ISBN 978-82-326-4612-8 (printed version)
ISBN 978-82-326-4613-5 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2020:132

Eric Bradford

Stochastic nonlinear model predictive control for chemical batch processes

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Doctoral theses at NTNU, 2020:132

NTNU

NTNU
Norwegian University of
Science and Technology

Eric Bradford

Stochastic nonlinear model predictive control for chemical batch processes

Thesis for the degree of Philosophiae Doctor

Trondheim, May 2020

Norwegian University of Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-326-4612-8 (printed version)
ISBN 978-82-326-4613-5 (electronic version)
ISSN 1503-8181

ITK Report 2020-4-W

Doctoral theses at NTNU, 2020:132

Printed by Skipnes Kommunikasjon as

Summary

The chemical industry is a vital part of the world economy transforming raw materials into crucial intermediary products. Batch processes are common in many sectors of the chemical industry, which are gaining in importance due to the increasing emphasis on fine and speciality chemicals that are mostly produced employing batch processes. The main advantage of batch processes is their relative ease of design and inherent flexibility to produce different products and product grades. The control of batch processes is challenging, since these are operated at unsteady state and are often highly nonlinear. Nonlinear model predictive control (NMPC) is therefore a promising approach that can handle nonlinearity and constraints on manipulated and controlled variables. Most batch process models are however affected by significant uncertainties, which need to be taken into account to prevent performance deterioration and constraint violations.

This thesis focuses on the development of NMPC formulations for batch processes that explicitly consider stochastic uncertainties to trade-off risk with economic performance. The work presented in this thesis is divided into two parts:

- Stochastic nonlinear model predictive control using uncertainty propagation
- Gaussian process dynamic modelling and nonlinear model predictive control

The first part deals with the formulation of stochastic NMPC (SNMPC) algorithms by propagating the stochastic uncertainty through nonlinear transformations within the optimal control problem (OCP). Stochastic uncertainties considered include both time invariant parametric and additive uncertainties. The estimated statistics at each time step, such as mean and variance are utilised to ensure the satisfaction of chance constraints, while optimizing a nonlinear economic objective in expectation.

Initially it is shown how the Unscented transformation and polynomial chaos expansions (PCEs) can be exploited to propagate stochastic uncertainties resulting

from state estimates, parametric uncertainties, and additive disturbances to formulate the SNMPC algorithm. In addition, these are exploited to update the state estimates given available measurements. Next a novel application of Gaussian processes (GPs) is introduced for uncertainty propagation in SNMPC similar to PCEs. Given the outstanding performance of GPs and PCEs to capture the required statistics, a new approach for uncertainty propagation is proposed next combining PCEs and GPs. This combination is shown to be superior to either GPs and PCEs alone. All of the proposed SNMPC algorithms are shown to lead to a robust and reliable solution for batch processes and show superior performance to their nominal NMPC counterparts.

The second part applies GPs to model batch processes from noisy input/output data. Commonly, dynamic models for batch processes are derived from first principles, which often have high development costs and are frequently too complex to be utilised online. Using GPs for black-box identification is therefore an appealing alternative. GPs in this regard are especially useful, since these quantify the residual uncertainty from the identification of a dynamic model. This uncertainty measure can be exploited to obtain more reliable optimization and control solutions.

Firstly, it is illustrated how GPs are able to accurately simulate a bioprocess from experimental data and capture the model uncertainty by propagation of the uncertainty measure. Further, it is shown that the predictive quality of GPs is overall comparable to that of ANNs. Given the excellent predictive quality of GPs, an algorithm is then proposed to employ the GP as an approximate plant model for NMPC. It is crucial to account for the plant-model mismatch of the GP to avoid constraint violations. The proposed algorithm generates Monte Carlo samples of the GP offline for constraint tightening using back-offs. It is shown how probabilistic guarantees can be derived based on the number of constraint violations of these samples. Further, the approach is able to account for both online learning and state dependency of the uncertainty explicitly to alleviate conservativeness. Lastly, computational times could be shown to be relatively low. Often for batch processes it is possible to derive major parts of the plant models from first principles, however certain parts of the model are very difficult to determine from physical laws alone. It is therefore proposed to extend the previous algorithm to this case, for which the GP is exploited to model only the parts of the dynamic system that are difficult to describe using first principles alone. A graphical abstract to summarize the algorithm is given in Figure 1.

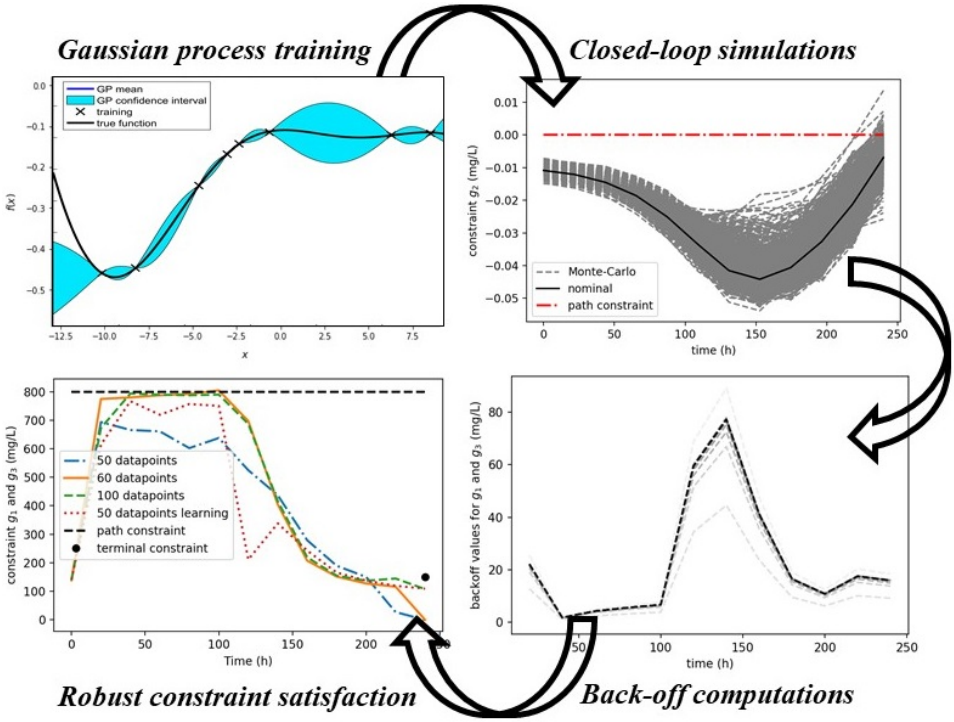


Figure 1: Graphical abstract for the GP NMPC algorithm proposed in Part II.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (PhD) at the Norwegian University of Science and Technology (NTNU). This project has been funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 675215, which I am thankful for.

The work in this dissertation is part of the Innovative Training Network (ITN) "PROcess NeTwork Optimization (PRONTO) for efficient and sustainable operation of Europe's process industries taking machinery condition and process performance into account" funded by the European Union. Each doctoral candidate in this European industrial doctorate programme needs to spend at least 18 months in the non-academic sector. The research has been conducted in equal parts at the Department of Engineering Cybernetics (ITK) at the Norwegian University of Science and Technology (NTNU) and at BASF SE in Ludwigshafen, Germany from September 2016 to April 2020. The project was supervised by Lars Imsland (NTNU), Bjarne Foss (NTNU), and Marcus Reble (BASF SE).

First and foremost I am extremely grateful to my supervisor Lars Imsland for his patient guidance and advice throughout my PhD. It has been a pleasure to have a supervisor, who cares so much about my work and always takes the time to answer my questions promptly. I also want to thank Bjarne Foss for his help at the beginning of my PhD. I also had the great pleasure of working with Marcus Reble, who has profound knowledge on both theoretical and practical problems of MPC. His constructive criticism and suggestions have been very helpful to improve the quality of my work.

I would also like to extend my deepest gratitude to my co-authors, who I had the pleasure of working with during my PhD. Artur Schweidtmann for his resilience to get our work from Cambridge published and helpful discussions for future research goals. Antonio Del Rio Chanona for our fruitful collaborations on many different papers, but also for the great company at many conferences and visits in London.

Dongda Zhang for his insight into bioprocesses and constructive feedback on our joint work. Panagiotis Petsagkourakis for the useful discussions on control and enjoyable work on reinforcement learning. Leif Erik Andersson for helping me get acclimated to Norway and the opportunity to help with his work on wind farm control.

I want to also thank my colleagues and friends at NTNU, BASF SE, and in the PRONTO consortium for making the last three years enjoyable. Thank you for all your input and good comradeship. Special thanks to Ouyang Wu for being a great friend during my time spent at both NTNU and BASF SE.

My deepest appreciation to my defence committee Melanie Zeilinger, Joel Paulson, and Morten Hovd for taking the time to revise my thesis and providing valuable and insightful feedback.

Lastly, I am forever grateful to my parents, Wulff and Pamela, for their continued support and encouragement.

Contents

List of Tables

List of Figures

Part I

Introduction and background

Chapter 1

Background

1.1 Batch processes

In this chapter some background information is given on batch processes. In particular, the chapter focuses on their importance and the challenges for their control to motivate the use of nonlinear model predictive control (NMPC). Note we use the term batch processes to include semi-batch processes.

The chemical industry is a crucial part of the world economy. In 2017 the chemical industry supported 120 million jobs and contributed US$5.7 trillion to the world's gross domestic product, an equivalent of 7% [122]. While continuous and batch operations are employed in all sectors of the chemical industry, high volume processes such as petroleum refining are typically continuous, while low volume processes such as the production of speciality chemicals exploit batch operations. In batch operations the raw materials are loaded in a vessel and processed without material addition or removal, while for semi-batch processes some of the reactants are continuously added. Continuous processes can be operated at economically desirable operating points, however require considerably more effort to design. Batch processes on the other hand enable higher flexibility by allowing for adjustment of the temperature profile, final time, and feeding profile in the case of semi-batch processes. Furthermore, batch reactors can be utilized for the production of different products and product grades, which allows the production to be adjusted based on market demand. The emphasis in the chemical industry has increasingly changed towards high technology and niche markets such as life sciences (pharmaceuticals and agrochemicals) as well as speciality chemicals. Consequently, batch processes are gaining in importance [29].

In general, batch process operation consists of the following steps:

- Loading the reactor vessel with raw material.
- Processing the raw material by following a recipe that outlines range of concentrations, flowrates, or temperatures for the desired reactions or separations to take place. During this stage key variables, such as temperature, need to be controlled to ensure satisfactory performance.
- Discharging the reactor vessel and measuring the quality of the final batch product.

In this regard, two different control problems arise for batch processes:

- End-point property control: the real economic objective of batch processes is usually related to the product quality at the end of the batch.
- Set-point tracking: offline or online determined time-varying setpoint trajectories need to be followed.

1.1.1 Control challenges

Batch process control poses some unique control challenges, which are mainly concerned with productivity, product quality, and safety. Important characteristics and challenges of batch processes are as follows [29]:

- *Unsteady state operation*: Batch processes do not operate according to a designed single operating point and instead dynamically proceed from an initial state to a very different final state. Set-points and control signals consequently correspond to time varying profiles. Implications are that the controller cannot be designed using approximate linearization around a single operating point and it is rarely possible to operate away from constraints due to wide operating ranges. Furthermore, available models are commonly incomplete and poor, since a large range of operating conditions needs to be covered.
- *Irreversible behaviour*: For batch processes it is often impossible to introduce remedial corrections to off-specification material once produced, while in contrast for continuous processes control actions can be used to move the process back to the desired operating point.
- *Limited feedback information*: The control of the product quality is vital, however online measurement can rarely be utilized to determine the product quality. It is therefore difficult to correlate specific inputs to the final product quality obtained.

- *Slow dynamics*: Compared to electrical or mechanical systems, chemical systems are rather slow. This allows for larger sampling times and therefore longer online computational times for MPC.
- *Repetitive nature*: Batch runs are repeated frequently and hence knowledge from previous runs can be exploited to improve subsequent runs.

1.1.2 Dynamic modelling and optimization of batch processes

Mechanistic first principles dynamic models can be derived from material and energy balances. The state variables are given by concentrations, pressure, temperature, and volume, while typical control inputs are given by flow rates or valve positions. Output measurements are often temperature and pressure. The model of a batch process can be expressed as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{1.1}$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)) \tag{1.2}$$

where t is the run time, $\mathbf{x}$ denotes the states, $\mathbf{u}$ represents the control inputs, and $\mathbf{y}$ are the online output measurements.

The main objective of batch operations is to determine an input profile that optimizes an objective function expressing the system performance. A typical optimization problem may be given as follows:

$$\begin{aligned}
&\underset{\mathbf{u}[0,T]}{\text{minimize}} \quad J(\mathbf{x}_0, \mathbf{u}[0,T]; T) = \phi(\mathbf{x}(T)) + \int_{t=0}^{T} \mathcal{L}(\mathbf{x}(t), \mathbf{u}(t))dt \\
&\text{subject to} \\
&\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \quad \forall t \in [0,T], \quad \mathbf{x}(0) = \mathbf{x}_0 \\
&\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max} \quad \forall t \in [0,T] \\
&\mathbf{g}_p(\mathbf{x}(t), \mathbf{u}(t)) \leq \mathbf{0} \quad \forall t \in [0,T] \\
&\mathbf{g}_T(\mathbf{x}(T)) \leq \mathbf{0}
\end{aligned} \tag{1.3}$$

where $J(\cdot)$ is the objective function to be minimized, $\mathbf{g}_p(\cdot)$ represents path constraints, and $\mathbf{g}_T(\cdot)$ are terminal constraints. Note: unlike continuous processes that have an infinite time horizon, OCPs of batch processes have a finite time horizon.

Batch process optimization involves both path constraints on inputs and states, as well as end-point constraints. Input path constraints are generally given by actuator limits, while state path constraints are the result of safety and operability considerations. Terminal constraints are often given by minimum product quality

requirements. The objective is commonly to maximize productivity. For example for tight temperature control we might aim to add as much of the reactant as possible to maximize productivity, while still adhering to the safety constraints on reactor temperature and pressure. This would lead to the following objective and constraints:

- Maximize the reactant dosage: $J(\mathbf{x}_0, \mathbf{u}[0,T]; T) = -F$, where F is the feed rate of the reactant.
- Path safety constraints: Keep temperature and pressure below certain safety limits, i.e. $\mathbf{g}_p(\mathbf{x}(t), \mathbf{u}(t)) = [T_R - T_{max}, P_R - P_{max}]$, where T_R and P_R are the reactor temperature and pressure respectively.
- Ensure that the total amount of reactant added remains below a certain limit: $g_T(\mathbf{x}(T)) = \int_{t=0}^{T} F dt - FT_{max}$.

1.1.3 Batch process control

The aim of the control strategy is generally to maximize the amount of product generated per unit time, while maintaining an acceptable product quality and respecting safety limits. In particular, side-reactions can lead to a poor product quality, while adiabatic temperature limits need to be followed to avoid a thermal runaway. Measurements related to product quality are often not available and hence it is common practice to track set-points of key process variables, such as reactor temperature, to ascertain the required product quality. The vast majority of batch processes are exothermic and therefore tight temperature control is vital.

Consider the problem of controlling a semi-batch reactor polymerization process, which is highly exothermic. The feed rate of monomers is often fixed. Typically a cascade of PI controllers is employed in industry to maintain a given set-point temperature, for which a slave controller tracks the set-point of the cooling jacket temperature. The master controller provides the set-point of the cooling jacket temperature to control the reactor temperature. The cascade control structure is illustrated in Figure 1.1. This cascade control structure has been shown to provide reliable operation, however the fixed monomer feed rates are kept low to ensure robustness [93]. Furthermore, it is industrial practice to operate at relatively low temperature levels for safety. Keeping the reactant feed rates constant and the reactor temperature low leads to low productivity and hence long batch times. Therefore, many different control schemes have been devised for semi-batch reactions, which optimize both the monomer dosage and temperature profile. For example, in [248] it has been proposed to determine optimal feeding and temperature profiles offline to be tracked online, while in [96] it is proposed to track necessary optimality conditions.

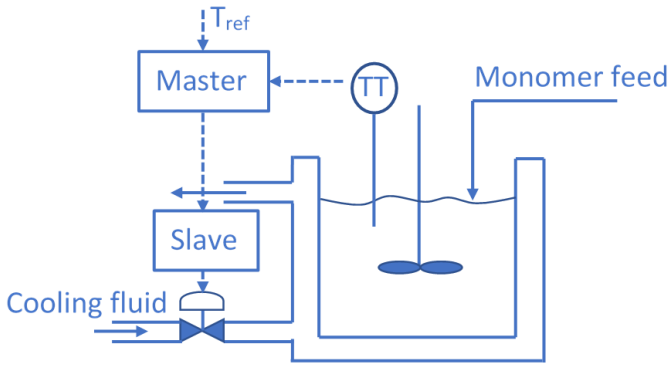


Figure 1.1: Typical cascade control structure for batch processes.

Among the most promising approaches is NMPC, which repeatedly solves an optimization problem based on a dynamic model of the batch process as in Equation 1.3. Many applications of NMPC for batch processes have been reported, such as for crystallization processes [176], semi-batch polymerization reactions [188], and bio processes [13]. NMPC can not only be used for improved temperature control, but also to optimize end-point quality constraints. It should be noted however that there is a lack of in situ sensors to measure product quality online. The main disadvantage of NMPC is its reliance on an accurate dynamic model of the batch process. Dynamic model predictions of production scale operations are often affected by high uncertainties as highlighted in Figure 1.2. It is therefore vital to account for these uncertainties to prevent constraint violations and performance deterioration of the NMPC algorithm. NMPC works that consider uncertainties are for example the multi-stage NMPC algorithm [164], an extended Kalman filter based NMPC approach [184], and polynomial chaos expansion based NMPC methods [177].

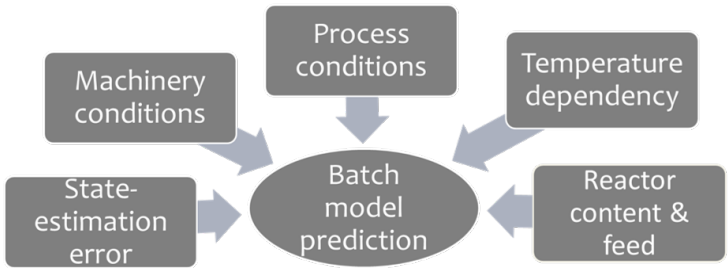


Figure 1.2: Batch process uncertainties.

1.2 Nonlinear model predictive control

Model Predictive Control (MPC), also referred to as receding horizon control, was developed in the late seventies and has progressed significantly since then. Model predictive control is the only advanced control methodology, which has made a significant impact on industrial control engineering. The main advantages of MPC are [165, 52]:

- It can be readily extended to multivariable problems.
- The formulation of the control problem takes actuator limitations into account.
- Process constraints can be included explicitly, which frequently leads to a more profitable operation.
- Can be used for processes with simple and complex dynamics.
- The resulting control law is easily implemented.
- Deals with dead time intrinsically.

Often MPC algorithms are categorised by the models that are implemented. Linear MPC refers to a family of MPC utilising linear models. Linear MPC approaches have found a multitude of successful applications in industry, in particular in the process industry [91]. Linear MPC theory is relatively mature and is well-established in practice. Many systems however display strong nonlinear behaviour. In addition, due to higher product quality specifications, higher productivity demands, tighter environmental regulations and demanding economic considerations the process industry is required to operate closer to the boundaries of the system. This necessitates the use of models that describe the dynamics of the process more accurately and hence the use of nonlinear models, which motivates the use of nonlinear model predictive control (NMPC) [92].

1.2.1 Principle and formulation of NMPC

In general, MPC is based on the online solution of a finite horizon open-loop optimal control problem (OCP) based on a dynamic model of the system to be controlled. Figure 1.3 depicts the basic principle of MPC. Given measurements at time t, the controller predicts the future states over a prediction horizon N and determines the inputs to optimize a specified open-loop performance objective subject to constraints on states and control inputs. Due to disturbances and plant-model mismatch, it is important to incorporate some feedback mechanism. This is achieved by implementing only the calculated open-loop control input of

the MPC until a new measurement becomes available, i.e. at each sampling time the OCP is solved again with the control and prediction horizon moving forward.

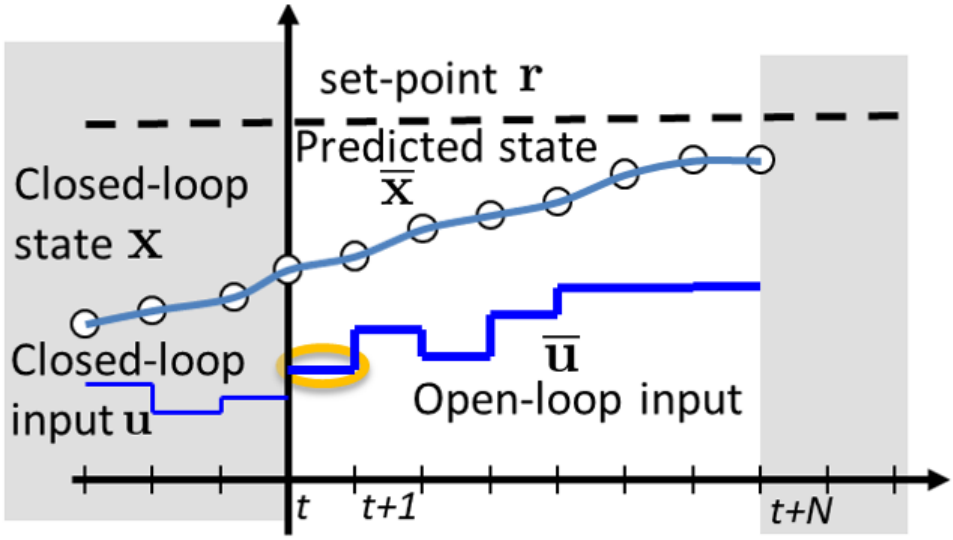


Figure 1.3: Principle of model predictive control.

We consider the system to be controlled to be given by a set of nonlinear discrete time equations:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t), \quad \mathbf{x}_0 = \hat{\mathbf{x}}_0 \tag{1.4}$$

where $\mathbf{x}_t \in \mathbb{R}^{n_\mathbf{x}}$ and $\mathbf{u}_t \in \mathbb{R}^{n_\mathbf{u}}$ are the vectors of states and inputs respectively,

The dynamic system is further assumed to be subject to constraints of the form:

$$\mathbf{u}_t \in \mathcal{U}\, \forall t \geq 0, \qquad \mathbf{x}_t \in \mathcal{X}\, \forall t \geq 0 \tag{1.5}$$

where $\mathcal{U} \subseteq \mathbb{R}^{n_\mathbf{u}}$ and $\mathcal{X} \subseteq \mathbb{R}^{n_\mathbf{x}}$ denote the set of feasible states and the set of feasible inputs respectively. Basic examples of $\mathcal{U}$ and $\mathcal{X}$ are given by box constraints:

$$\mathcal{U} := \{\mathbf{u} \in \mathbb{R}^{n_\mathbf{u}} | \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max}\} \tag{1.6}$$

$$\mathcal{X} := \{\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}} | \mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max}\} \tag{1.7}$$

where $\mathbf{u}_{min}, \mathbf{u}_{max}, \mathbf{x}_{min}$, and $\mathbf{x}_{max}$ are specified constant vectors.

A general discrete time open-loop OCP can be stated as follows:

$$\begin{aligned}
&\underset{\overline{\mathbf{U}}}{\text{minimize}} \quad J\left(\mathbf{x}_t, \overline{\mathbf{U}}; N\right) \\
&\text{subject to} \\
&\overline{\mathbf{x}}_{k+1} = \mathbf{f}(\overline{\mathbf{x}}_k, \overline{\mathbf{u}}_k) \quad \forall k \in \{t, \ldots, t+N-1\}, \quad \overline{\mathbf{x}}_t = \mathbf{x}_t \\
&\overline{\mathbf{u}}_k \in \mathcal{U} \quad \forall k \in \{t, \ldots, t+N\} \\
&\overline{\mathbf{x}}_k \in \mathcal{X} \quad \forall k \in \{t, \ldots, t+N\}
\end{aligned} \tag{1.8}$$

where $\overline{\mathbf{U}} = [\overline{\mathbf{u}}_t, \ldots, \overline{\mathbf{u}}_{t+N-1}]$, the time horizon is given by N and the objective function by $J(\cdot)$. To distinguish between the *real* system and the system model to predict the future, we denote the internal variables of the controller by a bar (e.g. $\overline{\mathbf{x}}_k$, $\overline{\mathbf{u}}_k$). This distinction is required, since even in the case with an exact system model, the predicted values will in general not be the same as the actual closed-loop values due to the recalculation of the OCP at every sampling time.

A common control objective is to steer the system to a desired set-point $\mathbf{r} = (\mathbf{x}_s, \mathbf{u}_s)$. A standard quadratic form for $J(\cdot)$ is the simplest and most often used:

$$J\left(\mathbf{x}_t, \overline{\mathbf{U}}; N\right) = \sum_{k=t}^{t+N} (\mathbf{x}_k - \mathbf{x}_s)^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}_s) + (\mathbf{u}_k - \mathbf{u}_s)^T \mathbf{R} (\mathbf{u}_k - \mathbf{u}_s) \tag{1.9}$$

where $\mathbf{Q}$ and $\mathbf{R}$ denote positive definite weighting matrices.

The objective of the MPC is usually setpoint stabilization, however the true objective is often to maximize profit. MPC using an objective to directly optimize a quantity of interest instead of setpoint stabilization is referred to as economic MPC (EMPC). Objectives for the EMPC may be to maximize a valuable product in the process industry, minimize energy consumption in building climate control, or cost efficient scheduling of production processes.

The open-loop OCP is repeatedly solved at each sampling time $t = 0, \ldots$ once new measurements become available. The closed-loop control is then given by the first optimal control action of Equation 1.8:

$$\mathbf{u}_t^* = \overline{\mathbf{u}}_t^*(\mathbf{x}_t; N) \tag{1.10}$$

where $\overline{\mathbf{u}}_t^*(\mathbf{x}_t, N)$ denotes the first control action of the optimal solution of Equation 1.8.

The corresponding value function as the function of the state is given by:

$$V(\mathbf{x}_t; N) = J(\mathbf{x}_t, \overline{\mathbf{U}}^*; N) \tag{1.11}$$

NMPC involves the online solution of a nonlinear OCP with a finite horizon N and finitely parameterized controls and states as in Equation 1.8. For linear MPC the solution of the OCP can be obtained as the optimization of a convex quadratic program, which can be efficiently solved online and may explain in part its wide usage in industry. NMPC on the other hand requires the solution of a non-convex nonlinear program (NLP), which can be computationally expensive. To allow for real-time solutions in general NMPC problems are solved only to local optimality and the special structure of the NLP needs to be exploited, such as sparsity. Two algorithms are most commonly utilized to solve NLPs, which are sequential quadratic programming (SQP) approaches and interior point (IP) methods [195]. The NLPs for NMPC in this work were solved using Casadi in Python, which was developed to allow for a large degree of flexibility compared to other algebraic modelling languages [9]. Casadi determines the exact gradients of the NLP utilizing automatic differentiation in forward and reverse modes on sparse matrix-valued computational graphs. These can subsequently be exploited in state-of-the-art NLP solvers such as IPOPT [258]. It should be noted that NMPC can be efficiently warm-started, since the NLP problem at one sampling is closely related to the NLP problem at the previous sampling time. For MPC problems the solution can be shifted by one sampling period and provide a reasonable solution for the next NLP.

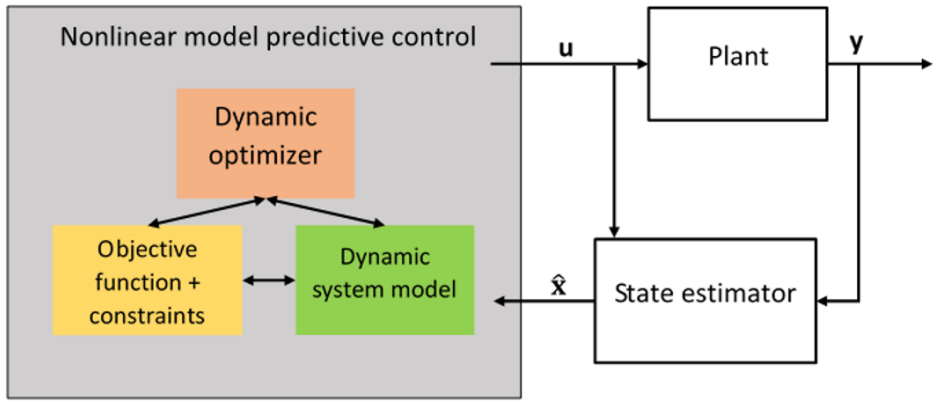


Figure 1.4: Basic NMPC control setup.

A basic NMPC control setup is depicted in Figure 1.4. The NMPC algorithm described so far is initialized using the actual system state at each sampling time, which often cannot be measured directly. Instead, the required state needs to com-

monly be estimated from output measurements. The basic steps for a NMPC scheme are as follows:

1. Obtain an estimate of the current system state using current measurements.
2. Compute an optimal control action sequence by solving a finite horizon OCP based on a dynamic model of the system.
3. Implement the first control action of this sequence and continue with 1.

1.2.2 Nominal stability of NMPC

A key question for NMPC is concerned with its closed-loop stability without uncertainties or disturbances, referred to as nominal stability. For simplicity it is assumed that the objective is regulation to a constant set-point $\mathbf{r} = (\mathbf{x}_s, \mathbf{u}_s)$. The state trajectory resulting from the solution of the infinite horizon problem using the principle of dynamic programming is known to be closed-loop stable. NMPC based on a finite horizon OCP on the other hand may not be stable, since the open-loop predictions and the closed-loop behaviour may be very different. In principle the guarantees on stability exploit the value function of the NMPC given in Equation 1.11 as a Lyapunov candidate using for example a quadratic objective as in Equation 1.9. Firstly, it is shown that feasibility at one sampling instance implies feasibility for the next sampling instance, which is known as recursive feasibility. Thereafter, it is established that the value function is strictly decreasing, which then implies asymptotic stability. Most techniques to guarantee stability modify the NMPC formulation by introducing additional constraints and a suitable penalty in the cost function. Several popular approaches are as follows [128]:

- A simple approach to guarantee stability is to add a so-called terminal equality constraint of the form $\overline{\mathbf{x}}_N = \mathbf{x}_s$, which ensures convergence in finite time. This leads to stability given feasibility at time $t = 0$, since it implies that after time N the value function is zero. The main disadvantage is that the system needs to be brought to the origin for a finite horizon, which can lead to small regions of attraction [134].
- Terminal set constraint of the type $\overline{\mathbf{x}}_N \in \Omega$ that ascertain the state to be regulated close enough to the set-point. In dual mode NMPC it is then assumed that there exists a feasible and stabilizing controller in Ω to switch to, which means once $\mathbf{x} \in \Omega$ it never leaves Ω and asymptotically goes to the set-point [178].
- The use of a terminal set constraint $\overline{\mathbf{x}}_N \in \Omega$ and a terminal cost function $E(\overline{\mathbf{x}}_N)$ that is added to the objective. The terminal set constraint ensures

the state to be in Ω for a finite horizon, while the terminal cost acts as a Lyapunov function to ensure local stability for all $\mathbf{x} \in \Omega$ from the NMPC. Terminal constraint sets and terminal costs can be obtained from local linearization, which is relatively practical [63].

- A sufficiently large time horizon N leads to closed-loop stability. Rules to choose a large enough time horizon N to guarantee stability were first introduced in [106].

1.2.3 Nonlinear state estimation

The implementation of NMPC requires knowledge of the current state of the nonlinear system at each sampling instance. Rarely is it possible to attain a measurement of the full state and hence nonlinear state estimation is crucial for the successful application of NMPC in practice. Furthermore, there may be substantial uncertainty on parameters of the nonlinear dynamic model utilized within the NMPC. It is common practice to add these uncertain parameters to the state vector to update them together with the states within the state estimator. Furthermore, to converge to a given set-point offset free a bias correction can be computed from output measurements to account for modelling errors and disturbances [224].

The most common approach for nonlinear state estimation is given by the Extended Kalman Filter (EKF), which applies the original Kalman filter equations to estimate the state by linearizing the nonlinear system [99]. This allows the mean and covariance of the Gaussian distribution to be propagated, which however achieves only a 1st order accuracy of these. To overcome this issue the Unscented Kalman filter (UKF) has been introduced, which uses a deterministic sampling approach instead to evaluate the mean and covariance accurately up to 3rd order [131]. Another popular stochastic approach is given by particle filters, which express the probability distributions of the states using weighted samples and propagate these through the nonlinear equation system. A major disadvantage of particle filters is that the number of samples required can be computationally prohibitive [12]. Alternatively, a least-squares optimal state estimation problem can be formulated defined on the full data history subject to the nonlinear dynamics. This may however be impractical due to the growing amount of data with time. Therefore, nonlinear moving horizon estimation has been proposed to make use of a finite memory moving window of both current and historical measurements for least-square estimation of the states [181].

1.2.4 Continuous time dynamic model

Most systems in chemical engineering are given by differential equations in continuous time. For example, as shown in Section 1.1.2 first principles models of

batch processes are in continuous time, which we assume can be expressed as:

$$\dot{\tilde{\mathbf{x}}}(\tilde{t}) = \tilde{\mathbf{f}}(\tilde{\mathbf{x}}(\tilde{t}), \tilde{\mathbf{u}}(\tilde{t})), \quad \tilde{\mathbf{x}}(0) = \hat{\mathbf{x}}_0 \tag{1.12}$$

where the continuous time variables are denoted by a tilde.

Let T denote the continuous time horizon over the finite horizon N, such that the sampling time is given by $h = \frac{T}{N}$ and $\tilde{t}_k$ denotes the continuous time at discrete time k. Unlike the OCP given in Equation 1.8, which has a finite number of optimization variables, the OCP based on a continuous system model has theoretically infinitely many optimization variables even for a finite time horizon. There are two approaches to solve the continuous time OCP, which are:

- *Indirect approach*: This method is based on the solution of Euler-Lagrange differential equations from the classical calculus of variations, which yields the control actions as a function of time and hence needs to be resolved for updated initial conditions. It can be seen as the application of the necessary conditions of optimization to an infinite dimensional optimization problem. The finite dimensional problem is however often too difficult to solve online.

- *Direct approach*: The direct approach parameterizes the infinite dimensional decision variables yielding a finite dimensional NLP. This NLP can be efficiently solved online by exploiting its sparse structure.

For online solutions of the NMPC problem the last approach is commonly used, which needs to be resolved at each sampling instance with the new initial state. The most common techniques for parameterization are single shooting, multiple shooting, and collocation. A further distinction within direct approaches is *sequential* or *simultaneous*. For sequential techniques the integrator is outside of the optimization problem, which only passes states and gradient information between each iteration. For simultaneous methods the required equations for discretization are added to the optimization problem as equality constraints. In this thesis most papers are written using discrete time notation, for which collocation has been applied to yield the required discrete time formulation. The collocation equations are added to the optimization problem, i.e. using a simultaneous formulation. The details for this are outlined here. For more information refer to [27].

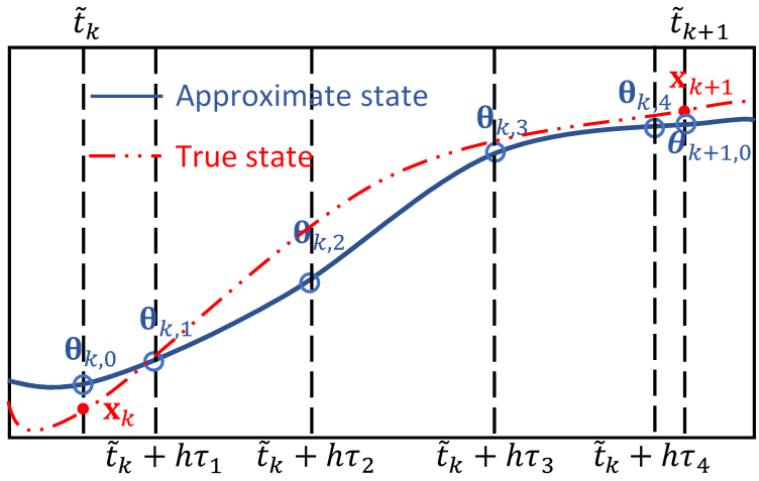


Figure 1.5: Polynomial approximation of state trajectory using collocation.

The infinite OCP is discretized in both controls and states on a finite grid $\tilde{t}_k$ with $k = 0,\dots,N$. This N corresponds to the time horizon length in discrete time. Let the controls be piecewise constant on each interval $[\tilde{t}_k, \tilde{t}_{k+1}]$, i.e. $\tilde{\mathbf{u}}(t) = \overline{\mathbf{u}}_k \; \forall \tilde{t} \in [\tilde{t}_k, \tilde{t}_{k+1}]$ equivalent to the previous discrete time formulation. Collocation approximates the state trajectory over the time interval utilizing a polynomial representation of order K with $K+1$ parameters. Lagrange polynomials are a common choice to represent the interpolation polynomials with $K+1$ interpolation points. This is illustrated in Figure 1.5. Let i refer to term i for each polynomial. The polynomial approximation of the state can then be stated as:

$$\tilde{t} = \tilde{t}_k + h\tau, \quad h = \frac{T}{N} \tag{1.13}$$

$$\hat{\mathbf{x}}(\boldsymbol{\theta}_k, \tilde{t}) = \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot P_{k,i}\left(\frac{\tilde{t} - \tilde{t}_k}{h}\right), \quad P_{k,i}(\tau) = \prod_{j=0, j\neq i}^{K} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{1.14}$$

where $\tilde{t} \in [\tilde{t}_k, \tilde{t}_{k+1}]$, $\tau \in [0,1]$ is the normalized time, $\tau_i \quad i = 0,\dots,K$ are the collocation points, h is the time interval length of one polynomial, $P_{k,i}$ are the Lagrange polynomials, $\hat{\mathbf{x}}(\cdot)$ is the approximate state trajectory, and $\boldsymbol{\theta}_k \in \mathbb{R}^{n_\mathbf{x} \times (K+1)}$ are the polynomial coefficients. The polynomial state approximation has the following property: $\hat{\mathbf{x}}(\boldsymbol{\theta}_k, \tilde{t}_k + h\tau_i) = \boldsymbol{\theta}_{k,i}$, i.e. $\hat{\mathbf{x}}(\cdot)$ is exactly equal to its parameters at the collocation points.

The initial collocation point is at $\tilde{t}_k$ to enforce continuity, i.e. $\tau_0 = 0$. The remaining K collocation points are chosen according to quadrature rules, such as Radau or Legendre quadrature.

The collocation parameters $\boldsymbol{\theta}_k$ need to be adjusted to approximate the continuous time equation $\tilde{\mathbf{f}}(\tilde{\mathbf{x}}(\tilde{t}), \mathbf{u}_k)$ in Equation 1.12. This is achieved by requiring the gradients of the polynomial at K collocation points to be equal to $\tilde{\mathbf{f}}(\tilde{\mathbf{x}}(\tilde{t}), \mathbf{u}_k)$:

$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \left.\frac{dP_{k,i}}{d\tau}\right|_{\tau=\tau_j} = h \cdot \tilde{\mathbf{f}}(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k) \quad j = 1, \ldots, K, \quad k = 1, \ldots, N-1 \tag{1.15}$$

Further, we require equations that ensure that the first state of the first polynomial matches the given initial condition and thereafter continuity conditions such that the final state prediction matches the initial state of the next polynomial:

$$\hat{\mathbf{x}}(\boldsymbol{\theta}_0, 0) = \boldsymbol{\theta}_{0,0} = \hat{\mathbf{x}}_0 \tag{1.16}$$

$$\hat{\mathbf{x}}(\boldsymbol{\theta}_k, t_{k+1}) = \hat{\mathbf{x}}(\boldsymbol{\theta}_{k+1}, \tilde{t}_{k+1}) = \boldsymbol{\theta}_{k+1,0} \quad k = 1, \ldots, N-1 \tag{1.17}$$

This then leads to $n_{\mathbf{x}} \times (K+1) \times N$ equality constraints to determine the unknown parameters $\boldsymbol{\theta}_k$. In essence the original discrete time system model $\mathbf{f}(\overline{\mathbf{x}}_k, \overline{\mathbf{u}}_k)$ in Equation 1.8 is replaced with discretization equations for Equation 1.12 using collocation. The reformulated OCP can then be given as:

$$\begin{aligned}
&\underset{\overline{\mathbf{U}}}{\text{minimize}} \quad J\left(\mathbf{x}_t, \overline{\mathbf{U}}; N\right) \\
&\text{subject to} \\
&\overline{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}(\boldsymbol{\theta}_k, \tilde{t}_{k+1}) \quad \forall k \in \{t, \ldots, t+N-1\}, \quad \boldsymbol{\theta}_{0,0} = \mathbf{x}_t \\
&\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \left.\frac{dP_{k,i}}{d\tau}\right|_{\tau=\tau_j} = h \cdot \tilde{\mathbf{f}}(\boldsymbol{\theta}_{k,j}, \overline{\mathbf{u}}_k) \quad j = 1, \ldots, K, \quad k = 1, \ldots, N-1 \\
&\hat{\mathbf{x}}(\boldsymbol{\theta}_k, t_{k+1}) = \boldsymbol{\theta}_{k+1,0} \quad k = 1, \ldots, N-1 \\
&\overline{\mathbf{u}}_k \in \mathcal{U} \quad \forall k \in \{t, \ldots, t+N\} \\
&\overline{\mathbf{x}}_k \in \mathcal{X} \quad \forall k \in \{t, \ldots, t+N\}
\end{aligned} \tag{1.18}$$

where the nonlinear discrete time equation is given by the polynomial approximation of the continuous time equation. Note using $\hat{\mathbf{x}}(\boldsymbol{\theta}_k, \tilde{t})$ additional state constraints can be introduced at arbitrary times $\tilde{t}$.

1.2.5 Infeasibility handling

Feasibility of the NMPC optimization problem is an important requirement to obtain reliable performance. For practical applications of NMPC this means that the optimization problem is relaxed to ensure feasibility at each sampling time, since accurate uncertainty information is commonly not available. A general way to reformulate an optimization problem is the use of soft constraints using slack

variables. Assume the state constraint set to be given by $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^{n_x} | \mathbf{g}(\mathbf{x}, \mathbf{u}) \leq \mathbf{0}\}$, slack variables can then be introduced as follows:

$$\begin{aligned}
&\underset{\overline{\mathbf{U}}, \boldsymbol{\lambda}}{\text{minimize}} \quad J\left(\mathbf{x}_t, \overline{\mathbf{U}}; N\right) + ||\mathbf{W}\boldsymbol{\lambda}||_1 \\
&\text{subject to} \\
&\overline{\mathbf{x}}_{k+1} = \mathbf{f}(\overline{\mathbf{x}}_k, \overline{\mathbf{u}}_k) \quad \forall k \in \{t, \ldots, t+N-1\}, \quad \overline{\mathbf{x}}_t = \mathbf{x}_t \\
&\overline{\mathbf{u}}_k \in \mathcal{U} \quad \forall k \in \{t, \ldots, t+N\} \\
&\mathbf{g}(\overline{\mathbf{x}}_k, \overline{\mathbf{u}}_k) \leq \boldsymbol{\lambda} \quad \forall k \in \{t, \ldots, t+N\} \\
&\boldsymbol{\lambda} \geq \mathbf{0}
\end{aligned} \tag{1.19}$$

where $\mathbf{W}$ is a positive definite weight matrix. The penalty term in the objective is usually chosen such that it dominates the original objective function to avoid undesirable relaxations.

1.2.6 NMPC under uncertainty

It was assumed so far that the actual plant is identical to the model employed in the NMPC algorithm, i.e. that there is no plant-model mismatch or unknown disturbances. In reality the dynamic model is going to be affected by many different uncertainties, which need to be addressed in the formulation of the NMPC. Assume the nonlinear discrete time system to be given by:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\omega}_t), \quad \mathbf{x}_0 = \hat{\mathbf{x}}_0 \tag{1.20}$$

where $\boldsymbol{\omega}_t$ describes possible uncertainties.

It has been shown that the nominal MPC approach ignoring uncertainties has inherent robustness under certain conditions, however this is usually not sufficient [224]. Instead, uncertainties need to be considered within the NMPC formulation. In principle approaches can be divided into two separate categories, which are robust MPC (RMPC) methods or stochastic MPC (SMPC) approaches. RMPC consider set-membership-type uncertainties, which are assumed to be deterministic and lie in a bounded set, i.e. it is assumed $\boldsymbol{\omega} \in \mathcal{W}$ and $\mathcal{W}$ is assumed to be compact. Uncertainties in the real-world are often of probabilistic nature and hence it may seem more natural to account for the probabilistic occurrence of these explicitly. SMPC therefore assumes uncertainties to be given by probability distributions, such that $\boldsymbol{\omega} \sim p_{\boldsymbol{\omega}}$ and $p_{\boldsymbol{\omega}}$ is a known probability distribution with either bounded or unbounded support.

The earliest RMPC approaches were focused on min-max MPC [53]. These methods focus on minimizing cost, while satisfying constraints under the worst-

case realization of the uncertainties. It has been shown however that, in many cases, these methods are unable to deal with the spread of state trajectories and hence are overly conservative or infeasible [232]. An example of a min-max OCP formulation is given in Equation 1.21. To overcome this issue closed-loop min-max MPC was proposed [155], for which the cost function is minimized over control policies as opposed to control actions. This alleviates the conservativeness by accounting for feedback. Optimizing over general control policies presents an infinite dimensional problem, such that it has been proposed to instead optimize over parameterized feedback policies [21]. Another RMPC approach are tube-based MPC methods, which were first introduced for linear systems [174] and later extended to nonlinear systems [174]. These methods introduce a so-called ancillary controller, which ensures that the evolution of the real system stays in a tube centred around the nominal solution. The tube is based on the pre-computation of a robust positive invariant set. For tube-based MPC the feedback control action reduces the influence of disturbances, however it does not address the optimal performance in the presence of uncertainties.

$$
\begin{aligned}
&\underset{\overline{\mathbf{U}}}{\text{minimize}} \left[\quad \underset{\overline{\boldsymbol{\omega}}_{t:t+N-1}}{\text{maximize}} \quad J\left(\mathbf{x}_t, \overline{\mathbf{U}}; N\right) \right] \\
&\text{subject to} \\
&\overline{\mathbf{x}}_{k+1} = \mathbf{f}(\overline{\mathbf{x}}_k, \overline{\mathbf{u}}_k, \overline{\boldsymbol{\omega}}_k) \quad \forall k \in \{t, \ldots, t+N-1\}, \quad \overline{\mathbf{x}}_t = \mathbf{x}_t \\
&\overline{\mathbf{u}}_k \in \mathcal{U} \quad \forall k \in \{t, \ldots, t+N\} \\
&\overline{\mathbf{x}}_k \in \mathcal{X} \quad \forall k \in \{t, \ldots, t+N\}
\end{aligned}
\tag{1.21}
$$

SMPC generally aims to maximize the expectation of the objective function subject to either chance constraints or expectation constraints. A possible formulation of a stochastic MPC problem is given in Equation 1.22. Stochastic tube MPC methods have been introduced extending the tube-based approaches in RMPC for linear systems affected by additive or multiplicative stochastic uncertainties [57]. There have also been several approaches proposed that optimize parameterized affine disturbance feedback policies [201]. Another class of popular methods for SMPC use scenario-based techniques, which draw several realizations of the stochastic uncertainties and then use these scenarios to approximate the probabilis-

tic constraints and objective [230].

$$
\begin{aligned}
&\underset{\overline{\mathbf{U}}}{\text{minimize}} \left[\underset{\overline{\boldsymbol{\omega}}_{t:t+N-1}}{\mathbb{E}} \left(J\left(\mathbf{x}_t, \overline{\mathbf{U}}; N\right)\right)\right] \\
&\text{subject to} \\
&\overline{\mathbf{x}}_{k+1} = \mathbf{f}(\overline{\mathbf{x}}_k, \overline{\mathbf{u}}_k, \overline{\boldsymbol{\omega}}_k) \quad \forall k \in \{t, \ldots, t+N-1\}, \quad \overline{\mathbf{x}}_t = \mathbf{x}_t \\
&\overline{\mathbf{u}}_k \in \mathcal{U} \quad \forall k \in \{t, \ldots, t+N\} \\
&\mathbb{P}(\mathbf{g}(\overline{\mathbf{x}}_k, \overline{\mathbf{u}}_k) \leq \mathbf{0}) \geq 1-\epsilon \quad \forall k \in \{t, \ldots, t+N\}
\end{aligned}
\tag{1.22}
$$

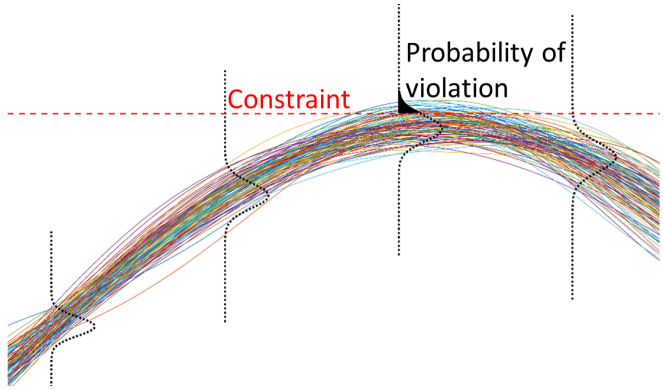


Figure 1.6: Illustration of chance constraints.

The probability of admissible constraint violations ϵ gives SMPC an additional tuning parameter, which can be utilized to trade-off the conservativeness of the SMPC controller with the control objective. Some scenarios of the uncertainty may have a diminishingly small probability of occurrence, which can lead to excessively conservative solutions if considered. RMPC requires the constraints to be fulfilled for all scenarios, while SMPC allows for some constraint violations. Chance constrained SMPC is therefore often motivated by alleviating this conservativeness of RMPC. An illustration of a chance constraint is shown in Figure 1.6. A recent development is also the use of stochastic EMPC, which often leads to improved performance by yielding higher objective values "on average" [19]. Several challenges that may arise for SMPC are as follows:

- The evaluation of chance constraints involves the solution of multivariate integrals, which is commonly computationally prohibitive. Instead, it is common practice to utilize inequalities to upper bound the original chance

constraint, such as the Chebyshev inequality. This however often leads to an overly conservative solution [88].

- Joint chance constraints are also difficult to evaluate exactly and therefore are frequently approximated exploiting individual chance constraints using Boole's inequality. This also leads to an overly conservative solution [88].
- To attain probabilistic guarantees for scenario-based approximations the central limit theorem is commonly utilized, which however requires the samples to be independent. This in turn prevents the use of more efficient sampling approaches, which introduce correlations between samples to improve convergence [119].

1.2.7 Stochastic nonlinear model predictive control

Most work in SMPC has been on linear systems, while stochastic NMPC (SNMPC) has received relatively little attention [175]. This can be in part explained by the difficulty of propagating stochastic uncertainties through a nonlinear system model without being prohibitively expensive. An exception are Markovian systems with finite possible realizations of the stochastic uncertainties, for which efficient algorithms are available [164]. Several methods have been proposed to propagate uncertainties through nonlinear systems, such as Monte Carlo sampling (MC), polynomial chaos expansions (PCEs), Gaussian closure, equivalent linearization, and stochastic averaging [148]. Chance constraints are commonly approximated using either estimates of the mean and variance of the states, or sample-based evaluations.

