncertainty

Because model parameters such as the activation energy or the pre-exponential collision factor are mostly fitted by laboratory experiments that exhibit some kind of measurement uncertainty, the parameters themselves cannot be exactly determined. They lie in a confidence interval. Furthermore, fitting results with counteracting mechanisms produce cross-correlations between parameters. For example, when a higher activation energy is determined, simultaneously the according pre-exponential factor must be also higher to compensate for the reduced reaction speed. In this case study, five first principle model parameters should be treated as uncertain. These are the radial bed heat conductivity λ_{bed} , the activation energies $E_{\text{A},1}, E_{\text{A},2}$ and their respective pre-exponential collision factors $k_{0,1}, k_{0,2}$ for the main, and the side reaction. The true values for all kinetic parameters are listed (tab. 4.1).

4.3.1 Radial bed heat conductivity

The radial bed heat conductivity is set to follow a normal distribution with a mean of $0.404 \text{ W m}^{-1} \text{ K}^{-1}$ calculated by a correlation by Zehner et al. [25] and a rather high standard deviation of $0.08 \text{ W m}^{-1} \text{ K}^{-1}$ (fig. 4.2). Its values range from 0.2 to $0.6 \text{ W m}^{-1} \text{ K}^{-1}$.

Definition 4.2. Normal distribution of the radial heat conductivity of the catalytic bed

$$\lambda_{\text{bed}} \sim \mathcal{N}(\mu_\lambda, \sigma_\lambda)$$

4.3.2 Activation Energies and Pre-exponential factors

The activation energies and pre-exponential factors are assumed to arise from a multi-variate normal distribution, which can be determined from a cross-correlated parameter fitting (def. 4.3).

Definition 4.3. Multi-variate normal distribution of cross-correlated kinetic parameters

$$\begin{pmatrix} \ln k_{0,1} \\ E_{A,1} \\ \ln k_{0,2} \\ E_{A,2} \end{pmatrix} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

The activation energy of the main reaction is plotted against its pre-exponential collision factor (fig. 4.3). While the true parameter combination is marked with a yellow point, the blue crosses indicate the possible parameter combinations by sampling the multi-variate normal distribution (eq. 4.3). The sampling is done 1000 times. Activation energies can range between $65\,000$ and $85\,000 \text{ J mol}^{-1}$, whereas the pre-exponential factor varies between 1.5×10^6 and $3.0 \times 10^7 \text{ mol kg}_{\text{cat}}^{-1} \text{ Pa}^{1.1} \text{ s}^{-1}$. To calculate the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$, an actual parameter fitting with emulated measurement noise has been executed using a *berty* reactor model. Mean and covariance can be consulted in the appendix (tab. ??).

4.4 Conservation of Atoms

To formulate the conservation of atoms (th. 2.2) for the ethene oxidation, the element-species-matrix \mathbf{B} needs to be set up (def. 4.4).

Definition 4.4. Element-species-matrix for the ethene oxidation.

$$\mathbf{B} = \begin{pmatrix} 4 & 0 & 4 & 2 & 0 \\ 2 & 0 & 2 & 0 & 1 \\ 0 & 2 & 1 & 1 & 2 \end{pmatrix}$$

Here, the columns are ordered as follows: ethene, oxygen molecule, ethylene oxide, water and carbon dioxide. The rows indicate elementary hydrogen, elementary carbon

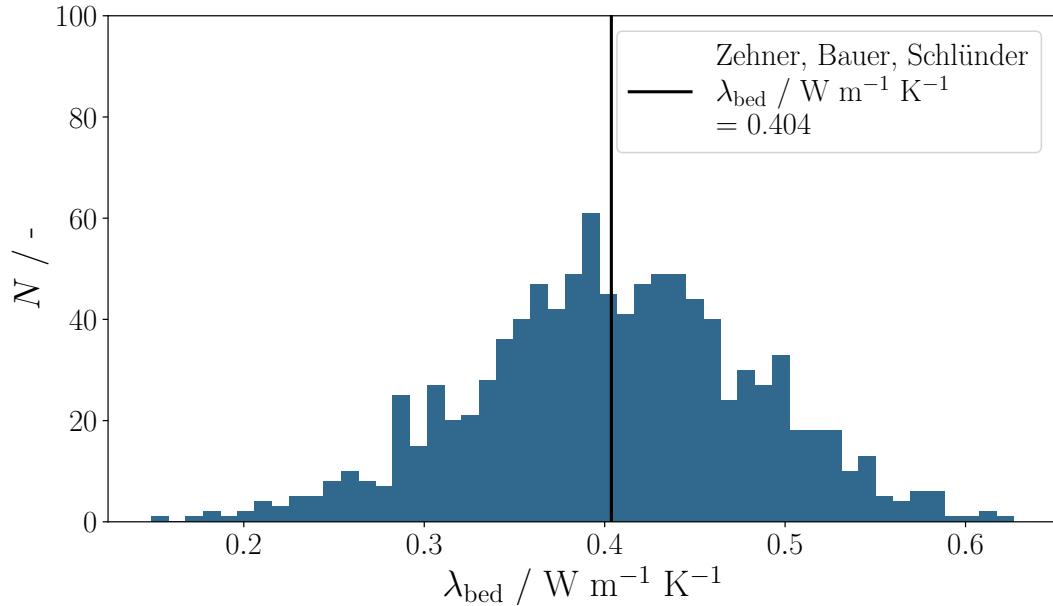


Figure 4.2: The radial heat conductivity of the packed bed follows a normal distribution with its mean calculated by the correlation of Zehner et al. [25] and a standard deviation of $\sigma = 0.08 \text{ Wm}^{-1}\text{K}^{-1}$, shown for 1000 sampling points.

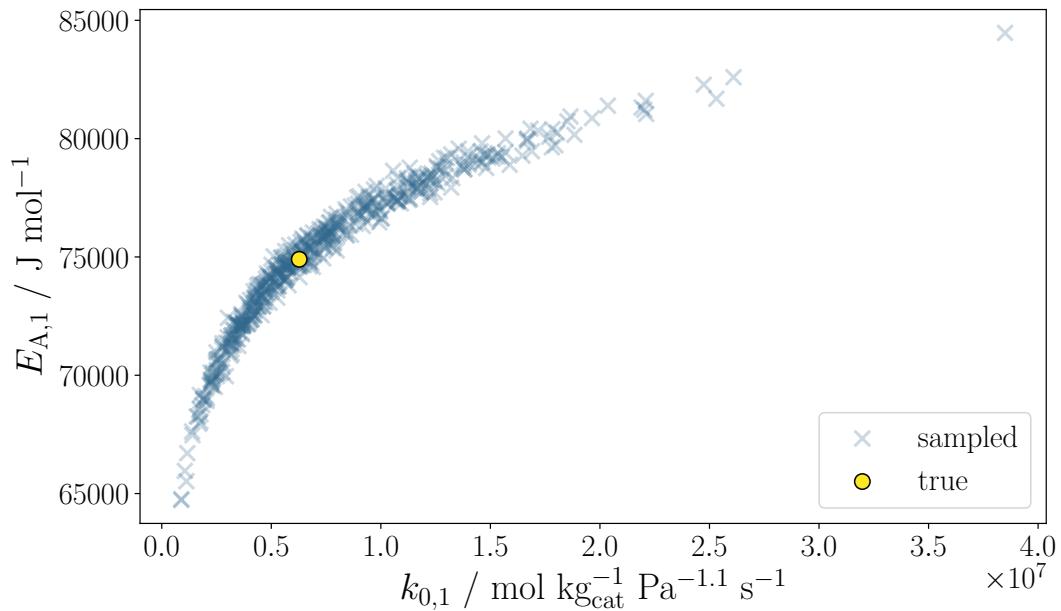


Figure 4.3: Due to the multi-variate normal distribution caused by an experimental parameter fitting and the non-linearity of the Arrhenius-Ansatz (eq. 4.4), the activation energy and the pre-exponential factor show a non-linear uncertainty. The nominal value is marked in yellow.

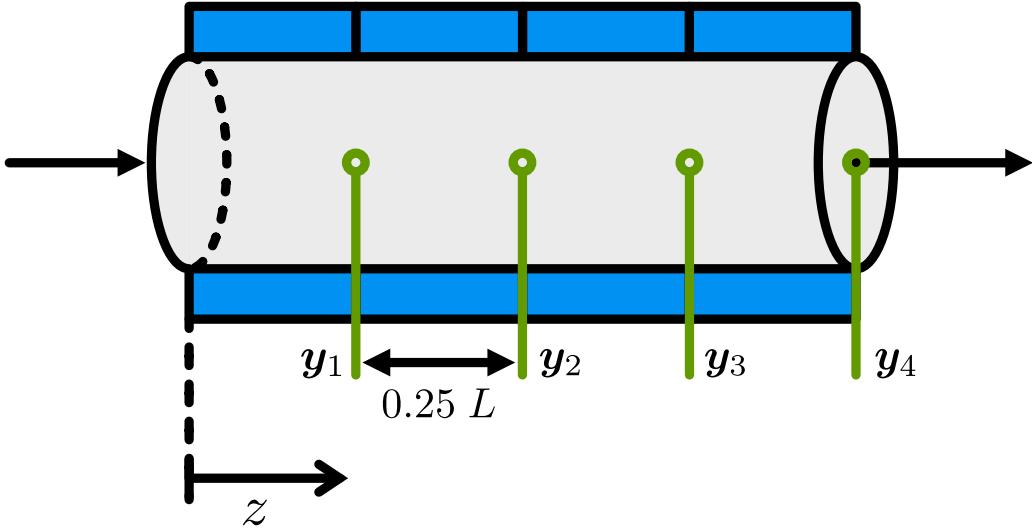


Figure 4.4: The four measurement positions $y_{(.)}$ are located equidistant in axial direction.

and elementary oxygen. Under the assumptions taken earlier, mainly as ideal gas behavior and incompressibility, the change in molar amount can be expressed in terms of the dimensionless state χ_i (eq. 4.13).

$$\Delta n_i = (\chi_{i,0} - \chi_i) \frac{pV}{R} \frac{x_*}{T_*} \quad (4.13)$$

Thus, the original conservation law is yielded as follows.

Corollary 4.1. *Conservation of atoms with a dimensionless concentration measure.*

$$B\Delta\chi = 0$$

Where, $\Delta\chi$ marks the vector of the dimensionless changes for all components divided by the temperature.