Several SNMPC approaches have been proposed. Assuming the uncertainties to have only finite number of realizations, multi-stage stochastic nonlinear programming approaches can be used to determine the exact solution [164]. These in general require adhering to the constraints for each scenario. Given this restriction, stochastic stability and recursive feasibility have been proven [205]. For continuous stochastic uncertainties on the other hand it is difficult to propagate the uncertainties without being prohibitively expensive. An easy solution to this problem is given by successive linearization of the nonlinear dynamic system as in extended Kalman filter based NMPC [154, 184]. Similarly, stochastic averaging can be utilized using the unscented transformation [160]. While both approaches are computationally cheap, they are only applicable to moderately nonlinear systems. In [233] the particle filter equations are used to estimate the required statistics. This SNMPC algorithm however becomes quickly prohibitive in complexity due to the required number of samples. Similarly, in [166] Markov Chain MC is proposed to solve constrained nonlinear stochastic optimization problems. A well-established framework exists for both particle filters and Markov Chain MC, which

do not rely on convexity assumptions and converge to a near-optimal solution for a sufficient number of samples. Nonetheless, the number of samples required is often computational prohibitive. For continuous-time dynamic systems Fokker-Planck equations have been used in [50] to propagate the pdfs of the uncertain variables. This approach is however quite expensive, since it requires the online solution of partial differential equations. In [261] it is proposed to use Gaussian mixtures for uncertainty propagation, in which most of the calculations are carried out offline. While this is generally quite efficient, it is only applicable for control actions from a discrete set and low dimensional systems. Conventional RMPC methods can also be applied to include chance constraints by defining the robust sets accordingly [30].

A number of methods in SNMPC rely on regression approaches, such as polynomial chaos expansions (PCEs) [177], supervised clustering [4], and Gaussian processes [36]. These methods replace the implicit mapping between uncertain variables and objective and constraint functions from realizations of the uncertain variable. This mapping can then be utilized to estimate the mean and variance of the objective and constraint functions [177], or sampled directly to estimate the chance constraints [242]. These methods are efficient for low dimensional uncertain variables and quickly become computationally expensive for higher dimensional uncertainties due to exponential scaling with the number of uncertain parameters. This poses a particular issue for time-varying uncertainties, which add to the dimension of the uncertain variables at each time instance. These approaches are therefore regarded as only applicable to time-invariant uncertainties.

There are several issues pertaining to many of the methods presented. Firstly, most methods rely on optimizing over scenarios of the uncertainties. Each scenario represents a nonlinear dynamic equation system and hence quickly leads to a very expensive non-convex optimization problem even for a moderate number of samples. In addition, most techniques rely on the optimization of open-loop control actions to limit the computational complexity and hence ignore feedback. This is however quite conservative. In addition, the estimation of the chance constraints is a very challenging issue in SMPC even in the linear case. Most approaches therefore rely on approximations using Chebyshev's inequality, which leads to further conservativeness. Lastly, there is a lack of guarantees for stability and recursive feasibility due to the difficulty of defining a worst-case scenario and propagating uncertainty sets for nonlinear systems. Lipschitz-based bounds for example quickly lead to infeasible problems [30].

1.2.8 Gaussian processes in NMPC

NMPC algorithms exploit numerous different models, commonly developed by first principles or by black-box identification. For example NMPC algorithms can be based on neural network models [212], fuzzy models [133], or support vector machines [123]. Another increasingly popular nonlinear regression method for black-box identification is based on GP regression. An example of GP regression is shown in Figure 1.7. GP models are probabilistic non-parametric models that also offer information about prediction uncertainties, which are difficult to evaluate appropriately for nonlinear parametric models [144]. This estimation of confidence is a significant advantage of GP models, because this information can be utilised in NMPC algorithms to attain a more robust and reliable control performance. Furthermore, the GP uncertainty measure can be exploited for active learning of dynamic models by steering the system into regions with large uncertainties [147]. Early work using GPs to identify dynamic models can be found in [100, 141].

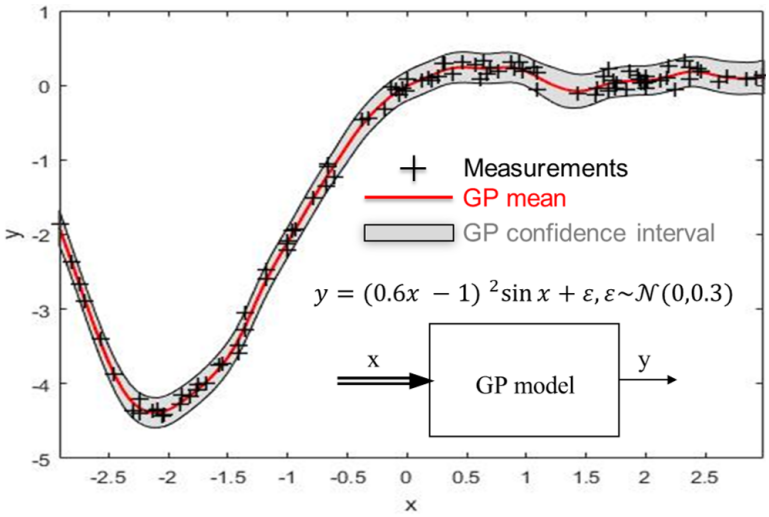


Figure 1.7: Gaussian process regression example with 95% confidence interval.

GPs have generally been applied in two distinct ways for MPC. Firstly, GPs have been employed to identify the required plant model based on input/output data pairs in some diverse areas of control. The use of GPs as approximate plant model was first introduced in [183], which continuously updates a GP for set-point tracking. In [143, 142] the GP plant is instead identified offline. The variance in this approach was constrained to prevent the controller from steering into regions with high uncertainty. [103] derived an explicit solution for GP-based NMPC,

while in [272] rise-sensitive cost functions are introduced for GP MPC for more effective online learning by balancing exploration with exploitation. Secondly, GPs have been exploited to overcome unmodelled errors. In [139] a GP is updated online to overcome unmodelled periodic errors, while in [239] a GP is trained to tackle unmodelled nonlinearities for linear MPC. In [167] it is proposed to apply GPs to update the dynamic model after a fault has occurred. There are also several papers that model the deviation of first principles models and the real system using GPs [113]. Applications of GP NMPC include the control of UAVs [60, 125], the control of a gas-liquid separation process [158], and the steering of autonomous miniature race cars [113]. These and other works show the feasibility of GP-based MPC, however the consideration of the uncertainty in an efficient manner remains a difficult problem. The issues concerning this uncertainty measure face similar issues as presented in the previous section, since GPs are nonlinear probabilistic models. Efficient uncertainty propagation and formulation of chance constraints is therefore difficult.

Most methods to account for the uncertainty measure are based on different uncertainty propagation methods with variance constraints or chance constraints, for example [143, 142, 113, 60, 103, 260]. In [114] an overview is given of the various stochastic propagation techniques. Recently, some methods from robust NMPC have been adopted to incorporate the uncertainty measure. In [147] the uncertainties are represented by ellipsoidal sets and propagated using linearization around the mean function to obtain closed-loop stability guarantees. [169] propose an approach that relies on bounds for the one-step ahead errors to obtain closed-loop stability guarantees. Lastly, [239] develops a linear MPC approach with stability guarantees.

Chapter 2

Thesis overview

2.1 Research objective

The aim of this thesis is the development of novel NMPC formulations that explicitly consider stochastic uncertainties for batch processes to trade-off risk with economic performance. The thesis is divided into two parts.

In the first part it is assumed that a first principles model is available with stochastic uncertainties arising from parametric uncertainties, time varying additive disturbances and state estimation errors. In particular, it aims to develop novel uncertainty propagation algorithms utilized within the NMPC formulation that use a small number of scenarios to remain computationally tractable.

In the second part it is instead assumed that a full first principles model is not available, and therefore part of this first principles model or the full state space model need to be identified from available data. To accomplish this a GP is built and the overall aim is to establish a GP NMPC algorithm that accounts for the stochastic non-parametric error quantified by the GP.

2.2 Outline and contributions

Part I of the thesis is the background and introduction to the thesis with chapters 1-2, while part IV concludes the thesis with some suggestions for future research in chapters 11-12. The main body of the thesis is divided thematically into two parts II-III. Part II consists of chapters 3-6 dealing with novel formulations of NMPC algorithms using uncertainty propagation, while part III is given by chapters 7-10 involving GP-based dynamic modelling and NMPC for batch processes.

Chapter 1 gives some background material for the main part of the thesis.

Chapter 2 gives of an overview of the thesis including the main research objective, outline and contributions, and a list of publications.

Chapter 3 (Paper A) introduces an algorithm using the Unscented transformation for uncertainty propagation for both the SNMPC formulation and state estimation considering noise from disturbances, parametric uncertainties, and state estimation errors. The approach is verified on a semi-batch reactor case study. The work extends existing unscented SNMPC formulations by accounting for parametric noise in addition to additive noise, adding a reformulation using log-transforms to ensure physical variables remain non-negative, and also uses the square-root unscented Kalman filter for improved numerical stability.

Chapter 4 (Paper B) describes an algorithm using PCEs to express the probability distributions of the uncertainties, which are updated using a PCE based state estimator from noisy output measurements and exploited in the NMPC formulation to adhere to chance constraints. Feedback is considered using time-invariant linear feedback gains. The method is verified on a polymerization semi-batch reactor case study. Previous work using PCE based SNMPC is extended to the output feedback case. PCEs are utilized, since for moderate dimensional problems these have been shown to give accurate estimates of the statistics required for relatively low number of samples. To alleviate conservativeness, a time-invariant linear feedback gain was added to the problem, which is optimized over in addition to the open-loop control actions. Lastly, the presence of time-varying additive disturbance noise is considered using the law of total expectation.

Chapter 5 (Paper C) shows how GPs can be utilized as an alternative to PCEs to propagate uncertainties in the NMPC formulation and formulate chance constraints with Gaussian parametric uncertainties. The approach is verified by showing the ability of the GP to accurately describe the probability density functions of the underlying system and the closed-loop behaviour of the resulting NMPC algorithm on a semi-batch reactor case study. The main advantage of using GPs over PCEs in SNMPC is the fact that the uncertainty involved from the approximation of the true model by a finite number of samples is taken into account, which is otherwise ignored by PCEs. In addition, GPs are not prone to unstable swings and are interpolating, i.e. pass exactly through all sample points provided, but otherwise suffer from the same drawbacks as PCEs. The novelty in this paper is the application of GPs to learn the mapping between uncertain parameters and model outputs for SNMPC applications. Terms are identified that can be calculated *offline* to significantly reduce the computational costs online.

Chapter 6 (Paper D) introduces a novel uncertainty propagation method by combining GPs and PCEs with Gaussian parametric uncertainties to obtain the re-

quired statistics, which mainly builds on Chapter 5 by employing PCEs as mean function for the GP. GPs are known to approximate the function well locally, but not as well globally. PCEs on the other hand are better suited for global function approximations, but may not have the same accuracy between data-points as GPs. The combination of both is therefore beneficial. The ability of this new approach to estimate the probability density function, and the mean and variance is extensively verified and compared to other techniques on a semi-batch reactor case study. It is shown that the approach outperforms PCEs, GPs, and the Unscented transformation. The computational times are kept moderately low by pre-computing the expensive terms involved in the method. Lastly, the new algorithm is shown to lead to superior constraint satisfaction over a nominal NMPC algorithm using soft constraints despite the stochastic uncertainties present.

Chapter 7 (Paper E) applies GP regression to obtain a dynamic model of a batch bioreactor from experimental data to show its potential application in this field. Dynamic modeling is an important tool to gain better understanding of complex bioprocesses and to determine optimal operating conditions for process control. In literature two modeling methodologies have been applied to bio systems: kinetic modeling, which necessitates deep mechanistic knowledge, and artificial neural networks (ANN), which in most cases cannot incorporate process uncertainty. To test the performance of this strategy, GPs were applied to model microalgae growth and lutein production based on existing experimental datasets and compared against the results of previous ANNs. The goal of this study is to introduce an alternative modeling strategy, namely Gaussian processes (GP), which incorporates uncertainty but does not require complicated kinetic information. The results show that GPs possess comparable prediction capabilities to ANNs for long-term dynamic bioprocess modeling, while accounting for model uncertainty. This strongly suggests their potential applications in bioprocess systems engineering. Furthermore, a dynamic optimization under uncertainty is performed, avoiding over-optimistic optimization outside of the model's validity.

Chapter 8 (Paper F) introduces an algorithm, which utilizes GP regression to obtain an approximate plant model for NMPC from input/output data given the excellent predictive quality of GP plant models shown in Chapter 7. To account for the uncertainty due to data sparsity the probabilistic nature of the GP is exploited to account for the plant-model mismatch. The majority of works for GP-based MPC consider the uncertainty measure provided using stochastic uncertainty propagation, which substantially increases the computational time due to the propagation approach and most works ignore feedback in the formulations. Due to the issues using uncertainty propagation for the NMPC formulation, we base the NMPC only on cheap evaluations of the GP. In particular, possible plant

models are sampled independently from the GP using recent sampling techniques and explicit back-offs are used to tighten constraints of the NMPC to guarantee satisfaction of chance constraints. Advantages of the approach over existing techniques is the consideration of closed-loop behaviour and online updates of the GP to alleviate conservativeness, while keeping the online computational time low. In addition, independence of samples allows some probabilistic guarantees to be given. Finally, through a comprehensive semi-batch bioprocess case study, the efficiency and potential of this method for the optimisation of complex stochastic systems (e.g. biological processes) is well demonstrated.

Chapter 9 (Paper G) further extends the algorithm introduced in the previous chapter. Apart from considering online updates, it is also shown how to take into account the inherent state dependency of the GP uncertainty measure to alleviate conservativeness. Furthermore, possible uncertainties from additive disturbance noise and the initial state were added. Lastly, a root-finding algorithm is exploited to obtain the required backoffs, which leads to improved convergence and satisfaction of the required probability bounds. All in all, the new algorithm is considerably more reliable than the algorithm in Chapter 8, while accounting for additive disturbance noise, noise on the initial state, and the ability to consider the state dependency of the uncertainty. The approach is extensively verified on a semi-batch bioreactor case study.

Chapter 10 (Paper H) further extends the technique from the previous two chapters to the hybrid modelling case, in which it is assumed that the GP is utilized to identify parts of a first principles model that are difficult to derive using physical laws alone as opposed to identifying the entire state space model. To accomplish this a new modelling approach is introduced using maximum a posteriori, since the required function values for the GP are now latent and need to be estimated. The approach is verified on a semi-batch bioreactor case study. Firstly, it is shown that the identified GP is able to both give accurate predictions and account for the uncertainty accurately. Further, the algorithm is compared to the previous algorithm in Chapter 9 using the same number of data-points. While the hybrid GP-based algorithm is able to determine a reasonable solution, the algorithm in Chapter 9 is evidently unable to establish a solution due to the very large spread of trajectories. This highlights the main advantage of the algorithm to take advantage of a pre-existing first principles model, which drastically reduces the amount of the data required to obtain an accurate plant model.

Chapter 11 concludes the thesis.

Chapter 12 gives some suggestions for future work.

2.3 Publications

2.3.1 Main part of the thesis

The following publications are given each as individual chapters in the main body of the thesis.

A E. Bradford and L. Imsland. Economic Stochastic Model Predictive Control Using the Unscented Kalman Filter. *IFAC-PapersOnLine*, 51(18):417–422, 2018.

B E. Bradford and L. Imsland. Output feedback stochastic nonlinear model predictive control for batch processes. *Computers & Chemical Engineering*, 126:434–450, 2019.

C E. Bradford and L. Imsland. Stochastic Nonlinear Model Predictive Control Using Gaussian Processes. In *2018 European Control Conference (ECC)*, pages 1027–1034, 2018.

D E. Bradford and L. Imsland. Combining Gaussian processes and polynomial chaos expansions for stochastic nonlinear model predictive control. *Journal of Process Control*, submitted, 2020.

E E. Bradford, A. M. Schweidtmann, D. Zhang, K. Jing, and E. A. del Rio-Chanona. Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate Gaussian processes. *Computers & Chemical Engineering*, 118:143–158, 2018.

F E. Bradford, L. Imsland, and E. A. del Rio-Chanona. Nonlinear model predictive control with explicit back-offs for Gaussian process state space models. In *58th Conference on decision and control (CDC)*, pages 4747–4754. IEEE, 2019.

G E. Bradford, L. Imsland, D. Zhang, and E. A. del Rio-Chanona. Stochastic data-driven model predictive control using Gaussian processes. *Computers & Chemical Engineering*, accepted, 2020.

H E. Bradford, L. Imsland, M. Reble, and E. A. del Rio-Chanona. Hybrid Gaussian process modelling applied to economic stochastic model predictive control of batch processes. In *Progress on Economic and Distributed Model Predictive Control and Applications*. Springer, submitted, 2020.

2.3.2 Appendix of the thesis

The following publications are not directly included in the thesis and instead are given in the appendix, since they describe intermediary work.

I E. Bradford and L. Imsland. Expectation constrained stochastic nonlinear model predictive control of a batch bioreactor. *Computer Aided Chemical Engineering*, 40:1621–1626, 2017.

J E. Bradford and L. Imsland. Stochastic NMPC of Batch Processes Using Parameterized Control Policies. *Computer Aided Chemical Engineering*, 44:625–630, 2018.

K E. Bradford and L. Imsland. Stochastic nonlinear model predictive control of a batch fermentation process. *Computer Aided Chemical Engineering*, 46:1237–1242, 2019.

L E. Bradford, M. Reble, A. Bouaswaig, and L. Imsland. Economic stochastic nonlinear model predictive control of a semi-batch polymerization reaction. *IFAC-PapersOnLine*, 52(1):667–672, 2019.

M E. Bradford, M. Reble, and L. Imsland. Output feedback stochastic nonlinear model predictive control of a polymerization batch process. In *2019 18th European Control Conference (ECC)*, pages 3144–3151. IEEE, 2019.

2.3.3 Not included in the thesis

The following are publications that are not directly relevant to the PhD thesis, but were written during the period of the PhD.

- E. A. del Rio-Chanona, X. Cong, E. Bradford, D. Zhang, and K. Jing. Review of advanced physical and data-driven models for dynamic bioprocess simulation: Case study of algae-bacteria consortium wastewater treatment. *Biotechnology and Bioengineering*, 116(2):342–353, 2018.
- E. A. del Rio-Chanona, J. E. A. Graciano, E. Bradford, and B. Chachuat. Modifier-Adaptation Schemes Employing Gaussian Processes and Trust Regions for Real-Time Optimization. *IFAC-PapersOnLine*, 52(1):52–57, 2019.
- P. Petsagkourakis, I. O. Sandoval, E. Bradford, D. Zhang, and E. A. del Rio-Chanona. Reinforcement Learning for Batch-to-Batch Bioprocess Optimisation. *Computer Aided Chemical Engineering*, 46:919–924, 2019.

- P. Petsagkourakis, I. Sandoval, E. Bradford, D. Zhang, and E. A. del Rio-Chanona. Reinforcement Learning for Batch Bioprocess Optimization. *Computers & Chemical Engineering*, 133:106649, 2019.
- P. Petsagkourakis, I. Sandoval, E. Bradford, D. Zhang, and E. A. del Rio-Chanona. Constrained Reinforcement Learning for Dynamic Optimization under Uncertainty. In *IFAC 2020 - 21st IFAC World Congress*, submitted, 2020.
- L. E. Andersson, E. Bradford, and L. Imsland. Gaussian processes modifier adaptation with uncertain inputs using distributed learning and optimization on a wind farm. In *IFAC 2020 - 21st IFAC World Congress*, submitted, 2020.
- L. E. Andersson, E. Bradford, and L. Imsland. Distributed learning for wind farm optimization with Gaussian processes. In *2020 American Control Conference (ACC)*, accepted, 2020.

Part II

Stochastic nonlinear model predictive control using uncertainty propagation

Chapter 3

Economic Stochastic Model Predictive Control Using the Unscented Kalman Filter

This chapter is based on **Paper A**: E. Bradford and L. Imsland. Economic Stochastic Model Predictive Control Using the Unscented Kalman Filter. *IFAC-PapersOnLine*, 51(18):417–422, 2018.

Summary

Economic model predictive control is a popular method to maximize the efficiency of a dynamic system. Often, however, uncertainties are present, which can lead to lower performance and constraint violations. In this paper, an approach is proposed that incorporates the square root Unscented Kalman filter directly into the optimal control problem to estimate the states and to propagate the mean and covariance of the states to consider noise from disturbances, parametric uncertainties and state estimation errors. The covariance is propagated up to a predefined "robust horizon" to limit open-loop covariances, and chance constraints are introduced to maintain feasibility. Often variables in chemical engineering are non-negative, which however can be violated by the Unscented Kalman filter leading to erroneous predictions. This problem is solved by log-transforming these variables to ensure consistency. The approach was verified and compared to a nominal nonlinear model predictive control algorithm on a semi-batch reactor case study with an economic objective via Monte Carlo simulations.

3.1 Introduction

Batch reactors are common in the chemical industry due to their flexibility. The control of batch processes is challenging, since these are operated at unsteady state and are frequently highly nonlinear. This motivates the use of nonlinear model predictive control (NMPC) [184]. The objective of the NMPC is usually to track a set point, but the true objective is to maximize profit. Therefore, in economic MPC (EMPC) the cost function is given by the quantity to be maximized [163], which has attracted significant attention in recent years [223]. For batch reactors the objective is usually a property of the final product and hence the control problem leads to a shrinking horizon implementation.

The performance of the NMPC algorithm depends on the accuracy of the model used. Models of real processes often involve substantial uncertainties, including parametric uncertainties, unaccounted disturbances and state estimation errors. In particular, economic MPC often drives the system to its constraints [163]. Most work to consider uncertainties has been in robust NMPC (RNMPC), which assumes that uncertainties are deterministic and bounded. Important methods for RNMPC are min-max NMPC [64] and tube-based NMPC [173]. An alternative to RNMPC is given by stochastic NMPC (SNMPC), which assumes that the uncertainties are given by known probability distributions. In SNMPC constraints are probabilistic and given by either chance or expectation constraints. The regulation of the probability of constraint violations allows the adjustment of the conservativeness of the solution [175].

In [59] a procedure for SNMPC is introduced based on successive linearization and application of a probabilistic tube method. [31] proposed to use a sampling average approach with variance reduction. A popular tool in SNMPC is given by the so-called polynomial chaos expansion (PCE), which is an efficient alternative to Monte Carlo simulations to propagate probabilistic uncertainties. A major disadvantage of this approach is that the complexity with respect to the number of uncertainty parameters scales exponentially [87]. In [36] a similar method is proposed using Gaussian processes instead. This has the advantage that it also considers the uncertainty of the approximation itself, but otherwise suffers from the same drawbacks. [166] proposed a method based on the Markov chain Monte Carlo approach, which is generally more efficient than common Monte Carlo sampling based techniques, but does not take gradient information into account. Lastly, multi-stage MPC has been used to solve SNMPC problems for discrete uncertainties, which however quickly becomes intractable due to the computational complexity scaling exponentially with the size of the time horizon, number of uncertainty parameters and uncertainty levels [101, 162].

In [32] the Unscented Kalman filter (UKF) is used to estimate the state for output feedback model predictive control and propagate the state estimation error and additive noise from disturbances forward in time. The predicted Gaussian distributions of the states were used to impose probabilistic constraints. A similar approach is given in [89] to propagate state estimation error and additive noise for the control of nonholonomic mobile robots. In addition, [160] and [257] show that the unscented transformation (UT) can be used to efficiently propagate additive disturbance errors. Lastly, in [109] the UT is used to propagate parametric uncertainties. A general advantage of the UT is the linear scaling with respect to the number of uncertain parameters.

In this paper the previous work is unified to take into account state estimation error from the square root UKF, noise from additive disturbances and lastly parametric uncertainties. Log-transformations are used to enforce positiveness of several variables and the square root UKF is used to guarantee positive semi-definiteness of the state covariances [253]. The approach was tested on a semi-batch reactor case study with an economic objective. The robustness of the approach was verified with 4 uncertain parameters and compared to a nominal NMPC approach via Monte Carlo simulations. The paper is divided into the following sections. In the next Section the general SNMPC problem is formulated. In the third Section the square root UKF is introduced and utilised to solve the SNMPC problem. Further, the concepts "robust horizon", log-transformation and linear joint state constraints are outlined. The case study to test the procedure is formulated in Section 4. Section 5 gives the results of the Monte Carlo simulations for the case study. In the last Section conclusions were drawn from the simulation results.

3.2 Nonlinear Model Predictive Control with Linear Chance Constraints

The dynamic system we consider is given by a discrete time stochastic nonlinear system with parametric uncertainties and additive noise. The states and the parameters enter the nonlinear equation system in a non-additive fashion, such that it is practical to write them jointly as x_a.

$$x(k+1) = f(x_a(k), u(k)) + w(k) \tag{3.1}$$

$$y(k) = h(x_a(k), u(k)) + v(k) \tag{3.2}$$

where k is the discrete time, $x_a = [x^T, \theta^T]^T \in \mathbb{R}^{n_x \times n_\theta = L}$ denotes the augmented state vector with a joint dimension of L, $x \in \mathbb{R}^{n_x}$ are the states, $u \in \mathbb{R}^{n_u}$ represents the inputs, $y \in \mathbb{R}^{n_y}$ are the measurements and $\theta \in \mathbb{R}^{n_\theta}$ denotes the parametric uncertainties; the additive disturbance term w lies in $\mathbb{R}^{n_x}$ and the additive measurement noise v lies in $\mathbb{R}^{n_y}$. The equations $f : \mathbb{R}^L \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_x}$ and

$h : \mathbb{R}^L \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_y}$ represent the dynamics of the states and the measurements respectively.

The parametric uncertainties are assumed to be Gaussian distributed with mean vector $m_\theta(k)$ and covariance matrix $\Sigma_\theta(k)$ at stage k. The additive disturbances $w(k)$ and $v(k)$ are assumed to be zero mean independent normal random variables with variances $\Sigma_w(k)$ and $\Sigma_v(k)$ at stage k respectively. The probability density of the initial state $x(0)$ is assumed to be normal with mean $\hat{x}(0)$ and covariance $\Sigma_x(0)$. Assuming that we are at stage n, let $\mathcal{Y}_n$ stand for the measurements collected thus far. Subsequently, $\mathbb{E}_{\mathcal{Y}_n}(\cdot)$ and $\mathbb{P}_{\mathcal{Y}_n}(\cdot)$ denote the expectation and probability conditioned on $\mathcal{Y}_n$ respectively [271]. The goal of the SNMPC algorithm at stage n is to determine a control sequence over a finite time horizon to adjust the probability distributions of the states to optimize an objective, while adhering to predefined probabilistic constraints, given imperfect information through $\mathcal{Y}_n$. A general SNMPC problem formulation at stage n can be given as follows, with deterministic constraints on the inputs and joint, linear chance constraints on the states:

Finite-horizon SNMPC problem with chance constraints

$$
\begin{aligned}
&\underset{\mathbf{u}_N}{\text{minimize}} \quad \mathbb{E}_{\mathcal{Y}_n}\left(J(N, x(n), \mathbf{u}_N)\right) \\
&\text{subject to} \\
&x(n+k+1) = f(x_a(k), u(k)) + w(n+k) \\
&y(n+k) = h(x_a(k), u(k)) + v(n+k) \\
&\mathbb{P}_{\mathcal{Y}_n}(l_k^{iT} x(n+k) \leq g_k^i) \geq 1 - p_k^i \\
&\qquad\qquad \forall (k,i) \in \{1,\ldots,N\} \times \{1,\ldots,n_g\} \\
&u(n+k) \in \mathbb{U}_k \quad \forall k \in \{0,\ldots,N-1\}
\end{aligned}
\tag{3.3}
$$

where the time horizon is given by N, n_g is the number of linear state constraints, $l_k^i \in \mathbb{R}^{n_x}$ and $g_k^i \in \mathbb{R}$ define each linear state constraint, the input constraints are represented by $\mathbb{U}_k \subset \mathbb{R}^{n_u}$, $\mathbf{u}_N := \{u(n), \ldots, u(n+N-1)\}$ is a collection of inputs over the finite horizon N from an initial stage n and $J(N, x(n), \mathbf{u}_N))$ is the objective function. The chance constraints are set to be violated by a probability of $p_k^i \in (0,1) \subset \mathbb{R}$.

3.3 Incorporation of the square root unscented Kalman filter

3.3.1 Transformation of variables with lower bound

First principle equations for batch processes are commonly given as continuous differential equations, which can be discretized using numerical integration techniques to obtain equations in the form of Eq. (3.1). In this work orthogonal

collocation was used for numerical integration [62]. The log-transformation is, however, directly applied to the continuous differential equation system. Many variables in chemical engineering are non-negative due to physical constraints, such as temperatures, concentrations, volumes, etc.. The methods available in literature to incorporate state constraints in the UKF are not suitable for our specific problem, since these are discontinuous [237]. Instead, we suggest to log-transform the variables to ensure a lower bound on the variables. Let x' be the variable that cannot be lower than a. Then define x as:

$$x = \log(x' - a) \tag{3.4}$$

If we now work with x in the problem rather than x', then x' is guaranteed to remain larger than a, i.e. we have implicitly introduced the following constraint:

$$x' > a \tag{3.5}$$

The differential equation of x' can then be transformed in the following way to obtain the required differential equation of x:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{\mathrm{d}x'}{\mathrm{d}t}\frac{1}{x' - a} \tag{3.6}$$

Lastly, given that we know x is normally distributed with mean $\hat{x}$ and covariance Σ_x, i.e. $x \sim \mathcal{N}(\hat{x}, \Sigma_x)$, x' then follows a log-normal distribution, with mean and covariance given by [108]:

$$\hat{x'}_i = \exp\left(\hat{x}_i + \frac{\Sigma_{xi,i}}{2}\right) \tag{3.7}$$

$$\Sigma_{x'i,j} = \exp\left(\hat{x}_i + \hat{x}_j + \frac{\Sigma_{xi,i} + \Sigma_{xj,j}}{2}\right)(\exp(\Sigma_{xi,j}) - 1) \tag{3.8}$$

where $\hat{x'}$ and $\Sigma_{x'}$ are the mean and covariance of x' respectively

Eq. (3.7) can be used to obtain the state estimate for the true variable x' from x with the corresponding covariance matrix from Eq. (3.8).

3.3.2 Propagation of state probability distributions using the square root Unscented Kalman filter

The problem in Eq. (3.3) is intractable, since it requires the propagation of conditional probability densities of the states through nonlinear transformations. Instead we use the UT to approximate the mean and covariance of this probability density. In our case the uncertain input is given by the states and uncertain parameters. The UT creates a set of sampling points, which depend on the mean and

covariance of the Gaussian distributed input. The mean and covariance of the UT are accurately estimated up to third order [236], which is an advantage of the UT over more conventional linearization approaches.

To obtain the state estimate the Kalman filter equations are used at stage n. For the propagation of the mean and covariance of the states the UT is repeatedly applied to Eq. (3.1), assuming at each stage that the output follows a Gaussian distribution. This approach is summarised in the Algorithm 3.1 box and illustrated in Fig. 3.1. The equation system is given such that $\hat{x}(n|n)$, the state estimate, is also calculated given the current measurement $y(n)$, the previous state estimate $\hat{x}(n-1|n-1)$, the previous state covariance $\Sigma_x(n-1|n-1)$ and the previous control input $u(n-1)$. Guidelines on how to set the scaling parameters ($\omega_i^{\mu}, \omega_i^{c}, \lambda$) and definitions of qr$(\cdot)$, $\cdot/\cdot$ and cholupdate$(\cdot,\cdot,\cdot)$ can be found in [253].

The considered OCP in Eq. (3.3) is open-loop, which does not account for the NMPC to have reduced covariances through feedback by the state and bias update. This leads to the predicted conditional covariances to increase with k and therefore the OCP becoming increasingly conservative with larger time horizons. Eventually the OCP becomes infeasible [271]. The "robust horizon" is utilised as in [32], up to which the covariances are propagated to address the problem of growing covariances. Hence, for the square root UKF in the Algorithm 3.1 box, equations were added, such that the covariance matrix is constant after a defined "robust horizon" t_R. This is similar to [271] who propagates the covariance at the first stage from a Kalman filter in linear MPC. [89] introduces "fake measurements". This is an interesting approach to take into account information gained to learn parameters, but is expensive since it requires additional equations.

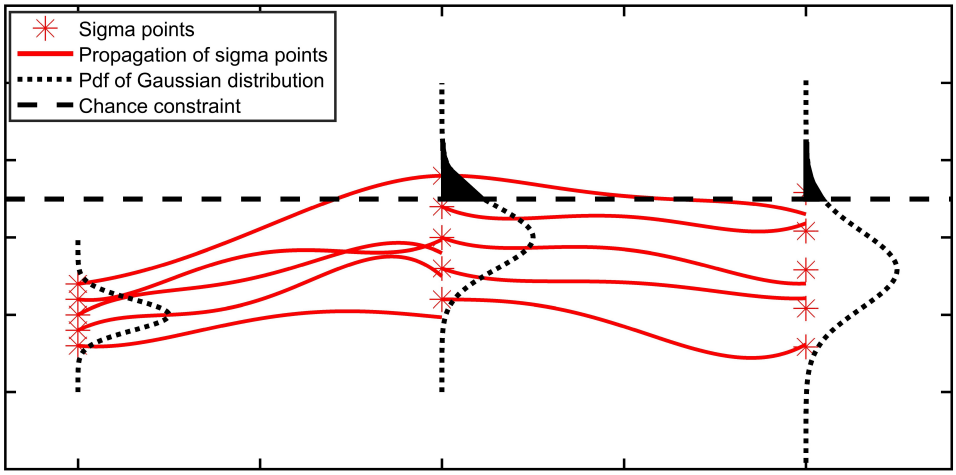


Figure 3.1: Illustration of UKF SNMPC algorithm: Each Sigma point resembles a different input, which are then propagated through the nonlinear transformation to the next stage as indicated by the red lines. These are then used to estimate the mean and covariance of the Gaussian distribution of the states at the next stage. The probability of violating the chance constraint shown is given by the area under the pdf.

Algorithm 3.1: Square root Unscented Kalman filter with additive noise and parametric uncertainty

Initialization

Input: $\hat{x}(n-1|s),\ \Sigma_x(n-1|s),$
$u(n-1),\ y(n),\ \lambda,\ \omega^{\mu},\ \omega^{c},\ t_R,\ n,\ N$
$m_\theta(n+k),\ \forall k \in \{1,\dots,N\},$
$\Sigma_\theta(n+k),\ \forall k \in \{1,\dots,N\}$
$\Sigma_w(n+k),\ \Sigma_v(n+k)\ \forall k \in \{1,\dots,N\}$
$f(\cdot),\ h(\cdot)$

For $s = n-1, k = 0$ and $s = n,\ k \in \{1,\dots,N\}$

Definition of Sigma points

$$\hat{x}_a(n+k-1|s) = [\hat{x}(n+k-1|s)^T\ m_\theta(n+k-1)^T]^T \tag{3.9a}$$

$$\Sigma_a^{1/2}(n+k-1|s) = \operatorname{diag}(\Sigma_x^{1/2}(n+k-1|s), \Sigma_\theta^{1/2}(n+k-1)) \tag{3.9b}$$

$$\mathcal{X}(n+k-1|s) = [\hat{x}_a(n+k-1|s)$$

$$\hat{x}_a(n+k-1|s) + \sqrt{L+\lambda}\,\Sigma_a^{1/2}(n+k-1|s) \tag{3.9c}$$
$$\hat{x}_a(n+k-1|s) - \sqrt{L+\lambda}\,\Sigma_a^{1/2}(n+k-1|s)]$$

Covariance and mean approximation of predictions

$$\mathcal{X}_i(n+k|s) = f(\mathcal{X}_i(n+k-1|s), u(n+k-1)) \tag{3.10a}$$

$$\hat{x}(n+k|s) = \sum_{i=0}^{2L} \omega_i^{\mu} \mathcal{X}_i(n+k|s) \tag{3.10b}$$

$$\forall k \le t_R \quad \Sigma_x^{1/2}(n+k|s) = \mathrm{qr}([\sqrt{\omega_1^c}(\mathcal{X}_{1:2L}(n+k|s) - \hat{x}(n+k|s)) \quad \Sigma_w^{1/2}(n+k)]) \tag{3.10c}$$

$$\forall k \le t_R \quad \Sigma_x^{1/2}(n+k|s) = \mathrm{cholupdate}(\Sigma_x^{1/2}(n+k|s), \mathcal{X}_0(n+k|s) - \hat{x}(n+k|s), \omega_0^c) \tag{3.10d}$$

$$\forall k > t_R \quad \Sigma_x^{1/2}(n+k|n) = \Sigma_x^{1/2}(n+k-1|n) \tag{3.10e}$$

Covariance and mean approximation of observations

$$\phi_i(n|n-1) = h(\mathcal{X}_i(n|n-1), u(n-1)) \tag{3.11a}$$

$$\hat{y}(n|n-1) = \sum_{i=0}^{2L} \omega_i^{\mu} \phi_i(n|n-1) \tag{3.11b}$$

$$\Sigma_{yy}^{1/2}(n|n-1) = \mathrm{qr}([\sqrt{\omega_1^c}(\phi_{1:2L}(n|n-1) - \hat{y}(n|n-1)) \quad \Sigma_v^{1/2}]) \tag{3.11c}$$

$$\Sigma_{yy}^{1/2}(n|n-1) = \mathrm{cholupdate}(\Sigma_{yy}(n|n-1), \phi_0(n|n-1) - \hat{y}(n|n-1), \omega_0^c) \tag{3.11d}$$

$$\Sigma_{xy}(n|n-1) = \sum_{i=0}^{2L} \omega_i^c (\mathcal{X}^{(i)}(n|n-1) - \hat{x}(n|n-1))(\phi^{(i)}(n|n-1) - \hat{y}(n|n-1))^T \tag{3.11e}$$

Update of states from available measurements

$$K(n) = (\Sigma_{xy}(n|n-1)/\Sigma_{yy}^{1/2T}(n|n-1))/\Sigma_{yy}^{1/2}(n|n-1) \tag{3.12a}$$
$$\hat{x}(n|n) = \hat{x}(n|n-1) + K(n)(y(n) - \hat{y}(n|n-1)) \tag{3.12b}$$
$$U = K(n)\Sigma_{yy}^{1/2}(n|n-1) \tag{3.12c}$$
$$\Sigma_x(n|n) = \mathrm{cholupdate}(\Sigma_x(n-1|n-1), U, -1) \tag{3.12d}$$

3.3.3 Probability constraints

The probability constraints on the states of interest are in the form of linear constraints as defined in Eq. (3.3):

$$\mathbb{P}(l^T x \leq g) \geq 1 - \epsilon \tag{3.13}$$

Using Chebychev's inequality the probability constraints in Eq. (3.13) can be robustly transformed to the following equation [160]:

$$l^T \hat{x} + \sqrt{\beta l^T \Sigma_x l} \leq g \tag{3.14}$$

where $\hat{x}$ and Σ_x are the mean and covariance of x respectively and $\beta = \frac{1-\epsilon}{\epsilon}$

For the untransformed variables Eq. (3.13) can be directly used with the mean and covariance predicted by the square root UKF. For the transformed variables we instead use the mean and covariance given by Eq. (3.7) and Eq. (3.8) respectively. This is also done so that we can define the robust horizon.

$$\forall k > t_R \quad \Sigma_{x'}(n+k|n) = \Sigma_{x'}(n+k-1|n) \tag{3.15}$$

3.3.4 Square root UKF SNMPC formulation

Given linear chance constraints of the form in Eq. (3.13), a simplified SNMPC formulation can be stated as follows:

Finite horizon SNMPC problem with incorporated square root UKF and chance constraints

$$\begin{aligned}
&\underset{\mathbf{u}_N}{\text{minimize}} \quad \mathbb{E}_{y_n}(J(N, x(n), \mathbf{u}_N)) \\
&\text{subject to} \\
&\mathbb{P}_{y_n}(l_k^{iT} x(n+k) \leq g_k^i) \geq 1 - p_k^i \\
&\qquad\qquad \forall (k,i) \in \{1,\ldots,N\} \times \{1,\ldots,n_g\} \\
&u(n+k) \in \mathbb{U}_k \quad \forall k \in \{0,...,N-1\} \\
&(3.9) - (3.12) \text{ square root UKF based on transformed variable } x \\
&(3.7), (3.8), (3.15) \text{ Variance and mean of untransformed variable } x'
\end{aligned} \tag{3.16}$$

where the probability constraints can be reformulated as shown in Section 3.3. The open-loop problem can then be used in a receding horizon fashion to obtain a SNMPC algorithm as shown in the box for Algorithm 3.2.

Algorithm 3.2: Square root UKF SNMPC with receding horizon

Initialize: Supply $\hat{x}(0|0)$, $\Sigma_x(0|0)$, $u(0)$ and define (3.16)
At each sampling time $n = 1, 2, 3, \ldots$

- Take measurements $y(n)$
- Solve (3.16) with $\hat{x}(n-1|n-1)$, $\Sigma_x(n-1|n-1)$, $u(n-1)$, $y(n)$ and obtain $u(n)$, $\hat{x}(n|n)$, $\Sigma_x(n|n)$
- Apply $u(n)$ to the real system

3.4 Semi-Batch Reactor Case Study

3.4.1 Semi-batch reactor model

To test the procedure the same case study as in [32] based on a DAE system in [94] is used, however parametric uncertainties were added and all the states were log-transformed. The following series reaction takes place in the reactor with H_2SO_4 as catalyst:

$$2A \xrightarrow[(1)]{k_{1A}} B \xrightarrow[(2)]{k_{2B}} 3C$$

The reactions are first order. The first reaction step is exothermic, while the second reaction step is endothermic. A heat exchanger is utilised to control the temperature. The following DAE system describes the dynamic behaviour of the semi-batch reactor:

$$\dot{C}_A = (-k_{1A}C'_A + (\theta_1 - C'_A)\frac{F}{V'})/C'_A, \tag{3.17a}$$

$$\dot{C}_B = (0.5k_{1A}C'_A - k_{2B}C'_B - C'_B\frac{F}{V'})/C'_B, \tag{3.17b}$$

$$\dot{C}_C = (3k_{2B}C'_B - C'_C\frac{F}{V'})/C'_C, \tag{3.17c}$$

$$\dot{T} = \left(\frac{(\theta_2(T_a - T') - F\theta_1 C_{P_A}(T' - T_0)}{(C'_A C_{P_A} + C'_B C_{P_B} + C'_C C_{P_C})V' + \theta_4 C_{P_{H_2SO_4}}} + \frac{(-\Delta H_{Rx1A}k_{1A}C'_A - \Delta H_{Rx2B}k_{2B}C'_B)V'}{(C'_A C_{P_A} + C'_B C_{P_B} + C'_C C_{P_C})V' + \theta_4 C_{P_{H_2SO_4}}}\right)/T', \tag{3.17d}$$

$$\dot{V} = F/V', \tag{3.17e}$$

$$k_{1A} = \theta_3 \exp\left(-E_{1A}\left(\frac{1}{420} - \frac{1}{T'}\right)\right), \tag{3.17f}$$

$$k_{2B} = A_2 \exp\left(-E_{2B}\left(\frac{1}{400} - \frac{1}{T'}\right)\right), \tag{3.17g}$$

$$C'_A = \exp(C_A),\ C'_B = \exp(C_B),\ C'_C = \exp(C_C) \tag{3.17h}$$

$$T' = \exp(T),\ V' = \exp(V) \tag{3.17i}$$

where C'_A, C'_B, C'_C are the concentrations in moldm^{-3} of species A, B and C respectively, T' is the temperature in K of the reactor and V' is the liquid volume in dm^3. C_A, C_B, C_C, T and V are the log-transformed states of C'_A, C'_B, C'_C, T' and V' respectively. The deterministic parameters were kept at their nominal values, which can be found in [94]. The uncertain parameters are jointly given by the vector θ, which are assumed to be normally distributed, with constant mean $m_\theta = [4, 45000, 0.08, 100]^T$ and constant covariance $\Sigma_\theta = \text{diag}([0.1, 2e7, 1.6\text{e}{-4}, 5])$. The inputs of the problem are given by the flow rate of pure A entering the reactor F in dm^{3}h^{-1} and the temperature of the heat exchanger T_a in K.

In compact form we can write $x' = [C'_A, C'_B, C'_C, T', V']^T$, $x = [C_A, C_B, C_C, T, V]^T$ and $u = [F, T_a]^T$. Using orthogonal collocation the continuous time equations can be given as a discrete time equation system in the form:

$$x(k+1) = f(x(k), u(k)) + w(k) \tag{3.18}$$

where $f(x(k), u(k))$ describes the DAE system in Eq. (3.17) and $w(k)$ is additive Gaussian noise with a constant covariance matrix $\Sigma_w = \text{diag}([1\text{e}{-3}, 1\text{e}{-3}, 1\text{e}{-3}, 1\text{e}{-6}, 1\text{e}{-6}])$.

Lastly, the measurement dynamics need to be defined, which are given by the following equation:

$$y(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} x'(k) + \nu(k) \tag{3.19}$$

where $\nu(k)$ is additive Gaussian noise with a constant covariance matrix $\Sigma_\nu = \text{diag}([1\text{e}{-3}, 1\text{e}{-3}, 1\text{e}{-3}])$.

3.4.2 SNMPC problem

The OCP to be solved is formulated below. The economic objective is to maximize the amount of C at a fixed final batch time with a penalty for excessive control actions. The feed rate can be varied between 0dm^{3}h^{-1} and 250dm^{3}h^{-1} and the heat exchanger temperature can be adjusted between $270K$ and $500K$. The liquid volume inside the semi-batch reactor is constrained to lie below 800dm^3, while the

temperature is constrained to lie below 440K. The OCP problem is given by:

$$
\begin{aligned}
&\underset{\mathbf{u}_N}{\text{minimize}} \quad -(\hat{x}_2(n+N|n)+\hat{x}_4(n+N|n))+\Delta U^T S \Delta U \\
&\text{subject to} \\
&\hat{x'}_3(n+k|n)+\sqrt{\beta \Sigma_{x'3,3}(n+k|n)} \leq 440 \quad \forall k \in \{1,\ldots,N\} \\
&\hat{x'}_4(n+k|n)+\sqrt{\beta \Sigma_{x'4,4}(n+k|n)} \leq 800 \quad \forall k \in \{1,\ldots,N\} \\
&u(n+k) \in [0,250]\times[270,500] \quad \forall k \in \{0,\ldots,N-1\} \\
&(3.9)-(3.12) \text{ square root UKF based on transformed variable } x \\
&(3.7)-(3.8),(3.15) \text{ variance and mean of true variable } x'
\end{aligned}
\tag{3.20}
$$

where $\beta = \frac{\epsilon}{1-\epsilon}$, $\Delta U = [u(n+k)-u(n+k-1)]_{k\in\{1,\ldots,N-1\}}$ and $S = \text{diag}([8\text{e}{-6}, 2\text{e}{-6}])$. For Eqs. (3.9) – (3.12),(3.15) the robust horizon t_R was set to 2, the required scaling parameters can be determined from [253] with $\alpha = 0.9$, $\beta = 2$ and $\kappa = 1$, $f(\cdot)$ is defined in Eq. (3.17) and Eq. (3.18) and $h(\cdot)$ in Eq. (3.19).

The problem objective is given at a fixed final time, such that a shrinking horizon implementation was used.

3.5 Simulation studies

The final batch time was set to $4h$ with the total number of sampling points set to $N_t = 20$. The OCP in Eq. (3.20) was solved repeatedly using Casadi [8] by employing direct collocation in Python. The degree of the polynomials was set to 4. The nonlinear programming problem was solved utilising IPOPT [258]. IDAS [116] simulated the "real" plant. At time $n = 1$, Algorithm 3 needs to be initialized by with the "previous" covariance matrix, mean and control action. These were set to $\hat{x}(0|0) = [\log(1\text{e}{-3}), \log(1\text{e}{-3}), \log(1\text{e}{-3}), \log(290), \log(100)]^T$, $\Sigma_x(0|0) = \text{diag}([1\text{e}{-3}, 1\text{e}{-3}, 1\text{e}{-3}, 1\text{e}{-3}, 1\text{e}{-3}])$ and $u(0) = [0, 290]^T$ respectively.

To test the robustness of the method 200 Monte Carlo simulations were performed, i.e. by sampling different realizations of parameters, additive disturbances and initial conditions for the “real system”, again with $\epsilon = 0.05$. The various trajectories can be seen in Fig. 3.4. For comparison purposes a nominal NMPC was run on 200 Monte Carlo simulations, for which the results are shown in Fig. 3.3. The UKF SNMPC overall performs well and leads to a relatively small number of constraint violations, while the nominal NMPC can be seen to violate both constraints substantially.

Lastly, the method was run for 3 different values of ϵ and for the nominal NMPC algorithm, with 100 Monte Carlo samples. The obtained objective values

are illustrated in Fig. 3.2 as a box plot to highlight the trade-off between conservativeness and performance. The red line in the box plot indicates the median of the objective values, while the blue lines represent the upper and lower quartiles. The black lines give the smallest and largest value attained from the simulations, excluding outliers, shown as red crosses. We can clearly see that the median amount of C at the final batch time consistently increases with ϵ as expected, since an increase in ϵ leads to less conservativeness. The nominal NMPC algorithm leads to the largest amount of C on average.

3.6 Conclusion

Overall a new algorithm is proposed for SNMPC with efficient formulation of the probability constraints, which has been shown to be an efficient means to account for uncertainties from state estimates, disturbances and parameters for an economic model predictive control problem of a semi-batch reactor. The algorithm was able to keep nearly all 200 Monte Carlo simulations within the constraints, while it was shown that a nominal NMPC algorithm leads to significant constraint violations. In addition, important issues such as the prevention of negative concentrations were addressed by log-transformations.