4.5 Measurement Positions

For surrogate modelling, four measurement positions are introduced into the TR, where the temperature and the dimensionless concentration measure χ_i should be measured and predicted (fig. 4.4). The distribution is done equidistantly. Taking all system states into account, meaning the concentration measure for all five components and the temperature, $6 \cdot 4 = 24$ measurements form. This denotes the output size of a surrogate model. Using the backward finite difference method with 128 points, the states at the positions 32, 64, 96 and 128 are considered as the measurements. Regarding the constraint satisfaction of the temperature, it is suitable to use the maximum temperature in the reactor as a measurement instead of discretizing spatially in a real-world scenario. The outlet states are of major interest in all cases.

4.6 Experimental Setup

The objective of this work is to derive, implement and analyze a physics consistent surrogate modelling strategy for the ethene oxidation flow reactor under parametric uncertainty. During the whole procedure, the previously defined NARX framework is employed (def. 3.1). It consists of a nominal prediction \mathbb{E} and two quantile predictions \hat{Q}_τ (fig. 4.5). All three predictors are separated in two parts. As the temperature is the critical state regarding the constraints, it is predicted separately from all other states. This allows for more accurate temperature predictions, as well as explicit quantile regression for the temperature measurements only. So the NARX models $Q_{0.9}$ (yellow) and $Q_{0.1}$ (purple) undergo quantile regression (2.4.1) in the training phase with the 0.9- and 0.1-quantile respectively. The expected value of the temperature $\mathbb{E}(\mathbf{T}_{k+1})$ (green) is learned using the MSE. The same is done for the expected values of all other states, namely the dimensionless concentration measures χ . To predict all other states, that *most likely* correspond to the temperature quantiles, NARX models are trained using a weighted MSE. These are marked as $\mathbb{E}_{0.9}$ and $\mathbb{E}_{0.1}$ to match their corresponding temperature quantile. Training efficiency and prediction quality rely on the choice of distance measure and weight function (alg. 1).

For one complete forward pass, the vector of past measurements and inputs ϕ_k is normalized with a standard scaler to have a mean of zero and a unit standard deviation. It is then encoded by the PCA encoder to a lower dimensional subspace (2.5.2). The compressed vector ξ_k needs to be scaled again with a maximum scaler before it can be fed into a NARX model. After the NARX models for the temperature and all other states have predicted the next instance, the complete measurement vector is constructed. For the next recursive forward pass, the complete measurement vector is inserted as the first element into the vector of past measurements and inputs (eq. 2.4). Additionally, the applied input \mathbf{u} is inserted as well at its corresponding position. This recursive insertion process is done by the function f_{shift} using the previous vector of past measurements and inputs ϕ_{k-1} (fig. 4.5). To start the simulation, an initial vector ϕ_{init} must be supplied.

4.6.1 Data Generation

To train and test the NARX models, reactor measurement data is generated at mentioned positions. For this sake, the normal distributions for the radial heat conductivity and the kinetic parameters are sampled in Monte-Carlo (MC) fashion to yield a parameter vector \mathbf{w} in the uncertainty space. For the inputs in the training data ($T_{w,1}$ to $T_{w,4}$), excited ramp signals are generated with an uniform distribution. The distribution ranges from 535 K to 665 K for each input temperature. Once a signal level is sampled, it is held for a random duration between 0.6 and 1.8 times of the system's time constant. The ramp duration is set to half of the system's time constant. The same is done for the time-varying inlet flow speed with uniform boundaries of 0.18 m s^{-1} to 0.42 m s^{-1} . Here, the hold duration is choosen to be longer with 1.75 to 3.5 times the system's time constant. To capture dynamic and stationary behavior of the reactor model, hold durations below

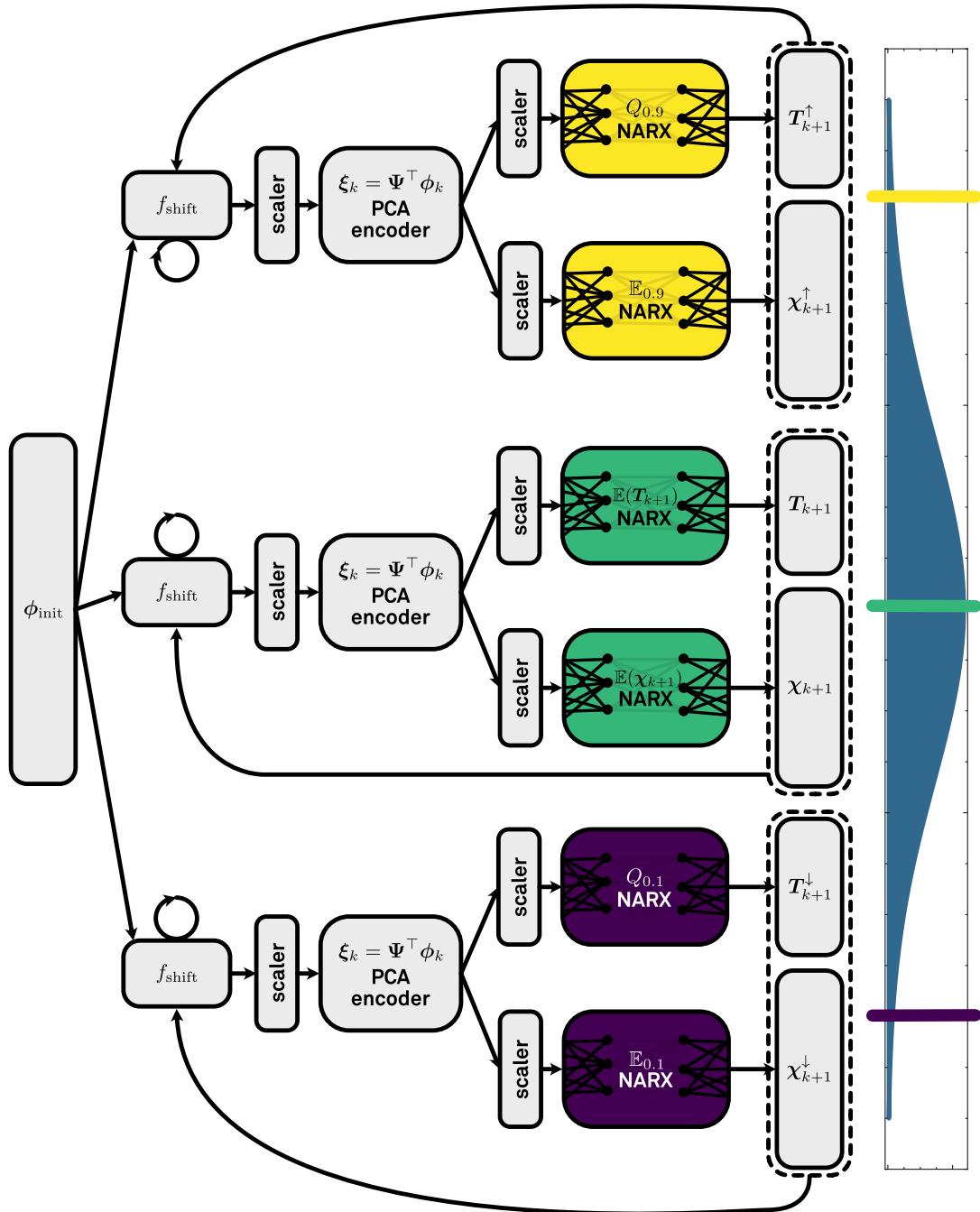


Figure 4.5: To simulate the distribution of system trajectories (blue bell curve), three NARX predictors are applied in form of the expected value (green), the 0.9-quantile (yellow) and the 0.1-quantile (purple). Each of them consists of two separate NARX models, one for the temperature measurements \mathbf{T} and one for all other states $\boldsymbol{\chi}$.

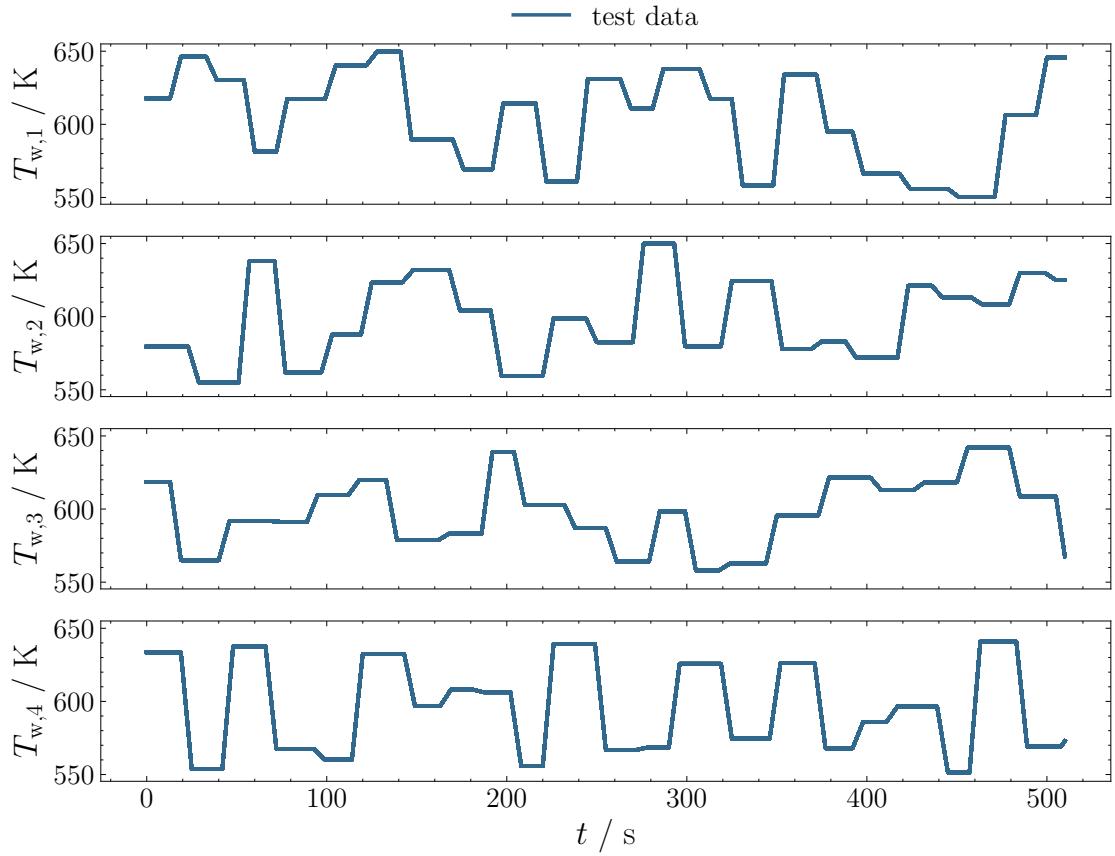


Figure 4.6: For data generation, the wall temperatures are excited using uniform distribution for the level and the hold time. The latter is a multiplier of the system's biggest time constant. Excited input trajectories of the test data are shown for a simulation horizon of 512 s.

and above the biggest time constant are picked. The heat transfer to the reactor wall is determined as the slowest time constant with approximately 20 s (eq. 4.14).

$$\tau = \frac{\rho c_p d_t}{4\alpha} \approx 20 \text{ s} \quad (4.14)$$

The excited input signals of the wall temperatures are shown (fig. 4.6) for 512 s with a sampling frequency of 1 s^{-1} .