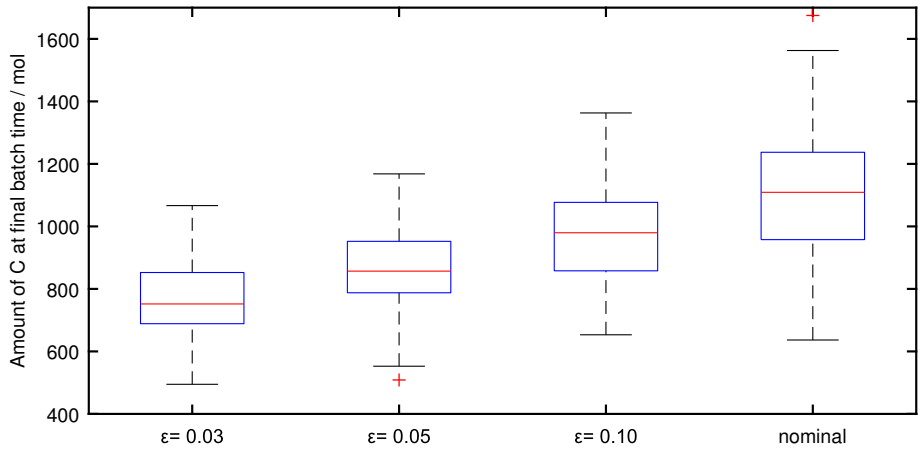


Figure 3.2: Box plot of 100 Monte Carlo simulations for different values of ϵ for the UKF SNMPC algorithm based on the OCP in Eq. (3.20) and for the nominal NMPC algorithm

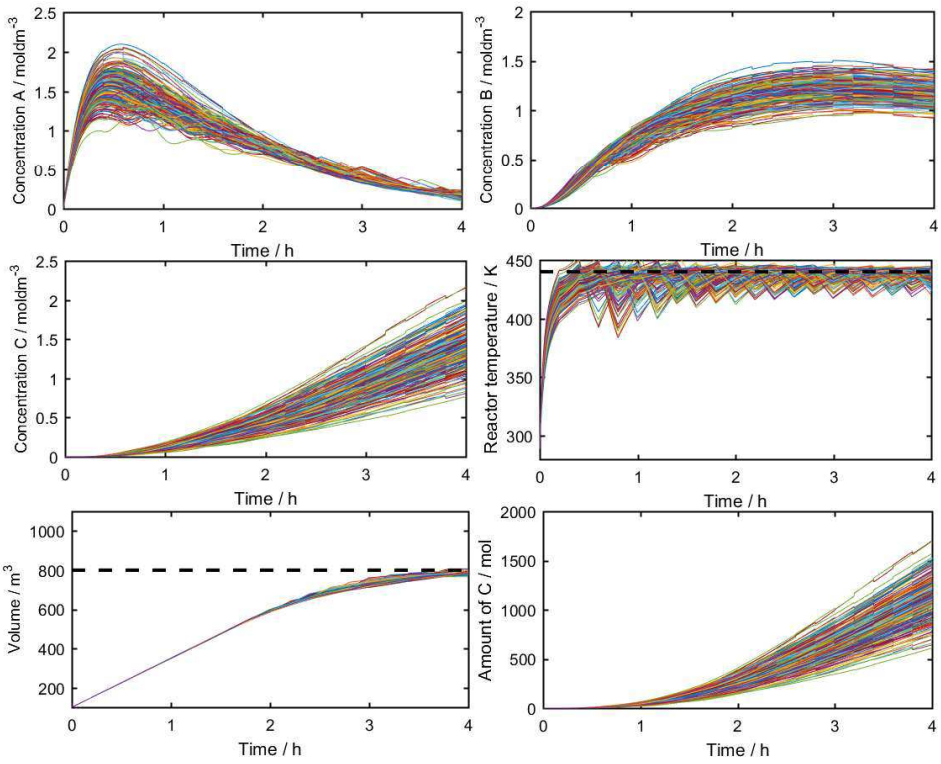


Figure 3.3: 200 Monte Carlo trajectories of the "real" system from a nominal NMPC algorithm

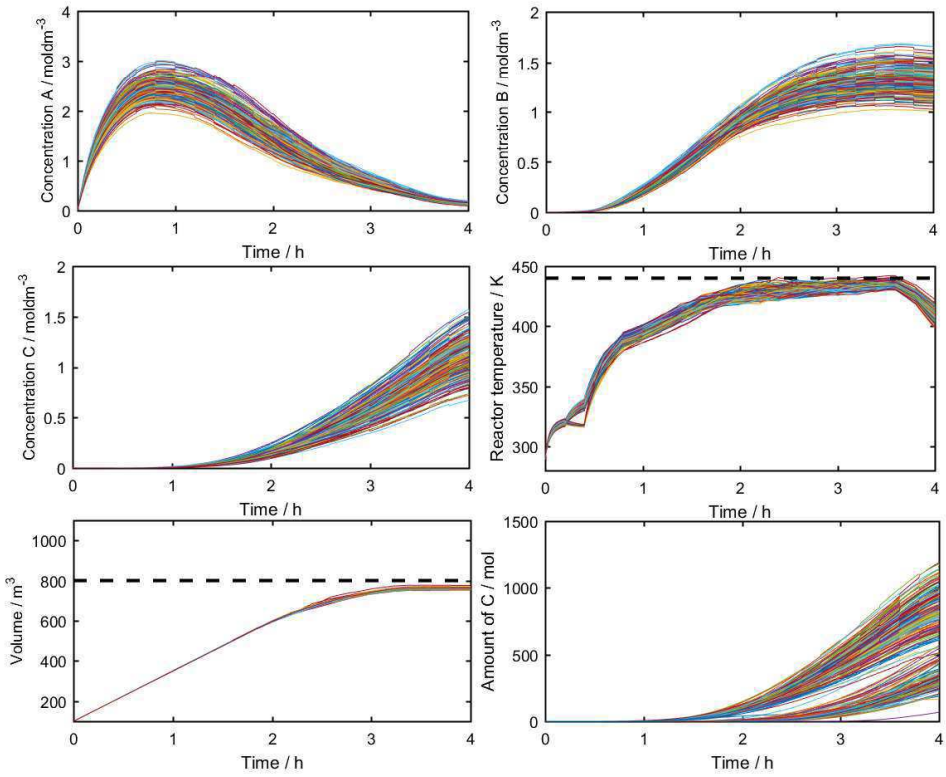


Figure 3.4: 200 Monte Carlo trajectories of the "real" system from the SNMPC algorithm based on the OCP in Eq. (3.20) with $\epsilon = 0.05$

Chapter 4

Output feedback stochastic nonlinear model predictive control for batch processes

This chapter is based on **Paper B**: E. Bradford and L. Imsland. Output feedback stochastic nonlinear model predictive control for batch processes. *Computers & Chemical Engineering*, 126:434–450, 2019.

The paper is an extension work from:

- **Paper K**: E. Bradford and L. Imsland. Stochastic nonlinear model predictive control of a batch fermentation process. *Computer Aided Chemical Engineering*, 46:1237–1242, 2019.
- **Paper L**: E. Bradford, M. Reble, A. Bouaswaig, and L. Imsland. Economic stochastic nonlinear model predictive control of a semi-batch polymerization reaction. *IFAC-PapersOnLine*, 52(1):667–672, 2019.
- **Paper M**: E. Bradford, M. Reble, and L. Imsland. Output feedback stochastic nonlinear model predictive control of a polymerization batch process. In *2019 18th European Control Conference (ECC)*, pages 3144–3151. IEEE, 2019.

Summary

Batch processes play a vital role in the chemical industry, but are difficult to control due to highly nonlinear behaviour and unsteady state operation. Nonlin-

ear model predictive control (NMPC) is therefore one of the few promising approaches. Batch process models are however often affected by uncertainties, which can lower the performance and cause constraint violations. In this paper we propose a shrinking horizon NMPC algorithm accounting for these uncertainties to optimize a probabilistic objective subject to chance constraints. At each sampling time only noisy output measurements are observed. Polynomial chaos expansions (PCE) are used to express the probability distributions of the uncertainties, which are updated at each sampling time using a PCE state estimator and exploited in the NMPC formulation. The approach considers feedback by using time-invariant linear feedback gains, which alleviates the conservativeness of the approach. The NMPC scheme is verified on a polymerization semi-batch reactor case study.

4.1 Introduction

Batch processes are used in many sectors in the chemical industry due to their inherent flexibility to produce multiple products and deal with variations in feedstock, product specifications, and market demand. Batch processes are difficult to control due to frequently highly nonlinear behaviour and unsteady state operation leading to an increased acceptance of advanced control methods in industry [184]. Model predictive control (MPC) is a popular advanced control method for multivariate plants with process constraints. At each sampling time MPC solves an optimal control problem (OCP) based on a dynamic plant model to evaluate a finite sequence of control actions [165]. Feedback is introduced to this procedure through the state update. Nonlinear MPC (NMPC) employs a nonlinear dynamic model to deal with systems that display strong nonlinear behaviour. In particular, the use of first principles models has become feasible due to the advent of improved optimization methods [61].

Many dynamic model predictions however are affected by significant uncertainties, such as parametric uncertainties, disturbance noise, or state estimation errors. This may have an adverse effect on the control performance of the MPC and may lead to constraint violations. While the feedback introduced from the state update gives MPC some degree of robustness with regards to uncertainties, this is often not enough and hence explicit consideration of these uncertainties in the MPC formulation is crucial [175]. Robust NMPC (RNMPC) assumes the uncertainties present to lie in a bounded set [22]. RNMPC approaches include tube-based NMPC [173] and min-max NMPC [64]. These approaches allow guarantees on stability and performance in the worst-case realization of the uncertainties, which however may have a very small chance of occurrence and hence the solution may be overly conservative. Alternatively, stochastic NMPC (SNMPC) methods have been proposed, which assume the uncertainties to follow known probability

density functions (pdf). Constraints and objective are formulated probabilistically. This allows for a pre-defined level of constraint violations in probability alleviating the issue of RNMPC by trading-off risk with closed-loop performance [177].

Several SNMPC approaches have been proposed. Assuming the uncertainties to have only finite number of realizations, multi-stage stochastic nonlinear programming approaches can be used to determine the exact solution [164]. Given this restriction, stochastic stability and recursive feasibility have been proven [205]. For continuous stochastic uncertainties on the other hand it is difficult to propagate the uncertainties without being prohibitively expensive. An easy solution to this problem is given by successive linearization of the nonlinear dynamic system as in extended Kalman filter based NMPC [154]. In [34] stochastic averaging is applied using the unscented transformation (UT). While both approaches are computationally cheap, they are only applicable to moderately nonlinear systems. In [233] the particle filter equations are used to estimate the required statistics. This SNMPC algorithm however becomes quickly prohibitive in complexity due to the required number of samples. Similarly, in [166] Markov Chain MC is used instead with similar restrictions. In [4] a supervised clustering algorithm is proposed to reduce the number of samples required to estimate the relevant statistical properties required, which however remains computationally expensive. For continuous-time dynamic systems Fokker-Planck equations have been used in [50] to propagate the pdfs of the uncertain variables. This approach is however quite expensive, since it requires the online solution of partial differential equations. In [261] it is proposed to use Gaussian mixtures for uncertainty propagation, in which most of the calculations are carried out offline. While this is generally quite efficient, it is only applicable for control actions from a discrete set and low dimensional systems. Alternatively, the statistics can be estimated using regression approaches such as polynomial chaos expansions (PCE) [87] or Gaussian processes [36]. While these methods are considerably more efficient than MC sampling, they are only applicable for moderate dimensional problems due to exponential scaling with the number of uncertain parameters. In a similar fashion power series expansions (PSE) have been employed, which use a Taylor expansion of the dynamic system to propagate the uncertainties with similar advantages and disadvantages as using PCEs. While PSEs have been shown to have comparable accuracy to the PCE approach with the same polynomial order, it is difficult to extend to polynomial approximation of PSEs to orders higher than 2 [138]. Much of the work in SNMPC has been restricted to full state feedback with some exceptions. The UT work in [34] uses the Unscented Kalman filter equations for state estimation, a probabilistic high-gain observer has been proposed in [120] for state estimation in conjunction with a continuous-time SNMPC formulation, and lastly in [233] particle filter equations were used for both propagation and state estimation.

In particular, PCE has received a lot of interest for SNMPC. PCEs are used to estimate the statistics online by building a stochastic surrogate model. This surrogate is then used in [87] to approximate objective and constraints in expectation. The work was then further extended in [177] to include chance constraints by using Chebychev's inequality. [242] samples instead the PCE directly and approximates the chance constraints using the empirical distribution function, which is less conservative, but also computationally more expensive. PCE is often computationally too expensive for time-invariant uncertainties. The problem of time-varying additive noise [18] and time-varying non-additive noise [207] has been addressed by employing conditional probability rules. In [35] feedback is considered in the control formulation using parameterized control policies to significantly lessen the conservativeness of the approach. PCE has also been used to great success for state estimation. [83] use linear update rules considering higher-order moments. By sampling the PCE it can be updated directly using Bayes' theorem as shown in [168]. Similarly, [16] applies Bayes' rule, however the approach accounts for time-varying additive disturbances and in addition uses the PCE for uncertainty propagation. In [44] a PCE SNMPC algorithm was extended to the case of output feedback by using the state estimator proposed in [168], while in [45] a similar approach is proposed considering in addition additive disturbance noise.

For batch processes we are often interested in constraints involving the end-product quality leading to a shrinking horizon NMPC (sh-NMPC) formulation, where the prediction horizon is equal to the final batch time. In [252] a sh-NMPC algorithm using min-max successive linearization is proposed. [184] use an extended Kalman filter based sh-NMPC approach to account for parametric uncertainties and state estimation errors. The real-time implementation of a sh-NMPC for industrial applications is studied in [188]. [176] compares various optimization algorithms for sh-NMPC applied to a crystallization process. [190] employs a sh-NMPC algorithm to control the crystal growth and size distribution of ibuprofen. The multi-stage SNMPC algorithm has been extensively applied to batch processes [164], while the PCE based SNMPC algorithms have been applied to sh-NMPC crystallization problems [177]. In [221] the evolution of a thin-film deposition process based on partial differential equations is controlled using a sh-NMPC implementation, for which the uncertainties are considered using PSEs.

In this paper we extend the previous work using a PCE based sh-SNMPC for batch processes [44, 45, 177] in the output feedback case. PCEs are utilised, since for moderate dimensional problems these have been shown to give accurate estimates of the statistics required for relatively low number of samples compared to other sampling approaches [157]. It was shown in [35] that feedback needs to be included in the nonlinear PCE MPC formulation to not be overly conservative,

which however has been otherwise ignored in such formulations. We propose to use time-invariant linear feedback gains to accomplish this, which are optimized over in addition to the open-loop control actions. The feedback only affects the predictions of the MPC, but are not themselves implemented outside of the MPC formulation. The parametric and state uncertainties are given by PCEs. It is assumed that at each sampling time only noisy output measurements are available, such that a nonlinear state estimator is used to update the PCE representations of the states and parameters. These representations are then efficiently exploited in the PCE based sh-SNMPC formulation to follow both path and end-point chance constraints to optimize an economic objective. The presence of time-varying additive disturbance noise is taken into account using the law of total expectation. The algorithm is verified on a challenging case study of a semi-batch reaction involving the production of the polymer polypropylene glycol. The aim is to directly minimize the required batch time subject to both safety and end-product constraints. The paper is comprised of the following sections. In Section 2 background information is given. In Section 3 a general problem definition is stated. In Section 4 the PCE state estimator is outlined, while in Section 5 we introduce the PCE sh-SNMPC formulation. Section 6 defines the algorithm using both the PCE state estimator and the PCE sh-SNMPC. Section 7 defines the case study to be solved, for which the results are shown and discussed in Section 8. Lastly, conclusions are given in Section 9.

4.2 Background

4.2.1 Introduction to polynomial chaos expansions

In this section we briefly outline PCEs specific for our purposes. For a more general review of PCEs, please refer to [270, 85, 199]. A PCE is a method to represent an arbitrary random variable γ with finite second order moments as a function of random variables $\boldsymbol{\xi}$ with a known distribution. The random variable γ is expanded onto an orthogonal polynomial basis, which can be expressed as follows:

$$\gamma(\boldsymbol{\xi}) = \sum_{\boldsymbol{\alpha} \in \mathbb{N}^{n_{\boldsymbol{\xi}}}} a_{\boldsymbol{\alpha}} \phi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) \tag{4.1}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{n_{\boldsymbol{\xi}}}$ is called the *germ*, $\phi_{\boldsymbol{\alpha}} : \mathbb{R}^{n_{\boldsymbol{\xi}}} \to \mathbb{R}$ are known multivariate polynomials comprising the basis with corresponding expansion coefficients $a_{\boldsymbol{\alpha}}$ and multidimensional summation indices $\boldsymbol{\alpha} \in \mathbb{N}^{n_{\boldsymbol{\xi}}}$.

The probability distribution of the *germ* $\boldsymbol{\xi}$ is a modelling choice. Without loss of generality we assume $\boldsymbol{\xi}$ to follow a standard normal distribution with zero mean and unit variance, i.e. $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The multivariate polynomials in Equation 4.1

are given as a tensor product of univariate polynomials of the components of $\boldsymbol{\xi}$:

$$\phi_{\boldsymbol{\alpha}} = \prod_{i=1}^{n_\xi} \phi_{\alpha_i}(\xi_i) \tag{4.2}$$

where $\phi_{\alpha_i} : \mathbb{R} \to \mathbb{R}$ are univariate polynomials of ξ_i of order α_i. The multidimensional index $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{n_\xi}]$ is hence used to define the degree of each univariate polynomial and the total order of the multivariate polynomial $\phi_{\boldsymbol{\alpha}}$ is consequently given as $|\boldsymbol{\alpha}| = \sum_{i=1}^{n_\xi} \alpha_i$.

The univariate polynomials ϕ_{α_i} are chosen to satisfy an orthogonality property according to the probability distribution of ξ_i, which in our case for standard normal distributions leads to Hermite polynomials:

$$\phi_{\alpha_i}(\xi_i) = (-1)^{\alpha_i} \exp\left(\frac{1}{2}\xi_i^2\right) \frac{d^{\alpha_i}}{d\xi_i^{\alpha_i}} \exp\left(-\frac{1}{2}\xi_i^2\right) \tag{4.3}$$

The multivariate polynomials built in this way according to Equations 4.2-4.3 have the following useful orthogonality property, which can be defined by the following inner product:

$$\langle \phi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}), \phi_{\boldsymbol{\beta}}(\boldsymbol{\xi}) \rangle = \mathbb{E}[\phi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\phi_{\boldsymbol{\beta}}(\boldsymbol{\xi})] = \int \phi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\phi_{\boldsymbol{\beta}}(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi} = \tau_{\boldsymbol{\alpha}}^2 \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}} \tag{4.4}$$

where $p(\boldsymbol{\xi})$ is the pdf of $\boldsymbol{\xi}$, $\delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}$ is the Kronecker delta, i.e. $\delta_{\boldsymbol{\alpha}\boldsymbol{\beta}} = 1$ iff $\boldsymbol{\alpha} = \boldsymbol{\beta}$ otherwise $\delta_{\boldsymbol{\alpha}\boldsymbol{\beta}} = 0$. The normalization constant $\tau_{\boldsymbol{\alpha}}^2$ is dependent on the chosen family of polynomials and often known in practice.

Generally to use Equation 4.1, it needs to be truncated. Keeping all terms up to a total order of m:

$$\gamma(\boldsymbol{\xi}) \approx \sum_{0 \le |\boldsymbol{\alpha}| \le m} a_{\boldsymbol{\alpha}} \phi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) = \mathbf{a}^T \boldsymbol{\phi}(\boldsymbol{\xi}) \tag{4.5}$$

where $\mathbf{a} \in \mathbb{R}^L$ and $\boldsymbol{\phi}(\boldsymbol{\xi}) : \mathbb{R}^{n_\xi} \to \mathbb{R}^L$ are vectors of the coefficients and polynomials of the truncated expansion respectively. The truncated series consists of $L = \frac{(n_\xi + m)!}{n_\xi! m!}$ terms.

Often the coefficients $\mathbf{a}$ of the truncated PCE expansion in Equation 4.5 are unknown and need to be determined using samples of γ. In this work we use the non-intrusive spectral projection approach based on the orthogonality property in Equation 4.4 and the definition of the truncated series in Equation 4.5:

$$a_j = \frac{\langle \gamma(\boldsymbol{\xi}), \phi_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi}) \rangle}{\tau_{\boldsymbol{\alpha}_j}^2} = \frac{1}{\tau_{\boldsymbol{\alpha}_j}^2} \int \gamma(\boldsymbol{\xi}) \phi_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{4.6}$$

where a_j refers to the j^{th} coefficient of $\mathbf{a}$ with a corresponding $\boldsymbol{\alpha}_j$ multidimensional summation index.

The integral in Equation 4.6 can be approximated using sampling. Quadrature methods are commonly used due to their improved convergence rates compared to crude MC. In the case of standard normal distributed germs Gauss-Hermite quadrature rules are employed, which approximate the integral in Equation 4.6 as:

$$\int \gamma(\boldsymbol{\xi})\phi_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi})p(\boldsymbol{\xi})d\boldsymbol{\xi} \approx \sum_{q=1}^{N_q} w_q \gamma(\boldsymbol{\xi}_q)\phi_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi}_q) \tag{4.7}$$

where N_q is the total number of quadrature points and $\boldsymbol{\xi}_q$ are the sample points with corresponding weights w_q given by the Gauss-Hermite quadrature rule.

This procedure leads to the following sample estimate $\hat{\mathbf{a}}$ of the coefficient vector $\mathbf{a}$:

$$\hat{\mathbf{a}} = (\mathbf{w}(\boldsymbol{\Gamma})^T \boldsymbol{\Phi}(\boldsymbol{\Xi}))^T * \boldsymbol{\tau}^{-2} \tag{4.8}$$

where $*$ denotes element-wise multiplication, $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{N_q}]^T \in \mathbb{R}^{N_q \times n_\xi}$ represents the quadrature sample design, the response vector is given by $\boldsymbol{\Gamma} = [\gamma(\boldsymbol{\xi}_1), \ldots, \gamma(\boldsymbol{\xi}_{N_q}]^T \in \mathbb{R}^{N_q}$, $\mathbf{w}(\boldsymbol{\Gamma}) = [w_1\gamma(\boldsymbol{\xi}_1), \ldots, w_{N_q}\gamma(\boldsymbol{\xi}_{N_q})]^T \in \mathbb{R}^{N_q}$, $\boldsymbol{\tau}^{-2} = [\tau_{\boldsymbol{\alpha}_1}^{-2}, \ldots, \tau_{\boldsymbol{\alpha}_L}^{-2}]^T \in \mathbb{R}^L$ and $\boldsymbol{\Phi}(\boldsymbol{\Xi}) = [\boldsymbol{\phi}(\boldsymbol{\xi}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{\xi}_{N_q})]^T \in \mathbb{R}^{N_q \times L}$.

So far we have limited ourselves to single dimensional random variable representations using PCEs, which can however easily be extended to multivariate random variables. Let a multivariate stochastic variable be given by $\boldsymbol{\gamma}(\boldsymbol{\xi}) = [\gamma_1(\boldsymbol{\xi}), \ldots, \gamma_{n_\gamma}(\boldsymbol{\xi})]^T \in \mathbb{R}^{n_\gamma = n_\xi}$ with coefficients collected in $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_{n_\gamma}] \in \mathbb{R}^{L \times n_\xi}$, where we have assumed that each PCE is parametrized in terms of standard normal variables $\boldsymbol{\xi}$ with the same dimension as $\boldsymbol{\gamma}(\cdot)$. We further let each component of $\boldsymbol{\gamma}(\boldsymbol{\xi})$ be given by a truncated PCE with the same truncation order m and hence the same number of terms L.

Statistical moments are an important characterization of random variables. Assuming the multivariate random variable to be given by PCEs as defined above, the statistical moments are functions of the PCE coefficients $\mathbf{A}$ and can be defined as:

$$M_{\mathbf{r}}(\mathbf{A}) = \int \prod_{i=1}^{n_\xi} \gamma_i^{r_i}(\boldsymbol{\xi})p(\boldsymbol{\xi})d\boldsymbol{\xi} \tag{4.9}$$

where $\mathbf{r} \in \mathbb{R}^{n_\xi}$ is a vector defining the moments with a total order $k = \sum_{i=1}^{n_\xi} r_i$.

Substituting the definition of the PCEs in Equation 4.5 into Equation 4.9 we arrive at [83]:

$$M_{\mathbf{r}}(\mathbf{A}) = \int \prod_{i=1}^{n_{\xi}} (\mathbf{a}_i^T \boldsymbol{\phi}(\boldsymbol{\xi}))^{r_i} p(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{4.10}$$

Using Equation 4.10 and the orthogonality property in Equation 4.4 it can be shown that the mean values and covariances of $\boldsymbol{\gamma}$ are given by:

$$\mu^{\gamma_i} = \mathbb{E}[\gamma_i] = a_{i1} \tag{4.11}$$

$$\Sigma_{ij}^{\boldsymbol{\gamma}} = \mathbb{E}[(\gamma_i - \mu^{\gamma_i})(\gamma_j - \mu^{\gamma_j})] = \sum_{s=2}^{L} \sum_{t=2}^{L} a_{is} a_{jt} \langle \phi_{\boldsymbol{\alpha}_s}, \phi_{\boldsymbol{\alpha}_t} \rangle \tag{4.12}$$

where μ^{γ_i} is the mean of γ_i and $\Sigma^{\boldsymbol{\gamma}}$ is the covariance matrix of $\boldsymbol{\gamma}$. Note that $\langle \phi_{\boldsymbol{\alpha}_s}, \phi_{\boldsymbol{\alpha}_t} \rangle$ does not depend on the coefficients $\mathbf{a}$ and can hence be pre-computed. The diagonal of Equation 4.12 define the variances of $\boldsymbol{\gamma}$.

4.2.2 Uncertainty propagation using PCEs

In this section we illustrate how PCEs can be used to efficiently propagate uncertainties through nonlinear functions using non-intrusive spectral projection as introduced in the previous section. Let an arbitrary nonlinear function $q(\cdot)$ of a random variable $\boldsymbol{\gamma}(\boldsymbol{\xi})$ be given by:

$$p = q(\boldsymbol{\gamma}(\boldsymbol{\xi})) \tag{4.13}$$

where $\boldsymbol{\xi}$ is the *germ* random variable parametrizing a random variable $\boldsymbol{\gamma}(\boldsymbol{\xi})$ as shown in the previous section.

The variable p is now a random variable as well parametrized by the *germ* $\boldsymbol{\xi}$ and the aim in this section is to determine its corresponding truncated PCE expansion. It should be noted that the PCE truncation order of p and $\boldsymbol{\gamma}(\boldsymbol{\xi})$ can be dissimilar. To accomplish this we generate samples of $\boldsymbol{\xi}$ with corresponding weights using the Gauss-Hermite rule and evaluate p at those points. The PCE coefficients of p can then be determined using Equation 4.8 as follows:

$$\hat{\mathbf{a}}_q = \mathbf{w}(\boldsymbol{\Gamma}_q)^T \boldsymbol{\Phi}(\boldsymbol{\Xi}) * \boldsymbol{\tau}^{-2} \tag{4.14}$$

where $\mathbf{w}(\boldsymbol{\Gamma}) = [w_1 q(\boldsymbol{\gamma}(\boldsymbol{\xi}_1)), \ldots, w_{N_q} q(\boldsymbol{\gamma}(\boldsymbol{\xi}_{N_q}))]^T$ with $\boldsymbol{\Gamma}_q = [q(\boldsymbol{\gamma}(\boldsymbol{\xi}_1)), \ldots, q(\boldsymbol{\gamma}(\boldsymbol{\xi}_{N_q})]^T$. The remaining terms are defined in Equation 4.8.

Once the approximate coefficients $\hat{\mathbf{a}}_q$ have been determined, we can use Equations 4.11-4.12 to obtain mean and variance estimates for p as follows:

$$\mu^p = \mathbb{E}[p] \approx \hat{a}_{q1} \tag{4.15}$$

$$\sigma^p = \mathbb{E}[(p - \mathbb{E}[p])^2] \approx \sum_{s=2}^{L} \hat{a}_{qs}^2 \langle \phi_{\boldsymbol{\alpha}_s}^2 \rangle \tag{4.16}$$

From the above procedure only the response vector $\boldsymbol{\Gamma}_q$ and $\mathbf{w}(\boldsymbol{\Gamma}_q) = [w_1 q(\boldsymbol{\gamma}(\boldsymbol{\xi}_1)), \ldots, w_{N_q} q(\boldsymbol{\gamma}(\boldsymbol{\xi}_{N_q}))]^T$ depend on the values of p and hence the remaining terms can be pre-computed. We can therefore view this as a function for which a response vector $\boldsymbol{\Gamma}_q$ returns estimates of mean and variance of a nonlinear transformation, which we will denote as:

$$\mu^p \approx \mu_\zeta^{PCE}(\boldsymbol{\Gamma}_q) \tag{4.17}$$

$$\sigma^p \approx \sigma_\zeta^{PCE}(\boldsymbol{\Gamma}_q) \tag{4.18}$$

where $\zeta = \{m, \mathbf{w}, n_{\boldsymbol{\xi}}, \boldsymbol{\Xi}\}$ is a collection of variables defining the mean and variance function, m is the total order of truncation of the PCE approximation of p, $\mathbf{w}$ is a vector of Gauss-Hermite weights, $n_{\boldsymbol{\xi}}$ the dimensionality of $\boldsymbol{\xi}$, and $\boldsymbol{\Xi}$ is the sample design of $\boldsymbol{\xi}$ from the Gauss-Hermite rule.

Note that this approach to determine the statistics of p can be employed as long as clearly defined input-output data pairs are available utilising Equation 4.16 to evaluate the required coefficients. A closed-form expression as shown in Equation 4.13 is not necessarily required.

4.2.3 Laws of total expectation and total variance

PCEs are an efficient way to represent time-invariant probabilities, but become quickly prohibitive in complexity for time-varying uncertainties. This is because each instance of a time-varying uncertainty would require its own dimension in the germ distribution $\boldsymbol{\xi}$, which scales exponentially with the number of terms L required in the PCE expansion, see Equation 4.5. We therefore use the laws of total expectation and covariance to deal with the uncertainties in turn, i.e. using PCEs for the time-invariant uncertainties and utilising linearization for the time-varying uncertainties. In this section we introduce these laws.

Let $\boldsymbol{\gamma}$ and $\boldsymbol{\omega}$ be arbitrary random variables and q an arbitrary function of these random variables, then according to the law of total expecation we have:

$$\mathbb{E}[q(\boldsymbol{\gamma}, \boldsymbol{\omega})] = \mathbb{E}_{\boldsymbol{\omega}}[\mathbb{E}_{\boldsymbol{\gamma}}[q(\boldsymbol{\gamma}, \boldsymbol{\omega})|\boldsymbol{\omega}]] \tag{4.19}$$

The law of total variance can be stated as:

$$\mathrm{Var}[q(\boldsymbol{\gamma}, \boldsymbol{\omega})] = \mathbb{E}_{\boldsymbol{\omega}}[\mathrm{Var}_{\boldsymbol{\gamma}}[q(\boldsymbol{\gamma}, \boldsymbol{\omega})|\boldsymbol{\omega}]] + \mathrm{Var}_{\boldsymbol{\omega}}[\mathbb{E}_{\boldsymbol{\gamma}}[q(\boldsymbol{\gamma}, \boldsymbol{\omega})|\boldsymbol{\omega}]] \tag{4.20}$$

Now we aim to use linearization to account for $\boldsymbol{\gamma}$ and PCE for $\boldsymbol{\omega}$ to approximate the expectation and variance of $q(\boldsymbol{\gamma}, \boldsymbol{\omega})$. Using linearization first we arrive at the following simplified expressions:

$$\mathbb{E}[q(\boldsymbol{\gamma}, \boldsymbol{\omega})] \approx \mathbb{E}_{\boldsymbol{\omega}}[q(\boldsymbol{\mu}_{\boldsymbol{\gamma}}, \boldsymbol{\omega})] \tag{4.21}$$

$$\mathrm{Var}[q(\boldsymbol{\gamma}, \boldsymbol{\omega})] \approx \mathbb{E}_{\boldsymbol{\omega}}[\mathbf{Q}(\boldsymbol{\omega})\boldsymbol{\Sigma}_{\boldsymbol{\gamma}}\mathbf{Q}(\boldsymbol{\omega})^T] + \mathrm{Var}_{\boldsymbol{\omega}}[\mathbb{E}_{\boldsymbol{\gamma}}[q(\boldsymbol{\mu}_{\boldsymbol{\gamma}}, \boldsymbol{\omega})|\boldsymbol{\omega}]] \tag{4.22}$$

where $\boldsymbol{\mu}_{\boldsymbol{\gamma}}$ is the mean of $\boldsymbol{\gamma}$, $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}}$ denotes the covariance of $\boldsymbol{\gamma}$ and $\mathbf{Q}(\boldsymbol{\omega}) = \frac{\partial q}{\partial \boldsymbol{\gamma}}\Big|_{\boldsymbol{\mu}_{\boldsymbol{\gamma}}, \boldsymbol{\omega}}$ is the Jacobian of $q(\cdot)$ with respect to $\boldsymbol{\gamma}$ evaluated at $\boldsymbol{\mu}_{\boldsymbol{\gamma}}$ and $\boldsymbol{\omega}$.

Next using the PCE Equations 4.17-4.18 for $\boldsymbol{\omega}$ we arrive at:

$$\mathbb{E}[q(\boldsymbol{\gamma}, \boldsymbol{\omega})] \approx \mu_{\zeta}^{PCE}(\boldsymbol{\Gamma}_{\mu}^{q}) \tag{4.23}$$

$$\mathrm{Var}[q(\boldsymbol{\gamma}, \boldsymbol{\omega})] \approx \mu_{\zeta}^{PCE}(\boldsymbol{\Gamma}_{\sigma}^{q}) + \sigma_{\zeta}^{PCE}(\boldsymbol{\Gamma}_{\mu}^{q}) \tag{4.24}$$

where $\boldsymbol{\Gamma}_{\mu}^{q} = [q(\boldsymbol{\mu}_{\boldsymbol{\gamma}}, \boldsymbol{\omega}_1), \ldots, q(\boldsymbol{\mu}_{\boldsymbol{\gamma}}, \boldsymbol{\omega}_{N_q})]$ and
$\boldsymbol{\Gamma}_{\sigma}^{q} = [\mathbf{Q}(\boldsymbol{\omega}_1)\boldsymbol{\Sigma}_{\boldsymbol{\gamma}}\mathbf{Q}(\boldsymbol{\omega}_1)^T, \ldots, \mathbf{Q}(\boldsymbol{\omega}_{N_q})\boldsymbol{\Sigma}_{\boldsymbol{\gamma}}\mathbf{Q}(\boldsymbol{\omega}_{N_q})^T]$.

Here we have shown how two separate random variables $\boldsymbol{\omega}$ and $\boldsymbol{\gamma}$ can be dealt with separately. In particular, regarding $\boldsymbol{\gamma}$ as time-varying it was accounted for using linearization, while assuming $\boldsymbol{\omega}$ as time-invariant was considered using PCEs.

4.2.4 Chance constraint reformulation using Chebyshev's inequality

Let γ be a random variable, for which we have a chance constraint as follows:

$$\mathbb{P}(\gamma \leq 0) \geq 1 - \epsilon \tag{4.25}$$

Often the exact evaluation of Equation 4.25 is difficult due to the integral definition of the probability function. Instead, we are however able to estimate the mean and variance of γ. Using Chebychev's inequality the probability constraints in Equation 4.25 can be robustly transformed to the following equation [177]:

$$\mu_{\gamma} + \kappa_{\epsilon}\sqrt{\sigma_{\gamma}} \leq 0 \tag{4.26}$$

where μ_{γ} and σ_{γ} are the mean and variance of γ respectively and $\kappa_{\epsilon} = \sqrt{\frac{1-\epsilon}{\epsilon}}$. Note the robust reformulation now only depends on the mean and variance of γ as required.

4.3 Problem definition

The dynamic system in this paper is assumed to be given by a discrete-time nonlinear equation system with stochastic parameters and additive disturbance

noise::

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\theta}_t) + \mathbf{w}_t^{\mathbf{x}}, \quad \mathbf{x}_0 = \mathbf{x}_0(\boldsymbol{\xi}) \tag{4.27}$$

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t, \boldsymbol{\theta}_t) + \boldsymbol{\nu}_t \tag{4.28}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \mathbf{w}_t^{\boldsymbol{\theta}}, \quad \boldsymbol{\theta}_0 = \boldsymbol{\theta}_0(\boldsymbol{\xi}) \tag{4.29}$$

where t is the discrete time, $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$ are the system states, $\boldsymbol{\theta} \in \mathbb{R}^{n_{\boldsymbol{\theta}}}$ are parametric uncertainties, $\mathbf{u} \in \mathbb{R}^{n_\mathbf{u}}$ denote the control inputs, $\mathbf{f} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_{\boldsymbol{\theta}}} \to \mathbb{R}^{n_\mathbf{x}}$ represents the nonlinear dynamic system for the states, $\mathbf{y} \in \mathbb{R}^{n_\mathbf{y}}$ denote the measurements and $\mathbf{h} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_{\boldsymbol{\theta}}} \to \mathbb{R}^{n_\mathbf{y}}$ are the output equations. Both the states $\mathbf{x}$ and the measurements $\mathbf{y}$ are assumed to be affected by normally distributed zero mean additive noise denoted by $\mathbf{w}^\mathbf{x}$ and $\boldsymbol{\nu}$ with known covariance matrices $\boldsymbol{\Sigma}_\boldsymbol{\nu}$ and $\boldsymbol{\Sigma}_\mathbf{w}^\mathbf{x}$ respectively. In addition, the parametric uncertainties $\boldsymbol{\theta}$ are also assumed to be affected by zero mean normally distributed additive disturbance noise to account for possible time variation denoted by $\mathbf{w}^{\boldsymbol{\theta}}$ with corresponding covariance matrix $\boldsymbol{\Sigma}_\mathbf{w}^{\boldsymbol{\theta}}$. The initial condition $\mathbf{x}_0$ and the initial parametric uncertainties $\boldsymbol{\theta}_0$ are assumed to follow PCEs represented by $\mathbf{x}_0(\boldsymbol{\xi})$ and $\boldsymbol{\theta}_0(\boldsymbol{\xi})$ respectively parametrized by $\boldsymbol{\xi} \in \mathbb{R}^{n_\mathbf{x}+n_{\boldsymbol{\theta}}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For more information refer to Section 4.2.1. These PCEs express the initial uncertainty of $\boldsymbol{\theta}$ and $\mathbf{x}$, which will usually represent a relatively broad distribution with large variances. Note by directly adding a disturbance term $\mathbf{d}_t \in \mathbb{R}^{n_\mathbf{x}}$ given by a PCE to Equation 4.27, we could account for plant-model mismatch and update this mismatch using the filter introduced in Section 4.4.

In the following sections we will work with joint vectors of states and parametric uncertainties for simplification, which we will denote by $\mathbf{x}' = [\mathbf{x}, \boldsymbol{\theta}]^T \in \mathbb{R}^{n_{\mathbf{x}'}=n_\mathbf{x}+n_{\boldsymbol{\theta}}}$. The nonlinear equation system for $\mathbf{x}'$ can then be expressed as:

$$\mathbf{x}'_{t+1} = \mathbf{f}'(\mathbf{x}'_t, \mathbf{u}_t) + \mathbf{w}_t, \quad \mathbf{x}'_0 = \mathbf{x}'_0(\boldsymbol{\xi}) \tag{4.30}$$

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}'_t) + \boldsymbol{\nu}_t \tag{4.31}$$

where $\mathbf{f}'(\mathbf{x}'_t, \mathbf{u}_t) = [\mathbf{f}(\mathbf{x}'_t, \mathbf{u}_t), \boldsymbol{\theta}_t]^T$, $\mathbf{w} = [\mathbf{w}^\mathbf{x}, \mathbf{w}^{\boldsymbol{\theta}}]^T$ with corresponding covariance matrix $\boldsymbol{\Sigma}_\mathbf{w} = \mathrm{diag}(\boldsymbol{\Sigma}_\mathbf{w}^\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{w}^{\boldsymbol{\theta}})$ and $\mathbf{x}'_0(\boldsymbol{\xi}) = [\mathbf{x}_0(\boldsymbol{\xi}), \boldsymbol{\theta}_0(\boldsymbol{\xi})]^T$.

The aim in this paper is the development of an algorithm for the dynamic equation system stated above for batch processes. Generally the objective to be minimized depends on the properties of the final product at the end of the batch, such that the control problem commonly has a finite-horizon leading to a sh-NMPC formulation [184]. We assume the objective to have the following form:

$$J(N, \mathbf{x}'_0(\boldsymbol{\xi}), \mathbf{U}_N) = \mathbb{E}[J^d(N, \mathbf{x}'_0(\boldsymbol{\xi}), \mathbf{U}_N)] \tag{4.32}$$

$$J^d(N, \mathbf{x}'_0(\boldsymbol{\xi}), \mathbf{U}_N) = \mathcal{M}(\mathbf{x}'_N) + \sum_{t=0}^{N-1} \mathcal{L}(\mathbf{x}'_t, \mathbf{u}_t) \tag{4.33}$$

where N is the time horizon, $\mathcal{M} : \mathbb{R}^{n_{\mathbf{x}'}} \to \mathbb{R}$ is the Mayer term, $\mathcal{L} : \mathbb{R}^{n_{\mathbf{x}'} \times n_{\mathbf{u}}} \to \mathbb{R}$ is the Lagrange term, and $\mathbf{U}_N = [\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}] \in \mathbb{R}^{n_{\mathbf{u}} \times N}$ are the control actions that need to be determined.

The objective is taken as the expectation of a nonlinear function with a Mayer and Lagrange term, i.e. the objective is to minimize the expected value of $J^d(N, \mathbf{x}'_0(\boldsymbol{\xi}, \mathbf{U}_N)$ given the initial PCE $\mathbf{x}'_0(\boldsymbol{\xi})$ and the dynamic system stated in Equation 4.30 and Equation 4.31.

The minimization of the objective is subject to both the adherence of path constraints and terminal constraints. The control inputs are subject to hard constraints expressed by the set $\mathbb{U}$. For batch processes common path constraints are safety limits on the reactor temperature and terminal constraints are commonly a minimum product quality to be reached. The constraints can be stated as follows:

$$\mathbb{P}[g_j(\mathbf{x}'_t, \mathbf{u}_t) \leq 0] \geq 1 - \epsilon \qquad \forall (t, j) \in \{1, \ldots, N\} \times \{1, \ldots, n_g\} \tag{4.34}$$

$$\mathbb{P}[g_j^N(\mathbf{x}'_N) \leq 0] \geq 1 - \epsilon \qquad \forall j \in \{1, \ldots, n_g^N\} \tag{4.35}$$

$$\mathbf{u}_t \in \mathbb{U} \qquad \forall t \in \{0, \ldots, N-1\} \tag{4.36}$$

where $g_j : \mathbb{R}^{n_{\mathbf{x}'} \times n_{\mathbf{u}}} \to \mathbb{R}$ are the path constraint functions, $g_j^N : \mathbb{R}^{n_{\mathbf{x}'}} \to \mathbb{R}$ are the terminal constraint functions and ϵ is the probability of constraint violation.

The constraints are given as so-called chance-constraints due to the presence of the stochastic uncertainties in both the initial condition $\mathbf{x}'_0$ and the disturbance noise. Each constraint in Equations 4.34 and 4.35 should be violated at most by a low probability of ϵ despite the stochastic uncertainties present to maintain feasibility.

4.4 Polynomial chaos expansion state estimation

In this section we introduce a nonlinear state estimator to update a prior probability distribution of the state $\mathbf{x}'$ given by a PCE using the available measurements from Equation 4.28. The nonlinear filter is similar to the one given in [168, 182], however these works do not consider additive disturbance noise. We assume we are at sampling time t and wish to update the PCE representation of the uncertainties given the newly available measurements. Let $D_t = \{\mathbf{y}_1 \ldots, \mathbf{y}_t\}$ be the available measurements up to time t. In particular Bayes' rule is used recursively for the update of $\mathbf{x}'$ using D_t:

$$p(\mathbf{x}'_t | D_t) = \frac{p(\mathbf{x}'_t | D_{t-1}) p(\mathbf{y}_t | \mathbf{x}'_t, D_{t-1})}{p(\mathbf{y}_t | D_{t-1})} \tag{4.37}$$

For convenience we denote $\mathcal{N}(\boldsymbol{\gamma} | \boldsymbol{\mu}_{\boldsymbol{\gamma}}, \boldsymbol{\Sigma}_{\boldsymbol{\gamma}})$ as the normal probability density of a

random variable $\boldsymbol{\gamma}$ with mean $\boldsymbol{\mu}_{\boldsymbol{\gamma}}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{\gamma}}$:

$$\mathcal{N}(\boldsymbol{\gamma}|\boldsymbol{\mu}_{\boldsymbol{\gamma}}, \boldsymbol{\Sigma}_{\boldsymbol{\gamma}}) = \det\left(2\pi\boldsymbol{\Sigma}_{\boldsymbol{\gamma}}\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{\gamma} - \boldsymbol{\mu}_{\boldsymbol{\gamma}})^T \boldsymbol{\Sigma}_{\boldsymbol{\gamma}}^{-1}(\boldsymbol{\gamma} - \boldsymbol{\mu}_{\boldsymbol{\gamma}})\right) \tag{4.38}$$

The terms on the RHS of Equation 4.37 are dependent on the dynamic system introduced in Section 4.3 and are defined below in turn.

$p(\mathbf{x}'_t|D_{t-1})$

Prior distribution of $\mathbf{x}'_t$ given the previous measurements D_{t-1}, which can be expressed as:

$$p(\mathbf{x}'_t|D_{t-1}) = \int p(\mathbf{x}'_t|\mathbf{x}'_{t-1})p(\mathbf{x}'_{t-1}|D_{t-1})d\mathbf{x}'_{t-1} \tag{4.39}$$

where $p(\mathbf{x}'_t|\mathbf{x}'_{t-1}) = \mathcal{N}(\mathbf{x}'_t|\mathbf{f}'(\mathbf{x}'_{t-1}, \mathbf{u}_{t-1}), \boldsymbol{\Sigma}_{\mathbf{w}})$ is a multivariate normal pdf with mean given by the dynamics defined in Equation 4.27 and the covariance by the disturbance noise evaluated at $\mathbf{x}'_t$. It should be noted that without disturbance noise $p(\mathbf{x}'_t|D_{t-1}) = \int \delta(\mathbf{x}'_t - \mathbf{f}'(\mathbf{x}'_{t-1}, \mathbf{u}_{t-1}))p(\mathbf{x}'_{t-1}|D_{t-1})d\mathbf{x}'_{t-1}$.

$p(\mathbf{y}_t|\mathbf{x}'_t, D_{t-1})$

The pdf of the current measurement $\mathbf{y}_t$ given $\mathbf{x}'_t$, which can be stated as follows:

$$p(\mathbf{y}_t|\mathbf{x}'_t, D_{t-1}) = \mathcal{N}(\mathbf{y}_t|\mathbf{h}(\mathbf{x}'_t), \boldsymbol{\Sigma}_{\mathbf{v}}) \tag{4.40}$$

$p(\mathbf{y}_t|D_{t-1})$

Total probability of observation $\mathbf{y}_t$ given previous measurements can be expressed as:

$$p(\mathbf{y}_t|D_{t-1}) = \int p(\mathbf{y}_t|\mathbf{x}'_t, D_{t-1})p(\mathbf{x}'_t|D_{t-1})d\mathbf{x}'_t \tag{4.41}$$

If we take both sides of Equation 4.37 times $\prod_{j=1}^{n_\xi}(x'_{tj})^{r_j}$ and integrate over both sides with respect to $\mathbf{x}'_t$ we obtain:

$$M_{\mathbf{r}}^+ = \frac{\int \prod_{j=1}^{n_\xi}(x'_{tj})^{r_j} p(\mathbf{y}_t|\mathbf{x}'_t, D_{t-1})p(\mathbf{x}'_t|D_{t-1})d\mathbf{x}'_t}{p(\mathbf{y}_t|D_{t-1})} \tag{4.42}$$

where from Equation 4.37 $M_{\mathbf{r}}^+ = \int \prod_{j=1}^{n_\xi}(x'_{tj})^{r_j} p(\mathbf{x}'_t|D_t)d\mathbf{x}'_t$ and $k = \sum_{j=1}^{n_\xi} r_j$. Now by definition $M_{\mathbf{r}}^+$ refers to the various k^{th} order moments of the updated distribution of $\mathbf{x}'_t$, $p(\mathbf{x}'_t|D_t)$.

In our case the uncertainties of $\mathbf{x}'_t$ are given by PCEs, which we have so far not taken advantage of. Let $\mathbf{x}'_{t-1}(\boldsymbol{\xi})$ refer to the previously estimated PCEs of $\mathbf{x}'_t$ using the measurements up to time $t-1$, i.e. $\mathbf{x}'_{t-1}(\boldsymbol{\xi})$ refers to $\mathbf{x}'_{t-1}|D_{t-1}$. The RHS of Equation 4.42 is approximated using sampling. The probability distribution $\mathbf{x}'_{t-1}(\boldsymbol{\xi})$ is readily sampled by generating samples of $\boldsymbol{\xi}$, which is known to follow a standard normal distribution. In addition, we also require samples of the disturbance $\mathbf{w}$ to estimate the integral in Equation 4.39. In this work we used Latin hypercube sampling with the inverse normal cumulative transformation, which has an improved convergence over crude MC, see [240] for more information. The sample approximation of the total probability in Equation 4.41 can be stated as:

$$\alpha = \frac{1}{N_s^{SE}} \sum_{s=1}^{N_s^{SE}} \mathcal{N}(\mathbf{y}_t|\mathbf{h}(\mathbf{x}_t'^{(s)}), \boldsymbol{\Sigma}_\mathbf{v}) \tag{4.43}$$

where α is the sample estimate of $p(\mathbf{y}_t|D_{t-1})$, $\mathbf{x}_t'^{(s)} = \mathbf{f}(\mathbf{x}_{t-1}(\boldsymbol{\xi}_s), \mathbf{u}_{t-1}) + \mathbf{w}_s$, N_s^{SE} is the sample size, $\boldsymbol{\xi}_s \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\mathbf{w}_s \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\mathbf{w})$ are the sample points.

Using the sample estimate in Eq.(4.43) and applying a further sample estimate to Equation 4.42 we obtain:

$$M_\mathbf{r}^{(s)+} = \frac{\sum_{s=1}^{N_s^{SE}} \prod_{j=1}^{n_\xi} (x_{tj}'^{(s)})^{r_j} \mathcal{N}(\mathbf{y}_t|\mathbf{h}(\mathbf{x}_t'^{(s)}), \boldsymbol{\Sigma}_\mathbf{v})}{\alpha N_s^{SE}} \tag{4.44}$$

where $M_\mathbf{r}^{(s)+}$ is an approximation of the RHS of Equation 4.42.

To update $\mathbf{x}'_{t-1}(\boldsymbol{\xi})$ to $\mathbf{x}'_t(\boldsymbol{\xi})$ we match the moments defined in Equation 4.44 with those of the PCE $\mathbf{x}'_t(\boldsymbol{\xi})$, which are a function of its coefficients as shown in Equation 4.10. The PCE is then fitted by solving a nonlinear least-squares optimization problem:

$$\hat{\mathbf{A}}_t = \arg\min_{\mathbf{A}_t} \sum_{k \le m^{SE}} ||M_\mathbf{r}^+(\mathbf{A}_t) - M_\mathbf{r}^{(s)+}||_2^2 \tag{4.45}$$

where $k = \sum_{j=1}^{n_\xi} r_j$ was defined above as the order of the moments and hence m^{SE} defines the total order of moments we aim to match. $M_\mathbf{r}^+(\mathbf{A}_t)$ is parametrized by $\mathbf{A}_t$ as shown in Equation 4.10. The estimated coefficients $\hat{\mathbf{A}}_t$ then define the updated PCE $\mathbf{x}'_t(\boldsymbol{\xi})$ as required, which approximately represents the probability distribution of $\mathbf{x}'_t|D_t$. The overall procedure to update an initial PCE expansion $\mathbf{x}'_{t-1}(\boldsymbol{\xi})$ to $\mathbf{x}'_t(\boldsymbol{\xi})$

is summarised in Algorithm 4.1 below.

Algorithm 4.1: PCE state estimation

Input : $\mathbf{y}_t$, $\mathbf{f}'(\mathbf{x}', \mathbf{u})$, $\mathbf{h}(\mathbf{x}')$, $\mathbf{\Sigma}_\mathbf{v}$, $\mathbf{\Sigma}_\mathbf{w}$, $\mathbf{x}'_{t-1}(\boldsymbol{\xi})$, m^{SE}, N_s^{SE}

1. Generate N_s^{SE} Gaussian distributed Latin hypercube samples of $\boldsymbol{\xi}$ and $\mathbf{w}$.
2. Using these samples approximate α in Equation 4.43.
3. Using the samples and α approximate the moments with an order of m^{SE} or less in Equation 4.44.
4. Solve the optimization problem in Equation 4.45 to obtain the updated coefficients that yield $\mathbf{x}'_t(\boldsymbol{\xi})$.