4.6.2 Training

Chapter 5

Results on Surrogate Model Performance

Please choose a more expressive title for this chapter. If it makes sense for your work you can also have multiple chapters presenting your results.

5.1 Training Effectivity

5.2 Physics Consistency

To analyze the physics consistency of the different NARX frameworks an open-loop simulation is performed. It propagates the state trajectories for the nominal case, the upper and the lower quantiles forward in time independently (def. 3.1). This is done for the three framework variations *vanilla*, *naive_pc* and *pc*. The main quality metric is the balance violation of the conservation of atoms due to stoichiometry (th. 4.1). When it is equal to the zero vector, it is perfectly fulfilled. The residual of the linear balance constraint is calculated (eq. 5.1) as the l_2 -norm and plotted as a function of time (fig. 5.1).

$$\|\mathbf{b}\|_2 = \|\mathbf{B}\Delta\xi\|_2 \quad (5.1)$$

To indicate maximum machine precision, horizontal dashed lines are drawn for 64 bit (gray) and 32 bit (black). The balance residual norm $\|\mathbf{b}\|_2$ is shown for the nominal NARX (green), trained with a standard MSE, and the conditioned NARX models, that are trained with a weighted MSE. Here, yellow colors the NARX conditioned by the 0.9-quantile and purple the NARX conditioned by the 0.1-quantile. After being initialized with an initial state, that obeys the balance up to an error of 1×10^{-14} . All NARX models abruptly increase the error to 1×10^{-3} and fluctuate in this region until the simulation horizon is reached. When the residual is averaged over the simulation horizon using the $\langle(\cdot)\rangle_t$ notation, the nominal NARX model produces a slightly smaller error of 6.39×10^{-4} . In contrast, the error of the quantile-conditioned NARX models is roughly twice as big with values of 1.12×10^{-3} and 1.42×10^{-3} for the 0.9 and 0.1 quantile respectively (fig. 5.1). It can be noted that neither an instability nor a divergence occurs that would lead to a drastic increase in the residual error. Additionally, the error does not add up over time.

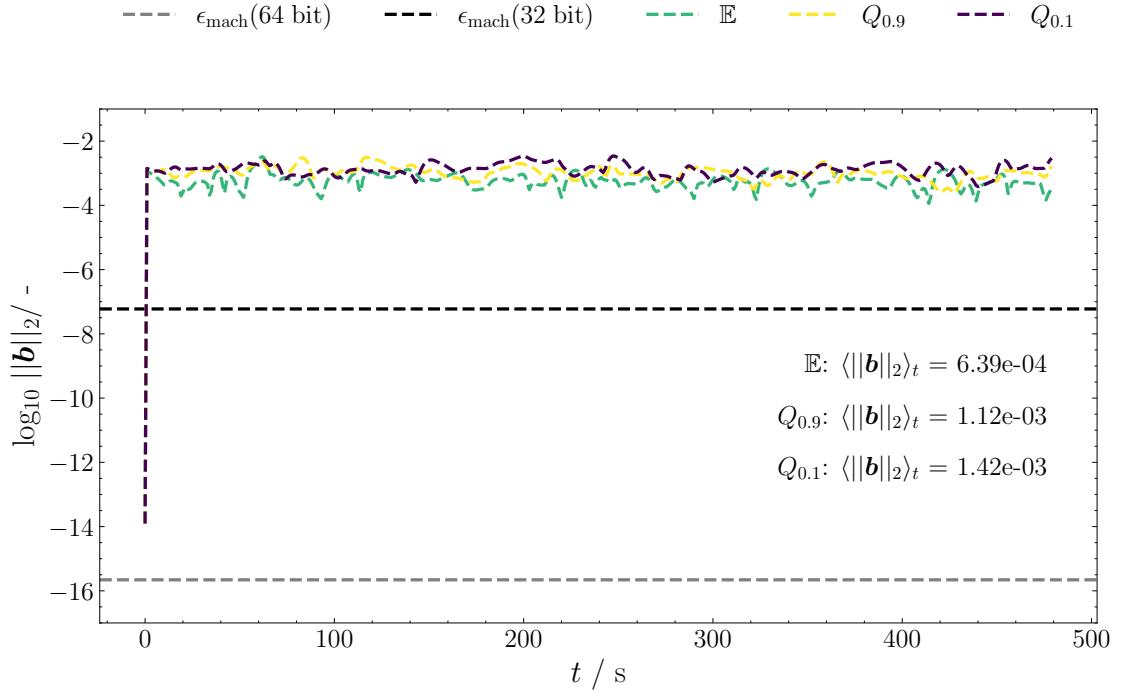


Figure 5.1: Balance equation violation as a function of time in a 480 s forward simulation of the vanilla NARX framework. The nominal trajectory \mathbb{E} (green) displays slightly better alignment with the balance than the models conditioned by quantiles $Q_{0.9}$ and $Q_{0.1}$.

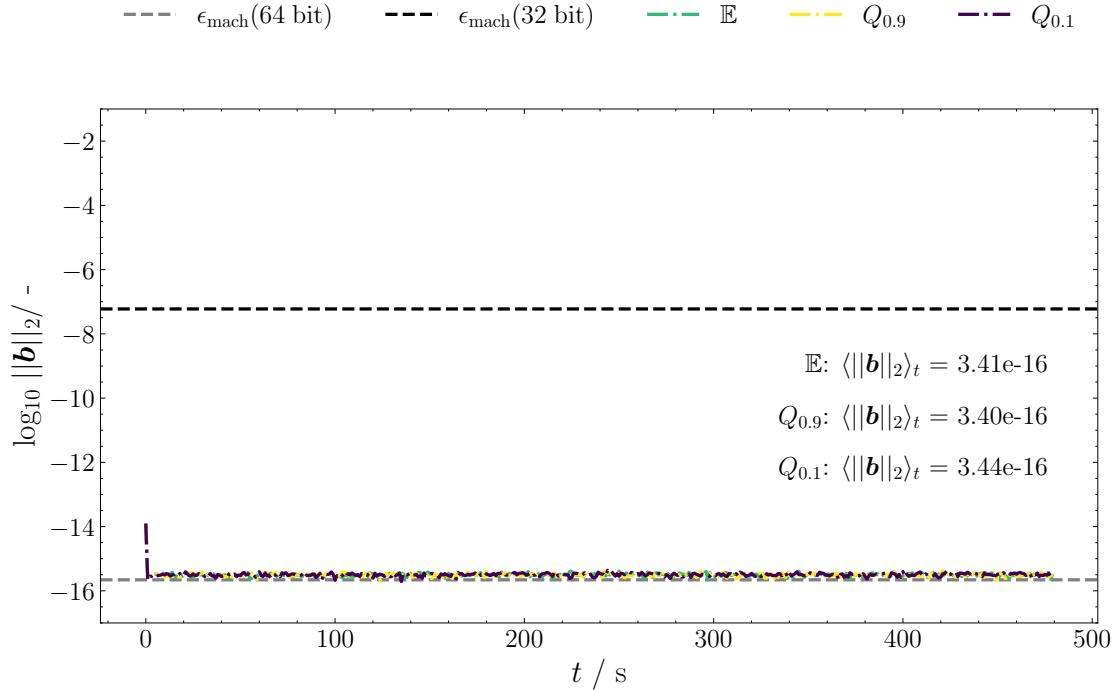


Figure 5.2: Balance equation violation as a function of time in a 480 s forward simulation of the vanilla NARX framework. The nominal trajectory \mathbb{E} (green) displays slightly better alignment with the balance than the models conditioned by quantiles $Q_{0.9}$ and $Q_{0.1}$.

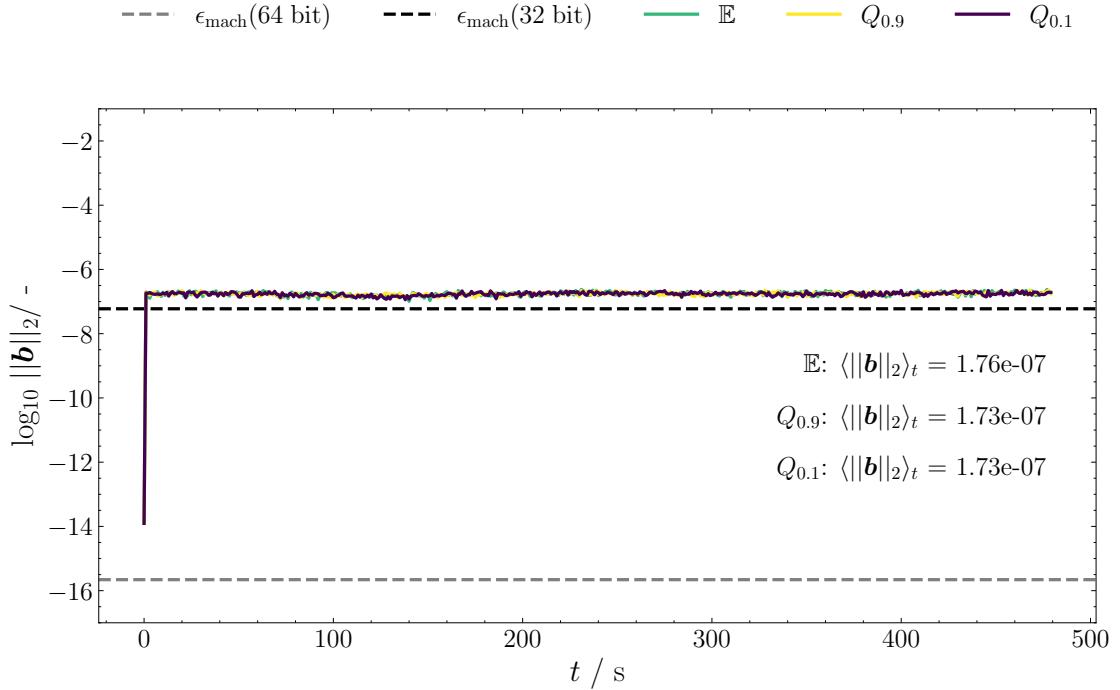


Figure 5.3: Balance equation violation as a function of time in a 480 s forward simulation of the vanilla NARX framework. The nominal trajectory \mathbb{E} (green) displays slightly better alignment with the balance than the models conditioned by quantiles $Q_{0.9}$ and $Q_{0.1}$.