Output : $\mathbf{x}'_t(\boldsymbol{\xi})$

4.5 Polynomial chaos expansion model predictive control

In this section we introduce the PCE based sh-SNMPC formulation used to solve the problem defined in Section 4.3 using the dynamic equation system for the joint state vector in Equations 4.30-4.31. We assume we are at sampling time t and we are given a current PCE approximation of $\mathbf{x}'_t|D_t$ denoted by $\mathbf{x}'_t(\boldsymbol{\xi})$ from the PCE state estimator, which accounts for our current uncertainty of the initial condition given the available measurements, see Section 4.4. To approximate the chance constraints and the objective it is not only necessary to propagate the uncertainty of the initial condition $\mathbf{x}'_t(\boldsymbol{\xi})$, but also the uncertainty of the additive disturbance noise.

4.5.1 Control policy parameterization

A common issue of MPC under uncertainty is the fact that open-loop uncertainties grow unboundedly leading to control actions that are exceedingly conservative, since the feedback through the state update is disregarded. Eventually the OCP will become infeasible with a large enough prediction horizon [271]. Therefore, to ensure reasonable predictions of the uncertainty feedback needs to be incorporated in the optimal control formulation. Optimization over general causal feedback policies is however often intractable [25], such that the optimization is restricted over a class of parameterized feedback policies [102]. In this paper we employ the following parameterization of the control input:

$$\mathbf{u}_k = \mathbf{v}_k + \mathbf{K}(\mathbf{y}_k - \boldsymbol{\mu}_{\mathbf{y},\mathbf{k}}) \tag{4.46}$$

where $\mathbf{v}_k \in \mathbb{R}^{n_\mathrm{u}}$ are the mean of the applied control inputs, $\mathbf{K} \in \mathbb{R}^{n_\mathrm{u} \times n_\mathrm{y}}$ are linear time-invariant feedback gains and $\boldsymbol{\mu}_{\mathbf{y},\mathbf{k}}$ denotes the mean of $\mathbf{y}_k$.

This is a relatively common form to parameterize the control policy, where $\mathbf{v}_k$ corresponds to the control inputs of the nominal system, whereas $\mathbf{K}$ is times by the difference of the real system outputs to the nominal system outputs for correction. This control parameterization is for example used in [185] to design a linear feedback gain for batch processes. The importance to account for feedback is highlighted in Figure 4.1. A stochastic OCP was solved twice once optimizing over a linear time-invariant feedback gain in addition to the open-loop control actions shown on the left-hand side and once over only the open-loop control actions shown on the right-hand side subject to an adiabatic temperature constraint. As can be seen on the left-hand side the adiabatic temperature trajectories are considerably narrower than on the right-hand side and hence accounting for feedback leads to a considerably less conservative OCP solution.

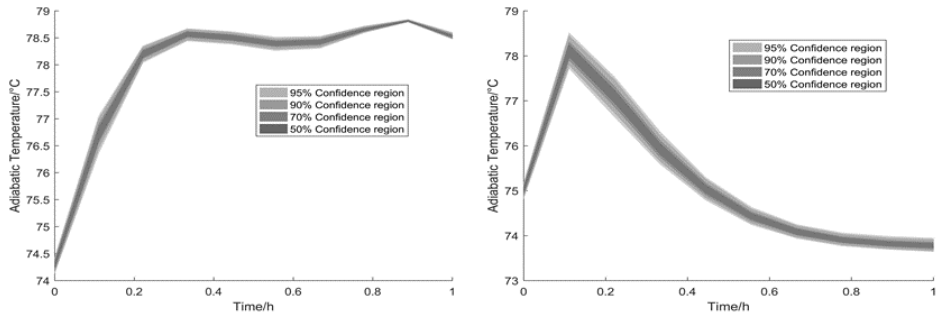


Figure 4.1: OCP adiabatic temperature trajectories accounting for linear time-invariant feedback on the left-hand side and not accounting for feedback on the right-hand side

4.5.2 Uncertainty propagation

For the uncertainty propagation in the sh-SNMPC formulation we use the results in Sections 4.2.2-4.2.3. In this section we outline recursive equations to obtain the required mean and variances of the objective, constraints and outputs from an initial time t to a final shrinking time horizon N^{sh} to formulate the MPC problem. The uncertainty represented by the PCE $\mathbf{x}'_l(\boldsymbol{\xi})$ is accounted for using the PCE Equations 4.17-4.18, while the additive disturbance noise from the states and measurements are considered using linearization. The overall mean and variance of the objective and chance constraints are then determined using the laws of total expectation and variance as shown in Section 4.2.3.

Let $\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi})$ correspond to the mean and $\boldsymbol{\Sigma}_{\mathbf{x}',k}(\boldsymbol{\xi})$ to the covariance of $\mathbf{x}'_k$ given

$\boldsymbol{\xi}$ defined as:

$$\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}) = \mathbb{E}[\mathbf{x}'_k|\boldsymbol{\xi}] \tag{4.47}$$

$$\Sigma_{\mathbf{x}',k_{ij}}(\boldsymbol{\xi}) = \mathbb{E}\left[(\mathbf{x}'_{k\,i} - \mathbb{E}[\mathbf{x}'_{k\,i}])(\mathbf{x}'_{k\,j} - \mathbb{E}[\mathbf{x}'_{k\,j}])\middle|\boldsymbol{\xi}\right] \tag{4.48}$$

We propagate the uncertainty of $\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi})$ and $\boldsymbol{\Sigma}_{\mathbf{x}',k}(\boldsymbol{\xi})$ from the additive disturbances using linearization for the dynamic equation system in Equations 4.30-4.31, which yields:

$$\boldsymbol{\mu}_{\mathbf{x}',k+1}(\boldsymbol{\xi}) = \quad \mathbf{f}'(\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}), \boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi})),\ \boldsymbol{\mu}_{\mathbf{x}',0}(\boldsymbol{\xi}) = \mathbf{x}'_t(\boldsymbol{\xi}) \tag{4.49}$$

$$\boldsymbol{\Sigma}_{\mathbf{x}',k+1}(\boldsymbol{\xi}) = \quad \mathbf{A}_k(\boldsymbol{\xi})\boldsymbol{\Sigma}_{\mathbf{x}',k}(\boldsymbol{\xi})\mathbf{A}_k(\boldsymbol{\xi})^T + \mathbf{B}_k(\boldsymbol{\xi})\boldsymbol{\Sigma}_{\mathbf{u},k}(\boldsymbol{\xi})\mathbf{B}(\boldsymbol{\xi})^T + \tag{4.50}$$
$$2\mathbf{A}_k(\boldsymbol{\xi})\boldsymbol{\Sigma}_{\mathbf{x}'\mathbf{u},k}(\boldsymbol{\xi})\mathbf{B}_k(\boldsymbol{\xi})^T,\ \boldsymbol{\Sigma}_{\mathbf{x}',0}(\boldsymbol{\xi}) = \mathbf{0}$$

where $\mathbf{A}_k(\boldsymbol{\xi}) = \frac{\partial \mathbf{f}'}{\partial \mathbf{x}'}\big|_{\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}),\boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi})}$, and $\mathbf{B}_k(\boldsymbol{\xi}) = \frac{\partial \mathbf{f}'}{\partial \mathbf{u}}\big|_{\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}),\boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi})}$ are Jacobian matrices of $\mathbf{f}'(\mathbf{x}', \mathbf{u})$ with respect to $\mathbf{x}'$ and $\mathbf{u}$ respectively evaluated at $\mathbf{x}' = \boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi})$ and $\mathbf{u} = \boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi}) = \mathbf{v}_k + \mathbf{K}(\mathbf{h}(\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi})) - \boldsymbol{\mu}_{\mathbf{y}_k})$. Let each component of $\boldsymbol{\mu}_{\mathbf{y}_k}$ be given by $\mu_{\mathbf{y}_{k\,i}} = \mu_\zeta^{PCE}(\boldsymbol{\Gamma}_{\mu,k}^{y_{k\,i}})$, which is defined later in Equation 4.62.

For the control policy parameterization in Equation 4.46 the covariance matrices $\boldsymbol{\Sigma}_{\mathbf{u},k}(\boldsymbol{\xi})$ and $\boldsymbol{\Sigma}_{\mathbf{x}'\mathbf{u},k}(\boldsymbol{\xi})$ can be expressed as:

$$\boldsymbol{\Sigma}_{\mathbf{u},k}(\boldsymbol{\xi}) = \mathbf{K}\left(\mathbf{H}_k(\boldsymbol{\xi})\boldsymbol{\Sigma}_{\mathbf{x}',k}(\boldsymbol{\xi})\mathbf{H}_k(\boldsymbol{\xi})^T + \boldsymbol{\Sigma}_\mathbf{v}\right)\mathbf{K}^T \tag{4.51}$$

$$\boldsymbol{\Sigma}_{\mathbf{x}'\mathbf{u},k}(\boldsymbol{\xi}) = \boldsymbol{\Sigma}_{\mathbf{x}',k}(\boldsymbol{\xi})\mathbf{H}_k(\boldsymbol{\xi})^T\mathbf{K}^T \tag{4.52}$$

where $\mathbf{H}_k(\boldsymbol{\xi}) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\big|_{\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi})}$ is the Jacobian matrix of $\mathbf{h}(\mathbf{x}')$ evaluated at $\mathbf{x}' = \boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi})$.

From Equation 4.49 we obtain $\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi})$ recursively for all $k \in \{1, \ldots, N^{sh}\}$, while Equation 4.50 gives us $\boldsymbol{\Sigma}_{\mathbf{x}',k}(\boldsymbol{\xi})$ recursively for all $k \in \{1, \ldots, N^{sh}\}$. The objective, output measurements, and constraint functions are however assumed to be nonlinear functions of these, see Section 4.3. We therefore require further simplification to estimate the required statistics to formulate the SNMPC problem. Using further linearization these quantities can be determined as follows:

$$\mathbb{E}[J^d(N^{sh}, \mathbf{x}'_t(\boldsymbol{\xi}), \mathbf{U}_{N^{sh}})|\boldsymbol{\xi}] \approx J^d(N^{sh}, \mathbf{x}'_t(\boldsymbol{\xi}), \mathbf{U}_{N^{sh}}) \tag{4.53}$$

$$\mathbb{E}[g_j(\mathbf{x}'_k, \mathbf{u}_k)|\boldsymbol{\xi}] \approx g_j(\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}), \boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi})) \tag{4.54}$$

$$\mathbb{E}[g_j^N(\mathbf{x}'_{N^{sh}})|\boldsymbol{\xi}] \approx g_j^N(\boldsymbol{\mu}_{\mathbf{x}',N^{sh}}(\boldsymbol{\xi})) \tag{4.55}$$

$$\mathbb{E}[\mathbf{y}_k|\boldsymbol{\xi}] \approx \mathbf{h}(\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi})) \tag{4.56}$$

$$\mathrm{Var}[g_j(\mathbf{x}'_k, \mathbf{u}_k)|\boldsymbol{\xi}] \approx \mathbf{G}_{j_k}(\boldsymbol{\xi})\boldsymbol{\Sigma}_{\mathbf{x}',k}(\boldsymbol{\xi})\mathbf{G}_{j_k}(\boldsymbol{\xi})^T \tag{4.57}$$

$$\mathrm{Var}[g_j^{N^{sh}}(\mathbf{x}'_{N^{sh}})|\boldsymbol{\xi}] \approx \mathbf{G}_j^N(\boldsymbol{\xi})\boldsymbol{\Sigma}_{\mathbf{x}',N^{sh}}(\boldsymbol{\xi})\mathbf{G}_j^N(\boldsymbol{\xi})^T \tag{4.58}$$

where $\mathbf{G}_{j_k}(\boldsymbol{\xi}) = \frac{\partial \mathbf{g}_j}{\partial \mathbf{x}'}\Big|_{\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}),\boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi})}$ is the Jacobian matrix of $\mathbf{g}_j(\mathbf{x}', \mathbf{u})$ evaluated at $\mathbf{x}' = \boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi})$ and $\mathbf{u} = \boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi})$ and $\mathbf{G}_j^N(\boldsymbol{\xi}) = \frac{\partial \mathbf{g}_j^N}{\partial \mathbf{x}'}\Big|_{\boldsymbol{\mu}_{\mathbf{x}',N^{sh}}(\boldsymbol{\xi})}$ is the Jacobian matrix of $\mathbf{g}_j^N$ evaluated at $\mathbf{x}' = \boldsymbol{\mu}_{\mathbf{x}',N^{sh}}(\boldsymbol{\xi})$.

We now have the means and variances of the objective, constraint functions and outputs accounting for the additive disturbance noise, but ignoring the uncertainty from the initial condition $\mathbf{x}_t'(\boldsymbol{\xi})$. As highlighted the means and variances, and their respective terms are all functions of $\mathbf{x}_t'(\boldsymbol{\xi})$ and in turn of $\boldsymbol{\xi}$ on which they are conditioned. To obtain the overall means and variances required we use the laws of total expectation and variance as shown in Section 4.2.3. This is accomplished by creating a Gauss-Hermite sample design of $\boldsymbol{\xi}$, which we will denote as $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{N_q}]$ with N_q quadrature points. Obtaining the respective response vectors $\boldsymbol{\Gamma}$ for the conditional means and variances in Equations 4.53-4.58 for these samples and applying the PCE mean and variance estimates from Section 4.2.2, the overall variances and expectations are given by:

$$\mathbb{E}[J^d(N^{sh}, \mathbf{x}_t'(\boldsymbol{\xi}), \mathbf{U}_{N^{sh}})] \approx \mu^{PCE}_{\zeta_{\text{NMPC}}}(\boldsymbol{\Gamma}_{\mu}^{J^d}) \tag{4.59}$$

$$\mathbb{E}[g_j(\mathbf{x}_k', \mathbf{u}_k)] \approx \mu^{PCE}_{\zeta_{\text{NMPC}}}(\boldsymbol{\Gamma}_{\mu,k}^{g_j}) \tag{4.60}$$

$$\mathbb{E}[g_j^N(\mathbf{x}_{N^{sh}}')] \approx \mu^{PCE}_{\zeta_{\text{NMPC}}}(\boldsymbol{\Gamma}_{\mu}^{g_j^N}) \tag{4.61}$$

$$\mathbb{E}[y_{k_i}] \approx \mu^{PCE}_{\zeta_{\text{NMPC}}}(\boldsymbol{\Gamma}_{\mu,k}^{y_{k_i}}) \tag{4.62}$$

$$\text{Var}[g_j(\mathbf{x}_k', \mathbf{u}_k)] \approx \mu^{PCE}_{\zeta_{\text{NMPC}}}(\boldsymbol{\Gamma}_{\sigma,k}^{g_j}) + \sigma^{PCE}_{\zeta_{\text{NMPC}}}(\boldsymbol{\Gamma}_{\mu,k}^{g_j}) \tag{4.63}$$

$$\text{Var}[g_j^N(\mathbf{x}_{N^{sh}}')] \approx \mu^{PCE}_{\zeta_{\text{NMPC}}}(\boldsymbol{\Gamma}_{\sigma}^{g_j^N}) + \sigma^{PCE}_{\zeta_{\text{NMPC}}}(\boldsymbol{\Gamma}_{\mu}^{g_j^N}) \tag{4.64}$$

where $\boldsymbol{\Gamma}_{\mu,k}^{y_{k_i}} = [\mathbf{h}(\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}_1)), \ldots, \mathbf{h}(\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}_{N_q}))]$,
$\boldsymbol{\Gamma}_{\mu}^{J^d} = [J^d(N^{sh}, \mathbf{x}_t'(\boldsymbol{\xi}_1), \mathbf{U}_{N^{sh}}), \ldots, J^d(N^{sh}, \mathbf{x}_t'(\boldsymbol{\xi}_{N_q}), \mathbf{U}_{N^{sh}})]^T$,
$\boldsymbol{\Gamma}_{\mu,k}^{g_j} = [g_j(\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}_1), \boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi}_1)), \ldots, g_j(\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}_{N_q}), \boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi}_{N_q}))]$,
$\boldsymbol{\Gamma}_{\mu}^{g_j^N} = [g_j^N(\boldsymbol{\mu}_{\mathbf{x}',N^{sh}}(\boldsymbol{\xi}_1)), \ldots, g_j^N(\boldsymbol{\mu}_{\mathbf{x}',N^{sh}}(\boldsymbol{\xi}_{N_q}))]$,
$\boldsymbol{\Gamma}_{\sigma,k}^{g_j} = [\mathbf{G}_{j_k}(\boldsymbol{\xi})\boldsymbol{\Sigma}_{\mathbf{x}',k}(\boldsymbol{\xi}_1)\mathbf{G}_{j_k}(\boldsymbol{\xi}_1)^T, \ldots, \mathbf{G}_{j_k}(\boldsymbol{\xi})\boldsymbol{\Sigma}_{\mathbf{x}',k}(\boldsymbol{\xi}_{N_q})\mathbf{G}_{j_k}(\boldsymbol{\xi}_{N_q})^T]$ and
$\boldsymbol{\Gamma}_{\sigma}^{g_j^N} = [\mathbf{G}_j^N(\boldsymbol{\xi}_1)\boldsymbol{\Sigma}_{\mathbf{x}',N^{sh}}(\boldsymbol{\xi}_1)\mathbf{G}_j^N(\boldsymbol{\xi})^T, \ldots, \mathbf{G}_j^N(\boldsymbol{\xi})\boldsymbol{\Sigma}_{\mathbf{x}',N^{sh}}(\boldsymbol{\xi}_{N_q})\mathbf{G}_j^N(\boldsymbol{\xi}_{N_q})^T]$.

Note that each sample of $\boldsymbol{\xi}$ corresponds to a separate initial condition and hence a separate nonlinear equation system given by Equation 4.30. In essence the PCE methodology generates separate samples of the initial condition, which each correspond to a distinctive nonlinear dynamic system that are propagated individually together with their linearized variance approximations. From these samples the overall means and variances are then determined from the coefficients of the PCE

expansion as shown in Sections 4.5.2 and 4.2.3. $\zeta_{\text{NMPC}} = \{m_{\text{NMPC}}, \mathbf{w}_{\text{NMPC}}, n_{\xi} = n_{\mathbf{x}'}, \Xi_{\text{NMPC}}\}$ defines in this context the variables of the PCE mean and variance function, see Section 4.5.2, which are the order of truncation of the PCE approximation, the Gauss-Hermite weights, the dimensionality of n_{ξ} given by the number of states and uncertain parameters, and lastly the Gaussian-Hermite sample design. Next we formulate the sh-SNMPC problem based on the above equations.

4.5.3 Chance constraint reformulation

In Section 4.5.2 we show how to obtain estimates of the mean and variance of the objective and constraint functions. While this is sufficient to approximate the objective defined in Section 4.3, we still require estimates for the chance constraints defined in Equations 4.34-4.35. In Section 4.2.4 it was shown how Chebyshev's inequality can be used to robustly reformulate chance constraints exploiting only mean and variance of the constrained variable, which leads to the following robust reformulations of Equations 4.34-4.35 using the estimates of mean and variances given in Equations 4.60-4.64:

$$\mu^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g_j}_{\mu,k}) + \kappa_{\epsilon}\sqrt{\mu^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g_j}_{\sigma,k}) + \sigma^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g_j}_{\mu,k})} \leq 0 \tag{4.65}$$

$$\mu^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g^N_j}_{\mu}) + \kappa_{\epsilon}\sqrt{\mu^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g^N_j}_{\sigma}) + \sigma^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g^N_j}_{\mu})} \leq 0 \tag{4.66}$$

where $\kappa_{\epsilon} = \sqrt{\frac{1-\epsilon}{\epsilon}}$.

4.5.4 Stochastic nonlinear optimal control formulation

In this section we formulate the stochastic optimal control problem to be solved in a shrinking horizon fashion at each sampling time t given a probability distribution of the initial condition represented by its PCE $\mathbf{x}'_t(\boldsymbol{\xi})$ and the dynamic equation system defined in Section 4.3. The formulation is based on the propagation equations outlined in Section 4.5.2 and the reformulations of the chance constraints in Section 4.5.3. We optimize over both open-loop control actions $\mathbf{v}_k$ and a time-invariant feedback control gain $\mathbf{K}$ to account for feedback. The overall formulation

can be stated as follows:

$$
\begin{aligned}
&\underset{\mathbf{V}_{N^{sh}},\mathbf{K}}{\text{minimize}} \quad \mu^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{J^d}_{\mu}) \\
&\text{subject to} \\
&\mu^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g_j}_{\mu,k}) + \kappa_\epsilon \sqrt{\mu^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g_j}_{\sigma,k}) + \sigma^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g_j}_{\mu,k})} \leq 0 \\
&\forall (k,j) \in \{1,\dots,N^{sh}\} \times \{1,\dots,n_g\} \\
&\mu^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g^N_j}_{\mu}) + \kappa_\epsilon \sqrt{\mu^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g^N_j}_{\sigma}) + \sigma^{PCE}_{\zeta_{\text{NMPC}}}(\mathbf{\Gamma}^{g^N_j}_{\mu})} \leq 0 \quad && \forall j \in \{1,\dots,n^N_g\} \\
&\boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi}_i) = \mathbf{v}_k + \mathbf{K}(\mathbf{h}(\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}_i)) - \boldsymbol{\mu}_{\mathbf{y}_k}) \in \mathbb{U} && \forall k \in \{0,\dots,N^{sh}-1\} \\
&\boldsymbol{\mu}^{(i)}_{\mathbf{x}',0} = \mathbf{x}'_t(\boldsymbol{\xi}_i) && \forall i \in \{1,\dots,N_q\}
\end{aligned}
\tag{4.67}
$$

where $\mathbf{V}_{N^{sh}} = [\mathbf{v}_0,\dots,\mathbf{v}_{N^{sh}-1}]^T$ is a matrix of open-loop control actions. It should be noted that the initial control input is given by $\mathbf{v}_0$, which is the output of the MPC algorithm. The initial measurement is known and hence feedback does not apply for the first control action in the problem above, i.e. $\mathbf{h}(\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}_i)) = \boldsymbol{\mu}_{\mathbf{y}_0} \forall i \in \{1,\dots,N_q\}$. The linear feedback gain $\mathbf{K}$ is only used to lead to more realistic future predictions.

4.6 Algorithm

In this section we state the algorithm to solve the problem defined in Section 4.3. Initially at time $t = 0$ we are given a probability distribution represented by a PCE of $\mathbf{x}'$ denoted by $\mathbf{x}'_0(\boldsymbol{\xi})$. In addition, we define the DAE system to be controlled by $\mathbf{f}(\mathbf{x}',\mathbf{u})$ and their measurement equation $\mathbf{h}(\mathbf{x}')$, and the corresponding additive disturbances defined by the covariance matrices $\mathbf{\Sigma}_\mathbf{v}$ and $\mathbf{\Sigma}_\mathbf{w}$. The overall time horizon N needs to be given, which is also equal to the number of control inputs. Further, we define the objective $J^d(N,\mathbf{x}'_0(\boldsymbol{\xi}),\mathbf{u}_N)$ with path $g_j(\mathbf{x}'_t,\mathbf{u}_t)$ and terminal constraints $g^N_j(\mathbf{x}'_N)$ with a corresponding chance of constraint violation ϵ given an overall time horizon of N. We define m_{SE} and N^{SE}_s as the order to be matched in the state estimator, while N^{SE}_s denotes the number of samples to approximate the moments in the state estimator, see Algorithm 4.1 in Section 4.4. Lastly, ζ_{NMPC} is required to define the PCE approximation of the sh-SNMPC in Section 4.5. Overall we suggest to represent the states $\mathbf{x}'$ at each sampling time t using PCEs introduced in Section 4.2.1. The SNMPC algorithm exploits this uncertainty description to control the dynamic system in Section 4.3. The available measurements are utilised to update the PCE representation recursive as outlined in Section 4.4. The overall algorithm is given below as Algorithm 4.2.

Algorithm 4.2: Output feedback PCE sh-SNMPC

Input : $\mathbf{f}'(\mathbf{x}', \mathbf{u})$, $\mathbf{h}(\mathbf{x}')$, $\mathbf{\Sigma}_{\mathbf{v}}$, $\mathbf{\Sigma}_{\mathbf{w}}$, $\mathbf{x}'_0(\boldsymbol{\xi})$, m_{SE}, N_s^{SE}, N, ζ_{NMPC}
Initialize: $N^{sh} := N$
for *each sampling time* $t = 0, 1, 2, \ldots, N-1$ **do**

1. Solve the PCE SNMPC problem in Equation 4.67 with $\mathbf{x}'_t(\boldsymbol{\xi})$ and time horizon N^{sh}.
2. Apply the first optimal control action $\mathbf{v}_0$ to the plant.
3. Measure $\mathbf{y}_{t+1}$.
4. Apply PCE filter to update $\mathbf{x}'_t(\boldsymbol{\xi})$ to $\mathbf{x}'_{t+1}(\boldsymbol{\xi})$ using $\mathbf{y}_{t+1}$.
5. Set $N^{sh} := N^{sh} - 1$.

4.7 Case study

The algorithm outlined in Section 4.6 is applied to a challenging polymerization semi-batch reactor case study. The reactor produces the polymer polypropylene glycol from the monomer propylene oxide (PO). A schematic of the process is shown below in Figure 4.2.

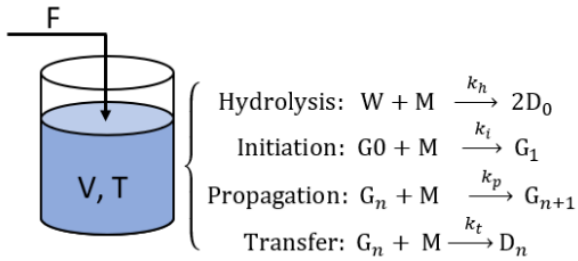


Figure 4.2: F is the monomer feedrate, V and T are the volume and temperature of the liquid in the reactor respectively, W is water, M is the monomer, D_n and G_n are the dormant and active product chains with length n respectively.

4.7.1 Semi-batch reactor model

A complex model for this process has been proposed in [193], which uses a separate balance equation for each specific chain length. This model was employed in [132] for crude NMPC and in [126] for multi-stage robust NMPC. The computational times reported were approximately in the range of 30 seconds to minutes, which is relatively high and can be attributed to the relative complexity of the dynamic model in [193]. In particular the high dimensionality of the state

would cause issues with the proposed methodology due to the scaling of PCEs, see Section 4.2.1. To reduce the dimensionality and complexity of the model we therefore applied the so-called "method of moments" [194]. This leads to balance equations describing the moments of the polymer as opposed to the concentration of each specific chain length, which is commonly sufficient to estimate key performance indicators. Further, we disregard the balance equations for the unsaturated polymer chain length and in addition assume that there are only trace amounts of water or methanol present. This means that hydrolysis only takes place in negligible amounts and hence can be ignored. In the original dynamic model perfect temperature control was assumed. Due to the relative importance of temperature control with regards to safety, we added an energy balance. The modified ordinary differential equation system consists of 4 balance equations and can be stated as follows:

$$\dot{m} = FMW_{\mathrm{PO}} \qquad m(0) = m_0(\boldsymbol{\xi}) \tag{4.68a}$$

$$\dot{T} = \frac{(-\Delta H_p)k_p n_C PO}{VmC_{pb}} - \frac{UA(T - T_C)}{mC_{pb}} - \frac{FMW_{\mathrm{PO}}C_{pf}(T - T_f)}{mC_{pb}} \qquad T(0) = T_0(\boldsymbol{\xi}) \tag{4.68b}$$

$$\dot{PO} = F - \frac{n_C(k_p + k_t)PO}{V} \qquad PO(0) = PO_0(\boldsymbol{\xi}) \tag{4.68c}$$

$$\dot{\gamma}_1 = \frac{k_p n_C PO}{V} \qquad \gamma_1(0) = \gamma_{10}(\boldsymbol{\xi}) \tag{4.68d}$$

where m is the liquid mass in the reactor in [kg], F is the feed rate of the monomer in [kmol/s], T is the temperature of the reactor in [K], PO is the amount of monomer in [kmol] and γ_1 is the first moment and hence the average molecular weight of the polymer chains in [kmol], MW_{PO} is the molecular weight of PO in [kg/kmol], ΔH_p is the enthalpy of the propagation reaction in [kJ/kmol], k_p is the kinetic constant of the propagation equation in [m^3/kmol/s], n_C is the amount of catalyst in [kmol], V is the volume of the liquid in the reactor in m^3, C_{pb} and C_{pf} are the heat capacities of the bulk liquid and the monomer feed respectively in [kJ/kg/K], k_t is the transfer kinetic constant in [m^3/kmol/s], T_C is the cooling water temperature in [K] and UA is the overall heat transfer coefficient in [kJ/K].

The kinetic constants are given by Arrhenius equations as functions of temperature, while the heat capacities C_{pb} and C_{pf} are also functions of temperature [193]:

$$k_p = A_p \exp(-E_{Ap}/RT) \tag{4.69a}$$

$$k_t = A_t \exp(-E_{At}/RT) \tag{4.69b}$$

$$C_{pf} = 0.92 + 8.871 \times 10^{-3}T - 3.1 \times 10^{-5}T^2 + 4.78 \times 10^{-8}T^3 \tag{4.69c}$$

$$C_{pb} = 1.1 + 2.72 \times 10^{-3}T \tag{4.69d}$$

Two parameters in the above model are assumed to be uncertain: A_p and UA. The remaining values of parameters to define Equations 4.68-4.69 are given in Table 4.1. Note that for the assumptions made the amount of the zeroth polymer moment γ_0 in the reactor is fixed, since no hydrolysis takes place.

Table 4.1: Parameter values for dynamic model defined in Equations 4.68 – 4.69 taken from [193]

Parameter	Value	Units	Description
MW_{PO}	58.08	kg/kmol	Molecular weight of PO
ΔH_p	−92048	kJ/kmol	Enthalpy for propagation reaction
A_t	950410	m^3/kmol/s	Pre-exponential factor of transfer reaction
E_{Ap}	69172	kJ/kmol	Activation energy of propagation reaction
E_{At}	105018	kJ/kmol	Activation energy of transfer reaction
n_C	2.0	kmol	Amount of catalyst
R	8.314	kJ/kmol/K	Universal gas constant
γ_0	10	kmol	Zeroth polymer moment (Methanol)

The control inputs are given by the monomer feed rate F and the cooling water temperature T_C. In compact form we can write $\mathbf{x} = [m, T, PO, \gamma_1]^T$, $\mathbf{u} = [F, T_C]^T$ and $\boldsymbol{\theta} = [A_p, UA]^T$. The corresponding joint vector is then given by $\mathbf{x}' = [m, T, PO, \gamma_1, A_p, UA]^T$. The continuous-time dynamic system we denote by $\bar{\mathbf{f}}'(\mathbf{x}', \mathbf{u}) = [\dot{m}, \dot{T}, \dot{PO}, \dot{\gamma}_1, 0, 0]^T$.

We assume that reactor temperature T and the amount of monomer PO are measured at each sampling time with $\mathbf{y} = [T, PO]^T$. The measurement equation $\mathbf{h}(\cdot)$ is given as follows:

$$\mathbf{h}(\mathbf{x}') = [T, PO]^T \tag{4.70}$$

The initial uncertainty of the states $\mathbf{x}$ and the uncertain parameters $\boldsymbol{\theta}$ is given by their respective initial PCEs, which are defined as:

$$m_0(\boldsymbol{\xi}) = 1538 \tag{4.71a}$$

$$PO_0(\boldsymbol{\xi}) = 10 + \xi_2 \tag{4.71b}$$

$$T_0(\boldsymbol{\xi}) = 378.15 + 4\xi_3 \tag{4.71c}$$

$$\gamma_{10}(\boldsymbol{\xi}) = 10 + 0.5\xi_4 \tag{4.71d}$$
$$A_{P0}(\boldsymbol{\xi}) = 10 + \xi_5 \tag{4.71e}$$
$$UA_0(\boldsymbol{\xi}) = 15 + 4\xi_6 \tag{4.71f}$$

The overall initial PCE is now given by $\mathbf{x}'_0(\boldsymbol{\xi}) = [m_0(\boldsymbol{\xi}), PO_0(\boldsymbol{\xi}), T_0(\boldsymbol{\xi}), \gamma_{10}(\boldsymbol{\xi}), A_{P0}(\boldsymbol{\xi}), UA_0(\boldsymbol{\xi})]^T$. The PCEs of $\mathbf{x}'$ given by $\mathbf{x}'(\boldsymbol{\xi})$ have a truncation order of 2.

The additive disturbance and measurement noise is defined by their respective covariance matrices, which were set to:

$$\boldsymbol{\Sigma}_{\mathbf{w}} = \mathrm{diag}(1, 10^{-3}, 1, 2.5 \times 10^{-1}, 5 \times 10^{-2}, 2 \times 10^{-1}) \tag{4.72}$$
$$\boldsymbol{\Sigma}_{\mathbf{v}} = \mathrm{diag}(0.25, 1 \times 10^{-3}) \tag{4.73}$$

4.7.2 Problem set-up

The time horizon N was set to 8 with a variable continuous time of t_{batch}. Given the dynamic system in Equation 4.68 we define the objective and constraints in this section to define a problem as the one given in Section 4.3. The control algorithm aims to minimize the required batch time t_{batch} with a penalty on the control change, which can be stated as follows:

$$J^d(N, \mathbf{x}'_0(\boldsymbol{\xi}), \mathbf{U}_N) = t_{\text{batch}} + \sum_{k=1}^{N} \boldsymbol{\Delta}_{\mathbf{u}_k}^T \mathbf{R} \boldsymbol{\Delta}_{\mathbf{u}_k} \tag{4.74}$$

where $\boldsymbol{\Delta}_{\mathbf{u}_k} = \mathbf{u}_k - \mathbf{u}_{k-1}$ and $\mathbf{R} = \mathrm{diag}(10^{-6}, 10^{-4})$.

The batch time is defined to describe the discrete-time equation for a control horizon N:

$$\mathbf{x}'_{t+1} = \int_0^{t_{\text{batch}}/N} \bar{\mathbf{f}}'(\mathbf{x}'_t, \mathbf{u}_t) dt + \mathbf{x}'_t \tag{4.75}$$

The minimum of the objective above is subject to two terminal constraints and a path constraint. The path constraint aims to keep the reactor temperature to remain below 420K for safety reasons:

$$g(\mathbf{x}'_t, \mathbf{u}_t) = T - 420 \leq 0 \tag{4.76}$$

The two terminal constraints are given by two product quality constraints. Firstly, the batch needs to reach a specified number average molecular weight

(*NAMW*) in [kg/mol] of 350 defined as $NAMW = MW_{\text{PO}}\frac{\gamma_1}{\gamma_0}$. Secondly, the amount of monomer PO in the final batch may not exceed 1000ppm. The two end-point product quality constraint can consequently be stated as:

$$g_1^N(\mathbf{x}'_N) = -MW_{\text{PO}}\frac{\gamma_1}{\gamma_0} + 350 \leq 0 \tag{4.77}$$

$$g_2^N(\mathbf{x}'_N) = 10^6 \times \left(\frac{POMW_{\text{PO}}}{m}\right) - 1000 \leq 0 \tag{4.78}$$

The chance of constraint violation was set to $\epsilon = 0.05$ for the constraints defined above. The monomer feed rate and the cooling water temperature were constrained as follows:

$$0 \leq F \leq 0.01 \tag{4.79}$$

$$298.15 \leq T_C \leq 423.15 \tag{4.80}$$

4.7.3 Solution approach

To solve the case study we need to discretize the continuous-time equation system outlined in Equation 4.68 to obtain the required discrete-time equation used in the problem definition in Section 4.3. For the discretization we applied orthogonal Radau collocation [27]. Each control interval is modelled by one polynomial with an overall degree of 5. Further, we require linearization matrices for the states and control inputs for the discrete time system to propagate the additive disturbance noise as outlined in Section 4.5.2, which are given as follows [151]:

$$\mathbf{A}_k(\boldsymbol{\xi}) = \exp\left(\left.\frac{\partial \bar{\mathbf{f}}'}{\partial \mathbf{x}'}\right|_{\mathbf{x}'=\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}),\mathbf{u}_k=\boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi})}\right) \tag{4.81a}$$

$$\mathbf{B}_k(\boldsymbol{\xi}) = \left(\left.\frac{\partial \bar{\mathbf{f}}'}{\partial \mathbf{x}'}\right|_{\mathbf{x}'=\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}),\mathbf{u}_k=\boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi})}\right)^{-1}(\mathbf{A}_k(\boldsymbol{\xi}) - \mathbf{I})\left(\left.\frac{\partial \bar{\mathbf{f}}'}{\partial \mathbf{u}}\right|_{\mathbf{x}'=\boldsymbol{\mu}_{\mathbf{x}',k}(\boldsymbol{\xi}),\mathbf{u}_k=\boldsymbol{\mu}_{\mathbf{u}_k}(\boldsymbol{\xi})}\right) \tag{4.81b}$$

The resulting optimization problems for both the PCE sh-SNMPC problem and the PCE state estimator were solved using Casadi [9] to obtain the gradients of the problem using automatic differentiation in conjunction with IPOPT [258]. The "real" plant model was simulated using IDAS [116]. The PCE state estimator aims to match up to 3 orders with overall 10000 samples to estimate the posterior moments, i.e. $m_{SE} = 3$ and $N_s^{SE} = 10000$. For the Gauss-Hermite samples we use a sparse rule for the weights $\mathbf{w}$ and sample design $\boldsymbol{\Xi}$ as outlined in [127] with a polynomial accuracy of 3. The PCE of the sh-SNMPC has a truncation order

of 2 with a dimensionality of $\boldsymbol{\xi}$ of 6 corresponding to the dimensionality of $\mathbf{x}'$. The parameters for the PCE sh-SNMPC algorithm are then given by $\zeta_{\text{NMPC}} = [2, \mathbf{w}, 6, \boldsymbol{\Xi}]$.

4.8 Results and discussions

In this section we present and discuss the results of the case study outlined in the previous section. For comparison purposes we compare three different sh-NMPC variations, which are as follows:

- SNMPC with feedback: The algorithm as outlined in Section 4.6 optimizing over both a linear feedback gain and open-loop control actions.
- SNMPC without feedback: The algorithm as outlined in Section 4.6, but optimizing over only open-loop control actions and setting the linear feedback gain to zero.
- Nominal NMPC: A NMPC algorithm based on the same dynamic model and discredization with the state estimate equal to the best-estimate given by the mean of the PCE state estimator. Objective and constraints are deterministic in this case with soft constraints used for feasibility.

The sh-SNMPC variations were each run for the economic MPC problem minimizing the objective given in Equation 4.74 subject to the safety path constraint in Equation 4.76, and end-point product quality constraints defined in Equations 4.77 and 4.78. Note that the time-invariant feedback gain is never actually implemented, since the first control action does not depend on it, see the SNMPC formulation in Equation 4.67. Instead, this inclusion of feedback only serves to obtain more realistic and less conservative predictions. The actual closed-loop response depends on the feedback from the state update.

4.8.1 Example scenario

We first run the three sh-NMPC variations on a single scenario, where we set the initial condition to the following: $\mathbf{x}_0' = [m_0, P0_0, T_0, \gamma_{10}, A_{P0}, UA_0]^T = [1538, 9.0, 385, 9.5, 7.5, 10]^T$. This can be seen as a single realization of the initial PCE stated in Equation 4.71. The results of this are summarised in Figures 4.3-4.11. In Figures 4.3-4.6 the trajectories of the four states are shown for each algorithm with the corresponding state estimate. In addition an error-bar is shown corresponding to a 95% confidence interval of the state estimate. In Figures 4.7 and 4.8 the two control inputs are shown respectively for each algorithm. Figure 4.9 shows the sampling times for each algorithm, which is minimized for the objective and changes at each sampling interval due to the improved estimates of the

uncertain parameters and states. Lastly, Figures 4.10 and 4.11 illustrate the evolution of the uncertain parameter probability density functions for the "SNMPC with feedback" variation. We can draw the following conclusions from the graphs depicted:

- Figure 4.5 shows the relative conservativeness of the SNMPC without feedback compared to the nominal NMPC and the SNMPC with feedback. While the latter two algorithms lead to trajectories that quickly approach the temperature constraint, the SNMPC without feedback first drastically reduces the temperature in the reactor due to the open-loop growth of the uncertainty. This is further highlighted in Figures 4.7 and 4.8, where the SNMPC without feedback keeps the Monomer feed rate and cooling water temperature at its lower bound for much longer than the other two algorithms. In addition it can be seen that the nominal NMPC overshoots the constraint slightly, while the SNMPC algorithms keep a reasonable distance to the constraint to prevent this.

- Figure 4.4 shows the evolution of the monomer concentration. While the nominal NMPC and the SNMPC with feedback operate at a rather steady amount of monomer until it is reduced due to the second terminal constraint, the SNMPC without feedback increases it considerably due to its initial conservativeness. In addition it can be seen that both SNMPC algorithms have a longer "reduction" phase, which is due to considering the uncertainty in the problem. In fact the nominal NMPC algorithm contains 90ppm too much of the monomer in the final batch, while the SNMPC algorithms overshoot the constraint slightly.

- In Figure 4.6 the evolution of the first polymer moment is shown, for which similar observations can be made as in Figure 4.4. The nominal and SNMPC with feedback approach the constraint steadily, while the SNMPC without feedback takes longer due to its conservativeness and consequently low feedrate initially, see Figure 4.7. Again while the nominal NMPC reaches the constraint exactly, both SNMPC algorithms overshoot the constraint to account for the uncertainty present. The final batch of the nominal NMPC has an NAMW that consequently is 0.1 kmol too small.

- It can be seen that the trajectories have different time lengths. The longest batch time is given by the SNMPC without feedback with a time of 6800 seconds due its relatively slow start resulting from the initially large open-loop uncertainties. By disregarding the uncertainties the nominal NMPC algorithm is able to have the shortest batch time with 5000 seconds, which

however leads to it violating both terminal constraints. The SNMPC with feedback is intermediary with a batch time of 5900 seconds, since it considers the uncertainties present but does not have the problem of open-loop growing uncertainties.

- In Figure 4.9 the variation of the sampling times is shown. The nominal NMPC has relatively consistent sampling times, since the reduction in uncertainty has no effect on its constraints. The two SNMPC algorithms on the other hand show an overall reduction of the sampling time, since the uncertainty is lowered from the PCE state estimator update. The SNMPC without feedback is as expected significantly more conservative initially and also shows a more dramatic response to the uncertainty reduction.

- In Figures 4.10 and 4.11 the probability density functions of the uncertain parameters A_p and UA are shown for SNMPC with feedback at different discrete times. In both cases the initial distribution is very broad and quickly approaches the true value. Nonetheless even at $t = 8$ a certain amount of biasness remains in particular for UA. While this is accounted for in the SNMPC algorithms, it is ignored by the nominal NMPC algorithm, which only considers the mean of the distribution.

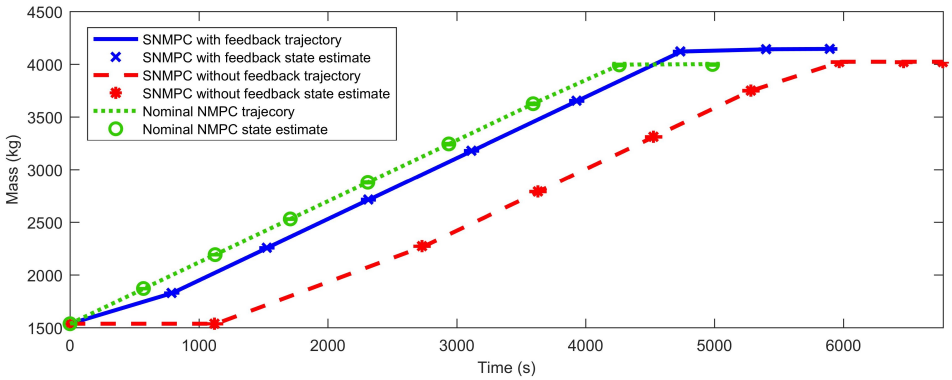


Figure 4.3: Scenario trajectories of the reactor mass for each algorithm variation

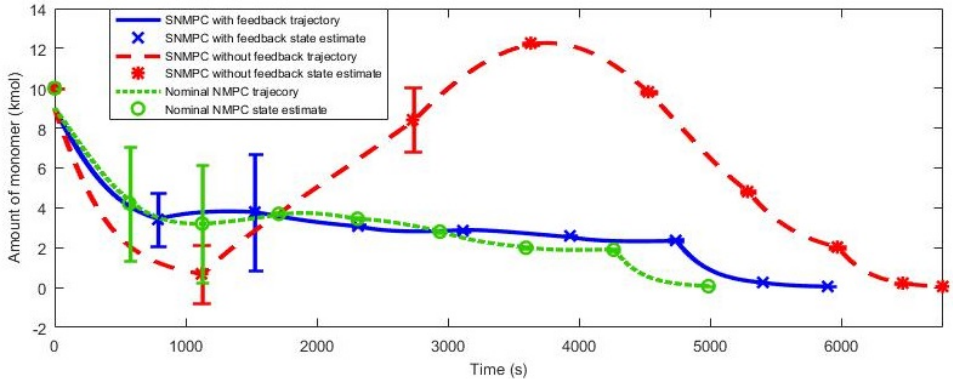


Figure 4.4: Scenario trajectories of the amount of monomer for each algorithm variation

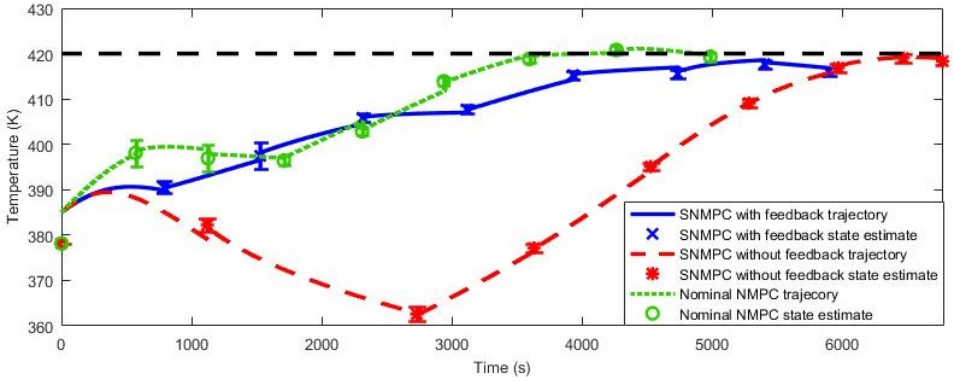


Figure 4.5: Scenario trajectories of the reactor temperature for each algorithm variation

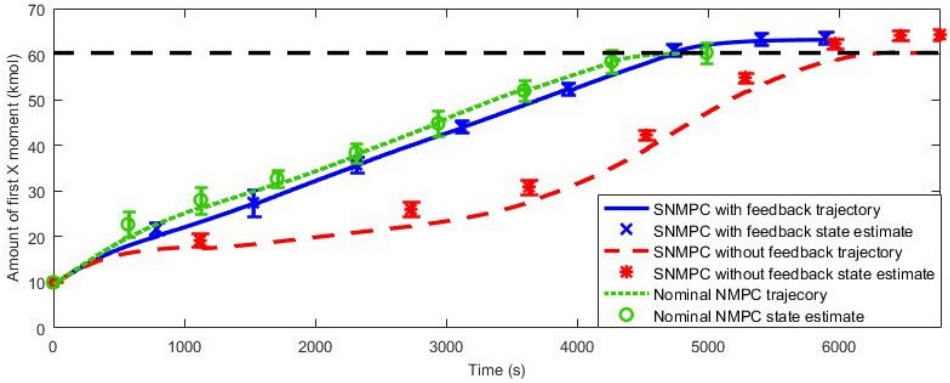


Figure 4.6: Scenario trajectories of the first polymer moment for each algorithm variation

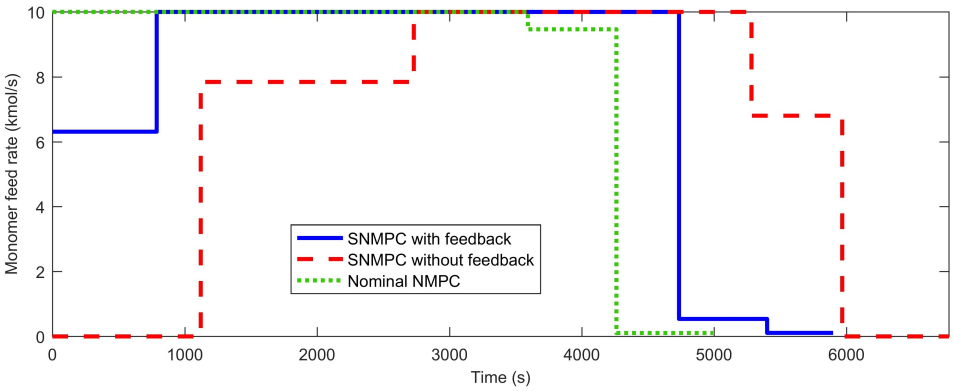


Figure 4.7: Monomer feed rate trajectory for each algorithm variation

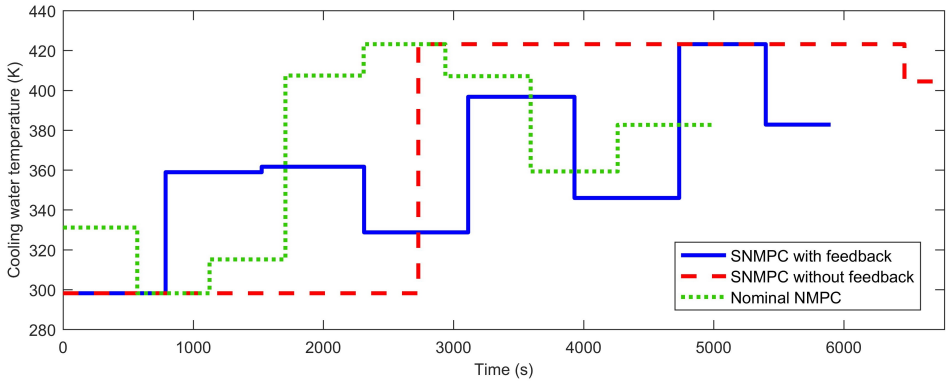


Figure 4.8: Cooling temperature trajectory for each algorithm variation

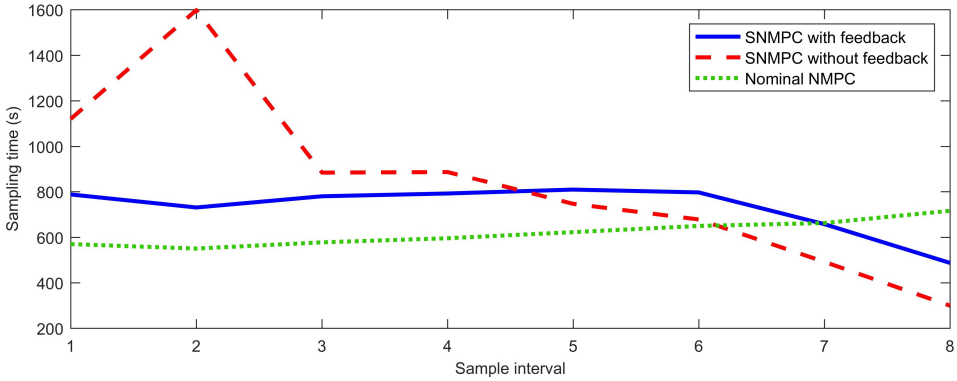


Figure 4.9: Changes in sampling time for each sampling interval and algorithm variation

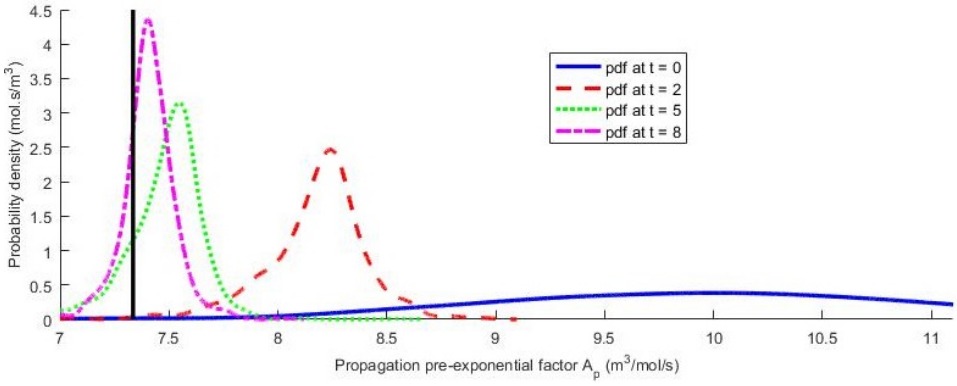


Figure 4.10: Probability density function evolution for the propagation pre-exponential factor in the case of SNMPC with feedback

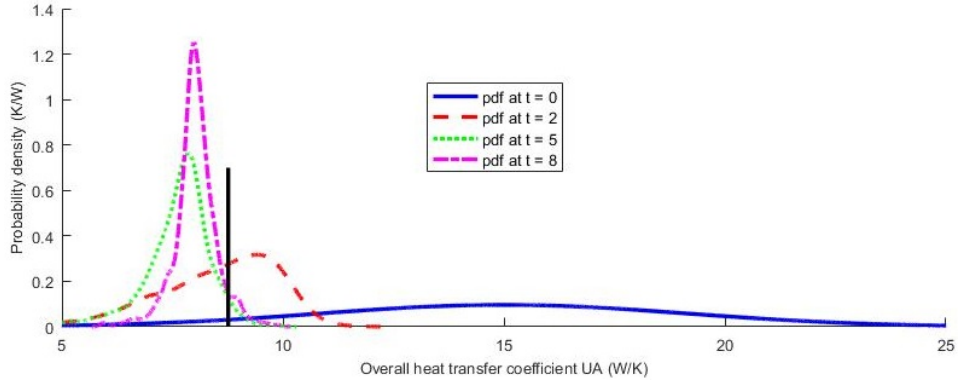


Figure 4.11: Probability density function evolution for the overall heat transfer coefficient in the case of SNMPC with feedback

4.8.2 Monte Carlo simulations

Next we run 100 closed-loop MC simulations of the same case study applying the outlined algorithms by sampling the initial condition and the disturbances independently. The results of these MC simulations are summarised in Figures 4.3-4.8. Figures 4.3, 4.4 and 4.5 show the probability density estimation of NAMW, the ppm of the monomer and the batch time respectively. Figures 4.6, 4.7 and 4.8 show the temperature trajectories of the 100 MC simulations for SNMPC with feedback, SNMPC without feedback and the nominal NMPC respectively. Based on these results the following observations can be made:

- Figure 4.12 shows that for both SNMPC variations the constraint on the NAMW given by the vertical black line is not violated by any of the MC simulations despite the uncertainty present. The nominal NMPC algorithm on the other hand violates the constraint in 46% of the closed-loop simulations.