The same is done with the *naive_pc* NARX models that are not trained with the physics constrained activation but simply apply it to their output (fig. 5.2). The balance error during the simulation shows a completely different picture. All models achieve vanishing balance errors at 64 bit precision of 3.4×10^{-16} . These appear consistent over the complete simulation horizon. The naively physics constrained models exhibit even smaller balance errors than the initial state itself, which stems from a first principle simulation, due to fewer rounding errors. There is no noticeable difference between the nominal predictions \mathbb{E} (green) and the conditioned quantile predictions $Q_{0.9}$ (yellow) and $Q_{0.1}$ (purple) in contrast to the vanilla versions. Additionally, it can be interpreted as a proof of concept for the physics constrained activation g_{pc} . It enforces linear equality constraints up to machine precision, even if it is not used during training.

Finally, the balance error is plotted for the *pc*-NARX, that uses the physics constrained activation during training and simulation (fig. 5.3). While it significantly improves training efficiency, it also enforces linear equality constraints in an open-loop simulation up to 32 bit precision. Similarly to the *naive_pc*, all three NARX models achieve the same constraint residual error of 1.7×10^{-7} . The difference between the physics constrained nominal trajectory (green) and the quantile conditioned (yellow, purple) trajectories is non-existent. It is worth noticing, that the minimum possible constraint residual differs between the *naive_pc* and the *pc*-NARX. The *naive* version is not trained with the physics consistent activation g_{pc} . It is applied *a posteriori* and implemented using a `casadi`-function object. The *pc* version owns a direct implementation of g_{pc} as a `torch.nn.Module`. Eventhough it is converted into a `casadi` expression to be

simulated in `do_mpc` afterwards, it still keeps the 32 bit precision of the `torch.float32`-object. Because the `casadi`-function computes in the 64 bit decimal space, *naive* and *pc* differ in maximum machine precision.

Chapter 6

Conclusions

CHAPTER 6. CONCLUSIONS

Appendix A

Datasheets

Did your work require long tables of parameters, e.g. to configure a system model. This table would distract the main part of the thesis but including it is an important aspect of scientific work. Please attach it here.

A.1 Parameter for ...

Table A.1: Mean and covariance matrix of the activation energies and pre-exponential collision factors.

μ		Σ			
		$\ln k_{0,1}$	$E_{A,1}$	$\ln k_{0,2}$	$E_{A,2}$
$\ln k_{0,1}$	15.621453	0.351544	1768.109910	0.380453	1913.304578
$E_{A,1}$	74856.339000	1768.109910	9031744.980000	1913.445056	9773020.700000
$\ln k_{0,2}$	16.272557	0.380453	1913.445056	0.411795	2070.851800
$E_{A,2}$	89853.839400	1913.304578	9773020.700000	2070.851800	10576522.400000

APPENDIX A. DATASHEETS

Appendix B

Proofs

$$\begin{aligned}\Delta n_i &= x_{i,0}n_0 - x_i n = x_{i,0} \frac{p_0 V_0}{R T_0} - x_i \frac{pV}{RT} \quad | \text{ isobaric, non-compressible} \\ &= \left(\frac{x_{i,0}}{T_0} - \frac{x_i}{T} \right) \frac{pV}{R} = \left(\frac{x'_{i,0}}{T'_0} - \frac{x'_i}{T'} \right) \frac{pV}{R} \frac{x_*}{T_*} \\ &= (\chi_{i,0} - \chi_i) \frac{pV}{R} \frac{x_*}{T_*}\end{aligned}\tag{B.1}$$

Did you introduce lemmas or theorems with lengthy proofs? Including them in the main part of the thesis might be distracting. You can also include them in the appendix.

B.1 Proof of Theorem X

...

APPENDIX B. PROOFS

Bibliography

- [1] JD, *Placeholder citation*, 0.
- [2] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: [1706.03762 \[cs.CL\]](https://arxiv.org/abs/1706.03762). [Online]. Available: <https://arxiv.org/abs/1706.03762>.
- [3] H. Chen, G. E. C. Flores, and C. Li, “Physics-informed neural networks with hard linear equality constraints,” *Computers & Chemical Engineering*, vol. 189, p. 108764, 2024. DOI: <https://doi.org/10.1016/j.compchemeng.2024.108764>.
- [4] B. Houska, “Robust optimization of dynamic systems,” Ph.D. dissertation, KU Leuven, 2011, ISBN: 978-94-6018-394-2.
- [5] O. Stein, “Bi-level methods for semi-infinite programming,” in *Bi-Level Strategies in Semi-Infinite Programming*. Boston, MA: Springer US, 2003, pp. 145–169, ISBN: 978-1-4419-9164-5. DOI: [10.1007/978-1-4419-9164-5_5](https://doi.org/10.1007/978-1-4419-9164-5_5). [Online]. Available: https://doi.org/10.1007/978-1-4419-9164-5_5.
- [6] M. Diehl, “Robust optimization,” Lecture slides, Summer School on Robust Model Predictive Control with CasADi, University of Freiburg, September 15–19, 2025. Slides jointly developed with Titus Quah, Katrin Baumgärtner, and Florian Messerer, based on joint work with B. Houska and some illustrations from his PhD thesis., University of Freiburg, Germany, 2025.
- [7] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991. DOI: [https://doi.org/10.1016/0893-6080\(91\)90009-T](https://doi.org/10.1016/0893-6080(91)90009-T).
- [8] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, “Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms,” *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397–2404, Apr. 2023. DOI: [10.1109/lra.2023.3246839](https://doi.org/10.1109/lra.2023.3246839).
- [9] S. Lucia and S. Engell, “Robust nonlinear model predictive control of a batch bioreactor using multi-stage stochastic programming,” in *2013 European Control Conference (ECC)*, 2013, pp. 4124–4129. DOI: [10.23919/ECC.2013.6669521](https://doi.org/10.23919/ECC.2013.6669521).
- [10] R. Güttel and T. Turek, *Chemische Reaktionstechnik*, 1st ed. 2021. Berlin, Heidelberg: Springer Berlin Heidelberg and Springer Spektrum, 2021, ISBN: 978-3-662-63150-8.
- [11] “Dynamic optimization methods with embedded dae solvers,” in *Nonlinear Programming*, ch. 9, pp. 251–286. DOI: [10.1137/1.9780898719383.ch9](https://doi.org/10.1137/1.9780898719383.ch9). eprint: <https://pubs.siam.org/doi/pdf/10.1137/1.9780898719383.ch9>. [Online]. Available: <https://pubs.siam.org/doi/abs/10.1137/1.9780898719383.ch9>.

BIBLIOGRAPHY

- [12] S. Lucia and S. Engell, “Potential and limitations of multi-stage nonlinear model predictive control,” *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 1015–1020, 2015. DOI: <https://doi.org/10.1016/j.ifacol.2015.09.101>.
- [13] K. McBride and K. Sundmacher, “Overview of surrogate modeling in chemical process engineering,” *Chemie Ingenieur Technik*, vol. 91, no. 3, pp. 228–239, 2019. DOI: <https://doi.org/10.1002/cite.201800091>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cite.201800091>.
- [14] T. Salzmann, J. Arrizabalaga, J. Andersson, M. Pavone, and M. Ryll, “Learning for casadi: Data-driven models in numerical optimization,” in *Learning for Dynamics and Control Conference (L4DC)*, 2024.
- [15] K. Benidis, S. S. Rangapuram, V. Flunkert, *et al.*, “Deep learning for time series forecasting: Tutorial and literature survey,” *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–36, Dec. 2022. DOI: <10.1145/3533382>.
- [16] N. Kemmerling and S. Lucia, “Uncertainty-aware reduced-order modeling for robust control of zeolite crystallization,” Manuscript in preparation, TU Dortmund University, Dortmund, Germany, 2025.
- [17] S. L. Brunton and J. N. Kutz, “Reduced order models (roms),” in *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, POD for model reduction, Cambridge University Press, 2019, pp. 375–402.
- [18] Anonymous, *Torch-incremental-pca: A pytorch implementation of incremental pca*, <https://pypi.org/project/torch-incremental-pca/>, PyTorch implementation inspired by scikit-learn’s IncrementalPCA and PCAonGPU. Provides GPU-accelerated, memory-efficient incremental PCA for large datasets., 2025.
- [19] M. Raissi, P. Perdikaris, and G. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019. DOI: <https://doi.org/10.1016/j.jcp.2018.10.045>.
- [20] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, “Differentiable convex optimization layers,” in *Advances in Neural Information Processing Systems*, 2019.
- [21] G. Lastrucci, T. Karia, Z. Gromotka, and A. M. Schweidtmann, *Picard-kkt-hpinn: Enforcing nonlinear enthalpy balances for physically consistent neural networks*, 2025. arXiv: [2501.17782 \[cs.LG\]](2501.17782). [Online]. Available: <https://arxiv.org/abs/2501.17782>.
- [22] A. Pietschak, M. Kaiser, and H. Freund, “Tailored catalyst pellet specification for improved fixed-bed transport characteristics: A shortcut method for the model-based reactor design,” *Chemical Engineering Research and Design*, vol. 137, pp. 60–74, 2018. DOI: <10.1016/j.cherd.2018.06.043>.
- [23] M.A. Al-Saleh M.S. Al-Ahmadi M.A. Shalabi, *Kinetic study of ethylene oxidation in a berty reactor*, 1988. DOI: [10.1016/0300-9467\(88\)80004-2](10.1016/0300-9467(88)80004-2). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0300946788800042>.

BIBLIOGRAPHY

- [24] K. L. Fischer and H. Freund, “Intensification of load flexible fixed bed reactors by optimal design of staged reactor setups,” *Chemical Engineering and Processing - Process Intensification*, vol. 159, p. 108183, 2021. DOI: [10.1016/j.cep.2020.108183](https://doi.org/10.1016/j.cep.2020.108183).
- [25] P. Zehner and E. U. Schlünder, “Wärmeleitfähigkeit von schüttungen bei mäßigen temperaturen,” *Chemie Ingenieur Technik*, vol. 42, no. 14, pp. 933–941, 1970. DOI: <https://doi.org/10.1002/cite.330421408>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cite.330421408>.