- Figure 4.13 shows again that both SNMPC algorithms do not violate the constraint on the ppm of the monomer frequently or to a significant extent. The algorithm SNMPC with feedback violated this constraint in 3% of the simulations, while without feedback this constraint was violated in 2% of the simulations. In contrast the nominal NMPC contained too much monomer in the final batch in 53% of the simulations.

- Figure 4.14 illustrates the trade-off for the more robust behaviour from the SNMPC algorithm compared to their deterministic counterpart. While the nominal NMPC requires an average batch time of 4900 seconds, the SNMPC with feedback needs approximately 7000 seconds on average. The SNMPC without feedback requires the largest batch times on average of 7400 seconds. In addition, this illustrates that the feedback leads to a reduction of batch times of about 5% with the same guarantees.

- Figures 4.15-4.17 illustrate the improved temperature control of the SNMPC algorithms compared to the nominal NMPC. While the nominal NMPC violates the temperature control 73% of the time, the SNMPC without feedback breaks it 30% and with feedback 14% of the time. Overall it can be said that the uncertainty has a small effect on the temperature constraint. The relatively large number of violations from the SNMPC variants can be possibly attributed to the inaccuracy of the linearization matrices in Equation 4.81, although the extent of the constraint violation is considerably lower than the nominal NMPC.

- In Table 4.2 the computational times of the SNMPC algorithms and the nominal NMPC are shown. While SNMPC with feedback is significantly less conservative, it does require on average twice the computational time as the SNMPC without feedback. The nominal NMPC is as expected 20 times faster, since it is based on a much smaller optimization problem without scenarios and linearization propagation matrices.

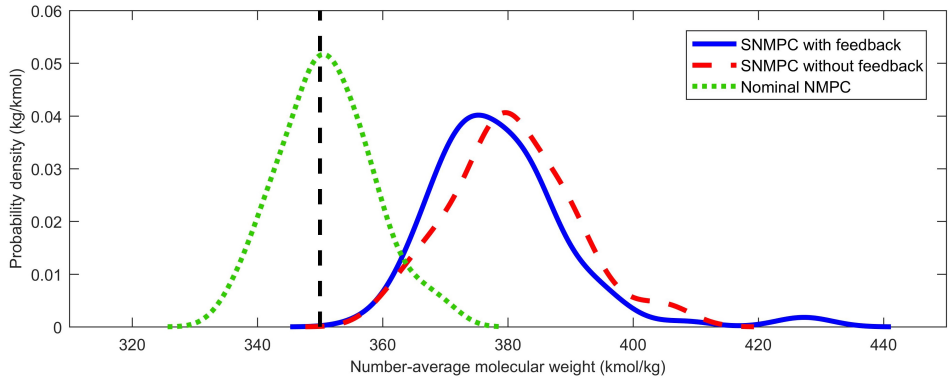


Figure 4.12: Probability density of NAMW based on 100 MC simulations

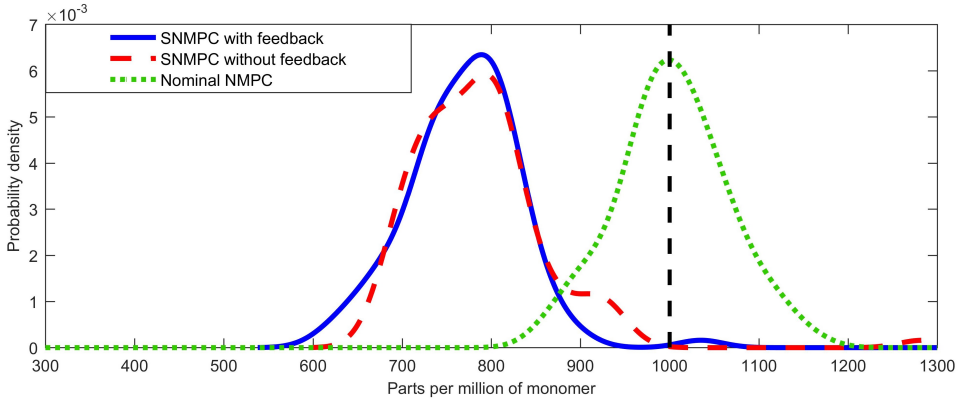


Figure 4.13: Probability density of part per million of the monomer based on 100 MC simulations

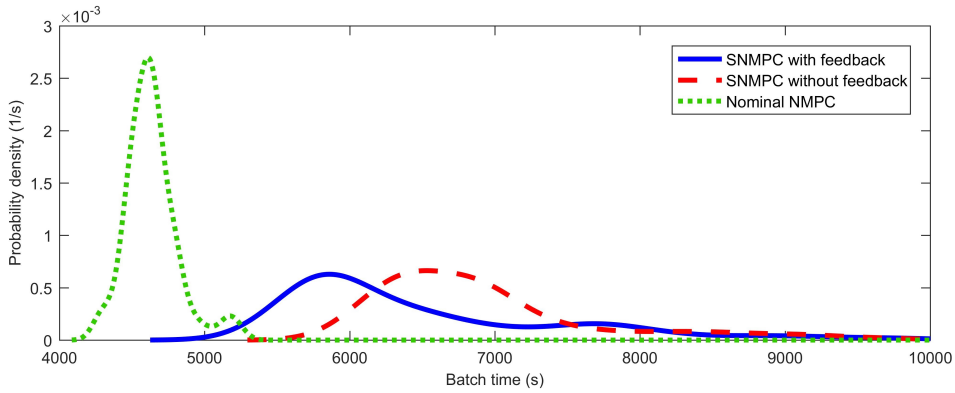


Figure 4.14: Probability density of the batch times based on 100 MC simulations

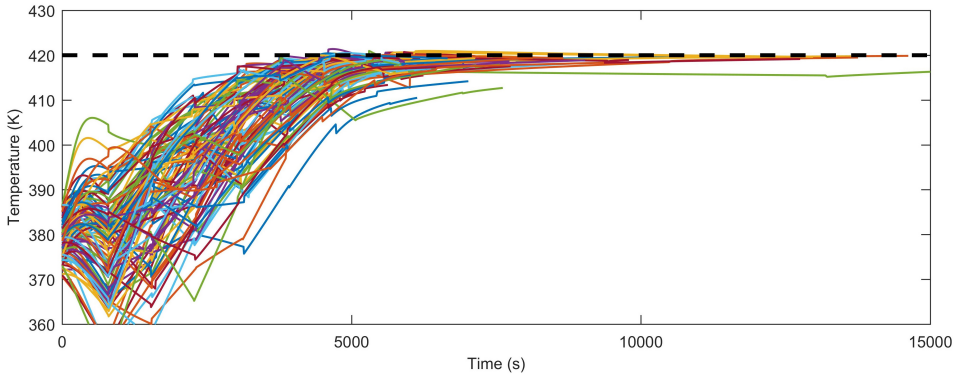


Figure 4.15: Temperature trajectories of 100 MC simulation for SNMPC with feedback

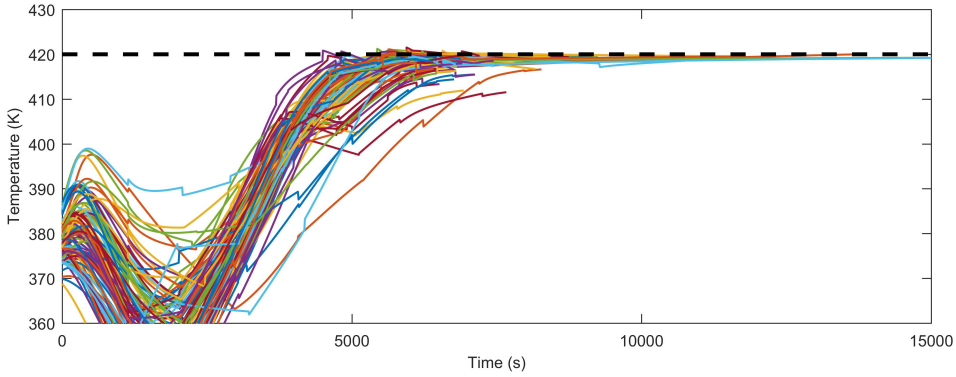


Figure 4.16: Temperature trajectories of 100 MC simulation for SNMPC without feedback

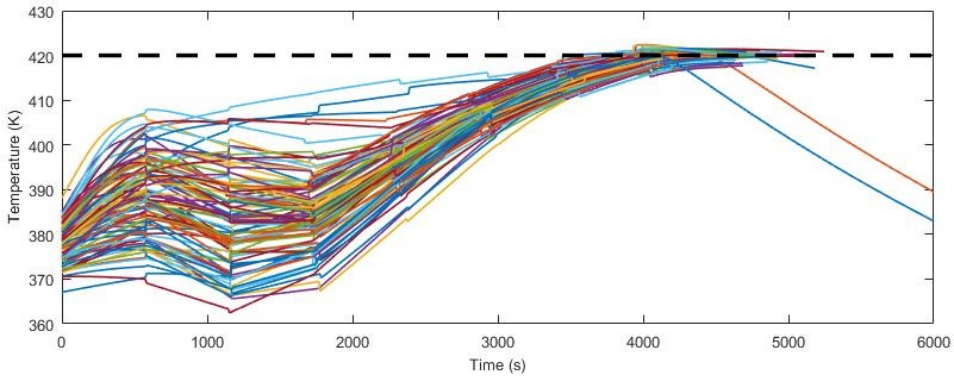


Figure 4.17: Temperature trajectories of 100 MC simulation for nominal NMPC

Table 4.2: The mean and standard deviation computational times for solving the sh-SNMPC OCP from 100 MC simulations

sh-NMPC variation	Mean (s)	Standard deviation (s)
SNMPC with feedback	11.5	9.8
SNMPC without feedback	5.7	4.5
Nominal NMPC	0.27	0.1

4.9 Conclusions

In conclusion a new algorithm has been proposed for output feedback sh-SNMPC for batch processes. The algorithm is able to account for parametric uncertainties, state estimation errors and additive disturbance noise. PCEs are utilised to represent the probability distributions of the states and uncertain parameters, which are updated at each sampling time using a PCE nonlinear state estimator employing noisy output measurements. This PCE representation is then exploited in a SNMPC formulation, which accounts for this uncertainty using PCEs and considers the additive disturbance noise by employing linearization in conjunction with the law of total probability. To reduce the conservativeness the algorithm optimizes not only over open-loop control actions, but also over a time-invariant linear feedback gain. The objective and constraints were based on general nonlinear functions. The aim was set to minimise the objective in expectation, while adhering the constraints in probability to maintain feasibility. For verification purposes a semi-batch reactor case study was used, which showed that the SNMPC framework is able to control the system despite the uncertainties on the initial condition and additive disturbance noise. In particular, ignoring the uncertainty information leads to approximately 50% constraint violations of the terminal constraints and

to 70% violation of the temperature constraint. Further, considering feedback in the SNMPC formulation leads to on average 5% lower batch times with the same guarantees.

Chapter 5

Stochastic Nonlinear Model Predictive Control Using Gaussian Processes

This chapter is based on **Paper C**: E. Bradford and L. Imsland. Stochastic Nonlinear Model Predictive Control Using Gaussian Processes. In *2018 European Control Conference (ECC)*, pages 1027–1034, 2018.

Summary

Model predictive control is a popular control approach for multivariable systems with important process constraints. The presence of significant stochastic uncertainties can however lead to closed-loop performance and infeasibility issues. A remedy is given by stochastic model predictive control, which exploits the probability distributions of the uncertainties to formulate probabilistic constraints and objectives. For nonlinear systems the difficulty of propagating stochastic uncertainties is a major obstacle for online implementations. In this paper we propose to use Gaussian processes to obtain a tractable framework for handling nonlinear optimal control problems with Gaussian parametric uncertainties. It is shown how this technique can be used to formulate nonlinear chance constraints. The method is verified by showing the ability of the Gaussian process to accurately approximate the probability density function of the underlying system and by the closed-loop behaviour of the algorithm via Monte Carlo simulations on an economic batch reactor case study.

5.1 Introduction

The only advanced control method that has been employed to a significant extent in industry is model predictive control (MPC). MPC refers to a control approach that explicitly uses a dynamic model to evaluate a sequence of control actions at each sampling time by solving an optimal control problem (OCP). The success of MPC can be largely attributed to its ability to deal with multivariable plants and process constraint [165].

Many problems however are affected by uncertainties, including inaccuracies from the parameters in the dynamic model and external disturbances. Robust MPC (RMPC) methods have been proposed to handle uncertain systems for which the uncertainties are assumed to be in a bounded set [22]. For robust nonlinear MPC (RNMPC), min-max NMPC [64] and tube-based NMPC [173] have been introduced among others. These approaches enable analysis of the stability and performance of the system in the worst-case, which may however have a very small chance of occurrence and hence lead to a too conservative solution [177].

An alternative to robust MPC is given by stochastic MPC (SMPC), which assumes the uncertainties to be described by known probability density functions (pdf). Constraints in this context are given by chance or expectation constraints. SMPC alleviates the previously described problem by allowing for a level of constraint violation in probability, which leads to a trade-off between risk of constraint violation and closed-loop control performance [175].

Most work in SMPC has been on linear systems, e.g. stochastic tube based MPC [55, 54], scenario-based MPC [213, 24] and affine-parameterization approaches [201, 118], while stochastic NMPC (SNMPC) has received relatively little attention [175]. This can be in part explained by the difficulty of propagating stochastic uncertainties through a nonlinear system model without being prohibitively expensive. An exception are Markovian systems with finite possible realizations of the stochastic uncertainties, for which efficient algorithms are available [205]. Several methods have been proposed to propagate uncertainties through nonlinear systems, such as Monte Carlo sampling (MC), generalized polynomial chaos expansions (gPCe), Gaussian closure, equivalent linearization and stochastic averaging [148].

A simple procedure to solve the SNMPC problem for moderately nonlinear systems is given by successive linearization and application of linear SMPC algorithms, such as stochastic tube based MPC [59]. In [33] stochastic averaging is applied using the unscented transformation, which is computationally efficient, but similar to the linearization approach only applicable to moderately nonlinear

systems. [31] used a sampling average approach to obtain a tractable OCP formulation. The required number of samples was reduced by employing variance reduction techniques. In [256] Markov Chain MC was used to solve the nonlinear MPC problem, which however quickly becomes prohibitive in complexity. In particular, the Markov Chain MC approach suggested tries to find the global optimum, which is only applicable to low dimensional problems. For continuous time the Fokker-Planck equations can be used to predict the pdf of the states over time, which has been used in [50]. A Lyapunov function is included in the SNMPC formulation to guarantee probabilistic stability, however feasibility is not ensured and the method is expensive due to the requirement of solving a partial differential equation system online.

Lastly, much of the research in SNMPC has been concerned with the application of gPCe, which describes a procedure of propagating uncertainties through a nonlinear model as an efficient alternative to MC sampling by utilising orthogonal polynomials [186]. For SNMPC these methods rely on running several realizations of the uncertain parameters, while solving a least-squares problem to calculate the coefficients of the orthogonal polynomials for every iteration of the control inputs [87, 177], which is known as non-intrusive gPCe. Alternatively, Galerkin projection may be used to determine the coefficients of the orthogonal polynomial, which however is only applicable to polynomial-type systems [242]. The chance constraints are either reformulated as second-order cone constraints or using direct MC sampling on the polynomial chaos expansion itself [242]. [18] shows how gPCe can in addition be used for additive disturbances of nonlinear systems, while [17] utilises gPCe to solve a MPC problem for model maintenance by designing experiments online. While gPCe constitute a promising approach for SNMPC, there are several disadvantages. The complexity of gPCe grows exponentially with the number of uncertain parameters, orthogonal polynomials of high orders are prone to unstable swings, time-varying disturbances are difficult to handle and lastly the expansion is only exact as the number of terms tends to infinity [199, 175].

In the statistics community the use of gPCe is rare and instead Gaussian processes (GP) are used for uncertainty quantification in "Bayesian calibration" by running different realizations of the uncertain parameters [135, 198]. An excellent comparison of gPCe to GPs is given in [199].

Gaussian process models are probabilistic, non-parametric models that not only provide predictions, but also prediction uncertainties. GPs have been shown to be a powerful tool in single- [130] and multi-objective optimization [46] by exploiting the uncertainty measure to sample functions efficiently. GPs have found various

applications in MPC. GPs have been shown to be an efficient alternative to neural network models to identify nonlinear models from data for NMPC [143] with successful application to a gas-liquid separation plant [158]. In addition, GPs have been used to identify disturbance models online. For example, in [139] the GP is used to overcome unmodelled periodic error and in [167] to update a model after a fault has occurred. In [114] it is shown how the GP can be used to model residual model uncertainty and formulate chance constraints based on the Gaussian distribution of the states.

In this paper we propose the use of GPs as an alternative to non-intrusive gPCe for SNMPC. The main advantage of using GPs over gPCe in SNMPC is the fact that the uncertainty involved from the approximation of the true model by a finite number of samples is taken into account, which is otherwise ignored by the gPCe. In addition, GPs are not prone to unstable swings and are interpolating, i.e. pass exactly through all sample points provided, but otherwise suffer from the same drawbacks as gPCe. The novelty in this paper is the application of GPs to learn the mapping between uncertain parameters and model outputs for SNMPC applications.

The remainder of the paper is structured as follows. In Sec. II a stochastic nonlinear OCP is formulated. Sec. III introduces Gaussian Process regression with equations to estimate the exact mean and variance from uncertain inputs. In Sec. IV it is shown how the stochastic nonlinear OCP can be approximately solved by employing GPs and how this can be applied in a receding horizon fashion for SNMPC. Lastly, Sec. V tests the approach on a batch reactor case study by comparing open-loop predictions of the pdf and closed-loop performance via Monte Carlo simulations.

5.2 Stochastic Nonlinear Optimal Control Problem Formulation

In this work we consider a general discrete time stochastic nonlinear system with parametric uncertainties:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k), \boldsymbol{\theta}) \tag{5.1}$$

where k represents the discrete time, $\mathbf{x}(k) \in \mathbb{R}^{n_x}$ are the states, $\mathbf{u}(k) \in \mathbb{R}^{n_u}$ are the control inputs, $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ denotes the uncertain model parameters and $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\theta} \rightarrow \mathbb{R}^{n_x}$ represents the nonlinear system dynamics. The parametric uncertainties are assumed to be jointly Gaussian distributed with known mean $\mathbf{m}_{\boldsymbol{\theta}} \in \mathbb{R}^{n_\theta}$ and known covariance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} \in \mathbb{R}^{n_\theta \times n_\theta}$, which fully specifies the pdf of $\boldsymbol{\theta}$.

Based on (5.1), we formulate an OCP. Assuming that the system states are measured at all times, a general OCP can be given as follows:

Finite-horizon nonlinear OCP with chance constraints

$$
\begin{aligned}
&\underset{\mathbf{u}_N}{\text{minimize}} \qquad \mathbb{E}\left(J(N, \mathbf{x}_t, \mathbf{u}_N, \boldsymbol{\theta})\right) + \omega \text{Var}\left(J(N, \mathbf{x}_t, \mathbf{u}_N, \boldsymbol{\theta})\right) \\
&\text{subject to} \\
&\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k), \boldsymbol{\theta}) \qquad \forall k \in \{0, ..., N-1\} \\
&\mathbb{P}(g_j^{(k)}(\mathbf{x}(k), \mathbf{u}(k), \boldsymbol{\theta}) \leq 0) \geq 1 - p_j^{(k)} \\
&\qquad\qquad \forall (k, j) \in \{1, \ldots, N\} \times \{1, \ldots, n_g^{(k)}\} \\
&\mathbf{u}(k) \in \mathbb{U}_k \qquad\qquad \forall k \in \{0, ..., N-1\} \\
&\mathbf{x}(0) = \mathbf{x}_t
\end{aligned}
\tag{5.2}
$$

where the length of the time horizon is given by N, the objective consists of the expectation and variance of a nonlinear function $J(N, \mathbf{x}_t, \mathbf{u}_N, \boldsymbol{\theta})$ weighted by ω, $g_j^{(k)} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\theta} \to \mathbb{R}$ are individual nonlinear chance constraints of which there are $n_g^{(k)}$ for each discrete time k, $p_j^{(k)} \in [0, 1] \subset \mathbb{R}$ is the desired probability of constraint violation with respect to $g_j^{(k)}$, the input constraints are represented by $\mathbb{U}_k$, $\mathbf{u}_N := \{\mathbf{u}(0), \ldots, \mathbf{u}(N-1)\}$ is a collection of inputs and lastly $\mathbf{x}_t$ is the initial state, which is assumed to be known.

The goal of the stochastic nonlinear OCP is to calculate an optimal control sequence over a finite time horizon that adjusts the pdfs of the states to obtain the optimal value of the probabilistic objective function, while allowing for a predefined violation of the stochastic nonlinear constraints.

5.3 Gaussian Process Regression

In this section we give a short introduction to GP regression specific for our purposes. For a more general review, please refer to [263, 220]. GP regression describes the inference of an unknown function $\xi : \mathbb{R}^{n_\theta} \to \mathbb{R}$ from data. The purpose of GPs in our case is to determine an unknown transformation $\xi(\boldsymbol{\theta})$ with respect to the uncertain parameters, hence the input dimension n_θ.

GPs describe a distribution over functions and can be seen as a generalisation of multivariate Gaussian distributions. A GP, $\xi \sim GP(m(\cdot), k(\cdot, \cdot))$, is specified by a mean function $m(\cdot)$ and a covariance function $k(\cdot, \cdot)$, defined as follows:

$$m(\boldsymbol{\theta}) := \mathbb{E}_\xi\left(\xi(\boldsymbol{\theta})\right) \tag{5.3}$$

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') := \mathbb{E}_\xi\left((\xi(\boldsymbol{\theta}) - m(\boldsymbol{\theta}))(\xi(\boldsymbol{\theta}') - m(\boldsymbol{\theta}'))\right) \tag{5.4}$$

where $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^{n_\theta}$ are arbitrary input vectors and $\mathbb{E}_\xi(\cdot)$ is the expectation over the function space. The mean function can be interpreted as the 'average' shape of the

function, while the covariance function specifies the covariance between any two function values computed at the corresponding inputs.

GPs in regression are used to place a prior on admissible functions in a Bayesian framework. The prior is given by the choice of the mean function and the covariance function. In this study we apply a constant mean function and the squared-exponential (SE) covariance function [263]:

$$m(\boldsymbol{\theta}) \coloneqq c \tag{5.5}$$

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \alpha^2 \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^T \boldsymbol{\Lambda}(\boldsymbol{\theta} - \boldsymbol{\theta}')\right) \tag{5.6}$$

where c is a constant, $\boldsymbol{\Lambda} = \frac{1}{\ell^2}\mathbf{I}$, ℓ is a width scaling parameter and α^2 is the signal variance. By selecting the SE covariance function we assume that the underlying transformation to be inferred is smooth and stationary.

Next we require data of the function, i.e. evaluations of the function at specific input values. We assume we are given n_s such function values at n_s different inputs. Let $\boldsymbol{\Theta} = \left[\widetilde{\boldsymbol{\theta}}_0, \widetilde{\boldsymbol{\theta}}_1, \ldots, \widetilde{\boldsymbol{\theta}}_{n_s-1}\right] \in \mathbb{R}^{n_\theta \times n_s}$ denote a data matrix with a collection of training inputs with a corresponding output vector of function evaluations defined by $\mathbf{y} = [\xi(\widetilde{\boldsymbol{\theta}}_0), \ldots, \xi(\widetilde{\boldsymbol{\theta}}_{n_s-1})]^T \in \mathbb{R}^{n_s}$.

The hyperparameters that define the prior of the GP in (5.5) and (5.6) are jointly given by the vector $\boldsymbol{\Psi} = [c, \ell, \alpha]^T$. The hyperparameters are generally unknown a priori, such that they need to be inferred from data. Maximum likelihood estimation (MLE) is commonly carried out to determine $\boldsymbol{\Psi}$. The log-likelihood of the observed data, ignoring constant terms, is given by:

$$\mathcal{L}(\boldsymbol{\Psi}) = -\frac{n_s}{2}\log(\alpha^2) - \frac{1}{2}\log(|\mathbf{K}|) - \frac{(\mathbf{y} - \mathbf{1}c)^T \mathbf{K}^{-1}(\mathbf{y} - \mathbf{1}c)}{2\alpha^2} \tag{5.7}$$

where $K_{ij} = \exp\left(-\frac{1}{2}(\widetilde{\boldsymbol{\theta}}_i - \widetilde{\boldsymbol{\theta}}_j)^T \boldsymbol{\Lambda}(\widetilde{\boldsymbol{\theta}}_i - \widetilde{\boldsymbol{\theta}}_j)\right)$ for all $i, j = [0, \ldots, n_s - 1]$

By setting the derivatives with respect to α^2 and c to zero, it is possible to obtain closed-form expressions for the optimal MLE values of α^2 and c as functions of $\mathbf{K}$ [129]:

$$\hat{c} = \frac{\mathbf{a}\mathbf{y}}{b} \tag{5.8}$$

$$\hat{\alpha}^2 = \frac{\boldsymbol{\nu}^T \mathbf{K}^{-1} \boldsymbol{\nu}}{n_s} \tag{5.9}$$

where $\mathbf{a} = \mathbf{1}^T\mathbf{K}^{-1}$, $b = \mathbf{1}^T\mathbf{K}^{-1}\mathbf{1}$, $\boldsymbol{\nu} = (\mathbf{y} - \mathbf{1}\hat{c})$, $\hat{c}$ and $\hat{\alpha}^2$ are the optimal MLE values of c and α^2 respectively.

The value of the scaling parameter ℓ, was fixed in this work due to the excessive cost of evaluating it online. This will however lead to a worse fit for the GP and a correspondingly larger uncertainty of the model. An alternative could be to find a value for ℓ using (5.7) for different nonlinear function values $\mathbf{y}$ that gives on average the best likelihood value. Possible function values $\mathbf{y}$ can be found by open-loop simulation with different control inputs. Fixing ℓ allows for the precomputation of various quantities. The heuristics that was used in this work for fixing ℓ can be found in [124] and is given as the median of all pairwise euclidean distances between the uncertain parameter values in the data matrix $\mathbf{\Theta}$:

$$\ell = \text{median}(||\widetilde{\boldsymbol{\theta}}_i - \widetilde{\boldsymbol{\theta}}_j||_2) \tag{5.10}$$

Finally, we require the mean and variance of $\xi(\boldsymbol{\theta})|\mathbf{\Theta}, \mathbf{y}$ at an arbitrary input $\boldsymbol{\theta}$, where the input follows a Gaussian distribution with mean $\mathbf{m}_{\boldsymbol{\theta}}$ and covariance matrix $\mathbf{\Sigma}_{\boldsymbol{\theta}}$. $\xi(\boldsymbol{\theta})|\mathbf{\Theta}, \mathbf{y}$ in this case refers to the posterior function of $\xi(\cdot)$, which corresponds to the prior that was updated using the data in $\mathbf{\Theta}$ and $\mathbf{y}$. GPs are commonly used for deterministic inputs. The case of Gaussian distributed uncertain inputs has, however, received extensive attention for the propagation of uncertainties in the case of multi-step ahead predictions [100]. It has been shown that for our choice of mean and covariance functions given in (5.5) and (5.6) respectively, it is possible to calculate the exact mean and variance. The expressions for the expectation and variance of $\xi(\boldsymbol{\theta})|\mathbf{\Theta}, \mathbf{y}$ are [71]:

$$\mathbb{E}\left(\xi(\boldsymbol{\theta})|\mathbf{\Theta}, \mathbf{y}\right) = \hat{c} + \hat{\alpha}^2 \mathbf{d}\boldsymbol{\nu} \tag{5.11}$$

$$\text{Var}\left(\xi(\boldsymbol{\theta})|\mathbf{\Theta}, \mathbf{y}\right) = \hat{\alpha}^2 + \boldsymbol{\nu}^T \mathbf{C}\boldsymbol{\nu} - \hat{\alpha}^2 e - \left(\hat{\alpha}^2 \mathbf{d}\boldsymbol{\nu}\right)^2 \tag{5.12}$$

where $\mathbf{d} = \mathbf{q}^T\mathbf{K}^{-1}$, $\mathbf{C} = \mathbf{K}^{-1}\mathbf{Q}\mathbf{K}^{-1}$, $e = \text{tr}(\mathbf{K}^{-1}\mathbf{Q})$, $q_j = |\mathbf{\Sigma}_{\boldsymbol{\theta}}\mathbf{\Lambda} + \mathbf{I}|^{-1/2}\exp(-\frac{1}{2}(\mathbf{m}_{\boldsymbol{\theta}} - \widetilde{\boldsymbol{\theta}}_j)^T(\mathbf{\Sigma}_{\boldsymbol{\theta}} + \mathbf{\Lambda}^{-1})^{-1}(\mathbf{m}_{\boldsymbol{\theta}} - \widetilde{\boldsymbol{\theta}}_j)$ and

$$\begin{aligned} Q_{ij} = &\exp\left(-\frac{1}{2}(\widetilde{\boldsymbol{\theta}}_i - \mathbf{m}_{\boldsymbol{\theta}})^T\mathbf{\Lambda}(\widetilde{\boldsymbol{\theta}}_i - \mathbf{m}_{\boldsymbol{\theta}})\right) \\ &\times \exp\left(-\frac{1}{2}(\widetilde{\boldsymbol{\theta}}_j - \mathbf{m}_{\boldsymbol{\theta}})^T\mathbf{\Lambda}(\widetilde{\boldsymbol{\theta}}_j - \mathbf{m}_{\boldsymbol{\theta}})\right)|\mathbf{R}|^{-1/2} \\ &\times \exp\left((\mathbf{x}_\text{b} - \mathbf{m}_{\boldsymbol{\theta}})^T\mathbf{R}^{-1}\mathbf{\Sigma}_{\boldsymbol{\theta}}\mathbf{\Lambda}(\mathbf{x}_\text{b} - \mathbf{m}_{\boldsymbol{\theta}})\right) \end{aligned}$$

where $\mathbf{R} = 2\mathbf{\Sigma}_{\boldsymbol{\theta}}\mathbf{\Lambda} + \mathbf{I}$ and $\mathbf{x}_\text{b} = \mathbf{\Lambda}^{-1}(\widetilde{\boldsymbol{\theta}}_i - \mathbf{m}_{\boldsymbol{\theta}}) + \mathbf{\Lambda}^{-1}(\widetilde{\boldsymbol{\theta}}_j - \mathbf{m}_{\boldsymbol{\theta}})$.

It should be noted that while we assume Gaussian distributed uncertainties in this paper, it is possible to determine (5.11) and (5.12) for other types of uncertainties, such as uniformly distributed uncertainties. Therefore, the method can be easily extended to other types of uncertainties.

In the GP-SNMPC algorithm the input training design $\boldsymbol{\Theta}$ is created offline and remains the same online. In addition, the hyperparameter ℓ is fixed once $\boldsymbol{\Theta}$ is fixed based on the heuristic in (5.10). The terms $\mathbf{a}$, b, $\mathbf{K}^{-1}$, $\mathbf{d}$, $\mathbf{C}$ and e in (5.8-5.12) are only functions of $\boldsymbol{\Theta}$ and ℓ, and hence can be pre-computed offline. This makes the use of GPs viable for SNMPC, since otherwise expensive calculations would have to be carried out for each iteration of the optimization algorithm, such as the inversion of the matrix $\mathbf{K}$ or the calculation of $\mathbf{C}$.

5.4 Gaussian Process Stochastic Nonlinear Model Predictive Control

In this section we show how GPs can be exploited for reformulating the OCP given in (5.2). Firstly, we outline how the chance constraints in (5.2) can be reformulated robustly in terms of the mean and variance of the random variable in question. Subsequently, the principle of the GP-SNMPC is highlighted. Lastly, the space-filling parameter design used is outlined and the reformulated OCP problem is given in terms of the resulting samples from the parameter design.

5.4.1 Robust chance constraints

The probabilistic control problem in (5.2) can be solved efficiently by robust reformulation of the chance constraints using the Chebyshev inequality, which results in the following theorem [177]:

Theorem 5.4.1. *Consider a generic probability constraint of the form:*

$$\mathbb{P}(\xi \leq 0) \geq 1 - \epsilon, \quad \epsilon \in (0,1) \subset \mathbb{R} \tag{5.13}$$

where $\xi \in \mathbb{R}$ is some random variable with known mean $\mathbb{E}(\xi) = \hat{\xi}$ and variance $Var(\xi) = \sigma_\xi^2$. Let Ω be the set of random variables with mean $\hat{\xi}$ and variance σ_ξ^2, then for any $\epsilon \in (0,1)$, the distributionally robust probability constraint

$$\inf_{\xi \in \Omega} \ \mathbb{P}(\xi \leq 0) \geq 1 - \epsilon \tag{5.14}$$

can be shown to be equivalent to:

$$\kappa_\epsilon \sigma_\xi + \hat{\xi} \leq 0, \quad \kappa_\epsilon = \sqrt{(1-\epsilon)/\epsilon} \tag{5.15}$$

where σ_ξ is the standard deviation of ξ. ■

The probability constraints in the stochastic OCP in (5.2) are given in the form of:

$$\mathbb{P}(g_j^{(k)}(\mathbf{x}(k), \mathbf{u}(k), \boldsymbol{\theta}) \leq 0) \geq 1 - p_j^{(k)} \tag{5.16}$$

Using Thm. 1 we arrive at the following deterministic constraints for the probability constraint in (5.16):

$$\kappa_{jk}\sqrt{\mathrm{Var}\left(g_j^{(k)}(\mathbf{x}(k),\mathbf{u}(k),\boldsymbol{\theta})\right)}+\mathbb{E}\left(g_j^{(k)}(\mathbf{x}(k),\mathbf{u}(k),\boldsymbol{\theta})\right)\leq 0 \tag{5.17}$$

where $\kappa_{jk}=\sqrt{(1-p_j^{(k)})/p_j^{(k)}}$

It should be noted that Chebyshev inequality leads to conservative constraints, in particular for pdfs that are close to Gaussian. An alternative to this has been used in gPCe based SNMPC by sampling the random variable instead to approximate the probability constraint [242].

5.4.2 GP-SNMPC Principle

Before outlining the exact equations necessary to simplify (5.2), we will first highlight the principle behind the approach. The main difficulty of the OCP in (5.2) is the determination of the statistics of the objective function $J(N,\mathbf{x}_t,\mathbf{u}_N,\boldsymbol{\theta})$ and the nonlinear functions constituting the probabilistic constraints $g_j^{(k)}(\mathbf{x}(k),\mathbf{u}(k),\boldsymbol{\theta})$. Given that $\boldsymbol{\theta}$ is assumed to be time invariant, once all control inputs $\mathbf{u}_N$ are fixed, the values of the states $\mathbf{x}(k)$ and consequently the values of the objective $J(N,\mathbf{x}_t,\mathbf{u}_N,\boldsymbol{\theta})$ and the constraints $g_j^{(k)}(\mathbf{x}(k),\mathbf{u}(k),\boldsymbol{\theta})$ depend solely on the value of $\boldsymbol{\theta}$. The functions can consequently be expressed in the following form:

$$y=\xi(\boldsymbol{\theta}) \tag{5.18}$$

where $\xi(\boldsymbol{\theta})$ denotes an arbitrary transformation of $\boldsymbol{\theta}$.

The problem may now be expressed as the requirement to estimate the pdf of the random variable y given the distribution of $\boldsymbol{\theta}$. In many cases it is sufficient to determine the expectation (first moment) and variance (second moment) of y. SNMPC using GPs is a scenario based approach similar to the non-intrusive polynomial chaos method proposed in [177, 87]. This involves creating several realizations of the uncertain variable $\boldsymbol{\theta}$. Each of these realizations corresponds to a separate nonlinear dynamic equation system given by (5.1) with $\boldsymbol{\theta}$ replaced by the respective realization.

The principle on which the procedure works is illustrated in Fig. 5.1. Each sample of $\boldsymbol{\theta}$ creates a separate trajectory from the initial state $x(0)$, as shown by the lines on the left-hand side graph. These trajectories each have distinct values of x at each discrete time, highlighted by the markers on the same graph. If we are now interested in the statistics of a nonlinear transformation $g(x)$, these values need to be transformed as shown by the arrows. This gives us several values

for each uncertain parameter realization, which we can represent as an unknown transformation of $\boldsymbol{\theta}$, $\xi(\boldsymbol{\theta})$, as is illustrated on the right-hand side graph. GP regression is then used to estimate the unknown transformation $\xi(\boldsymbol{\theta})$. The resulting GP surrogate of $\xi(\boldsymbol{\theta})$ is then used to estimate the mean and variance from the closed-form expressions in (5.11) and (5.12). The GP needs to be rebuilt each time $\mathbf{u}_N$ changes using the data from the different scenarios, i.e. for each iteration step of the optimization algorithm.

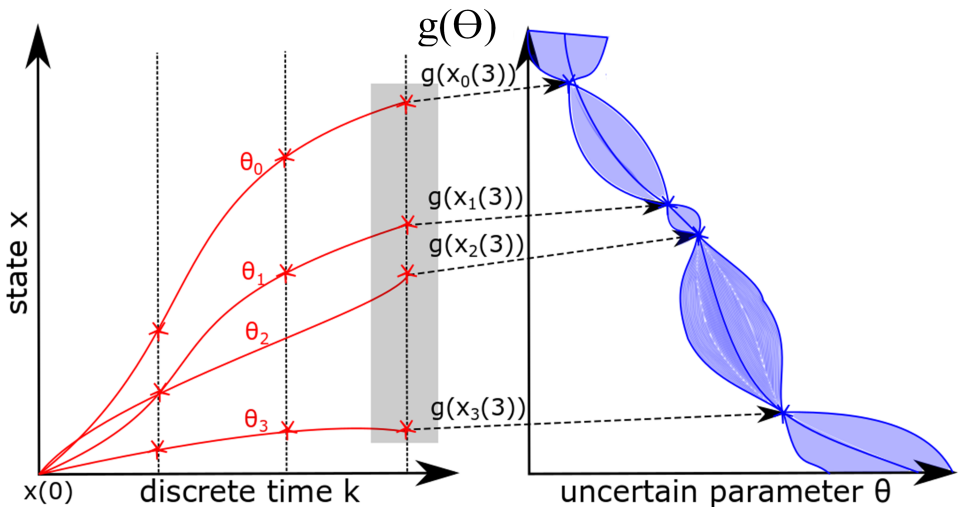


Figure 5.1: Illustration of GP-SNMPC algorithm: On the left-hand side graph the trajectories are shown for each realization of $\boldsymbol{\theta}$ with markers highlighting the different values of the state x. For the final discrete time the values of the states are transformed through $g(x)$, which gives us several values, which are plotted on the right-hand side graph against the realization values of $\boldsymbol{\theta}$. It is then shown on the right-hand side graph that the unknown relationship of the transformation with $\boldsymbol{\theta}$ can be approximated by GP regression

5.4.3 Gaussian process optimal control formulation

The GP-SNMPC algorithm is a sampling-based algorithm, i.e. we create a set of different values of $\boldsymbol{\theta}$, which was represented by $\boldsymbol{\Theta} := \left[\widetilde{\boldsymbol{\theta}}_0, \widetilde{\boldsymbol{\theta}}_1, \ldots, \widetilde{\boldsymbol{\theta}}_{n_s-1}\right]$ in the previous section, where each $\widetilde{\boldsymbol{\theta}}$ implies a separate nonlinear dynamic system given by (5.1). From these separate simulations we then obtain the output values $\mathbf{y}$, which are used to estimate the necessary statistics in (5.19). The parameter design needs to ensure a good spread of $\widetilde{\boldsymbol{\theta}}$ values in the region of significant probability densities. In this work we used min-max Latin hypercube sampling from the Gaussian distribution of $\boldsymbol{\theta}$ to generate the necessary parameter set by using the procedure described in [240]. Based on this parameter design $\boldsymbol{\Theta}$, the GP-SNMPC

OCP problem may be given as follows:

Finite horizon GP-SNMPC problem with chance constraints

$$
\begin{aligned}
&\underset{\mathbf{u}_N}{\text{minimize}} \quad \mathbb{E}\left(\xi_J|\boldsymbol{\Theta},\mathbf{y}_J\right)+\omega\text{Var}\left(\xi_J|\boldsymbol{\Theta},\mathbf{y}_J\right)\\
&\text{subject to}\\
&\mathbf{x}_i(k+1)=\mathbf{f}(\mathbf{x}_i(k),\mathbf{u}(k),\widetilde{\boldsymbol{\theta}}_i)\\
&\qquad\qquad \forall(k,i)\in\{1,\ldots,N-1\}\times\{0,\ldots,n_s-1\}\\
&\kappa_{jk}\sqrt{\text{Var}\left(\xi_{g_{jk}}|\boldsymbol{\Theta},\mathbf{y}_{g_{jk}}\right)}+\mathbb{E}\left(\xi_{g_{jk}}|\boldsymbol{\Theta},\mathbf{y}_{g_{jk}}\right)\leq 0\\
&\kappa_{jk}=\sqrt{(1-p_j^{(k)})/p_j^{(k)}}\quad \forall(k,j)\in\{1,\ldots,N\}\times\{1,\ldots,n_g^{(k)}\}\\
&\hat{c}_J=\frac{\mathbf{a}\mathbf{y}_J}{b},\qquad \hat{\alpha}_J^2=\frac{\boldsymbol{\nu}_J^T\mathbf{K}^{-1}\boldsymbol{\nu}_J}{n_s}\\
&\hat{c}_{g_{jk}}=\frac{\mathbf{a}\mathbf{y}_{g_{jk}}}{b},\quad \hat{\alpha}_{g_{jk}}^2=\frac{\boldsymbol{\nu}_{g_{jk}}^T\mathbf{K}^{-1}\boldsymbol{\nu}_{g_{jk}}}{n_s}\\
&\mathbb{E}\left(\xi_J|\boldsymbol{\Theta},\mathbf{y}_J\right)=\hat{c}_J+\hat{\alpha}_J^2\mathbf{d}\boldsymbol{\nu}_J\\
&\text{Var}\left(\xi_J|\boldsymbol{\Theta},\mathbf{y}_J\right)=\hat{\alpha}_J^2+\boldsymbol{\nu}_J^T\mathbf{C}\boldsymbol{\nu}_J-\hat{\alpha}_J^2e-\left(\hat{\alpha}_J^2\mathbf{d}\boldsymbol{\nu}_J\right)^2\\
&\mathbb{E}\left(\xi_{g_{jk}}|\boldsymbol{\Theta},\mathbf{y}_{g_{jk}}\right)=\hat{c}_{g_{jk}}+\hat{\alpha}_{g_{jk}}^2\mathbf{d}\boldsymbol{\nu}_{g_{jk}}\\
&\text{Var}\left(\xi_{g_{jk}}|\boldsymbol{\Theta},\mathbf{y}_{g_{jk}}\right)=\hat{\alpha}_{g_{jk}}^2+\boldsymbol{\nu}_{g_{jk}}^T\mathbf{C}\boldsymbol{\nu}_{g_{jk}}-\hat{\alpha}_{g_{jk}}^2e-\left(\hat{\alpha}_{g_{jk}}^2\mathbf{d}\boldsymbol{\nu}_{g_{jk}}\right)^2\\
&\mathbf{u}(k)\in\mathbb{U}_k\qquad \forall k\in\{0,...,N-1\}\\
&\mathbf{x}_i(0)=\mathbf{x}_t\qquad \forall i\in\{0,\ldots,n_s-1\}
\end{aligned}
\tag{5.19}
$$

where $\mathbf{x}_i$ corresponds to the state vector of scenario i with uncertain parameter $\widetilde{\boldsymbol{\theta}}_i$, as mentioned in Sec. III the terms $\mathbf{a}$, b, $\mathbf{K}^{-1}$, $\mathbf{d}$, $\mathbf{C}$ and e can be pre-computed offline from $\boldsymbol{\Theta}$, $\mathbf{y}_J=[J(N,\mathbf{x}_t,\mathbf{u}_N,\widetilde{\boldsymbol{\theta}}_i),\ldots,J(N,\mathbf{x}_t,\mathbf{u}_N,\widetilde{\boldsymbol{\theta}}_i)]^T$ is a vector of values of the objective function for each scenario,
$\mathbf{y}_{g_{jk}}=[g_j(\mathbf{x}_0(k),\mathbf{u}(k),\widetilde{\boldsymbol{\theta}}_0),\ldots,g_j(\mathbf{x}_{n_s-1}(k),\mathbf{u}(k),\widetilde{\boldsymbol{\theta}}_{n_s-1})]^T$ is a vector of values for each nonlinear chance constraint for each scenario, $\boldsymbol{\nu}_{g_{jk}}=\mathbf{y}_{g_{jk}}-\mathbf{1}\hat{c}_{g_{jk}}$ and $\boldsymbol{\nu}_J=\mathbf{y}_J-\mathbf{1}\hat{c}_J$.

Algorithm 5.1: GP-SNMPC

Initialize:

- Supply uncertain parameter description: $\boldsymbol{m}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_\theta$

- Create uncertain parameter design $\mathcal{Z}$
- Calculate $\mathbf{a}$, b, $\mathbf{K}^{-1}$, $\mathbf{d}$, $\mathbf{C}$ and e from (5.8-5.12)
- Define OCP in (5.19)

At each sampling time $t = 0, 1, 2, \ldots$

- Take measurement $\mathbf{x}_t$
- Solve (5.19) to obtain $\mathbf{u}_N$
- Apply the first control input from $\mathbf{u}_N$, $\mathbf{u}(0)$ to the real system

A few remarks regarding the computational complexity of the algorithm. Firstly, the expressions involving the expectation and variance are either linear or quadratic with respect to the scenario output, which yields an overall well-posed optimization problem that is smooth everywhere and makes implementation relatively easy. The algorithm is the most effective when there are a small number of constraints compared to the number of states, since then only a small number of GPs are required, the computational cost of which are likely negligible.

5.5 Stochastic Nonlinear Model predictive control of a batch reactor

5.5.1 Dynamic model equations and OCP formulation

In this section the algorithm is verified on a semi batch reactor with a cooling jacket. The reaction is the saponification of ethyl acetate, which is a good example of an exothermic reaction for which safety concerns are paramount to prevent a thermal runaway. The dynamic model was taken from [94]:

$$\dot{C}_A = -k(C_A C_B - C_C C_D / K_C) - F C_A / V, \tag{5.20a}$$
$$\dot{C}_B = -k(C_A C_B - C_C C_D / K_C) + F(\exp(\theta_2) - C_B)/V, \tag{5.20b}$$
$$\dot{C}_C = k(C_A C_B - C_C C_D / K_C) - F C_C / V, \tag{5.20c}$$
$$\dot{N}_W = F C_{W0}, \tag{5.20d}$$
$$\dot{T} = \frac{\dot{Q}_{gs} - \dot{Q}_{rs}}{N C_p}, \tag{5.20e}$$
$$\dot{V} = F \tag{5.20f}$$

where $k = 0.0039175 \exp(5472.7(1/273 - 1/T))$, $K_C = 10^{\exp(\theta_3)/T}$, $C_D = C_C$, $C_w = N_W/V$, $\dot{Q}_{gs} = Vk(C_A C_B - C_C C_D/K_C)\Delta H_{RX}$, $\dot{Q}_{rs} = F \exp(\theta_1)(\exp(\theta_2) + C_{W0})(T - T_0) + UA(T - T_j)$,
$NCP = V\left(C_{P_A} C_A + \exp(\theta_1)(C_B + C_C + C_D + C_W)\right)$,$C_A, C_B, C_C, C_D$ are the concentration of species A, B, C and D respectively in kmol.m^{-3}, T is the reactor temperature in K, N_W the amount of water in the reactor in kmol and V the reactor volume in m^3.

Three parameters were overall assumed to be uncertain indicated by components of $\boldsymbol{\theta}$, which are assumed to be Gaussian distributed with mean and covariance given by:

$$\mathbf{m}_{\boldsymbol{\theta}} = \begin{bmatrix} 4.40 \\ 0.59 \\ 8.26 \end{bmatrix}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}} = \begin{bmatrix} 2 \cdot 10^{-3} & 5 \cdot 10^{-6} & -2 \cdot 10^{-4} \\ 5 \cdot 10^{-6} & 2.5 \cdot 10^{-3} & -2 \cdot 10^{-4} \\ -2 \cdot 10^{-4} & -2 \cdot 10^{-4} & 1 \cdot 10^{-2} \end{bmatrix} \tag{5.21}$$

For the missing parameters refer to [94], example 13-3.

The control input is the feedrate to the semi-batch reactor given by F. In compact form we can write $\mathbf{x} = [C_A, C_B, C_C, N_W, T, V]^T$ and $u = F$. Using discretization the equations can be given as a discrete time equation system with the discretization time set to $15s$. Direct orthogonal collocation was used for the discretization of the dynamic equation system in (5.20) with 4th order polynomials placed according to the Radau quadrature rule.

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), u(k), \boldsymbol{\theta}) \tag{5.22}$$

Based on this equation system we formulate an OCP as follows:

$$\begin{aligned} &\underset{\mathbf{u}_N}{\text{minimize}} && -\mathbb{E}\left(C_C(N)V(N)\right) + 1.5\text{Var}\left(C_C(N)V(N)\right) \\ &\text{subject to} \\ &\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), u(k), \boldsymbol{\theta}) && \forall k \in \{0, ..., N-1\} \\ &\mathbb{P}(T_{adiabatic}(k) - 320 \leq 0) \geq 0.95 && \forall k \in \{1, \ldots, N\} \\ &u(k) \in [0, 8 \times 10^{-3}] && \forall k \in \{0, ..., N-1\} \\ &\mathbf{x}(0) = \mathbf{x}_t \end{aligned} \tag{5.23}$$

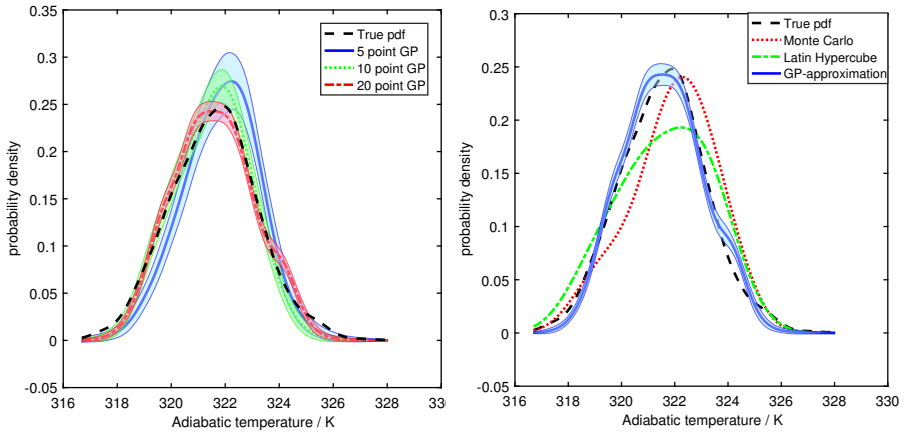
where $T_{adiabatic}(k)$ refers to the adiabatic temperature change at discrete time k defined as $T_{adiabatic}(k) = T(k) - \Delta H_{RX} C_B(k) V(k)/NCP(k)$, assuming that C_A is in excess. The adiabatic temperature change refers to the maximum attainable temperature in the reactor under a cooling failure.

The OCP in (5.23) aims to maximize the mean production of C with trade-off towards the variation of the production of C, while keeping the adiabatic temperature below 320K in probability to prevent a thermal runaway. The stochastic OCP in (5.23) is converted using GPs. Algorithm 5.1 shows how this OCP can then be used for a receding-horizon implementation to obtain a stochastic MPC. The time horizon was set to $N = 15$ with a final batch time of $240s$. The OCP was solved utilising Casadi [8]. The resulting nonlinear programming problem was solved by employing IPOPT [258]. Lastly, IDAS [116] was applied to simulate the "real" nonlinear equation system. At time $t = 0$, the the initial state to solve the OCP was set to $\mathbf{x}_0 = [25, 1, 0, 6.6, 0.2, 310]$.

5.5.2 Open-loop predictions of statistics

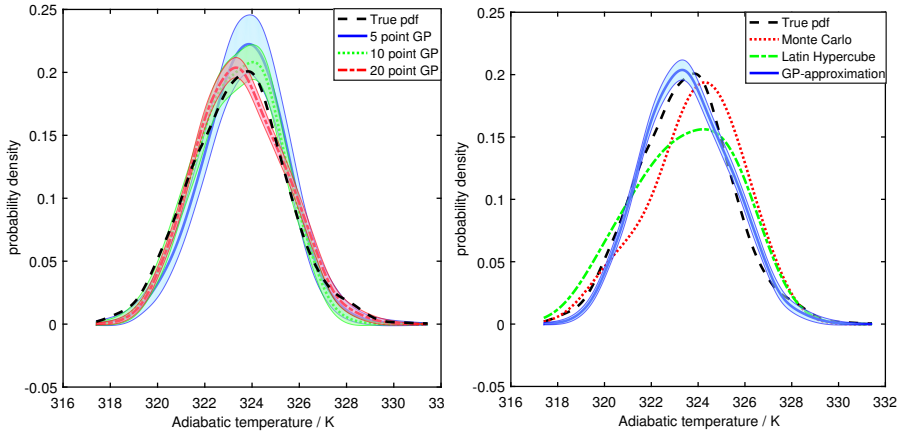
We first evaluate the ability of the GPs to estimate the statistics of the relevant random variables. The procedure was taken from [203]. In particular, the random variables "adiabatic temperature" at discrete times $k = 5$ and $k = 15$ and "amount of species C" at the end of the batch were chosen for the analysis, since these represent the nonlinear probabilistic constraint and the objective respectively. To carry out the comparison the feed rate was fixed at $F = 8 \times 10^{-3}\mathrm{m}^3/\mathrm{s}$ and the uncertain parameter distribution was sampled 1000 times to obtain the "true" pdf. This is contrasted to the approximations of the pdfs obtained from the GP-approximations built from 5-points, 10-points and 20-points, which is shown in the graphs in Fig. 2 on the left-hand side. The pdfs were obtained from kernel density estimation. In addition, it is possible for GPs to calculate a confidence region for the pdfs, since a GP corresponds to not just one function value, but a distribution of possible function values. We can see that for all 3 pdfs the approximations get closer to the "true" pdf as the number of points increases. In addition, it can be seen that the confidence region is largest for the GP with 5 points and smallest for the GP with 20 points, which reflects the observations made. For 20 points the GP more or less matches the pdf in all 3 cases.

Furthermore, the 20-point GP approximations were compared on the right-hand side graphs to the pdf estimates obtained from 20 Monte Carlo samples and the 20 "maximin" Latin Hypercube samples the GP was built from. We can see that the GP approximation gives the most accurate representation of all pdfs, while the "maximin" Latin Hypercube tends to overestimate the variance and the Monte Carlo tends to underestimate the variance.



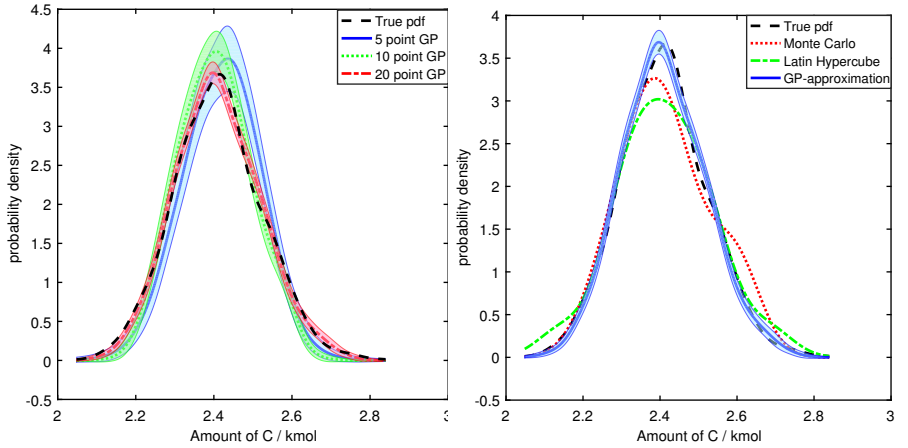
(a)

Adiabatic temperature at $k = 5$



(b)

Adiabatic temperature at $k = 15$



(c)

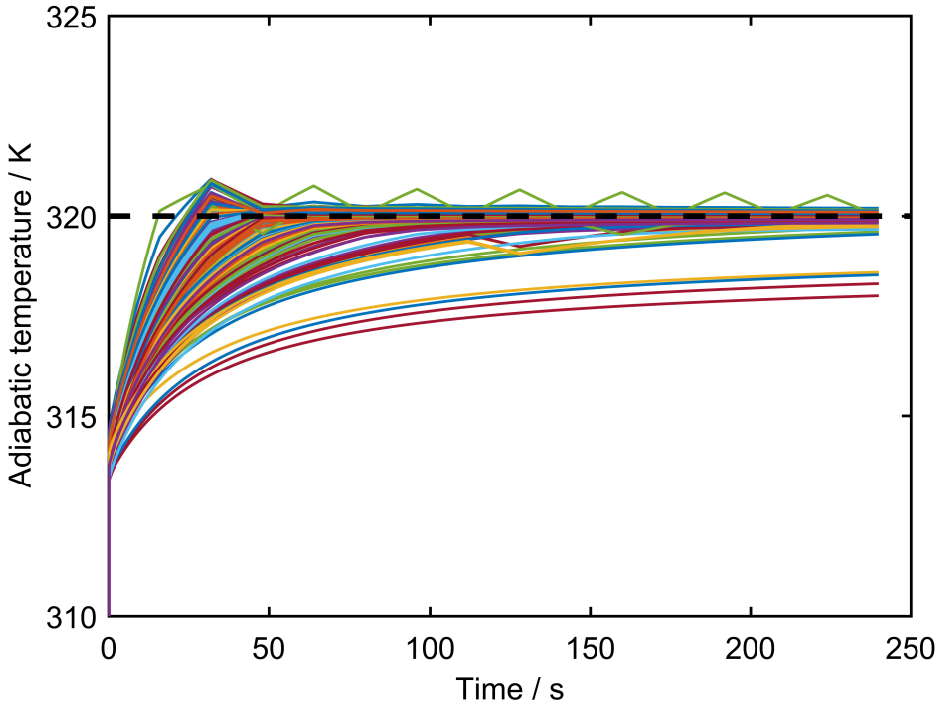
Amount of C at $k = 15$

Figure 5.2: pdf estimation comparison results for the OCP in (5.23) with fixed $F = 8 \times 10^{-3}\mathrm{m}^3/\mathrm{s}$. Left graph: true pdf compared to GP approximations with 5 points, 10 points and 20 points. For the GP plots a 99% confidence region is shown. Right graph: true pdf compared to Monte Carlo, Latin hypercube and GP-approximation built on a sample size of 20 points.

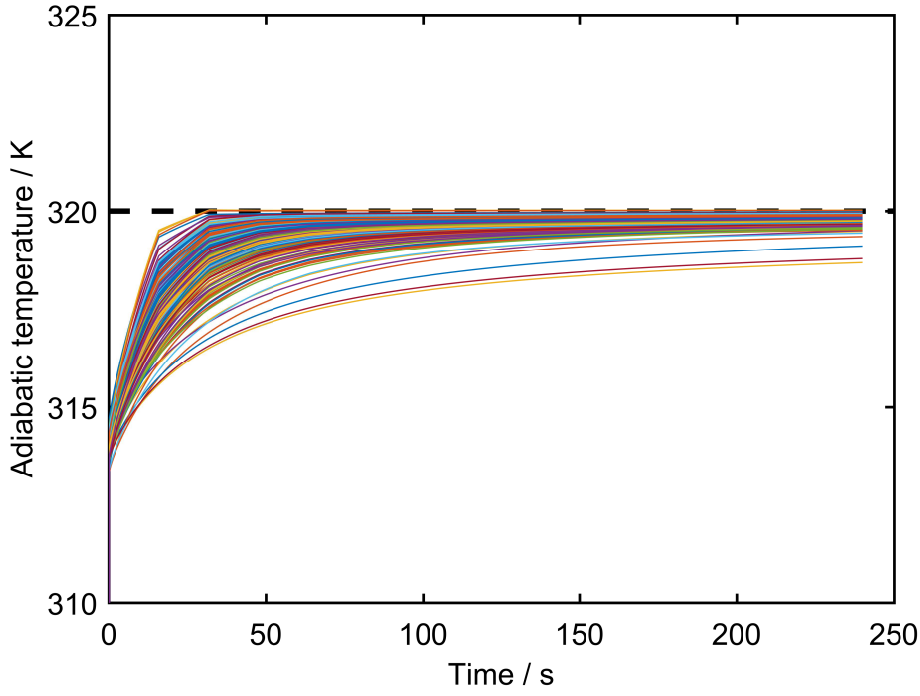
5.5.3 Closed-loop implementation

Next, the OCP in (5.23) was solved in a shrinking horizon fashion to gauge the closed-loop performance of the algorithm with the size of the uncertain parameter design set to 20. To verify the robustness of the approach 200 Monte Carlo simulations were performed by sampling 200 independent realizations of the uncertain parameter from the Gaussian multivariate distribution. These uncertain parameters are used for the simulation of the "real" system to which the control inputs of the GP-SNMPC algorithm were applied. The algorithm is compared to a nominal NMPC, again using a shrinking horizon implementation. The OCP of the nominal NMPC implementation is similar to the OCP in (5.23), except that the predictions are taken to be deterministic by fixing the uncertain variables to their nominal values. Therefore, the objective is given by the predicted amount of C at the final time. To avoid infeasibilities the adiabatic temperature constraint was formulated using soft constraints.

In Fig. 5.3 the trajectories of the adiabatic temperature constraint are shown. It can be seen that the nominal NMPC algorithm violates the upper bound significantly for many of the 200 simulations. Overall, 51% of the simulations for the NMPC algorithm lead to constraint violations. It can be seen that the violations mostly occur at the beginning of the reaction due to the initial concentration of B being high. Once the concentration of B becomes lower towards the later parts of the reaction, the probability of constraint violation is lower, since then the adiabatic temperature change corresponds more or less to the actual temperature of the reactor with less uncertainty. The GP-SNMPC on the other hand fulfilled the constraint in all simulations.



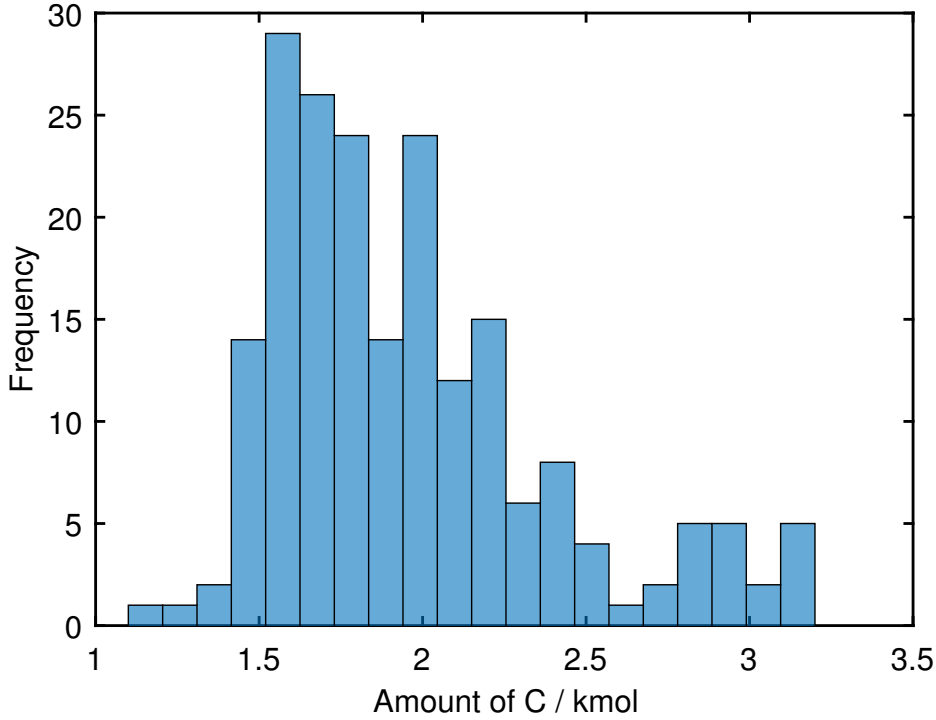
(a) Nominal nonlinear model predictive control



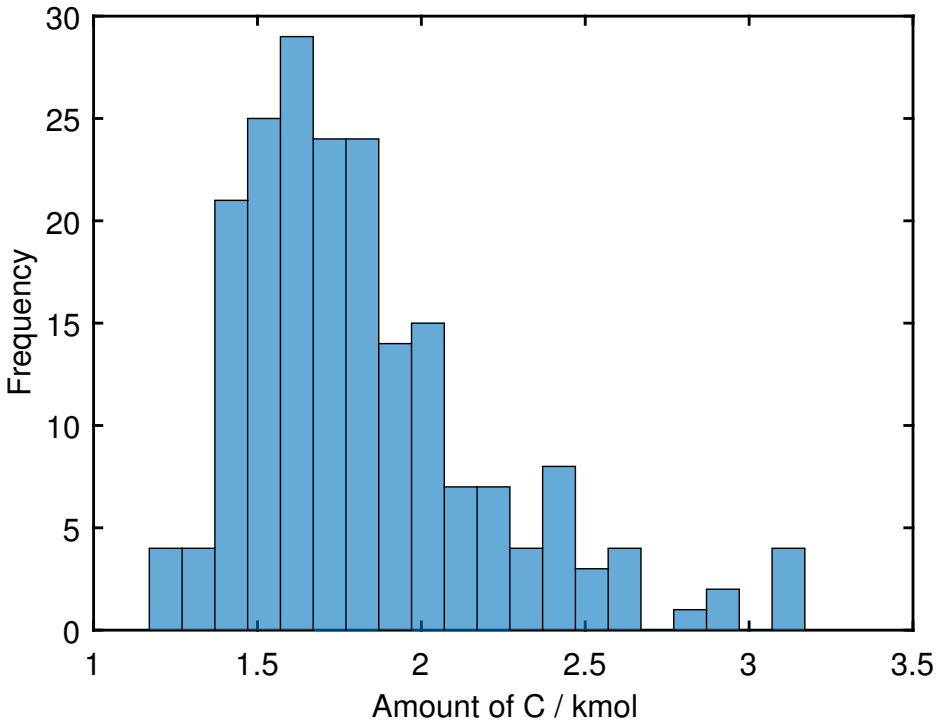
(b) Stochastic nonlinear model predictive control

Figure 5.3: 200 Monte Carlo trajectories of the adiabatic temperature constraint. The constraint at 320K is highlighted by a dashed black line.

In Fig. 5.4 the histograms of the amount of C produced at the end of the batch are shown based on the 200 Monte Carlo simulations for both methods. The objective of the GP-SNMPC was given in terms of both the mean and variance, while the nominal NMPC was geared towards maximizing the amount of C only. The GP-SNMPC produced on average 1.83 kmol of C, while the nominal NMPC algorithm produced 1.96 kmol on average, which is approximately 7% more. On the other hand the variance of the GP-SNMPC is 25% less, hence the performance differences may be due to a different control objective or due to the conservativeness of the GP-SNMPC algorithm.



(a) Nominal nonlinear model predictive control



(b) Stochastic nonlinear model predictive control

Figure 5.4: Histograms of the amount of C at the end of the batch based on the 200 Monte Carlo simulations

5.6 Conclusions

In conclusion, a novel framework for SNMPC has been proposed by employing GPs to handle Gaussian distributed parametric uncertainties with known mean and covariance. The SNMPC problem involved both objective and probability constraints based on general nonlinear functions. The probability constraints were reformulated requiring only mean and variance. GP-SNMPC is a scenario-based MPC algorithm that uses the data from the various realizations of the uncertain parameter to build a surrogate, from which the variance and mean of the nonlinear objective and constraint functions are estimated efficiently. In addition, GP-SNMPC not only takes into account the variance induced by the uncertain parameters but also, unlike gPCe, the uncertainty of the surrogate itself. A semi-batch reactor case study showed that GP-SNMPC is able to provide predictions on the statistics of the problem more accurately than either Monte Carlo or Latin hypercube samples. Lastly, a shrinking horizon application showed excellent closed-loop performance by ensuring the fulfilment of a nonlinear chance constraint, while optimizing a probabilistic objective.

Chapter 6

Combining Gaussian processes and polynomial chaos expansions for stochastic nonlinear model predictive control

This chapter is based on **Paper D**: E. Bradford and L. Imsland. Combining Gaussian processes and polynomial chaos expansions for stochastic nonlinear model predictive control. *Journal of Process Control*, submitted, 2020.

Summary

Model predictive control is an advanced control approach for multivariable systems with constraints, which is reliant on an accurate dynamic model. Most real dynamic models are however affected by uncertainties, which can lead to closed-loop performance deterioration and constraint violations. In this paper we introduce a new algorithm to explicitly consider time-invariant stochastic uncertainties in optimal control problems. The difficulty of propagating stochastic variables through nonlinear functions is dealt with by combining Gaussian processes with polynomial chaos expansions. The main novelty in this paper is to use this combination in an efficient fashion to obtain mean and variance estimates of nonlinear transformations. Using this algorithm, it is shown how to formulate both chance-constraints and a probabilistic objective for the optimal control problem. On a batch reactor case study we firstly verify the ability of the new approach to accurately approximate the probability distributions required. Secondly, a tractable

stochastic nonlinear model predictive control approach is formulated with an economic objective to demonstrate the closed-loop performance of the method via Monte Carlo simulations.

6.1 Introduction

Model predictive control (MPC) was developed in the late seventies and refers to a popular control approach that has been applied in the process industry to deal with multivariable control systems with important constraints. MPC solves at each sampling time a finite horizon optimal control problem (OCP) to determine a control action to take, by exploiting a dynamic model directly. Feedback enters this process by updating the initial state at each sampling time with the available measurements. Nonlinear MPC (NMPC) refers to MPC that utilize nonlinear dynamic models, which is particularly relevant for highly nonlinear problems operated at unsteady state, such as batch processes [165]. While most NMPC algorithms are based on data-driven nonlinear models, NMPC using first-principles is becoming increasingly used due to the availability of more efficient optimization algorithms [27]. Commonly MPC algorithms are applied for set-point tracking, while economic MPC employs as cost function the quantity to be maximized directly, such as profits [223]. Many dynamic models include significant uncertainties, such as parametric deviations or unaccounted disturbances. These may negatively affect the performance of the MPC algorithm and lead to constraint violations. For economic MPC the system is often driven close to its constraints [163]. It is therefore crucial to account for significant uncertainties in the formulation of the MPC problem.

Assuming uncertainties to be deterministic and bounded leads to robust MPC (RMPC). Min-max MPC frameworks are among the first methods proposed and focus on minimizing the cost while satisfying constraints under worst-case realizations [232]. Min-max MPC has also been applied to nonlinear systems, for example in [64]. These methods however were found to be often unable to deal with the spread of the state trajectories or be overly conservative. Tube-based approaches were subsequently developed to address these limitations, which uses a feedback controller explicitly to ensure that the real system remains in a tube computed offline centred around the nominal solution [174]. Several nonlinear tube-based MPC algorithms have been proposed including [173, 171, 145]. RMPC allows for the analysis of stability and performance of the system in the worst case, which may however have a diminishingly small chance of occurrence and hence can be overly conservative.

An alternative to RMPC is given by stochastic MPC (SMPC). In SMPC the uncertainties are described by known probability density functions (pdf). In SMPC

constraints are given by either expectation or chance constraints. Therefore, SMPC allows for a controlled rate of constraint violations, which avoids the previously mentioned problem of RMPC and leads to a trade-off between risk of constraint violation and performance [175]. Recent reviews on SMPC can found in [88, 175]. SMPC has predominantly been developed for linear systems. An important group of algorithms are given by probabilistic tube-based approaches as counter-part to their robust versions for additive and multiplicative noise [55, 57]. Alternatively several approaches have been suggested using affine parameterization of either state or disturbance feedback [149, 201, 118], which apply exact propagation methods for the mean and covariance. Lastly, scenario-based MPC methods have been put forward that simulate separate realizations of the uncertain parameters and use Monte Carlo estimations of chance constraints and objectives [230, 213, 70]. Stochastic NMPC (SNMPC) has on the other hand received significantly less attention, which may be in part explained by the difficulty of propagating stochastic variables through nonlinear transformations. An exception to this is given by the case of only discrete realizations of uncertainties for which efficient algorithms have been developed using multi-stage stochastic programming [162, 205]. These propagate each possible scenario and ensure that none of the constraints are violated. Several procedures are used in literature to analyse stochastic uncertainties of nonlinear systems, including Monte Carlo (MC) sampling, polynomial chaos expansions (PCE), Gaussian closure, equivalent linearization, and stochastic averaging [148].

A straight-forward approach for SNMPC is given by successive linearization, such as in [184] which uses an extended Kalman filter approach to propagate the stochastic uncertainties or as in [55] that applies the probabilistic tube-based approach on the successively linearized system. An alternative is given by applying the Unscented transformation [34]. Both linearization or Unscented transformations, while being computationally cheap, are only applicable to moderately nonlinear systems. [225, 31] utilize a sampling average approach to approximate chance constraints and objective. The required number of samples to obtain accurate predictions however quickly becomes prohibitive. In [233] an output feedback SNMPC approach is introduced by using the particle filter equations for both prediction of future state distributions and for updating the states from available measurements. Again the required number of samples can be prohibitive. [166] proposed to use Markov chain MC sampling to propagate the uncertainties and solve the optimization problem. The computational cost of the suggested method is high, since it aims to find the global optimum at each sampling instant. In the case of continuous time the Fokker-Planck partial differential equation system can be used to describe the evolution over time of the pdfs of the states, which is used in [50] for SNMPC. A Lyapunov function is included to guarantee stochastic stabil-

ity. Much of the work in SNMPC has been concerned with the application of PCEs. PCEs are an efficient alternative to MC sampling for approximating the probability distribution of a nonlinear transformation of stochastic variables by employing orthogonal polynomials [186]. PCEs in this context are a scenario-based SNMPC algorithm that uses least-squares estimation online for every iteration of inputs to approximate the coefficients of an orthogonal polynomial approximation, known as non-intrusive PCE [87]. For polynomial-type systems Galerkin projection is used instead to determine the coefficients, which is called intrusive PCE [242]. Chance constraints can either be given using Chebychev's inequality [177] or applying a MC sampling approximation on the orthogonal polynomials themselves [242]. The PCE based SNMPC algorithm has been extended to the case of output feedback in [37, 45] by combining the approach with a PCE nonlinear state estimator. The usefulness and generality of PCE can be seen for example by its use in [17] to formulate a SNMPC formulation for design of experiments online to maintain the dynamic model or in [110] for discriminating between different dynamic models for fault-diagnosis.

While PCE leads to useful SNMPC algorithms, it does have a few disadvantages:

- Computational complexity grows exponentially with the number of uncertain parameters.
- Orthogonal polynomials of high-order are prone to unstable swings.
- Time-varying disturbances are difficult to handle.
- Expansion is only exact for infinitely many terms.

In the statistics community PCEs are rarely used. Gaussian processes (GP) are employed instead for uncertainty analysis in "Bayesian calibration" [135, 198]. A comparison of GPs to PCEs is given in [199]. Gaussian processes are stochastic processes that are used as non-parametric models, which unlike other popular regression methods such as neural networks not only provide predictions, but also prediction uncertainties. GPs have been applied in several MPC papers. In [144] GPs were used to identify a nonlinear dynamic model from data as the prediction model in a NMPC algorithm. This methodology has been successfully applied in [158] to control a gas-liquid separation plant. Further, in [137] the GP models are updated efficiently online using recursive formulas and sparse approximations. Furthermore, GPs have been shown to be a powerful tool to model disturbances. In [167] a GP is used to correct a dynamic model online for fault-tolerant control, while in [139] a GP is employed to learn a function for an unmodelled periodic error term. Similarly, [114] proposes to model residual model uncertainty by GPs. In

[42] a GP-based algorithm is proposed that tightens the constraints offline to maintain feasibility online. GPs have been in addition employed in multiple works to approximate the mean and covariance of nonlinear transformations for the Kalman filter equations [73, 215, 229], which bear some similarity to this paper's use of GPs.

In this paper we propose a new method using GPs and PCEs jointly for SNMPC. In this regard we employ PCEs as mean function for the GP. The combination will be referred to as "GPPCE". GPs are well-known to approximate the function well locally, but not as well globally. PCEs on the other hand are better suited for global function approximations, but may not have the same accuracy between data-points as GPs [203]. The combination of both is therefore beneficial. Another advantage over regular PCEs apart from better local fits is that the uncertainty introduced through the sample approximation can be taken into account from the GPPCE, which is otherwise ignored. Furthermore, GPs are not prone to unstable swings and are interpolating, i.e. pass through all sample points exactly. Otherwise GPs suffer from similar drawbacks as PCEs. Combining GPs and PCEs for uncertainty quantification has been previously proposed in [231]. The main novelty in this paper is to show how to use this GPPCE to obtain cheap approximations of mean and variance using closed-form expressions derived in this paper. In addition, terms are identified that can be calculated *offline* to significantly reduce computational costs *online*. For SNMPC the terms are utilised directly in the optimization problem and hence it is paramount that the mean and variance estimator are fast. Lastly, we show how the GPPCE expressions can be utilised to approximate the SNMPC problem. Using GPs for SNMPC was first introduced in [36]. The remainder of the paper is structured as follows. In the first three sections of the paper we show how GPPCE can be formulated and used in an efficient fashion to propagate uncertainties. In Section 6.2 GPs with a PCE mean function (GPPCE) are introduced. Thereafter, in Section 6.3 terms are derived to obtain posterior mean and variance estimates given a noisy input. Section 6.4 shows how these expressions can be utilised efficiently to propagate uncertainties. Next we show how GPPCE can be exploited to formulate a SNMPC algorithm. Section 6.5 defines the general problem to be solved using the GPPCE SNMPC algorithm, while Section 6.6 introduces the GPPCE SNMPC algorithm to accomplish this task. A challenging semi-batch reactor case study is outlined in Section 6.7. Results and discussions to this case study are presented in Section 6.8. The paper is concluded in Section 6.9.

6.2 Gaussian processes with polynomial chaos expansion mean function

This section presents GPs and PCEs for our purposes and is not meant as a general introduction. Please refer to [220, 226, 130] for general descriptions of GPs and refer to [136, 199, 186] for a general outline on PCEs.

GP regression is utilized to identify an unknown function $\zeta : \mathbb{R}^{n_\theta} \to \mathbb{R}$ from data. GPs are a generalization of the multivariate normal distribution and can be viewed as distributions over functions. These can hence be used as prior functions in a Bayesian framework. The posterior update of this prior then gives us the required function approximation. A GP is fully described by a mean function $m(\cdot)$ and a covariance function $k(\cdot,\cdot)$ as follows:

$$m(\boldsymbol{\theta}) := \mathbb{E}_\zeta\left[\zeta(\boldsymbol{\theta})\right] \tag{6.1a}$$

$$k(\boldsymbol{\theta},\boldsymbol{\theta}') := \mathbb{E}_\zeta\left[(\zeta(\boldsymbol{\theta}) - m(\boldsymbol{\theta}))(\zeta(\boldsymbol{\theta}') - m(\boldsymbol{\theta}'))\right] \tag{6.1b}$$

where $\boldsymbol{\theta},\boldsymbol{\theta}'$ are arbitrary inputs and $\mathbb{E}_\zeta[\cdot]$ denotes the expectation taken over the function space $\zeta(\cdot)$. The mean function can be seen as the "average" shape of the function, while the covariance function defines the covariance between any two output values at their corresponding inputs.

The prior is specified by the mean function and the covariance function, which need to be chosen based on the *prior knowledge* of the function to be inferred. The mean function can be chosen as any function, but in general should be chosen close to the function to be learnt. In this work we propose to use as mean function a linear regression term as in universal Kriging [136]:

$$m(\boldsymbol{\theta}) := \sum_{i=1}^{n_\phi} \beta_i \phi_i(\boldsymbol{\theta}) := \boldsymbol{\beta}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{\theta}) \tag{6.2}$$

where $\boldsymbol{\beta} \in \mathbb{R}^{n_\phi}$ is a vector containing n_ϕ trend coefficients β_i and $\phi_i : \mathbb{R}^{n_\theta} \to \mathbb{R}$ are a set of basis functions collected in $\boldsymbol{\phi}(\boldsymbol{\theta}) = [\phi_1(\boldsymbol{\theta}), \ldots, \phi_{n_\phi}(\boldsymbol{\theta})]^\mathsf{T}$. The exact choice of the mean function is motivated by the noise assumed on $\boldsymbol{\theta}$, which in this paper is for $\boldsymbol{\theta}$ to follow a standard normal distribution with zero mean and unit variance, i.e. $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The mean function in our case are then given by multivariate Hermite polynomials. This selection of the mean function is motivated by their successful use as PCEs, see [136, 199, 186] for more information. These can be expressed as a tensor product of univariate Hermite polynomials of the components of $\boldsymbol{\theta}$:

$$\boldsymbol{\phi}_{\boldsymbol{\alpha}} = \prod_{i=1}^{n_\theta} \phi_{\alpha_i}(\theta_i) \tag{6.3}$$

where $\phi_{\alpha_i} : \mathbb{R} \to \mathbb{R}$ are univariate polynomials of θ_i of order α_i. The multidimensional index $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{n_\theta}]$ is hence used to define the degree of each univariate polynomial and the total order of the multivariate polynomial $\phi_{\boldsymbol{\alpha}}$ is consequently given as $|\boldsymbol{\alpha}| = \sum_{i=1}^{n_\theta} \alpha_i$.

The univariate polynomials ϕ_{α_i} are chosen to satisfy an orthogonality property according to the probability distribution of θ_i, which in our case for standard normal distributions leads to Hermite polynomials:

$$\phi_{\alpha_i}(\theta_i) = (-1)^{\alpha_i} \exp\left(\frac{1}{2}\theta_i^2\right) \frac{d^{\alpha_i}}{d\theta_i^{\alpha_i}} \exp\left(-\frac{1}{2}\theta_i^2\right) \tag{6.4}$$

Keeping all polynomial terms up to a total order of m leads then to the following expression for the mean function in Equation 6.2:

$$m(\boldsymbol{\theta}) = \sum_{0 \leq |\boldsymbol{\alpha}| \leq m} \beta_{\boldsymbol{\alpha}} \phi_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) = \boldsymbol{\beta}^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{\theta}) \tag{6.5}$$

where $\boldsymbol{\beta} \in \mathbb{R}^{n_\phi}$ and $\boldsymbol{\phi}(\boldsymbol{\theta}) : \mathbb{R}^{n_\theta} \to \mathbb{R}^{n_\phi}$ are vectors of the coefficients and polynomials of the truncated expansion respectively. The truncated series consists of $n_\phi = \frac{(n_\theta + m)!}{n_\theta! m!}$ terms. Note the number of terms grows exponentially with the input dimension of $\boldsymbol{\theta}$ and the truncation order of the polynomials.

For the covariance function we utilise the anisotropic squared-exponential (SE) [220]:

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \alpha^2 r(\boldsymbol{\theta}, \boldsymbol{\theta}'), \quad r(\boldsymbol{\theta}, \boldsymbol{\theta}') = \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^\mathsf{T} \boldsymbol{\Lambda}^{-1} (\boldsymbol{\theta} - \boldsymbol{\theta}')\right) \tag{6.6}$$

where $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1^2, \ldots, \lambda_{n_\theta}^2)$ is a diagonal matrix with n_θ separate width scaling parameters λ_i for each input dimension i and α^2 is the covariance magnitude. The SE is infinitely differentiable and therefore assumes the underlying function to be smooth. In addition, the SE covariance function is stationary, i.e. $k(\boldsymbol{\theta}, \boldsymbol{\theta}') = k(\boldsymbol{\theta} - \boldsymbol{\theta}', \mathbf{0})$.

Let the hyperparameters of the GP prior defined in Equation 6.2 and Equation 6.6 be denoted as $\boldsymbol{\xi} = [\beta_1, \ldots, \beta_{n_\phi}, \alpha, \lambda_1, \ldots, \lambda_{n_\theta}]^\mathsf{T}$. By choosing the mean function and covariance function the prior is now specified, however in general the hyperparameters are unknown. We therefore need to infer these from data. The data is given as *noiseless samples* of the function $\zeta(\boldsymbol{\theta})$ at separate inputs. Given n_s such responses, let $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{n_s}]^\mathsf{T} \in \mathbb{R}^{n_s \times n_\theta}$ be a vector of the input design and $\mathbf{z} = [\zeta(\boldsymbol{\theta}_1), \ldots \zeta(\boldsymbol{\theta}_{n_s})]^\mathsf{T} \in \mathbb{R}^{n_s}$ be a vector of the corresponding function values.

Most commonly maximum likelihood estimation (MLE) is carried out to determine reasonable hyperparameter values. The log-likelihood of the observations $\mathbf{z}$ is:

$$L(\boldsymbol{\xi}) = -\frac{n_s}{2}\log(2\pi) - \frac{n_s}{2}\log(\alpha^2) - \frac{1}{2}\log(|\boldsymbol{\Sigma}_\mathbf{z}|) - \frac{\boldsymbol{\nu}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{z}^{-1}\boldsymbol{\nu}}{2\alpha^2} \tag{6.7}$$

where $[\boldsymbol{\Sigma}_\mathbf{z}]_{ij} = r(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$, $\boldsymbol{\nu} = \mathbf{z} - \mathbf{m}_\mathbf{z}$, $\mathbf{m}_\mathbf{z} = \boldsymbol{\Phi}\boldsymbol{\beta}$, and $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\boldsymbol{\theta}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{\theta}_{n_s})]^\mathsf{T} \in \mathbb{R}^{n_s \times n_\phi}$ is a matrix of the regression terms of the mean function in Equation 6.5 evaluated at the inputs of the data.

By setting the derivatives with respect to α^2 and $\boldsymbol{\beta}$ to zero, the following closed-form expressions for the optimal MLE values of α^2 and $\boldsymbol{\beta}$ as functions of $\boldsymbol{\Sigma}_\mathbf{z}$ and $\mathbf{z}$ can be determined [136]:

$$\hat{\boldsymbol{\beta}} = \mathbf{a}^\mathsf{T}\mathbf{z}, \quad \mathbf{a} = (\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{z}^{-1}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{z}^{-1} \tag{6.8a}$$

$$\hat{\alpha}^2 = \frac{\boldsymbol{\nu}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{z}^{-1}\boldsymbol{\nu}}{n_s} \tag{6.8b}$$

As pointed out in [36] the evaluation of the scaling parameters λ_i is too expensive for online implementations and will therefore be fixed in this work. This will however lead to a worse fit of the GP and hence a larger uncertainty with regards to the model fit. We show two different approaches for fixing this parameter in this paper. A simple but effective heuristic has been suggested in [124], where all the width scales are fixed to the median of all pairwise euclidean distances in the data matrix $\boldsymbol{\Theta}$:

$$\hat{\lambda}_i = \text{median}(||\boldsymbol{\theta}_i - \boldsymbol{\theta}_j||_2) \quad \forall i \in \{1, \ldots, n_\boldsymbol{\theta}\} \tag{6.9}$$

While this in general can lead to good solutions, it ignores the response values $\mathbf{z}$ and sets all λ_i to the same value. In the GP-based Kalman filter the width scaling parameters are fixed instead using qualitative reasoning on the *importance* of the inputs on the output from $\zeta(\cdot)$ [215]. A small λ_i corresponds to an important input dimension θ_i to the value of $\zeta(\cdot)$, while a large value conversely indicates less significance of this input dimension. Often in these applications it is simple to generate several *representative* datasets $\mathbf{z}$ *offline* and from these we can obtain optimal scaling values as follows [95]:

$$\hat{\boldsymbol{\lambda}} = \arg\max \left[-\frac{n_s}{2}\log(\hat{\alpha}^2) - \frac{1}{2}\log(|\boldsymbol{\Sigma}_\mathbf{z}|) \right] \tag{6.10}$$

From these different values we can then choose values for $\hat{\boldsymbol{\lambda}}$ that account for the *importance* of the different inputs. Let the corresponding scaling matrix be given by $\hat{\boldsymbol{\Lambda}} = \text{diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_{n_\boldsymbol{\theta}})$.

Once the hyperparameters are fixed the posterior GP is utilised to obtain predictions and corresponding uncertainty values. The posterior distribution is given by the prior distribution taking the observations $\mathbf{z}$ into account. Due to the GP prior assumptions, the observations follow a multivariate Gaussian distribution. Similarly, the value of the latent function at an arbitrary input $\boldsymbol{\theta}$ also follows a Gaussian distribution. The conditional distribution of $\zeta(\boldsymbol{\theta})$ given the observations $\mathbf{z}$ can be stated as [231]:

$$\zeta(\boldsymbol{\theta})|\mathbf{z} \sim \mathcal{N}\left(m_\zeta(\boldsymbol{\theta})|\mathbf{z}, \sigma_\zeta^2(\boldsymbol{\theta})|\mathbf{z}\right) \tag{6.11a}$$

$$m_\zeta(\boldsymbol{\theta})|\mathbf{z} = m(\boldsymbol{\theta}) + \mathbf{r}_{\zeta,\mathbf{z}}^\mathsf{T}(\boldsymbol{\theta})\boldsymbol{\Sigma}_\mathbf{z}^{-1}(\mathbf{z} - \mathbf{m}_\mathbf{z}) \tag{6.11b}$$

$$\sigma_\zeta^2(\boldsymbol{\theta})|\mathbf{z} = \hat{\alpha}^2\left(1 - \boldsymbol{\kappa}_{\zeta,\mathbf{z}}^\mathsf{T}(\boldsymbol{\theta})\mathbf{K}^{-1}\boldsymbol{\kappa}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\right) \tag{6.11c}$$

where $m_\zeta(\boldsymbol{\theta})|\mathbf{z}$ and $\sigma_\zeta^2(\boldsymbol{\theta})|\mathbf{z}$ are the mean and variance function of $\zeta(\boldsymbol{\theta})|\mathbf{z}$ at an arbitrary input $\boldsymbol{\theta}$ given the observations $\mathbf{z}$, $\mathbf{r}_{\zeta,\mathbf{z}}(\boldsymbol{\theta}) = [r(\boldsymbol{\theta},\boldsymbol{\theta}_1),\ldots,r(\boldsymbol{\theta},\boldsymbol{\theta}_{n_\mathbf{z}})]^\mathsf{T}$, and $\boldsymbol{\kappa}_{\zeta,\mathbf{z}}(\boldsymbol{\theta}) = [\boldsymbol{\phi}^\mathsf{T}(\boldsymbol{\theta}), \mathbf{r}_{\zeta,\mathbf{z}}^\mathsf{T}(\boldsymbol{\theta})]^\mathsf{T}$, and $\mathbf{K} = \begin{bmatrix} \mathbf{0} & \boldsymbol{\Phi}^\mathsf{T} \\ \boldsymbol{\Phi} & \boldsymbol{\Sigma}_\mathbf{z} \end{bmatrix}$. The mean $m_\zeta(\boldsymbol{\theta})|\mathbf{z}$ can be seen as the best-estimate of $\zeta(\boldsymbol{\theta})|\mathbf{z}$, while the variance $\sigma_\zeta^2(\boldsymbol{\theta})|\mathbf{z}$ can be viewed as a measure of uncertainty of this prediction.

An example of a GP prior and posterior is shown in Figure 6.1. Firstly, it can be seen that the posterior has significantly lower uncertainty than the prior due to the data available, especially close to the data points. Secondly, it can be seen from the samples that the SE covariance function yields smooth functions.

6.3 Posterior mean and variance estimates from GPPCE

So far we have assumed that the input $\boldsymbol{\theta}$ is deterministic, often however the input $\boldsymbol{\theta}$ is given by a probability distribution. The aim of this section is to use the GP posterior introduced in Section 6.2 to estimate the mean and variance of $\zeta(\boldsymbol{\theta})|\mathbf{z}$ given that $\boldsymbol{\theta}$ follows a standard normal distribution. In particular, the case of Gaussian distributed inputs has been addressed extensively due to its importance when using GP state space models for multi-step ahead predictions [100].

It is possible to give equations for the exact mean and variance of $\zeta(\boldsymbol{\theta})|\mathbf{z}$ for certain choices of mean and covariance function, which were made in this work. The law of iterated expectations can be used to find the exact posterior mean and variance as follows [217]:

$$\mathbb{E}_{\boldsymbol{\theta}}[\zeta(\boldsymbol{\theta})|\mathbf{z}] = \mathbb{E}_{\boldsymbol{\theta}}[\mathbb{E}_\zeta[\zeta(\boldsymbol{\theta})|\mathbf{z},\boldsymbol{\theta}]] = \mathbb{E}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})|\mathbf{z}] \tag{6.12a}$$

$$\mathbb{V}_{\boldsymbol{\theta}}[\zeta(\boldsymbol{\theta})|\mathbf{z}] = \mathbb{E}_{\boldsymbol{\theta}}[\mathbb{V}_\zeta[\zeta(\boldsymbol{\theta})|\mathbf{z},\boldsymbol{\theta}]] + \mathbb{V}_{\boldsymbol{\theta}}[\mathbb{E}_\zeta[\zeta(\boldsymbol{\theta})|\mathbf{z},\boldsymbol{\theta}]] = \tag{6.12b}$$
$$\mathbb{E}_{\boldsymbol{\theta}}[\sigma_\zeta^2(\boldsymbol{\theta})|\mathbf{z}] + \mathbb{V}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})|\mathbf{z}]$$

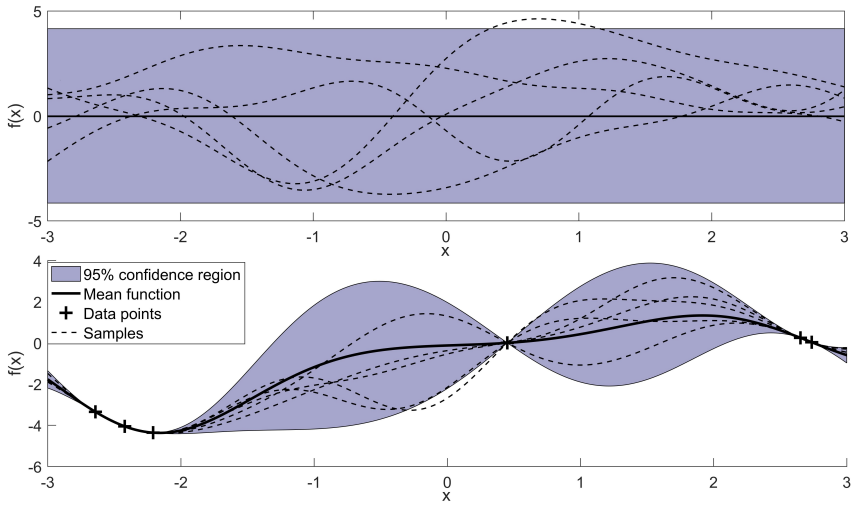


Figure 6.1: Illustration of a GP prior is shown on the top, while the corresponding GP posterior is shown below, updated with 8 observations of a one dimensional function. The prior has a mean function of 0 and a SE kernel given by Equation 6.6. It can be observed that for the GP posterior points close to the data have low uncertainty, while data far away from the observations have significantly higher uncertainty.

where $m_\zeta(\boldsymbol{\theta})|\mathbf{z}$ and $\sigma^2_\zeta(\boldsymbol{\theta})|\mathbf{z}$ are given in Equation 6.11b and Equation 6.11c respectively. Note this marks a major advantage of GPPCE modelling, since this leads to analytically tractable expressions for the expectation and variance with respect to the function $\zeta(\cdot)|\mathbf{z}$ given the observations $\mathbf{z}$.

To evaluate the posterior mean and variance we require expressions for the terms: $\mathbb{E}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})|\mathbf{z}]$, $\mathbb{E}_{\boldsymbol{\theta}}[\sigma^2_\zeta(\boldsymbol{\theta})|\mathbf{z}]$, and $\mathbb{V}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})|\mathbf{z}]$. By substituting the definitions of $m_\zeta(\boldsymbol{\theta})|\mathbf{z}$ and $\sigma^2_\zeta(\boldsymbol{\theta})|\mathbf{z}$ in Equation 6.11, we arrive at:

$$\mathbb{E}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})|\mathbf{z}] = \mathbb{E}_{\boldsymbol{\theta}}[m(\boldsymbol{\theta})] + \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbf{r}^\mathsf{T}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\right]\boldsymbol{\Sigma}_\mathbf{z}^{-1}\mathbf{v} \tag{6.13a}$$

$$\mathbb{E}_{\boldsymbol{\theta}}[\sigma^2_\zeta(\boldsymbol{\theta})|\mathbf{z}] = \hat{\alpha}^2\left(1 - \mathrm{tr}\left(\mathbb{E}_{\boldsymbol{\theta}}\left[\boldsymbol{\kappa}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\boldsymbol{\kappa}^\mathsf{T}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\right]\mathbf{K}^{-1}\right)\right) \tag{6.13b}$$

$$\begin{aligned}\mathbb{V}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})|\mathbf{z}] = \mathbb{E}_{\boldsymbol{\theta}}\left[m(\boldsymbol{\theta})^2\right] + 2\mathbb{E}_{\boldsymbol{\theta}}\left[m(\boldsymbol{\theta})\mathbf{r}^\mathsf{T}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\right]\boldsymbol{\Sigma}_\mathbf{z}^{-1}\mathbf{v} \\ + \mathbf{v}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{z}^{-1}\mathbb{E}_{\boldsymbol{\theta}}\left[\mathbf{r}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\mathbf{r}^\mathsf{T}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\right]\boldsymbol{\Sigma}_\mathbf{z}^{-1}\mathbf{v} - \left(\mathbb{E}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})]\right)^2\end{aligned} \tag{6.13c}$$

Note these expressions are given by a series of expectations and variances on the covariance function and mean function. The idea here is to choose these such that the integrals given above can be evaluated exactly. The expressions for the

expectations can be found in Section 6.10. Substituting these values into Equation 6.13:

$$\mathbb{E}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})|\mathbf{z}] = \mu_m + \boldsymbol{\mu}_{\mathbf{r}_{\zeta,\mathbf{z}}}^\mathsf{T} \boldsymbol{\Sigma}_\mathbf{z}^{-1} \boldsymbol{\nu} \tag{6.14a}$$

$$\mathbb{E}_{\boldsymbol{\theta}}[\sigma_\zeta^2(\boldsymbol{\theta})|\mathbf{z}] = \hat{\alpha}^2 \left(1 - \operatorname{tr}\left(\mathbf{M}_{\boldsymbol{\kappa}_{\zeta,\mathbf{z}}\boldsymbol{\kappa}_{\zeta,\mathbf{z}}^\mathsf{T}} \mathbf{K}^{-1}\right)\right) \tag{6.14b}$$

$$\mathbb{V}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})|\mathbf{z}] = \mu_{m^2} + 2\boldsymbol{\mu}_{m\mathbf{r}_{\zeta,\mathbf{z}}}^\mathsf{T} \boldsymbol{\Sigma}_\mathbf{z}^{-1} \boldsymbol{\nu} + \boldsymbol{\nu}^\mathsf{T} \boldsymbol{\Sigma}_\mathbf{z}^{-1} \mathbf{M}_{\mathbf{r}_{\zeta,\mathbf{z}}\mathbf{r}_{\zeta,\mathbf{z}}^\mathsf{T}} \boldsymbol{\Sigma}_\mathbf{z}^{-1} \boldsymbol{\nu} - \left(\mathbb{E}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})]\right)^2 \tag{6.14c}$$

6.4 Uncertainty propagation using GPPCE

In this section we outline how the GPPCE methodology can be used to efficiently propagate uncertainties through nonlinear functions. Let an arbitrary nonlinear function $\zeta(\cdot)$ be given by:

$$z = \zeta(\boldsymbol{\theta}) \tag{6.15}$$

where $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \mathbf{I})$ follows a standard normal distribution.

The aim of this section is to estimate the mean and variance of z using GPPCE as introduced in Section 6.2. The estimate should be as computationally cheap as possible, since it is used *online*. Therefore, the section is divided into two parts: "*Offline computation*" and "*Online computation*". "*Offline computation*" outlines terms that do not directly depend on the response values z and can hence be determined offline based on the sample design of $\boldsymbol{\theta}$ alone to save significant computational time, while "*Online computation*" shows how to obtain the posterior mean and variance estimates given the pre-computed terms.

6.4.1 Offline computation

First we need to decide on the number of samples n_s for the approximation. Thereafter, a sample design denoted by $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{n_s}]^\mathsf{T} \in \mathbb{R}^{n_s \times n_{\boldsymbol{\theta}}}$ needs to be chosen. This sample design should lead to a reasonable function approximation of $\zeta(\cdot)$ in regions that have significant probability densities. Regions of diminishing probability densities do not require good function approximations, since they do not contribute to the expectation values. This is accomplished by ensuring that the sample design is generated according to a standard normal distribution. The most obvious, but arguably worst approach uses crude MC to obtain these samples, which however may lead to poor convergence. Alternatively, so-called space-filling designs can be used to generate the necessary points in a unit hypercube $[0,1]^{n_{\boldsymbol{\theta}}}$. These sample designs can then be converted to follow a standard normal distribution by using the probit function, see for example [240]. Popular

space-filling sampling designs include Latin hypercube designs [240], or Quasi MC designs, such as Sobol [238].

Given this sample design $\boldsymbol{\Theta}$ we next set the hyperparameters λ_i according to Section 6.2, which gives us $\hat{\boldsymbol{\Lambda}}$. These are treated differently than the other hyperparameters, since they do not have a closed-form solution and hence cannot be evaluated *online* without incurring large computational times.

Next there are several terms that only depend on the sample design $\boldsymbol{\Theta}$ and the hyperparameters $\hat{\lambda}_i$, which can hence be pre-computed. These are as follows:

$$\mathbf{a} = \left(\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{z}^{-1}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{z}^{-1} \tag{6.16a}$$

$$\mathbf{b} = \boldsymbol{\mu}_{\mathbf{r}_{\zeta,\mathbf{z}}}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{z}^{-1} \tag{6.16b}$$

$$c = \operatorname{tr}\left(\mathbf{M}_{\boldsymbol{\kappa}_{\zeta,\mathbf{z}}\boldsymbol{\kappa}_{\zeta,\mathbf{z}}^\mathsf{T}}\mathbf{K}^{-1}\right) \tag{6.16c}$$

$$\mathbf{d} = \boldsymbol{\mu}_{m\mathbf{r}_{\zeta,\mathbf{z}}}^\mathsf{T}\boldsymbol{\Sigma}_\mathbf{z}^{-1} \tag{6.16d}$$

$$\mathbf{E} = \frac{\boldsymbol{\Sigma}_\mathbf{z}^{-1}}{n_s} \tag{6.16e}$$

$$\mathbf{F} = \boldsymbol{\Sigma}_\mathbf{z}^{-1}\mathbf{M}_{\mathbf{r}_{\zeta,\mathbf{z}}\mathbf{r}_{\zeta,\mathbf{z}}^\mathsf{T}}\boldsymbol{\Sigma}_\mathbf{z}^{-1} \tag{6.16f}$$

where the required parameters can be evaluated using their definitions in Section 6.2 and Section 6.10 for the expectations represented by $\boldsymbol{\mu}_{\mathbf{r}_{\zeta,\mathbf{z}}}$, $\mathbf{M}_{\boldsymbol{\kappa}_{\zeta,\mathbf{z}}\boldsymbol{\kappa}_{\zeta,\mathbf{z}}^\mathsf{T}}$, $\boldsymbol{\mu}_{m\mathbf{r}_{\zeta,\mathbf{z}}}$, and $\mathbf{M}_{\mathbf{r}_{\zeta,\mathbf{z}}\mathbf{r}_{\zeta,\mathbf{z}}^\mathsf{T}}$.

6.4.2 Online computation

Given these pre-computed values in Equation 6.16 we then have efficient formulas to estimate mean and variance using the posterior GP. For this we need to evaluate the function $\zeta()$ at the points defined in the sample design $\boldsymbol{\Theta}$ according to Equation 6.15, which gives us the response vector $\mathbf{z} = [\zeta(\boldsymbol{\theta}_1), \ldots, \zeta(\boldsymbol{\theta}_{n_\boldsymbol{\theta}})]^\mathsf{T}$. The posterior GP then represents a fitted model using the data in $\mathbf{z}$, for which we have efficient formulas to obtain the mean and variance. Note that this is the first time we actually use the function $\zeta()$ and hence can be repeated for different functions $\zeta()$ without incurring too high computational costs. Based on the pre-computed values we obtain the following estimates for the mean and variance of the nonlinear transformation defined in Equation 6.15:

$$\hat{\boldsymbol{\beta}} = \mathbf{a}^\mathsf{T}\mathbf{z} \tag{6.17a}$$

$$\boldsymbol{\nu} = \mathbf{z} - \hat{\boldsymbol{\beta}}\boldsymbol{\Phi}, \quad \boldsymbol{\Phi} = [\boldsymbol{\phi}(\boldsymbol{\theta}_1), \ldots, \boldsymbol{\phi}(\boldsymbol{\theta}_{n_s})]^\mathsf{T} \tag{6.17b}$$

$$\hat{\alpha}^2 = \boldsymbol{\nu}^\mathsf{T}\mathbf{E}\boldsymbol{\nu} \tag{6.17c}$$

$$\mathbb{E}_{\boldsymbol{\theta}}[\zeta(\boldsymbol{\theta})] \approx \mu^{z}_{GP}(\cdot;\boldsymbol{\tau}) = \mu_m + \mathbf{b}\boldsymbol{\nu} \tag{6.17d}$$

$$\mathbb{V}_{\boldsymbol{\theta}}[\zeta(\boldsymbol{\theta})] \approx \sigma^2_{m_\zeta}(\mathbf{z};\boldsymbol{\tau}) = \mathbb{V}_{\boldsymbol{\theta}}[m_\zeta(\boldsymbol{\theta})|\mathbf{z}] = \mu_{m^2} + 2\mathbf{d}\boldsymbol{\nu} + \boldsymbol{\nu}^\top\mathbf{F}\boldsymbol{\nu} - (\mu_m + \mathbf{b}\boldsymbol{\nu})^2 \tag{6.17e}$$

$$\sigma^2_{GP}(\mathbf{z};\boldsymbol{\tau}) = \mathbb{V}_{\boldsymbol{\theta}}[\zeta(\boldsymbol{\theta})|\mathbf{z}] = \sigma^2_{m_\zeta}(\mathbf{z};\boldsymbol{\tau}) + \hat{\alpha}^2(1-c) \tag{6.17f}$$

where $\mu^{z}_{GP}(\cdot;\boldsymbol{\tau})$ is the mean estimate for the GPPCE, $\sigma^2_{m_\zeta}(\cdot;\boldsymbol{\tau})$ is the variance estimate of the mean function from the GPPCE and can be seen as the best-estimate of the *true* variance from the GPPCE, while $\sigma^2_{GP}(\cdot;\boldsymbol{\tau})$ is the variance of the GP accounting for the uncertainty due to using only a *finite sample* approximation. The variance estimate $\sigma^2_{GP}(\cdot;\boldsymbol{\tau})$ therefore has a larger variance by adding the term $\hat{\alpha}^2(1-c)$. The variable $\boldsymbol{\tau} = \{\boldsymbol{\Theta}, \hat{\boldsymbol{\lambda}}, n_{PCE}\}$ summarizes the different choices made, that define the mean and variance estimate outside of the direct dependency on the training data $\mathbf{z}$. Firstly, the sample design $\boldsymbol{\Theta}$ and the truncation order of the PCE n_{PCE} should be chosen. Thereafter, the hyperparameters $\hat{\boldsymbol{\lambda}}$ need to be determined either using heuristics or available data as shown in Section 6.2. From this the terms in Equation 6.16 can be pre-computed and used for different values of $\mathbf{z}$ to obtain estimates of the mean and variance.

6.5 Problem formulation

We consider a general discrete-time stochastic nonlinear dynamic equation system with parametric uncertainties:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\theta}), \quad \mathbf{x}_0 = \hat{\mathbf{x}}_0 \tag{6.18}$$

where t is the discrete time, $\mathbf{x}_t \in \mathbb{R}^{n_\mathbf{x}}$ represents the states, $\mathbf{u}_t \in \mathbb{R}^{n_\mathbf{u}}$ denotes the control inputs, $\boldsymbol{\theta} \in \mathbb{R}^{n_{\boldsymbol{\theta}}}$ represents time-invariant parametric uncertainties, and $\mathbf{f} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_{\boldsymbol{\theta}}} \to \mathbb{R}^{n_\mathbf{x}}$ is the nonlinear dynamic equation system. The parametric uncertainties $\boldsymbol{\theta}$ are assumed to follow a standard normal distribution, i.e. $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note this is not restrictive, since the uncertain model parameters can be parametrized in terms of $\boldsymbol{\theta}$ and this way obtain the required probability distribution, see for example [45]. The initial condition is given by a known value $\hat{\mathbf{x}}_0$.

Given the dynamic system defined in Equation 6.18 we aim to minimize a finite horizon objective function:

$$J(N, \hat{\mathbf{x}}_0, \boldsymbol{\theta}, \mathbf{U}_N) = \mathbb{E}[J^d(N, \hat{\mathbf{x}}_0, \boldsymbol{\theta}, \mathbf{U}_N)] + \omega \mathrm{Var}[J^d(N, \hat{\mathbf{x}}_0, \boldsymbol{\theta}, \mathbf{U}_N)] \tag{6.19a}$$

$$J^d(N, \hat{\mathbf{x}}_0, \boldsymbol{\theta}, \mathbf{U}_N) = \mathcal{M}(\mathbf{x}_N) + \sum_{t=0}^{N-1} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) \tag{6.19b}$$

where N is the time horizon, $\mathcal{M} : \mathbb{R}^{n_\mathbf{x}} \rightarrow \mathbb{R}$ denotes the Mayer term, $\mathcal{L} : \mathbb{R}^{n_\mathbf{x} \times n_\mathbf{u}} \rightarrow \mathbb{R}$ represents the Lagrange term, and $\mathbf{U}_N = [\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}] \in \mathbb{R}^{n_\mathbf{u} \times N}$ are the control actions that need to be determined.

The objective is taken as the expectation with a weighted variance added to it of a nonlinear function, i.e. the aim is to minimize the objective in Equation 6.19b given the dynamic system in Equation 6.18. The weighted variance can be exploited to penalize excessive uncertainty on the objective values. The case-in-point of this paper is the control of batch processes. The objective to be minimized generally depends on the final product at the end of the batch, which leads to a shrinking horizon NMPC formulation [184]. The objective depending on the final state is represented by the Mayer term.

The control problem is subject to hard constraints on the control inputs expressed by the set $\mathbb{U}$. In addition, the control problem is subject to both nonlinear path chance constraints and terminal chance constraints. For batch processes common path chance constraints are given by safety limits, such as upper bounds on the adiabatic temperature or reactor pressure. Terminal chance constraints on the other hand often describe a minimum product quality to be reached. The constraints are formulated as follows:

$$\mathbb{P}[g_j(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\theta}) \leq 0] \geq 1 - \epsilon \qquad \forall (t, j) \in \{1, \ldots, N\} \times \{1, \ldots, n_g\} \tag{6.20a}$$

$$\mathbb{P}[g_j^N(\mathbf{x}_N, \boldsymbol{\theta}) \leq 0] \geq 1 - \epsilon \qquad \forall j \in \{1, \ldots, n_g^N\} \tag{6.20b}$$

$$\mathbf{u}_t \in \mathbb{U} \qquad \forall t \in \{0, \ldots, N-1\} \tag{6.20c}$$

where $g_j : \mathbb{R}^{n_\mathbf{x} \times n_\mathbf{u} \times n_{\boldsymbol{\theta}}} \rightarrow \mathbb{R}$ are the path constraint functions, $g_j^N : \mathbb{R}^{n_\mathbf{x} \times n_{\boldsymbol{\theta}}} \rightarrow \mathbb{R}$ are the terminal constraint functions, and ϵ is the probability of constraint violations.

The constraints are given as chance constraints due to the presence of the parametric uncertainties $\boldsymbol{\theta}$. Each constraint in Equations 6.20a and Equation 6.20b should be violated by at most a low probability ϵ despite the stochastic uncertainties present to maintain feasibility.

6.6 GPPCE stochastic nonlinear model predictive control

In this section we introduce the GPPCE based SNMPC algorithm to solve the problem outlined in Section 6.5 employing the dynamic equation system in Equation 6.18. Assume we are at time t and we have a full state measurement $\mathbf{x}_t$ of the current state. The GPPCE equations are utilised in the optimization algorithm to obtain accurate estimates of the mean and variances of both objective and constraint functions to approximate the probabilistic objective and chance constraints.

6.6.1 Uncertainty propagation

For the uncertainty propagation we apply the results outlined in Section 6.4. Our aim is to approximate the mean and variance of the objective and chance constraints to formulate the GPPCE SNMPC optimization problem. For this we first create a sample design $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{n_s}]^\mathsf{T}$. Each realization of $\boldsymbol{\theta}$ then represents its own nonlinear dynamic equation system:

$$\tilde{\mathbf{x}}_{k+1}^{(s)} = \mathbf{f}(\tilde{\mathbf{x}}_k^{(s)}, \tilde{\mathbf{u}}_k^{(s)}, \boldsymbol{\theta}_s), \quad \tilde{\mathbf{x}}_0^{(s)} = \hat{\mathbf{x}}_0 \qquad \forall s \in \{1, \dots, n_s\} \tag{6.21}$$

where $\tilde{\mathbf{x}}_k^{(s)}$ and $\tilde{\mathbf{u}}_k^{(s)}$ denotes the states and control inputs for the sample s.

Using Equation 6.21 we then have separate state values for each $\boldsymbol{\theta}$, for which we obtain different values for the constraint functions and objective:

$$\mathbf{z}_{J^d} = [J^d(N, \hat{\mathbf{x}}_0, \boldsymbol{\theta}_1, \mathbf{U}_N), \dots, J^d(N, \hat{\mathbf{x}}_0, \boldsymbol{\theta}_{n_s}, \mathbf{U}_N)]^\mathsf{T} \tag{6.22a}$$

$$\mathbf{z}_{g_j}^{(k)} = [g_j(\tilde{\mathbf{x}}_k^{(1)}, \tilde{\mathbf{u}}_k^{(1)}, \boldsymbol{\theta}_1), \dots, g_j(\tilde{\mathbf{x}}_k^{(n_s)}, \tilde{\mathbf{u}}_k^{(n_s)}, \boldsymbol{\theta}_{n_s})]^\mathsf{T}$$

$$\forall (k, j) \in \{1, \dots, N\} \times \{1, \dots, n_g\} \tag{6.22b}$$

$$\mathbf{z}_{g_j^N} = [g_j^N(\tilde{\mathbf{x}}_N^{(1)}, \boldsymbol{\theta}_1), \dots, g_j^N(\tilde{\mathbf{x}}_N^{(n_s)}, \boldsymbol{\theta}_{n_s})]^\mathsf{T} \quad \forall j \in \{1, \dots, n_g^N\} \tag{6.22c}$$

Using the sample design $\boldsymbol{\Theta}$ we define the mean and variance estimate GPPCE functions by determining the hyperparameters $\hat{\boldsymbol{\lambda}}$ and setting the truncation order for the PCE n_{PCE}, which defines $\boldsymbol{\tau} = \{\boldsymbol{\Theta}, \hat{\boldsymbol{\lambda}}, n_{PCE}\}$. The hyperparameters $\hat{\boldsymbol{\lambda}}$ in general will be set to different values for the constraint and objective functions, since the inputs have varied importance. The posterior mean and variance estimates of the constraints and objective are then given by the mean and variance function as defined in Equation 6.17.

The principle of the GPPCE estimates is shown in Figure 6.2. Each sample of $\boldsymbol{\theta}$ corresponds to a separate trajectory according to Equation 6.21, which is shown by the red lines. Each of these trajectories then leads to distinct values of the state $\mathbf{x}$ at each discrete-time k shown by the red markers. These in turn are then transformed using the objective and constraint definitions to obtain the "data" required for the mean and variance estimates, which leads to the data vectors shown in Equation 6.22. This is highlighted by the arrows for a particular constraint. Thereafter, GP regression is applied leading to the blue line. The closed-form expressions from Equation 6.17 are thereafter applied, which return the exact mean and variance of the GP surrogate. Note that for each iteration of the control vector $\mathbf{U}_N$ this procedure needs to be repeated, i.e. for each step in the optimization algorithm. GPs are probabilistic models and hence also include a confidence region, which can be accounted for in the variance estimate.

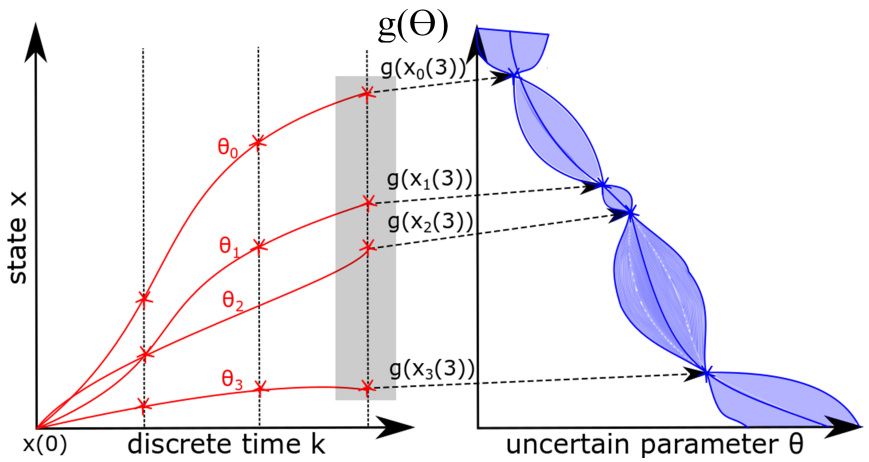


Figure 6.2: Illustration of the mean and variance estimate from the GPPCE algorithm. The red trajectories on the left-hand side graph represent different realizations of $\boldsymbol{\theta}$, which each lead to different state values. These are then transformed using a constraint function $g(\cdot)$ to obtain a series of values, which are used as data to build a GP model as shown on the right-hand side graph.

6.6.2 Chance constraint approximation

In Section 6.6.1 we show how to obtain mean and variance estimates employing GPPCE for the constraint and objective function. These can then be utilised directly to approximate the objective in Equation 6.19b. The chance constraints in Equation 6.20a and Equation 6.20*b* on the other hand are robustly reformulated using Chebychev's inequality, since the exact evaluation of the chance constraint in Equation 6.23 is notoriously difficult due to the integral definition of the probability function.

Assume we have a chance constraints on an arbitrary random variable γ:

$$\mathbb{P}\{\gamma \leq 0\} \geq 1 - \epsilon \tag{6.23}$$

Chebychev's inequality can then be used as follows to robustly reformulate the probability constraint in Equation 6.23 [177]:

$$\mu_\gamma + \kappa_\epsilon \sqrt{\sigma_\gamma^2} \leq 0, \quad \kappa_\epsilon = \sqrt{\frac{1-\epsilon}{\epsilon}} \tag{6.24}$$

where μ_γ and σ_γ^2 are the mean and variance of γ respectively. The robust reformulation now only requires the mean and standard deviation of γ.

Now applying the robust reformulation introduced above and using the mean and variance estimates from the GPPCE we can reformulate the chance constraints in Equation 6.20a and Equation 6.20b as follows:

$$\mu_{GP}(\mathbf{z}_{g_j}^{(k)}; \boldsymbol{\tau}_{g_j}) + \kappa_\epsilon \sqrt{\sigma_{GP}^2(\mathbf{z}_{g_j}^{(k)}; \boldsymbol{\tau}_{g_j})} \leq 0 \quad \forall (k, j) \in \{1, \dots, N\} \times \{1, \dots, n_g\} \tag{6.25a}$$

$$\mu_{GP}(\mathbf{z}_{g_j^N}; \boldsymbol{\tau}_{g_j^N}) + \kappa_\epsilon \sqrt{\sigma_{GP}^2(\mathbf{z}_{g_j^N}; \boldsymbol{\tau}_{g_j^N})} \leq 0 \quad \forall j \in \{1, \dots, n_g^N\} \tag{6.25b}$$

where $\mu_{GP}(\mathbf{z}_{g_j}^{(k)}; \boldsymbol{\tau}_{g_j})$ and $\mu_{GP}(\mathbf{z}_{g_j^N}; \boldsymbol{\tau}_{g_j^N})$ are the GPPCE mean estimates of the constraints $g_j^{(k)}$ and g_j^N respectively, while $\sigma_{GP}^2(\mathbf{z}_{g_j}^{(k)}; \boldsymbol{\tau}_{g_j})$ and $\sigma_{GP}^2(\mathbf{z}_{g_j^N}; \boldsymbol{\tau}_{g_j^N})$ represent the GPPCE variance estimates as introduced in Section 6.4. Note we are using the GP variance that is larger, since it accounts for the error using only *finite number* of samples.

6.6.3 GPPCE SNMPC formulation

In this section we formulate the stochastic optimal control problem to be solved using the mean and variance approximations of the objective and constraint functions as introduced in Section 6.6.1. We optimize over the control actions $\mathbf{u}_k$ given the objective and constraints defined in Section 6.5:

$$\begin{aligned}
&\underset{\tilde{\mathbf{U}}_N^{(t)}}{\text{minimize}} \quad \mu_{GP}(\mathbf{z}_{Jd}; \boldsymbol{\tau}_{Jd}) + \omega \cdot \sigma_{GP}^2(\mathbf{z}_{Jd}; \boldsymbol{\tau}_{Jd}) \\
&\text{subject to} \\
&\tilde{\mathbf{x}}_{k+1}^{(s)} = \mathbf{f}(\tilde{\mathbf{x}}_k^{(s)}, \tilde{\mathbf{u}}_k^{(s)}, \boldsymbol{\theta}_s), \quad \tilde{\mathbf{x}}_k^{(s)} = \hat{\mathbf{x}}_t && \forall (k, s) \in \{t, \dots, N\} \times \{1, \dots, n_s\} \\
&\mu_{GP}(\mathbf{z}_{g_j}^{(k)}; \boldsymbol{\tau}_{g_j}) + \kappa_\epsilon \sqrt{\sigma_{GP}^2(\mathbf{z}_{g_j}^{(k)}; \boldsymbol{\tau}_{g_j})} \leq 0 && \forall (k, j) \in \{t, \dots, N\} \times \{1, \dots, n_g\} \\
&\mu_{GP}(\mathbf{z}_{g_j^N}; \boldsymbol{\tau}_{g_j^N}) + \kappa_\epsilon \sqrt{\sigma_{GP}^2(\mathbf{z}_{g_j^N}; \boldsymbol{\tau}_{g_j^N})} \leq 0 && \forall j \in \{1, \dots, n_g^N\}
\end{aligned} \tag{6.26}$$

where $\hat{\mathbf{x}}_t$ is the current state measurement and $\tilde{\mathbf{U}}_N^{(t)} = [\tilde{\mathbf{u}}_t, \dots, \tilde{\mathbf{u}}_N]^\mathsf{T}$.

6.6.4 GPPCE SNMPC algorithm

In this section we outline the algorithm to solve the problem defined in Section 6.5 using the GPPCE SNMPC optimization problem from the previous section. At each time t we are given the current state measurement $\hat{\mathbf{x}}_t$, from which we aim to determine the best control action to take. To formulate the problem the dynamic equation system in Equation 6.18 needs to be defined together with the initial conditions $\hat{\mathbf{x}}_0$ and the time horizon N. Further, the objective function

$J^d(N, \mathbf{x}_0(\boldsymbol{\theta}), \boldsymbol{\theta}, \mathbf{U}_N)$ with the variance factor ω need to be defined, together with the input constraint set $\mathbb{U}$, path constraint functions $g_j(\cdot)$ and terminal constraint functions $g_j^N(\cdot)$. The corresponding probability of feasibility ϵ needs to be set. Next we specify $\boldsymbol{\tau}_{J^d}$, $\boldsymbol{\tau}_{g_j}$, and $\boldsymbol{\tau}_{g_j^N}$. Lastly, the terms required for the GPPCE estimator are pre-computed according to Equation 6.16. The overall algorithm is stated in Algorithm 6.1.

Algorithm 6.1: GPPCE SNMPC algorithm

Offline Computations

1. Choose time horizon N, initial condition $\hat{\mathbf{x}}_0$, stage costs $\mathcal{L}(\cdot)$ and terminal cost $\mathcal{M}(\cdot)$, variance weighting factor ω, path constraint functions $g_j(\cdot)$, terminal constraint functions $g_j^N(\cdot)$, input constraint set $\mathbb{U}$, and the chance constraint probability ϵ.
2. Specify the GPPCE estimator by setting reasonable values to $\boldsymbol{\tau}_{J^d}$, $\boldsymbol{\tau}_{g_j}$, and $\boldsymbol{\tau}_{g_j^N}$.
3. Pre-compute terms required for the GPPCE estimator given in Equation 6.16.

Online Computations

for $t = 0, \ldots, N-1$ **do**

1. Solve the SNMPC problem in Equation 6.26 with the current state $\hat{\mathbf{x}}_t$.
2. Apply the first control input of the optimal solution to the real plant.
3. Measure the state $\hat{\mathbf{x}}_t$.

6.7 Semi-batch reactor case study

The GPPCE SNMPC algorithm introduced in Section 6.6 is applied to a semi-batch reactor case study for the production of the polymer polypropylene glycol from the monomer propylene oxide (PO) as illustrated in Figure 6.3.

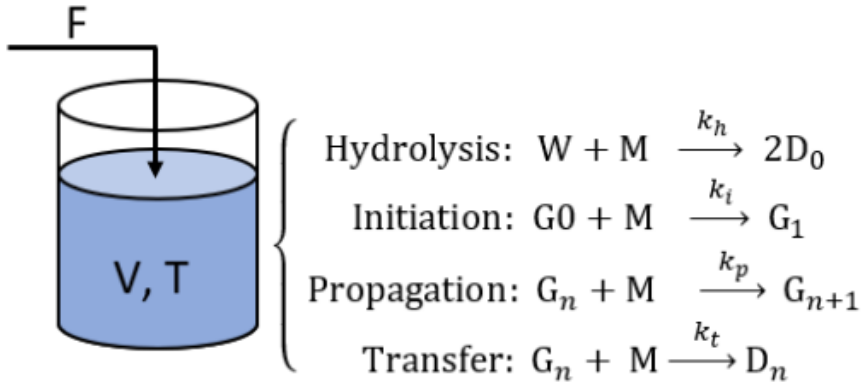


Figure 6.3: Figure summarizing the main variables of the semi-batch reactors with the main reactions taking place. F is the monomer feedrate, V and T are the volume and temperature of the liquid in the reactor respectively, W represents water, M denotes the monomer, D_n and G_n are the dormant and active product chains with length n respectively.

6.7.1 Dynamic model

For this batch process polymerization reaction an extensive model has been proposed in [193]. This model has a separate dynamic equation for each chain length separately, which we simplified using the "method of moments" [194]. The ordinary differential equation (ODEs) then describe the moments of the polymer as opposed to the amount of each specific chain length. This is often sufficient to estimate important performance criteria. In addition, an energy balance was added due to the importance of temperature control. The dynamic model consists of 7 ODEs. The dynamic model can be stated as:

$$\dot{m} = FMW_{\text{PO}}, \quad m_0 = \hat{m}_0 \tag{6.27a}$$

$$\dot{T} = \frac{(-\Delta H_p)k_p\gamma_{G0}M - UA(T - T_C)V - FMW_{\text{PO}}C_{pf}(T - T_f)V}{mC_{pb}V}, \quad T_0 = \hat{T}_0 \tag{6.27b}$$

$$\dot{W} = -\frac{k_h W M}{V}, \quad W_0 = \hat{W}_0 \tag{6.27c}$$

$$\dot{M} = F - \frac{\left(k_h W + k_i G_0 + k_p\gamma_{G0} + k_t(\gamma_{G0} + G_0)\right) M}{V}, \quad M_0 = \hat{M}_0 \tag{6.27d}$$

$$\dot{X}_0 = \frac{(2k_h W - k_i G_0)M}{V}, \quad X_{00} = \hat{X}_{00} \tag{6.27e}$$

$$\dot{\gamma}_{X0} = \frac{k_i G_0 M}{V}, \quad \gamma_{X00} = \gamma_{\hat{X}00} \tag{6.27f}$$

$$\dot{\gamma}_{X1} = \frac{(k_i G_0 + k_p\gamma_{G0})M}{V}, \quad \gamma_{X10} = \gamma_{\hat{X}10} \tag{6.27g}$$

where m is the liquid mass in kg, T is the reactor temperature in K, W is the amount of water in kmol, M is the amount of monomer in kmol, X_0 is the concentration of Methanol in kmol, γ_{X0} is the zeroth polymer moment in kmol, and γ_{X1} is the first polymer moment in kmol. F is the feed rate of the monomer in kmol/s and T_C is the cooling water temperature in K. k_p, k_h, k_i, and k_t in m^3kmol/s are the kinetic constants of the propagation, hydrolysis, initiation, and transfer reactions respectively. C_{pb} and C_{pf} are the heat capacities of the bulk liquid and the monomer feed respectively in kJ/kg/K. G_0, γ_{G0}, and γ_{G1} are the active concentrations of Methanol, the zeroth polymer moment, and the first polymer moment in kmol. V is the liquid volume in the reactor. The kinetic constants and the heat capacities are given as functions of temperature [194]:

$$k_p = A_p \exp(-E_{Ap}/RT) \tag{6.28a}$$
$$k_h = A_h \exp(-E_{Ah}/RT) \tag{6.28b}$$
$$k_i = A_i \exp(-E_{Ai}/RT) \tag{6.28c}$$
$$k_t = A_t \exp(-E_{At}/RT) \tag{6.28d}$$
$$C_{pf} = 0.92 + 8.871 \times 10^{-3} T - 3.1 \times 10^{-5} T^2 + 4.78 \times 10^{-8} T^3 \tag{6.28e}$$
$$C_{pb} = 1.1 + 2.72 \times 10^{-3} T \tag{6.28f}$$

G_0, γ_{G0}, and γ_{G1} depend on X_0, γ_{X0}, and γ_{X1} as follows:

$$G_0 = X_0 n_C/(X_0 + \gamma_{X0}) \tag{6.29a}$$
$$\gamma_{G0} = \gamma_{X0} n_C/(X_0 + \gamma_{X0}) \tag{6.29b}$$
$$\gamma_{G1} = \gamma_{X1} n_C/(X_0 + \gamma_{X0}) \tag{6.29c}$$

where n_C is the amount of catalyst in the reactor in kmol.

Three parameters in the dynamic model were assumed to be uncertain: The pre-exponential factor of the propagation reaction A_p in m^3/kmol/s, the overall heat transfer coefficient UA in kW/K, and the total amount of catalyst n_C in kmol. The remaining parameters including the initial conditions are given in Table 6.1.

Table 6.1: Parameter values for dynamic model taken from [193] and operating conditions as defined in Equations 6.27a – 6.29a.

Parameter	Value	Units	Description
MW_{PO}	58.08	kg/kmol	Molecular weight of PO
ΔH_p	−92048	kJ/kmol	Enthalpy of reaction for propagation reaction
A_h	2.4×10^8	$\mathrm{m}^3/\mathrm{kmol/s}$	Pre-exponential factor of hydrolysis kinetic constant
A_i	4.0×10^8	$\mathrm{m}^3/\mathrm{kmol/s}$	Pre-exponential factor of initiation kinetic constant
A_t	9.5×10^8	$\mathrm{m}^3/\mathrm{kmol/s}$	Pre-exponential factor of transfer kinetic constant
E_{Ap}	6.9×10^4	kJ/kmol	Activation energy of propagation reaction
E_{Ah}	8.2×10^4	kJ/kmol	Activation energy of hydrolysis reaction
E_{Ai}	7.8×10^4	kJ/kmol	Activation energy of initiation reaction
E_{At}	1.05×10^5	kJ/kmol	Activation energy of transfer reaction
R	8.314	kJ/kmol/K	Universal gas constant
$\hat{m}_0$	1.56×10^3	kg	Initial reactor mass
$\hat{T}_0$	400	K	Initial reactor temperature
$\hat{W}_0$	1.0	kmol	Initial amount of water
$\hat{M}_0$	10.0	kmol	Initial amount of monomer
$\hat{X}_{00}$	0.0	kmol	Initial amount of methanol
$\gamma_{\hat{X}00}$	10.0	kmol	Initial zeroth polymer moment
$\gamma_{\hat{X}10}$	10.0	kmol	Initial first polymer moment

The control inputs are given by the monomer feed rate F and cooling water temperature T_C. In compact form we can write $\mathbf{x} = [m, T, W, M, X_0, \gamma_{X0}, \gamma_{X1}]^\mathsf{T}$ and $\mathbf{u} = [F, T_C]^\mathsf{T}$. The uncertain model parameters are given as functions of $\boldsymbol{\theta}$, which can be used to attain complex probability distributions:

$$A_p(\boldsymbol{\theta}) = \exp\left(-5 + 0.05\theta_3 + 0.05\theta_2 + 0.03\theta_1\right) \times 10^9 + 5 \times 10^6 \tag{6.30a}$$

$$UA(\boldsymbol{\theta}) = 40 \cos\left(1.26 + 0.09\theta_3 - 0.09\theta_2 + 0.09\theta_1\right) + 40 \tag{6.30b}$$

$$n_C(\boldsymbol{\theta}) = |-0.05\theta_3 + 0.05\theta_2 - 0.05\theta_1| + 1 \tag{6.30c}$$

This allows for the uncertain parameters to attain nearly arbitrary complex probability distributions as can be seen from their respective pdfs plotted in Figure 6.4.

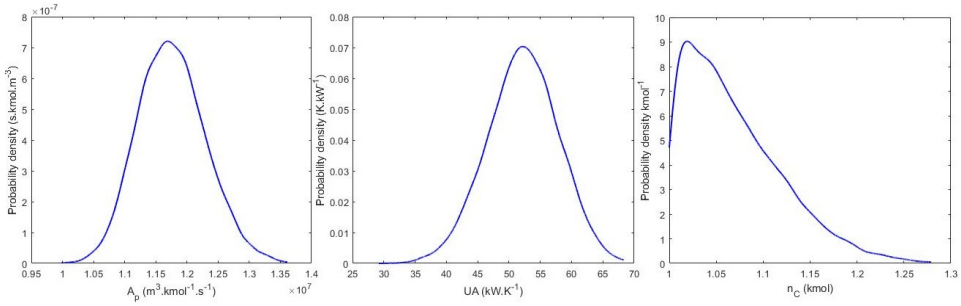


Figure 6.4: Plots of the probability density functions of the uncertain parameters (from left to right for: A_p, UA, n_C)

6.7.2 Problem set-up

The time horizon N was set to 12 with a variable continuous batch time t_{batch} with equal sampling times. The state at each discrete time $t + 1$ can be expressed as follows employing Equation 6.27a:

$$\mathbf{x}_{t+1} = \int_0^{t_{batch}/N} \bar{\mathbf{f}}(\mathbf{x}_t, \mathbf{u}_t)dt + \mathbf{x}_t \tag{6.31}$$

where $\bar{\mathbf{f}}(\cdot) = [\dot{m}, \dot{T}, \dot{W}, \dot{M}, \dot{X}_0, \dot{\gamma}_{X0}, \dot{\gamma}_{X1}]^\mathsf{T}$ as defined in Equation 6.27a.

The required discrete-time system for Equation 6.18 is then obtained using orthogonal Radau collocation. Each control interval is simulated by a polynomial with an overall degree of 5. The objective for the control algorithm is aimed to minimize the required batch time t_{batch} with a penalty on changes in the control input:

$$J^d(N, \hat{\mathbf{x}}_0, \boldsymbol{\theta}, \mathbf{U}_N) = t_{batch} + \sum_{t=1}^{N} \boldsymbol{\Delta}_{\mathbf{u}_t}^\mathsf{T} \mathbf{R} \boldsymbol{\Delta}_{\mathbf{u}_t} \tag{6.32}$$

where $\boldsymbol{\Delta}_{\mathbf{u}_t} = \mathbf{u}_t - \mathbf{u}_{t-1}$ and $\mathbf{R} = \mathrm{diag}(10^{-6}, 10^{-4})$.

The minimization is subject to two terminal constraints and a path constraint. The path constraint aims to keep the reactor temperature below 420K for safety reasons, which can be stated as follows:

$$g(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\theta}) = T - 420 \leq 0 \tag{6.33}$$

The two terminal constraints state batch product quality properties to be reached. The first terminal constraint requires the batch to reach a number average molecular weight ($NAMW$) in kg/kmol of 1500 defined as $NAMW = MW_{PO}\frac{\gamma_{X1}}{\gamma_{X1}}$. The

second terminal constraint requires the final monomer concentration to not exceed 1000ppm. These terminal constraints are:

$$g_1^N(\mathbf{x}_N, \boldsymbol{\theta}) = -MW_{PO}\frac{\gamma_{X1}}{\gamma_{X0}} + 1500 \leq 0 \tag{6.34a}$$

$$g_2^N(\mathbf{x}_N, \boldsymbol{\theta}) = 10^6 \times \frac{MW_{PO}M}{m} - 1000 \leq 0 \tag{6.34b}$$

The chance of constraint violation was to $\epsilon = 0.05$ for the constraints defined above. The control inputs are constrained as:

$$0 \leq F \leq 0.1 \tag{6.35a}$$

$$298.15 \leq T_C \leq 423.15 \tag{6.35b}$$

For the GPPCE approximation the scaling variable $\hat{\boldsymbol{\lambda}}$ was set to the following for the different constraint functions:

$$\hat{\boldsymbol{\lambda}}_g = [0.22, 0.77, 0.55] \tag{6.36a}$$

$$\hat{\boldsymbol{\lambda}}_{g_1^N} = [0.31, 0.87, 0.44] \tag{6.36b}$$

$$\hat{\boldsymbol{\lambda}}_{g_2^N} = [0.30, 0.75, 0.45] \tag{6.36c}$$

The values were determined using Equation 6.10 by generating different trajectories by setting $\mathbf{U}$ to values between its upper and lower bound. The various $\hat{\boldsymbol{\lambda}}$ values obtained then allowed us to set $\hat{\boldsymbol{\lambda}}$ to reasonable values for the different constraints.

6.8 Results and discussions

In this section we present the results of the case study outlined in Section 6.7. The aim of this section is two-fold. First in Section 6.8.1 we compare the accuracy of the GPPCE mean and variance estimates by comparing it to other important approaches that have been utilised to formulate SNMPC problems. Thereafter, in Section 6.8.2 we verify the GPPCE SNMPC algorithm defined in Section 6.6 and compare it to a nominal SNMPC algorithm with soft constraints.

6.8.1 GPPCE accuracy

In this section we verify the GPPCE approach to obtain mean and variance estimates of nonlinear transformations as outlined in Section 6.2. To accomplish this we ran the following tests:

- Set $\mathbf{U}$ to its upper bound and compare the pdfs of PCE, GP, and GPPCE for the two terminal constraint functions and the path constraint function at the

final time $t = N$ with the *true* pdfs. Each model has 15 training data points according to a Sobol design and the PCE and GPPCE polynomial order was set to 2. Note the pdfs are obtained using kernel density estimation (KDE) of the respective models [235]. The models are obtained using the data and polynomial order as outlined in the previous section. The results for this are shown in Figure 6.5 with the corresponding mean and variance estimates given in Table 6.2.

- Set $\mathbf{U}$ to its upper bound and compare the pdfs of GPPCE with 15, 25, and 40 training data-points according to a Sobol design with polynomial order of 2 throughout for the two terminal constraint functions and the path constraint function at the final time $t = N$. The results for this are shown in Figure 6.6.
- Lastly, $\mathbf{U}$ was set to 100 random values. For these the mean and variance of the terminal constraint functions and the path constraint at $t = N$ were estimated using GPs with 15 data-points, PCEs with 15 data-points, the Unscented transformation with 7 data-points ($2n_{\boldsymbol{\theta}} + 1$), and GPPCE based on 15, 25, and 40 data-points. The relative absolute error for these is illustrated in Figure 6.7 for the mean estimates and in Figure 6.8 for the standard deviation estimates as box plots.

Based on the tests outlined above we can draw the following observations and conclusions:

- Generally speaking from the plots in Figure 6.5 and Table 6.2 it can be said that all three approaches are able to represent the pdfs reasonably well with good approximations to the mean and variance. Nonetheless, PCEs are seen to outperform GPs considerably for the mid-plot, while for the graph on the RHS GPs outperform PCEs. GPs are expected to be able to handle more complex responses due to interpolating between data-points, while PCEs often capture better the overall trend, i.e. lead to a better global fit. GPPCE seems to approximate both well, since it is based on both methods, which highlights its major advantage over both.
- The confidence bound for GPPCEs and GPs in Figure 6.5 and Figure 6.6 corresponds to a 95% confidence region. It can be seen that the region is able to capture the relative uncertainty well, since it is larger if the fit is poor and smaller if the fit is better. Nonetheless, it does seem to be often overconfident. Furthermore, the variance accounting for the *finite samples* is given in Table 6.2 as *stochastic variance*. It can be seen that this variance is only once smaller then the *true* variance, further highlighting its potential use as a more conservative variance estimate.

- In Figure 6.7 the box plots highlight the absolute error from 100 mean approximations. It can be seen that in general the Unscented transformation performs the worst, which is not surprising since it is based on *only* 7 data-points. Furthermore, PCEs are seen to perform rather poorly as well. For PCEs it should be noted that for several variations it performed very well as can be seen in Figure 6.5 for example, however for several variations it performed poorly. This can in part be explained by the difficulty of determining reasonable regularization parameters, which can viewed as a significant disadvantage. GPs on the other hand often perform worse than PCEs, but manage to never perform very poorly due their nature of interpolating between data-points. Therefore, GPs on average perform much better than PCEs as can be seen in Figure 6.7. Lastly, GPPCE can again be seen to outperform both PCEs and GPs, which is in line with previous observations that GPPCE captures the *best* of both techniques. Interestingly the GPPCE mean approximation does not seem to improve with more data-points, however its *worst* performance is already small at 3% with 15 data-points.

- In Figure 6.8 the box plots show the absolute error from 100 variance approximations. It should be noted that accurate variance estimates are more difficult to achieve. The PCE performs the worst in this case, which similar to previously is down to it performing very poorly on a few test cases. This is further exacerbated from the square variance definition. Unscented performs poorly again, except for the second terminal constraint. GPs also do not perform particularly well leading to a up to nearly 40% error for the second terminal constraint. GPPCE on the other hand performs much better with 15 data-points leading to an error of at most 15% for the first terminal constraint. Further, it can be seen that GPPCE variance approximations steadily improves with more data-points, with 40 data-point GPPCE never exceeding a 5% error threshold. Overall it can be said that GPPCE is a vast improvement over its GP and PCE counterparts for estimating variances.

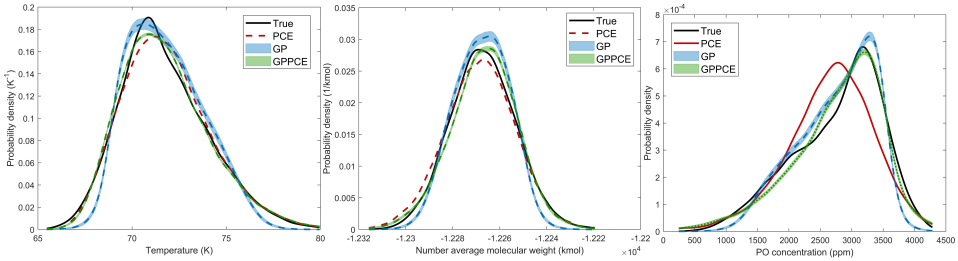


Figure 6.5: Plots of the probability density functions of the models *PCE*, *GP*, and *GPPCE* for the path constraint function $g(\cdot)$ at $t = N$ and the two terminal constraint functions $g_1^N(\cdot)$, $g_2^N(\cdot)$ from left to right respectively for $\mathbf{U}$ set to its upper bound.

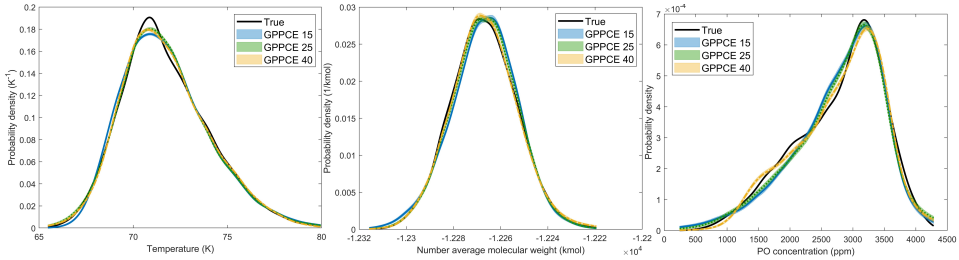


Figure 6.6: Plots of the probability density functions of the *GPPCE* model with 15, 25, and 40 training data-points for the path constraint function $g(\cdot)$ at $t = N$ and the two terminal constraint functions $g_1^N(\cdot)$, $g_2^N(\cdot)$ from left to right respectively for $\mathbf{U}$ set to its upper bound.

Table 6.2: Mean, variance, and *stochastic* variance for the PCE, GP, GPPCE with 15, 25 and 40 training data-points. From left to right the values in each field refer to the path constraint $g(\cdot)$ at $t = N$, the first terminal constraint $g_1^N(\cdot)$, and the second terminal constraint $g_2^N(\cdot)$ respectively.

Estimator	Mean ($g(\cdot)$, $g_1^N(\cdot)$, $g_2^N(\cdot)$)	Variance ($g(\cdot)$, $g_1^N(\cdot)$, $g_2^N(\cdot)$)	Stochastic variance ($g(\cdot)$, $g_1^N(\cdot)$, $g_2^N(\cdot)$)
True	$71.7,\ -1.23\times10^4,\ 2780$	$5.50,\ 195.1,\ 4.96\times10^5$	
PCE	$71.9,\ -1.23\times10^4,\ 2660$	$5.59,\ 222.0,\ 4.96\times10^5$	
GP	$71.8,\ -1.23\times10^4,\ 2760$	$3.57,\ 118.5,\ 3.40\times10^5$	$4.01,\ 203.4,\ 5.73\times10^5$
GPPCE 15	$71.8,\ -1.23\times10^4,\ 2778$	$5.60,\ 199.1,\ 5.43\times10^5$	$5.67,\ 203.4,\ 5.73\times10^5$
GPPCE 25	$71.7,\ -1.23\times10^4,\ 2809$	$5.67,\ 189.2,\ 5.51\times10^5$	$5.74,\ 196.8,\ 5.85\times10^5$
GPPCE 40	$71.7,\ -1.23\times10^4,\ 2803$	$5.50,\ 202.0,\ 5.12\times10^5$	$5.55,\ 204.4,\ 5.33\times10^5$

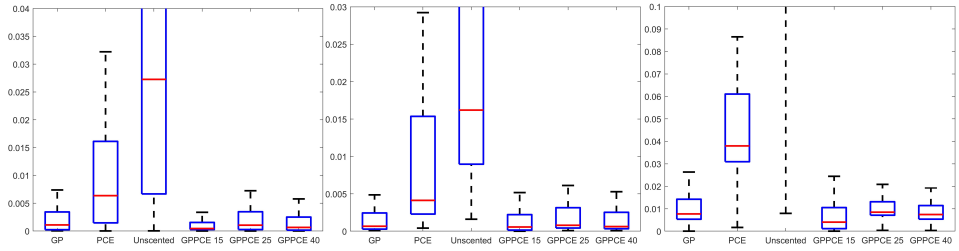


Figure 6.7: Box plots of absolute relative error of mean estimates from 100 random $\mathbf{U}$ values. From left to right the values in each field refer to the path constraint $g(\cdot)$ at $t = N$, the first terminal constraint $g_1^N(\cdot)$, and the second terminal constraint $g_2^N(\cdot)$ respectively.

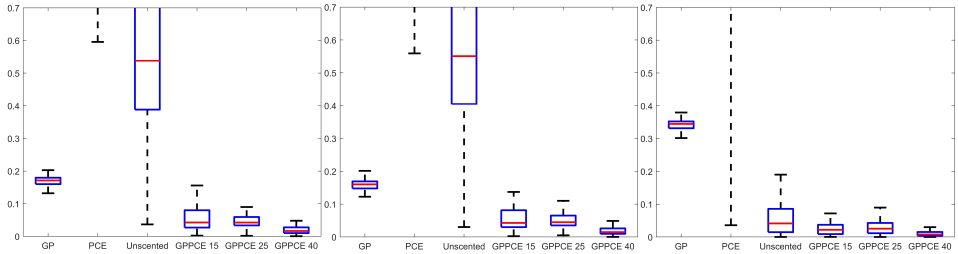


Figure 6.8: Box plots of absolute relative error of standard deviation estimates from 100 random $\mathbf{U}$ values. From left to right the values in each field refer to the path constraint $g(\cdot)$ at $t = N$, the first terminal constraint $g_1^N(\cdot)$, and the second terminal constraint $g_2^N(\cdot)$ respectively.

6.8.2 SNMPC verification

To verify the SNMPC algorithm given in Section 6.6.4 we run 400 closed-loop MC simulations of the case study outlined in 6.7 by sampling the uncertain parameters $\boldsymbol{\theta}$ independently. For comparison purposes the 400 MC simulations of the SNMPC algorithm are compared to 400 MC simulations of a nominal NMPC algorithm with soft constraints. The results of these MC simulations are highlighted in Figure 6.9 and Figure 6.10. Figure 6.9 depicts the probability densities using KDE of the 400 MC simulations for the two terminal constraint functions and the final batch time required for the SNMPC and nominal NMPC algorithm. Figure 6.10 shows the temperature trajectories of all 400 MC simulations for the SNMPC algorithm on the RHS and for the nominal NMPC algorithm on the LHS. In Table 6.3 the average and standard deviation computational times are given for both the SNMPC and the nominal NMPC algorithm. Based on these results we can draw the following observations and conclusions:

- In Figure 6.9 on the LHS we can see from the pdf the nominal NMPC violates the constraint considerably more than the SNMPC algorithm reaching often not the required *NAMW*. For all 400 MC simulations the SNMPC algorithm does not violate this constraint even once, while the nominal NMPC algorithm violates the constraint in 70% of the simulations.

- In Figure 6.9 the mid plot shows the pdfs of the ppm of the monomer. Again it can be seen that the nominal NMPC violates this constraint frequently, while the SNMPC is considerably more robust. Again the SNMPC algorithm does not violate this constraint for all 400 MC simulations, while the nominal NMPC algorithm violates it 62% of the time.

- From Figure 6.10 it can be seen that the temperature control of the SNMPC algorithm is considerably improved over the nominal NMPC, which violates the constraint in many of the MC simulations. The SNMPC was found to not violate the temperature constraint at all, while the nominal NMPC algorithm violates the path constraint in 95% of cases.

- The rightmost plot in Figure 6.9 shows the trade-off of the improved constraint satisfaction for the SNMPC algorithm. The pdf of the batch time for the SNMPC algorithm is more skewed towards longer batch times to be able to adhere the constraints. While the nominal NMPC algorithm on average has a batch time of 5550s, the SNMPC algorithm takes on average 7380s to complete a batch. This is expected, since superior constraint satisfaction leads to a worse objective.

- Table 6.3 shows the computational times of both algorithms. It can be seen that the SNMPC algorithm has considerably longer computational times compared to the nominal NMPC algorithm, which is expected. Nominal NMPC is based on a much smaller optimization problem without scenarios and has less strict constraints, which are often easier to adhere. Also the absence of hard constraints can lead to considerably faster computational times.

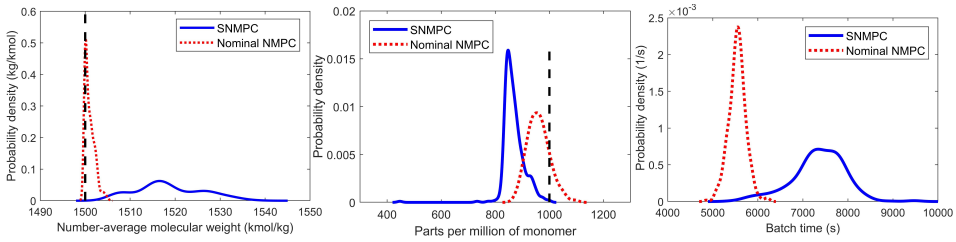


Figure 6.9: Plots of probability densities of *NAMW* (kg/kmol), parts per million of monomer, and batch time (s) based on 400 MC simulations from left to right respectively.

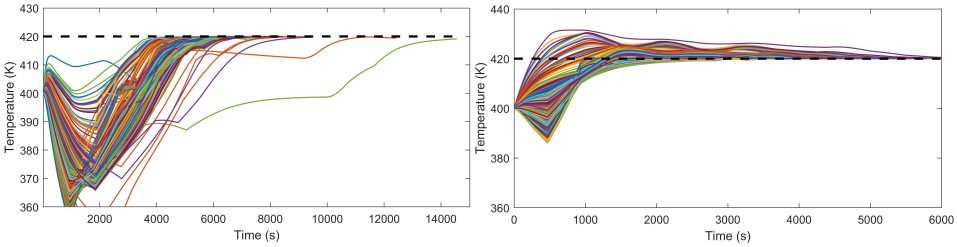


Figure 6.10: Plots of temperature trajectories of 400 MC simulations for SNMPC (left) and nominal NMPC (right).

Table 6.3: Optimal control problem average and standard deviation of computational times.

Algorithm	average OCP time (s)	standard deviation OCP time (s)
SNMPC	2.868	9.277
Nominal NMPC	0.045	0.001

6.9 Conclusions

In conclusion, we proposed a new approach to approximate the mean and variance of a nonlinear transformation given a standard normally distributed input by combining GPs and PCEs. It was shown that the method in general is able to capture better the shape of pdfs and leads to much improved approximations of both mean and variance. This can in part be explained by the approach leading to a good approximation if either GP or PCE leads to a good fit. Further, the GP-PCE SNMPC algorithm is shown to lead to superior constraint satisfaction over a nominal NMPC algorithm using soft constraints despite the stochastic uncertainties present. The computational times are kept moderately low by pre-computing the expensive terms involved in the GPPCE approach.

6.10 Posterior mean and variance derivation

In Section 6.2 we show how to derive the posterior mean and variance given a fitted GPPCE. The covariance function is given by the SE covariance function as given in Equation 6.6 and for the mean function we use Hermite polynomials, as is done in PCE [136]. Given these choices we need to determine several expectations for Equation 6.16 with $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \mathbf{I})$ following a standard normal distribution. Given these choices we can derive the expectations we require in turn. In essence we will exploit the fact that the SE covariance function represents an unnormalized Gaussian pdf:

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}_i) = \frac{\hat{\alpha}^2 \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_i, \hat{\boldsymbol{\Lambda}})}{|2\pi\hat{\boldsymbol{\Lambda}}|^{-\frac{1}{2}}} \tag{6.37}$$

and exploit the following identity:

$$\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p) \cdot \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) \coloneqq r\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r) \tag{6.38}$$

$$\int \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \mathbf{I}) \exp\left(-\frac{1}{2}\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\Lambda}^{-1}\boldsymbol{\theta}\right) d\boldsymbol{\theta} \coloneqq |\mathbf{I} + \boldsymbol{\Lambda}^{-1}|^{-\frac{1}{2}} \tag{6.39}$$

where $\boldsymbol{\Sigma}_r = (\boldsymbol{\Sigma}_p^{-1} + \boldsymbol{\Sigma}_q^{-1})^{-1}$, $\boldsymbol{\mu}_r = \boldsymbol{\Sigma}_r\left(\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\mu}_p + \boldsymbol{\Sigma}_q^{-1}\boldsymbol{\mu}_q\right)$, and $r = |2\pi(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q)|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^\mathsf{T}(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q)^{-1}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)\right)$.

Expectation of $m(\boldsymbol{\theta})$

The expectation of $m(\boldsymbol{\theta})$ is given by the first expansion coefficient in Equation 6.2 due to the orthogonality properties of the Hermite polynomials [177]:

$$\mu_m = \hat{\beta}_0 \tag{6.40}$$

where $\mu_m = \mathbb{E}_{\boldsymbol{\theta}}[m(\boldsymbol{\theta})]$.

Expectation of $\mathbf{k}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})$

The expectation of $\mathbf{k}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})$ can be derived by expressing the SE covariance function as a multivariate normal distribution. The full derivation can be found in [71] and leads to the following:

$$[\boldsymbol{\mu}_{\mathbf{k}_{\zeta,\mathbf{z}}}]_i = \hat{\alpha}^2 |\mathbf{I} + \boldsymbol{\Lambda}^{-1}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\boldsymbol{\theta}_i^\mathsf{T}(\mathbf{I} + \boldsymbol{\Lambda})^{-1}\boldsymbol{\theta}_i\right) \tag{6.41}$$

where $\boldsymbol{\mu}_{\mathbf{k}_{\zeta,\mathbf{z}}} = \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbf{k}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\right]$.

Expectation of $(m(\boldsymbol{\theta}))^2$

Note this is by definition the second moment of $m(\boldsymbol{\theta})$ and hence has received considerable attention. Again due to the orthogonality properties of the Hermite polynomials utilized the expectation of $(m(\boldsymbol{\theta}))^2$ is considerably simplified [177]:

$$\mu_{m^2} = \sum_{i=0}^{L-1} \beta_i^2 \mathbb{E}_{\boldsymbol{\theta}} \left[\phi_i^2(\boldsymbol{\theta})\right] \tag{6.42}$$

where $\mu_{m^2} = \mathbb{E}_{\boldsymbol{\theta}} \left[m(\boldsymbol{\theta})^2\right]$.

Expectation of $\mathbf{k}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\mathbf{k}_{\zeta,\mathbf{z}}^{\mathsf{T}}(\boldsymbol{\theta})$

The expectation of the outer product of $\mathbf{k}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})$ can be found in [71] and is as follows:

$$[\mathbf{M}_{\mathbf{k}_{\zeta,\mathbf{z}}\mathbf{k}_{\zeta,\mathbf{z}}^{\mathsf{T}}}]_{ij} = k(\boldsymbol{\theta}_i, \mathbf{0})k(\boldsymbol{\theta}_j, \mathbf{0})|\mathbf{R}|^{-\frac{1}{2}} \exp\left(\mathbf{l}^{\mathsf{T}}\mathbf{R}^{-1}\mathbf{l}\right) \tag{6.43}$$

where $\mathbf{M}_{\mathbf{k}_{\zeta,\mathbf{z}}\mathbf{k}_{\zeta,\mathbf{z}}^{\mathsf{T}}} = \mathbb{E}\left[\mathbf{k}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\mathbf{k}_{\zeta,\mathbf{z}}^{\mathsf{T}}(\boldsymbol{\theta})\right]$, $\mathbf{R} = 2\hat{\boldsymbol{\Lambda}}^{-1} + \mathbf{I}$, and $\mathbf{l} = \hat{\boldsymbol{\Lambda}}\boldsymbol{\theta}_i + \hat{\boldsymbol{\Lambda}}\boldsymbol{\theta}_j$.

Expectation of $m(\theta)\mathbf{k}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})$

This term is somewhat more difficult to deal with, since it is a cross-term between the mean function and the covariance function. Unfortunately we cannot exploit the orthogonality properties of $m(\boldsymbol{\theta})$, nonetheless the term has a closed-form solution as we will show here:

$$\left[\boldsymbol{\mu}_{m\mathbf{k}_{\zeta,\mathbf{z}}}\right]_i = \int \mathcal{N}(\boldsymbol{\theta}; \mathbf{0}, \mathbf{I}) \frac{\hat{\alpha}^2 \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\theta}_i, \boldsymbol{\Lambda})}{|2\pi\boldsymbol{\Lambda}|^{-\frac{1}{2}}} m(\boldsymbol{\theta}) d\boldsymbol{\theta} = r \int m(\boldsymbol{\theta}) \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r) d\boldsymbol{\theta} = \tag{6.44}$$

$$r\boldsymbol{\beta}^{\mathsf{T}} \int \boldsymbol{\phi}(\boldsymbol{\theta}) \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r) d\boldsymbol{\theta} = r\boldsymbol{\beta}^{\mathsf{T}} \mathbb{E}_{\boldsymbol{\theta}_r} \left[\boldsymbol{\phi}(\boldsymbol{\theta}_r)\right]$$

where $\mathbb{E}_{\boldsymbol{\theta}} \left[m(\theta)\mathbf{k}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\right] = \boldsymbol{\mu}_{m\mathbf{k}_{\zeta,\mathbf{z}}}$, $r = \alpha^2 |\boldsymbol{\Lambda}^{-1} + \mathbf{I}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\boldsymbol{\theta}_i^{\mathsf{T}}(\boldsymbol{\Lambda} + \mathbf{I})^{-1}\boldsymbol{\theta}_i\right)$, $\boldsymbol{\theta}_r \sim \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$ follows a multivariate Gaussian distribution, $\boldsymbol{\mu}_r = \boldsymbol{\theta}_i(\boldsymbol{\Lambda}+\mathbf{I})$, and $\boldsymbol{\Sigma}_r = (\mathbf{I} + \boldsymbol{\Lambda}^{-1})^{-1}$. Note that $\boldsymbol{\phi}(\boldsymbol{\theta}_r)$ is a vector of polynomial terms, for which the expectations are given by statistical moments, which have a closed form solution according to a multivariate Gaussian distribution see [262].

Expectation of $\boldsymbol{\kappa}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\boldsymbol{\kappa}_{\zeta,\mathbf{z}}^{\mathsf{T}}(\boldsymbol{\theta})$

This term is again a cross-term and is dealt with in a similar way.

$$\mathbf{M}_{\boldsymbol{\kappa}_{\zeta,\mathbf{z}}\boldsymbol{\kappa}_{\zeta,\mathbf{z}}} = \begin{bmatrix} \mathbb{E}_{\boldsymbol{\theta}}\left[\boldsymbol{\phi}(\boldsymbol{\theta})\boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{\theta})\right] & \mathbb{E}_{\boldsymbol{\theta}}\left[\boldsymbol{\phi}(\boldsymbol{\theta})\mathbf{k}_{\zeta,\mathbf{z}}^{\mathsf{T}}(\boldsymbol{\theta})\right] \\ \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbf{k}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{\theta})\right] & \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbf{k}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\mathbf{k}_{\zeta,\mathbf{z}}^{\mathsf{T}}(\boldsymbol{\theta})\right] \end{bmatrix} \tag{6.45}$$

where $\mathbf{M}_{\boldsymbol{\kappa}_{\zeta,\mathbf{z}}\boldsymbol{\kappa}_{\zeta,\mathbf{z}}} = \mathbb{E}_{\boldsymbol{\theta}}\left[\boldsymbol{\kappa}_{\zeta,\mathbf{z}}(\boldsymbol{\theta})\boldsymbol{\kappa}_{\zeta,\mathbf{z}}^{\mathsf{T}}(\boldsymbol{\theta})\right]$.

The bottom right expectation was determined previously. The top left can be expressed as follows:

$$\left[\mathbb{E}_{\boldsymbol{\theta}}\left[\boldsymbol{\phi}(\boldsymbol{\theta})\boldsymbol{\phi}^{\mathsf{T}}(\boldsymbol{\theta})\right]\right]_{ij} = \mathbb{E}_{\boldsymbol{\theta}}\left[\phi_i(\boldsymbol{\theta})\phi_j(\boldsymbol{\theta})\right] \tag{6.46}$$

where the RHS expectation has a closed-form solution and is a common occurrence to determine the covariance employing PCE, see for example [83].

The last remaining term can be determined as follows:

$$\begin{aligned}\left[\mathbb{E}_{\boldsymbol{\theta}}\left[\boldsymbol{\phi}(\boldsymbol{\theta})\mathbf{k}_{\zeta,\mathbf{z}}^{\mathsf{T}}(\boldsymbol{\theta})\right]\right]_{ij} &= \int \mathcal{N}(\boldsymbol{\theta};\mathbf{0},\mathbf{I})\frac{\hat{\alpha}^2\mathcal{N}(\boldsymbol{\theta};\boldsymbol{\theta}_i,\boldsymbol{\Lambda})}{|2\pi\boldsymbol{\Lambda}|^{-\frac{1}{2}}}\phi_j(\boldsymbol{\theta})d\boldsymbol{\theta} = \\ r\int \phi_j(\boldsymbol{\theta})\mathcal{N}(\boldsymbol{\theta}_r;\bar{\boldsymbol{\mu}}_r,\boldsymbol{\Sigma}_r)d\boldsymbol{\theta} &= r\mathbb{E}_{\boldsymbol{\theta}_r}\left[\phi_j(\boldsymbol{\theta}_r)\right]\end{aligned} \tag{6.47}$$

where $\boldsymbol{\theta}_r \sim \mathcal{N}(\boldsymbol{\theta};\boldsymbol{\mu}_r,\boldsymbol{\Sigma}_r)$ follows a multivariate Gaussian distribution. Note that $\phi_j(\boldsymbol{\theta}_r)$ is a multivariate polynomial, for which the expectations are given by statistical moments, which have a closed form solution according to a multivariate Gaussian distribution [262].

Part III

Gaussian process dynamic modelling and nonlinear model predictive control

Chapter 7

Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate Gaussian processes

This chapter is based on **Paper E**: E. Bradford, A. M. Schweidtmann, D. Zhang, K. Jing, and E. A. del Rio-Chanona. Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate Gaussian processes. *Computers & Chemical Engineering*, 118:143–158, 2018.

Summary

Dynamic modeling is an important tool to gain better understanding of complex bioprocesses and to determine optimal operating conditions for process control. Currently, two modeling methodologies have been applied to biosystems: kinetic modeling, which necessitates deep mechanistic knowledge, and artificial neural networks (ANN), which in most cases cannot incorporate process uncertainty. The goal of this study is to introduce an alternative modeling strategy, namely Gaussian processes (GP), which incorporates uncertainty but does not require complicated kinetic information. To test the performance of this strategy, GPs were applied to model microalgae growth and lutein production based on existing experimental datasets and compared against the results of previous ANNs.

Furthermore, a dynamic optimization under uncertainty is performed, avoiding over-optimistic optimization outside of the model's validity. The results show that GPs possess comparable prediction capabilities to ANNs for long-term dynamic bioprocess modeling, while accounting for model uncertainty. This strongly suggests their potential applications in bioprocess systems engineering.

7.1 Introduction

The synthesis of sustainable bioproducts from microalgae through photosynthetic related metabolic pathways has become a promising research field because of its outstanding advantages over traditional fossil fuel based processes [49, 274]. Specifically, in the energy and food sectors the development and deployment of microalgae based technologies have seen substantial interests within the last decade [152, 172]. For example, these emerging technologies represent a variety of promising alternatives for the next generation of renewable and environmentally friendly transportation fuels such as biodiesel and biohydrogen [246, 1]. Meanwhile, they have been recently adopted by different countries such as the United States, China and Mexico to produce nutritious food supplements and animal feeds, of which the global market has been predicted to undergo considerable growth [66, 275]. Furthermore, they have been successfully industrialized to produce different high-value bioproducts that are widely used in the cosmetic, pharmaceutical and food industries (*e.g.* lutein, C-phycocyanin and astaxanthin), which are otherwise produced from expensive, energy intensive and low efficient manufacturing routes using non-renewable sources [86, 243].

In particular, the biorenewable product investigated in the current study is lutein, which is of great interest to the health, pharmaceutical, and food sectors. In the United States the demand of lutein is predicted to increase from $150 million in 2000 to $309 million in 2018, with an annual growth rate of over 6% up to 2024 [117, 77]. However, the current feedstock for lutein production is marigold, a plant with extremely low lutein content (0.02-0.1% wt (fresh flowers)) and low growth rate requiring large separation costs [273]. A promising alternative to produce lutein is from microalgae due to their rapid growth rate, higher lutein content (up to 0.5% wt) and capability of utilizing abundant sustainable resources such as solar energy, atmospheric CO_2 and waste water for their growth and product synthesis [268].

To drive the industrialization of sustainable lutein production from microalgae, a robust mathematical model is required for precise process control over a long-term time horizon, so that both process safety and efficiency can be guaranteed. Moreover, by utilizing state-of-the-art process optimization strategies for mathematical models, dynamic optimization can be further carried out to increase

process profitability [75]. As bioprocesses are in general sensitive to changes of operating conditions, it is expected that by implementing dynamic optimization a significant improvement on product yield can be obtained compared to the recent literature results [170, 268, 79].

So far, two modeling methodologies have been employed to simulate dynamic behavior of the underlying biosystem for microalgal biomass growth and lutein synthesis, namely kinetic modeling and artificial neural networks (ANN) [98, 79]. In this paper, however, a third methodology, Gaussian process (GP) regression, is proposed to simulate this system and compared against the previous two, so that its feasibility and capability for bioprocess modeling can be thoroughly explored for the first time. Recently, GPs have become an increasingly popular non-parametric method for both regression and classification problems [220]. GP regression was first proposed by [200] and then popularized in [191].

The GP regression framework not only provides a prediction for unknown outputs, but also provides a measure for prediction uncertainty, which is a distinct advantage compared to other commonly used black-box methods. Furthermore, in [219] it was shown that GPs are able to forecast outputs with comparable performance to other modeling approaches like ANNs or local learning methods. GPs have been shown to be a powerful tool for derivative-free optimization, since the uncertainty measure can be exploited to evaluate functions more efficiently for both single-objective optimization [130, 234] and multi-objective optimization [46]. Although GPs have been predominantly used to model static nonlinearities, it is notable that they have also been demonstrated and applied to simulate dynamic systems in several studies [141, 48, 100, 251, 259, 36]. Particularly for studies of long-term bioprocess modeling and optimization, despite the fact that GPs have never been adopted in this domain, they are expected to possess two outstanding advantages over the most commonly used methods (*i.e.* kinetic model and ANN), which are:

1. Compared to ANNs, GPs provide a clear measure of prediction uncertainty which is crucial when modeling complex biological systems. In addition, although ANNs might be preferred over GPs in some cases (*e.g.* when there is a large amount of training data) because of the matrix inversion that is required for the construction of GP predictions, in macro-scale biomanufacturing studies it is infeasible to obtain datasets in the order of hundreds of thousands or millions. Hence, GPs are clearly a comparable or even superior tool to ANNs in this domain.

2. Compared to kinetic models, GPs do not need a full understanding of the complex metabolic mechanisms that take place in the specific biosystem.

> Specific to bioprocess applications, in most cases, intensive collaborative effort of the scientific community is required to identify the essential biochemical kinetic information before a kinetic model is ready to be constructed. On the contrary, GPs are black-box models that can be used to simulate and optimize processes in the early stage of research, hence efficiently forwarding the assessment and prototyping stages for process design and scale-up.

As a result, in the current research GPs are set as the default black-box modeling strategy for algal lutein production. Given that for bioprocesses there is a need to predict long time horizons (in the order of days), two approaches have been proposed in literature to accomplish this [77]:

One approach is to train a model on **all** control inputs and the initial state to obtain predictions at the full-time horizon. While this approach is easy-to-implement, it has several considerable disadvantages. Once trained, the model cannot be extended to make predictions at time horizons with different lengths. In addition, for large time horizons the number of inputs quickly becomes large, requiring a high dimensionality of the GP to be learnt and hence too many data-points. Alternatively, the iterative method can be used, which trains a GP to predict one-step ahead and applies this GP recursively to obtain a prediction for the full time horizon. The iterative method has several advantages compared to one-step ahead predictions. It can be easily used for different time horizons with different lengths and provide any k-step ahead forecast including the joint probability distribution of the states at the desired points. In addition, the dimensionality of GPs is much smaller in the iterative method when control inputs are present and therefore more data efficient.

In this paper, a general procedure to execute dynamic modeling using GPs is outlined. For this the *iterative method* was selected and applied to predict the evolution of multivariate states for lutein production. In [100] it was shown that the noise in the one-step ahead prediction needs to be propagated to be conservative enough. Therefore, we propagated the resulting probability densities of the GPs using exact moment matching (by moments we refer to the mean and covariance) for the squared exponential covariance function. The methodology exhibited in [141] given for single variate state systems was extended to the multivariate case by using results from machine learning (*e.g.* reinforcement learning) in [71].

The paper is structured as follows. In Section 7.2 we introduce the reader to Gaussian processes. Section 7.3 outlines the experimental set-up of the algal lutein production process with various operating conditions and shows how this data can be used to build GPs for the dynamic modeling of biosystems. In Section 7.4 a description of the recently constructed ANNs for comparison to the GP is given.

In the Results and Discussion section (Section 7.5) the GP regression results are presented and compared against the ANNs. Meanwhile, a dynamic optimization with stochastic constraints was performed in this section to maximize lutein yield by varying flow rate and light intensity, while taking advantage of the probabilistic nature of the GP.

7.2 Introduction to Gaussian process regression

7.2.1 Multivariate Gaussian distribution

To explain the principle of the GP modeling framework, we need to first introduce some concepts commonly used in probability theory. A random variable follows a univariate Gaussian distribution if its probability density function is given by Equation 7.1, where $x \in \mathbb{R}$ is the result of a single test. A Gaussian distribution is defined by its mean μ (expectation), and its variance σ^2. This is generally written as $x \sim \mathcal{N}\left(\mu, \sigma^2\right)$.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{7.1}$$

Let us generalize this definition into higher dimensions. For a given random n-dimensional vector $\mathbf{x} = [x_1, ..., x_n]^T$ with mean $\boldsymbol{\mu} = [\mu_1, ..., \mu_n]^T$, its covariance matrix $\boldsymbol{\Sigma} = \text{cov}(\mathbf{x})$ is defined as a $n \times n$ matrix of which the entry at the i^{th} row and j^{th} column is calculated by Equation 7.2, where μ_i and μ_j denote the expectation of x_i and x_j, respectively. In general, any symmetric positive semidefinite matrix can be used as a covariance matrix. This necessitates the diagonal elements to be non-negative, since these are variances. The vector $\mathbf{x} \in \mathbb{R}^n$ is said to have a multivariate Gaussian distribution if every linear combination of its components $(x_1, ..., x_n)$ is a univariate Gaussian distribution. The probability density function of this multivariate Gaussian distribution is written as Equation 7.3, and commonly denoted as $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean and covariance matrix of these random variables, respectively. It is worth mentioning that the covariance $\text{cov}\left(x_i, x_j\right)$ is a measure of the correlation between the component x_i and the component x_j. Thus, if these components are independent, the covariance coefficient becomes 0. For example, a standard multivariate Gaussian distribution is defined such that each component is independent, the mean $\boldsymbol{\mu} = \mathbf{0}$ and the covariance matrix is given by $\boldsymbol{\Sigma} = \mathbf{I}_{n\times n}$.

$$\text{cov}\left(x_i, x_j\right) = E\left[\left(x_i - \mu_i\right)\left(x_j - \mu_j\right)\right] \tag{7.2}$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2} \mid \boldsymbol{\Sigma} \mid^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{7.3}$$

where $|\cdot|$ denotes the determinant

There are two important identities for multivariate Gaussian distributions, which are essential in GP regression. Let us assume that we are given a joint vector of $\mathbf{x}$ and $\mathbf{y}$ that are distributed as a multivariate normal distribution as shown in Equation 7.4.

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_\mathbf{x} \\ \boldsymbol{\mu}_\mathbf{y} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_\mathbf{x} & \boldsymbol{\Sigma}_{\mathbf{x},\mathbf{y}} \\ \boldsymbol{\Sigma}_{\mathbf{y},\mathbf{x}} & \boldsymbol{\Sigma}_\mathbf{y} \end{bmatrix} \right) \tag{7.4}$$

The marginal distribution of $\mathbf{x}$ (the distribution of $\mathbf{x}$ alone) is then given by Equation 7.5.

$$\mathbf{x} \sim \mathcal{N}\left(\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}\right) \tag{7.5}$$

The conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ can be denoted by Equation 7.6.

$$\mathbf{x}|\mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_\mathbf{x} + \boldsymbol{\Sigma}_{\mathbf{x},\mathbf{y}}\boldsymbol{\Sigma}_\mathbf{y}^{-1}\left(\mathbf{y}-\boldsymbol{\mu}_\mathbf{y}\right), \boldsymbol{\Sigma}_\mathbf{x} - \boldsymbol{\Sigma}_{\mathbf{x},\mathbf{y}}\boldsymbol{\Sigma}_\mathbf{y}^{-1}\boldsymbol{\Sigma}_{\mathbf{y},\mathbf{x}}\right) \tag{7.6}$$

where $\mathbf{x}|\mathbf{y}$ denotes the probability distribution of $\mathbf{x}$ given that we know the value of $\mathbf{y}$. Therefore, the distribution on the right-hand side of Equation 7.6 has lower variances (the diagonal elements of the covariance matrix) than the marginal distribution of $\mathbf{x}$ in Equation 7.5, since we are exploiting the knowledge that the value of $\mathbf{y}$ gives us on $\mathbf{x}$.

7.2.2 Introduction to Gaussian process regression

In this section we give a short introduction to GP regression. For a more detailed and complete overview please refer to [220, 84, 129]. GPs generalize multivariate Gaussian distribution to infinite dimensions defining a functional space and hence describe a distribution over infinite dimensional vector functions. Formally, a GP is a collection of random variables of which any finite subset follows a Gaussian distribution. GP regression aims to model an unknown latent function $f(\mathbf{x})$ using noisy observations y of $f(\mathbf{x})$, which are related as follows:

$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2) \tag{7.7}$$

where $\mathbf{x} \in \mathbb{R}^n$ denotes an arbitrary input vector and ϵ is Gaussian distributed measurement noise with a variance of σ_ϵ^2.

Assume we want to make a prediction of $f(\mathbf{x})$ at some arbitrary input $\mathbf{x}$. Before we have sampled the function at this point, *i.e.* before we obtain observations of $f(\mathbf{x})$ at this input $\mathbf{x}$, this value will be uncertain. For GPs we model this uncertainty of the value of the function at $\mathbf{x}$ as the realization of a normally distributed random variable $f(\mathbf{x})$ with mean μ and variance σ^2. Intuitively, we are assuming the function value at $\mathbf{x}$ to have a typical value of μ, which can be expected to lie with a probability of 99.7% in the range $[\mu - 3\sigma, \mu + 3\sigma]$. The mean μ of $f(\mathbf{x})$ in the most general case may be given by an arbitrary function $m(\mathbf{x})$, which defines the "average" shape of the function.

To define a covariance function, we consider two arbitrary input vectors $\mathbf{x}$ and $\mathbf{x}'$, which again have not been sampled and consequently the values of the function at these points are uncertain. However, if we assume the unknown function, which we wish to model, to be continuous, then the function values $f(\mathbf{x})$ and $f(\mathbf{x}')$ will be close if the distance between $\mathbf{x}$ and $\mathbf{x}'$ is small. This prior information can be modeled statistically by assuming that the random variables $f(\mathbf{x})$ and $f(\mathbf{x}')$ are strongly correlated if the distance $||\mathbf{x}-\mathbf{x}'||$ is small. Correlation means that $f(\mathbf{x})$ will tend to be large if $f(\mathbf{x}')$ is large, as long as $\mathbf{x}$ and $\mathbf{x}'$ are close together. On the other hand if $\mathbf{x}$ and $\mathbf{x}'$ are far apart, then the values of $f(\mathbf{x})$ and $f(\mathbf{x}')$ are virtually independent. In particular, in this paper we assume the correlation between the random variables is given by the squared-exponential (SE) covariance function, which is a stationary covariance function. A stationary covariance function is a function of $\mathbf{x} - \mathbf{x}'$, such that the covariance function is translation invariant $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}', \mathbf{0})$. The SE covariance function can be defined as follows [220]:

$$\operatorname{cov}_f\left(f(\mathbf{x}), f(\mathbf{x}')\right) = \mathbb{E}_f\left(\left(f(\mathbf{x}) - m(\mathbf{x})\right)\left(f(\mathbf{x}') - m(\mathbf{x}')\right)\right) = k(\mathbf{x}, \mathbf{x}') = \alpha^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{\Lambda} (\mathbf{x} - \mathbf{x}')\right) \tag{7.8}$$

where $\mathbf{x}$,$\mathbf{x}'$ are arbitrary inputs, $\mathbf{\Lambda} = \operatorname{diag}([\lambda_1^{-2}, \ldots, \lambda_n^{-2}])$ is a diagonal matrix with a length scale λ_i for each input and α^2 is the signal variance. $\mathbb{E}_f$ is the expectation over the function space. The mean function can be viewed as the 'typical' shape of the function, while the covariance function specifies the covariance between any two function values at two separate inputs. The SE covariance function is such that if $\mathbf{x} = \mathbf{x}'$ then $k(\mathbf{x}, \mathbf{x}') = \alpha^2$ reaches the maximum; while if $||\mathbf{x} - \mathbf{x}'|| \rightarrow \infty$ the correlation tends to zero as required. The parameters λ_i determine how fast the correlation tends to zero as one moves in the i^{th} dimension of the input

vector. Small values of λ_i model functions that are significantly dependent on the i^{th} dimension, *i.e.* the function value can rapidly change when varying the i^{th} variable of such functions. Conversely large values of λ_i lead to functions that are close to invariant with respect to the i^{th} variable. In addition, the SE covariance function not only assumes the latent function to be continuous, but also smooth since it is infinitely differentiable.

A GP generalizes the Gaussian distribution to infinite dimensions and describes a distribution over functions. It is fully specified by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$. We write f is distributed as a GP as follows:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{7.9}$$

The noisy observations y also follows a GP due to the additive property of Gaussian distributions with the same mean, but with a different covariance function to account for the measurement noise:

$$y \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}') + \sigma_\epsilon^2 \delta(\mathbf{x}, \mathbf{x}')) \tag{7.10}$$

where $\delta(\mathbf{x}, \mathbf{x}') = 1$ **iff** $\mathbf{x} = \mathbf{x}'$ and else $\delta(\mathbf{x}, \mathbf{x}') = 0$, known as the Kronecker-delta.

Equations 7.9 and 7.10 define the prior of the function, since no data has been used yet. Afterwards this prior is updated using input-output data available from observations. For GP regression we need to first define the prior of the GP by choosing the mean function and covariance function, which encapsulate our prior beliefs, if available, about the function to be modeled. In this report we assume a mean function of zero as given in Equation 7.11, which is a common choice in Machine Learning [220]. A zero-mean of the data is achieved in this report by scaling the data. In essence this means that we are assuming the function to be overall zero mean, such as a sine function.

$$m(\mathbf{x}) = 0 \tag{7.11}$$

We assume the covariance function to be given by the SE defined in Equation 7.8. As previously described this encapsulates our belief that the function to be modeled is smooth.

Next we assume that N observations are available at N different inputs given by the following two quantities:

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N], \qquad \mathbf{y} = [y_1, \ldots, y_N]^T \tag{7.12}$$

We can then represent the uncertainty of n function values based on the prior from the mean and covariance functions with the help of a random vector $\mathbf{F} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^T$ at n separate input vectors given by the matrix $\mathbf{X}$. This random vector has a mean vector defined as $\mathbf{0}$ and a covariance matrix equal to:

$$\boldsymbol{\Sigma}_{\mathbf{F}} = \left[k\left(\mathbf{x}_i, \mathbf{x}_j\right)\right]_{N \times N} \tag{7.13}$$

where $\boldsymbol{\Sigma}_{\mathbf{F}}$ is a $N \times N$ matrix with (i, j) element given by Equation 7.8.

Equation 7.13 gives us the covariance matrix for the latent function values. We however observe y and not $f(\mathbf{x})$, which is perturbed by Gaussian distributed measurement noise with a variance of σ_ϵ^2 as shown in Equation 7.7. The uncertainty of the observation matrix $\mathbf{y}$ can then be expressed in the same way as $\mathbf{F}$ with a mean function of $\mathbf{0}$ and a covariance matrix given by:

$$\boldsymbol{\Sigma}_{\mathbf{y}} = \left[k\left(\mathbf{x}_i, \mathbf{x}_j\right) + \sigma_\epsilon^2 \delta(\mathbf{x}_i, \mathbf{x}_j)\right]_{N \times N} \tag{7.14}$$

The hyperparameters defining the prior GP are commonly unknown *a priori*, and hence an important step in GP regression is the determination of the hyperparameters from the available data. The hyperparameters that define the GP are given by the parameters of the covariance function in Equation 7.8 and by the noise of y in Equation 7.7. These are jointly denoted by the vector $\boldsymbol{\Theta} = [\log(\lambda_1), \ldots, \log(\lambda_n), \log(\alpha), \log(\sigma_\epsilon)]^T$, where the parameters were log-transformed to ensure positiveness. The hyperparameters of the GPs in this study were efficiently found using a *maximum a posteriori* (MAP) estimate, which is more efficient for smaller data sets than the more commonly used *maximum likelihood* (ML) approach [220]. This is due to the prior introduced in MAP preventing overfitting compared to ML [245]. In this work, we assume independent Gaussian distributions on the hyperparameters:

$$\Theta_j \sim \mathcal{N}(\mu_{\Theta_j}, \sigma_{\Theta_j}^2) \tag{7.15}$$

where μ_{Θ_j} is the mean and $\sigma_{\Theta_j}^2$ the variance of the prior Gaussian distribution for the hyperparameter Θ_j.

Following from the uncertainty expression of the data as multivariate Gaussian distribution with covariance matrix as in Equation 7.14 and the prior distribution of the hyperparameters in Equation 7.15, the log-likelihood of the posterior density

of the hyperparameters can be stated as follows [220]:

$$\mathcal{L}(\boldsymbol{\Theta}) = -\frac{1}{2}\log(|\boldsymbol{\Sigma}_{\mathbf{y}}|) - \frac{1}{2}\mathbf{y}^T\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{y} - \frac{N}{2}\log(2\pi) + \sum_j(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma_{\Theta_j}^2) - \frac{1}{2\sigma_{\Theta_j}^2}(\Theta_j - \mu_{\Theta_j})^2) \tag{7.16}$$

Notice that function $\mathcal{L}(\boldsymbol{\Theta})$ is still a function of the training targets $\mathbf{y}$, and hence the difference between the predicted outputs and the target outputs (the $\mathbf{y}$s) will be minimized in a similar fashion as would happen with an ANN training framework. The covariance matrix $\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}$ can be efficiently factorized using Cholesky decomposition, since it is a symmetric positive semidefinite matrix. Once the hyperparameters are fixed, the inverse matrix product can then be solved efficiently. Since the elements of the covariance are however nonlinear functions of the hyperparameters, the factorization needs to be recalculated for each iteration of these.

The MAP estimate of $\boldsymbol{\Theta}$ is then given by:

$$\boldsymbol{\Theta}_{MAP} \in \arg\max_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\Theta}) \tag{7.17}$$

Once we have calculated the MAP estimate of $\boldsymbol{\Theta}$, we can use GPs to predict the value of $f(\mathbf{x})$ and y at unknown inputs. First consider the joint distribution of the data and the function value $f(\mathbf{x})$, which can be established using the mean and covariance function of the prior:

$$\begin{bmatrix} f(\mathbf{x}) \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \Sigma_f & \boldsymbol{\Sigma}_{f,\mathbf{y}} \\ \boldsymbol{\Sigma}_{\mathbf{y},f} & \boldsymbol{\Sigma}_{\mathbf{y}} \end{bmatrix}\right)$$

where $\boldsymbol{\Sigma}_{\mathbf{y}}$ is given in Equation 7.14, $\Sigma_f = k(\mathbf{x},\mathbf{x})$, $\boldsymbol{\Sigma}_{f,\mathbf{y}} = [k(\mathbf{x},\mathbf{x}_1),\ldots,k(\mathbf{x},\mathbf{x}_N)]$ and $\boldsymbol{\Sigma}_{\mathbf{y},f} = \boldsymbol{\Sigma}_{f,\mathbf{y}}^T$.

As set out at the beginning, we want to know the value of $f(\mathbf{x})$ given the data available to us, which is represented by the random vector $\mathbf{y}$. We can now apply the identity given in Equation 7.6 that gives us the distribution of $f(\mathbf{x})$ given the observations of $\mathbf{y}$, which can be stated as follows:

$$f(\mathbf{x})\,|\mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\Sigma}_{f,\mathbf{y}}\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{y}, \Sigma_f - \boldsymbol{\Sigma}_{f,\mathbf{y}}\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\boldsymbol{\Sigma}_{f,\mathbf{y}}^T\right) \tag{7.18}$$

The mean in this case is the best estimate of $f(\mathbf{x})$ given the data available, while the variance gives us a measure of uncertainty to this estimate. To obtain the

posterior of the observation of $f(\mathbf{x})$ we simply need to add the observation noise to the variance:

$$y|\mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\Sigma}_{f,\mathbf{y}}\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{y}, \Sigma_f - \boldsymbol{\Sigma}_{f,\mathbf{y}}\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\boldsymbol{\Sigma}_{f,\mathbf{y}}^{T} + \sigma_\epsilon^2\right) \tag{7.19}$$

where y is the observation of $f(\mathbf{x})$ according to Equation 7.7.

We have now shown how GPs can be used to obtain predictions at arbitrary inputs. The overall procedure involves these three steps:

1. Choose mean and covariance function depending on the prior knowledge of the underlying function.

2. Determine the hyperparameter values by maximum a posteriori likelihood estimation using observations of the underlying function.

3. Make predictions at arbitrary inputs using Equations 7.18 and 7.19, where the mean represents the prediction and the variance the corresponding uncertainty.

An example of GP regression can be seen in Figure 7.1 with the prior and posterior shown.

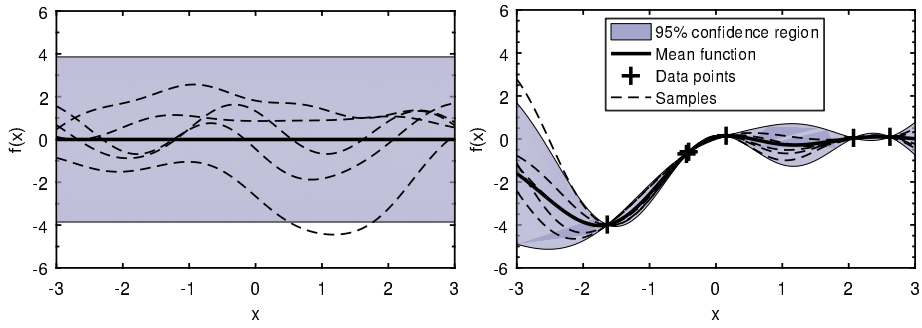


Figure 7.1: Illustration of a GP of a 1-dimensional function perturbed by noise. On the left the prior of the GP is shown with mean 0 and standard deviation of ~ 2 with 5 samples drawn from the GP prior, each of which corresponds to a separate function. On the right the GP was given additional information (8 observations of the latent function) and fitted to these observations to obtain the posterior. Again the mean and 5 samples are shown. One can notice that close to these observations the uncertainty is greatly reduced, however areas far from observations exhibit greater uncertainty.

7.3 Gaussian process dynamic modeling for bioprocesses

7.3.1 Algal lutein production process experimental set-up

The experiment of microalgal lutein production consists of 3 states and 2 control variables. The 3 states are biomass concentration, nitrate concentration and lutein production and the 2 control variables are incident light intensity and nitrate inflow rate. The microalgae species *Desmodesmus* sp. F51 was used for lutein production and experimental temperature was fixed at 35°C. A 1 L photobioreactor (15.5 cm in length and 9.5 cm in diameter) was used in these experiments with an external light source on both sides. Initial biomass concentrations were kept constant and incident light intensities were varied between 150 μmol $m^{-2}s^{-1}$ to 600 μmol $m^{-2}s^{-1}$. Nitrate influent was supplied to the reactor to compensate for the culture nitrate consumption from the 60^{th} hour until the end of the experiment with a fixed inflow rate of 3 mL hr^{-1}. Influent nitrate concentration was chosen as 0.1M or 0.5M. All the runs were carried out over 6 days.

The states were measured every 12 hours over 144 hours, hence 12 measurements were taken for each experimental run. In total 7 different experiments were conducted. All of the experiments were replicated twice and the detailed presentation of experimental design and measurement techniques can be found in [80]. The operating conditions of these experiments are summarized in Table 7.1.

Table 7.1: Operating conditions of 7 algal lutein production experiments

Operating conditions	Exp1	Exp2	Exp3	Exp4	Exp5	Exp6	Exp7
Initial Biomass (g L^{-1})	0.07	0.07	0.07	0.07	0.07	0.07	0.07
Initial Nitrate (mM)	8.8	30	8.8	8.8	8.8	30	8.8
Inflow rate (mL h^{-1})	3.0	3.0	3.0	3.0	3.0	3.0	3.0
Influent nitrate (M)	0.5	0.5	0.1	0.1	0.1	0.5	0.1
Light intensity (μmol $m^{-2}s^{-1}$)	300	600	150	480	600	480	300

The aim of this section is to introduce GP regression in the context of a discrete time, dynamic black-box model for a biosystem given a set of time series measurements (data sets). It is emphasized that the measurements are taken at a constant sampling frequency. Therefore, in this paper we consider the dynamic system to be in the form of:

$$\mathbf{x}(t) = \boldsymbol{F}(\mathbf{x}(t-1), \mathbf{u}(t-1)) \tag{7.20}$$

where t is the discrete time, $\mathbf{x} \in \mathbb{R}^n$ denotes the states, $\mathbf{u} \in \mathbb{R}^m$ denotes the control inputs and $\boldsymbol{F} : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$ resembles the nonlinear transition dynamics. It is assumed that the control inputs $\mathbf{u}$ are deterministic.

In simple words Equation 7.20 means that the system at time step t will be predicted using measurements and inputs at the previous time step $t-1$. This is the general approach when a real experiment is conducted, where past data are used to predict and hence optimize the process at a future time.

For a biosystem the states are commonly given by concentrations, while a common input is the feed rate of a substrate. For example for the lutein case study the state vector is given by $\mathbf{x} = [C_X, C_N, C_L]^T$, where C_X represents the concentration of algal biomass, C_N the concentration of nitrate and C_L the concentration of lutein; while the control inputs are given by $\mathbf{u} = [Li, F_N]^T$, where Li denotes the light intensity and F_N the inflow rate of nitrate.

7.3.2 Data preparation

To model the multi-input, multi-output system in Equation 7.20, we employ independent GPs for each output, *i.e.* each output is modeled by a separate GP. The training procedure is therefore the same as outlined in Section 7.2. The data is consequently given, such that each GP can be trained with the same multivariate inputs, but with different single dimensional outputs.

GPs are identified from input-output data pairs, and can be adopted to approximate the dynamic behavior described by Equation 7.20 given a set of measurements. The first step consists of preparing available data points for the GP training. Assuming several laboratory experiments have been conducted, we are commonly given the initial conditions $\mathbf{x}(0)$ and the measurements of $\mathbf{x}(t)$ at constant time intervals over different experimental runs with known controls $\mathbf{u}(t)$. From s distinct experimental runs, we obtain data over s time series (data sets), which gives us data in the form of $\left\{\mathbf{x}^{(i)}(0), \ldots, \mathbf{x}^{(i)}(T_i)\right\}_{i\in\{1,\ldots,s\}}$ and $\left\{\mathbf{u}^{(i)}(0), \ldots, \mathbf{u}^{(i)}(T_i-1)\right\}_{i\in\{1,\ldots,s\}}$, where T_i denotes the number of time intervals in experimental run i. We assume that the sampling rate of all experiments remains constant. In this research we use the augmented vector $\mathbf{x}_a(t) = [\mathbf{x}(t), \mathbf{u}(t)]^T \in \mathbb{R}^{n+m}$ as inputs and the differences $\boldsymbol{\Delta}_{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}(t-1) + \boldsymbol{\epsilon} \in \mathbb{R}^n$ as regression targets, where $\boldsymbol{\epsilon}$ denotes measurement noise. The regression targets define what we want to predict,*i.e.* we aim to predict the change of the states at each stage using GPs.

The input-output data is consequently given by collecting the measurements of

all experiments in the matrices $\mathbf{X}'$ and $\mathbf{Y}'$:

$$\mathbf{X}' = \begin{bmatrix} [\mathbf{x}^{(1)}(0)^T, \mathbf{u}^{(1)}(0)^T] \\ \vdots \\ [\mathbf{x}^{(1)}(T_1-1)^T, \mathbf{u}^{(1)}(T_1-1)^T] \\ [\mathbf{x}^{(2)}(0)^T, \mathbf{u}^{(2)}(0)^T] \\ \vdots \\ [\mathbf{x}^{(2)}(T_2-1)^T, \mathbf{u}^{(2)}(T_2-1)^T] \\ \vdots \\ [\mathbf{x}^{(s)}(0)^T, \mathbf{u}^{(s)}(0)^T] \\ \vdots \\ [\mathbf{x}^{(s)}(T_s-1)^T, \mathbf{u}^{(s)}(T_s-1)^T] \end{bmatrix}^T, \mathbf{Y}' = \begin{bmatrix} \mathbf{x}^{(1)}(1)^T - \mathbf{x}^{(1)}(0)^T \\ \vdots \\ \mathbf{x}^{(1)}(T_1)^T - \mathbf{x}^{(1)}(T_1-1)^T \\ \mathbf{x}^{(2)}(1)^T - \mathbf{x}^{(2)}(0)^T \\ \vdots \\ \mathbf{x}^{(2)}(T_2)^T - \mathbf{x}^{(2)}(T_2-1)^T \\ \vdots \\ \mathbf{x}^{(s)}(1)^T - \mathbf{x}^{(s)}(0)^T \\ \vdots \\ \mathbf{x}^{(s)}(T_s)^T - \mathbf{x}^{(s)}(T_s-1)^T \end{bmatrix}^T \tag{7.21}$$

where $\mathbf{X}' \in \mathbb{R}^{(n+m)\times N}$ are the training inputs, $\mathbf{Y}' \in \mathbb{R}^{n\times N}$ are the training targets and is the total number N of input-output data pairs.

Note that what we are proposing is for the GP to predict the change of the states over a fixed time interval given previous states and inputs (*e.g.* given biomass concentration, lutein concentration, nitrate concentration, light intensity and nitrate input at time $t-1$, we can predict the increase or decrease on biomass concentration, lutein concentration and nitrate concentration from time $t-1$ to time t).

The GPs were trained with transformed data. To train the GPs, the inputs were scaled to lie in $[0,1]$. The input scaling was chosen as a popular feature scaling procedure that have been shown to improve the prediction quality [3]. Unlike the output, the equations used for the input do not assume zero mean and instead account for the mean of the input, consequently a zero mean scaling is not required. The outputs were scaled to have mean 0 to match the zero mean assumption introduced in Section 7.2 and a standard deviaiton of 1. Transformations also help to set the priors of the hyperparameters introduced in Section 7.2 (Equation 7.15), since normalized data behave in a more predictable fashion. The described transformations are accomplished as follows:

$$\mathbf{X}^{(i)} = \mathbf{A}\mathbf{X}'^{(i)} - \mathbf{b} \tag{7.22}$$

where $\mathbf{X}_i$ and $\mathbf{X}^{(i)}$ are the i^{th} row and column of matrix $\mathbf{X}$ respectively, $\mathbf{X}'_i$ and $\mathbf{X}'^{(i)}$ are the i^{th} row and column of matrix $\mathbf{X}'$ respectively,
$\mathbf{b} = [\min \mathbf{X}'_1, \ldots, \min \mathbf{X}'_{n+m}]^T$ and

$$\mathbf{A} = \text{diag}([1/(\max \mathbf{X'}_1 - \min \mathbf{X'}_1), \ldots, 1/(\max \mathbf{X'}_{n+m} - \min \mathbf{X'}_{n+m})])$$

$$\mathbf{Y}^{(i)} = \mathbf{C}\mathbf{Y'}^{(i)} - \mathbf{d} \tag{7.23}$$

where $\mathbf{Y}^{(i)}$ is the i^{th} column of matrix $\mathbf{Y}$, $\mathbf{Y'}^{(i)}$ the i^{th} column of the matrix $\mathbf{Y'}$, $\mathbf{d} = [\overline{\mathbf{Y'}}_1, \ldots, \overline{\mathbf{Y'}}_n]^T$ and $\mathbf{C} = \text{diag}([1/std_1, \ldots, 1/std_n])$, where $\overline{\mathbf{Y'}}_i$ is the sample mean and std_i the sample standard deviation of row i of matrix $\mathbf{Y'}$.

7.3.3 Training of Gaussian processes

The inputs of the GP is the concatenated vector of states and deterministic control inputs $[\mathbf{x}^T, \mathbf{u}^T]^T$, where $\mathbf{x} = [C_X, C_N, C_L]^T$ represents the concentration of biomass (C_X), concentration of nitrate (C_N) and concentration of lutein (C_L), while $\mathbf{u} = [Li, F_N]^T$ denotes the light intensity (Li) and inflow rate of nitrate (F_N). The training outputs are given by the differences of the states at each time step in the training data. The three independent GPs, one for each state, were constructed based on the procedure outlined in the subsequent sections. The parameters were optimized over their log-values with priors set on their log-values as well to ensure positiveness as was shown in Section 7.2 in Equation 7.15. The same Gaussian priors were used for all states with mean and variances given in Table 7.2, which were set to ensure that the parameters do not take too large or too small values and hence to prevent overfitting. It is possible to use the same Gaussian priors for all states due to the data transformation outlined in Section 7.2. The standard deviation of the initial state was taken to be 5% of the initial state, based on the current experimental measurement accuracy. The initial covariance matrix is hence given by $\mathbf{\Sigma_x}(0) = \text{diag}([(0.05C_X(0))^2, (0.05C_N(0))^2, (0.05C_L(0))^2])$.

Table 7.2: Variance and mean of Gaussian prior distributions on the log of the hyperparameters

Hyperparameter	Mean	variance
$\log(\lambda_1), \ldots, \log(\lambda_n)$	0.0	1.0
$\log(\alpha)$	0.0	2.0
$\log(\sigma_\epsilon)$	-6.0	4.0

The detailed training procedure is explained in the following subsections.

7.3.4 Gaussian process prior

The GP regression framework is designed for one-step ahead predictions by identifying a latent function $\mathbf{f}(\cdot)$ to predict $\mathbf{\Delta_x}(t)$ given $\mathbf{x}_a(t-1)$:

$$\mathbf{\Delta_x}(t)|\mathbf{x}_a(t-1) = \mathbf{f}(\mathbf{x}_a(t-1)) + \boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_\epsilon}) \tag{7.24}$$

to approximate the subsequent states by:

$$\mathbf{x}(t)|\mathbf{x}_a(t-1) \approx \mathbf{x}(t-1) + \mathbf{f}(\mathbf{x}_a(t-1)) \tag{7.25}$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^n$ represents the measurement noise, which is assumed to be normally distributed with covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} = \text{diag}([\sigma_{\epsilon}^{(1)^2}, \ldots, \sigma_{\epsilon}^{(n)^2}])$.

In Figure 7.2 an illustration is shown for a latent function representing a time series of state x that is modeled by a GP using several noisy observations.

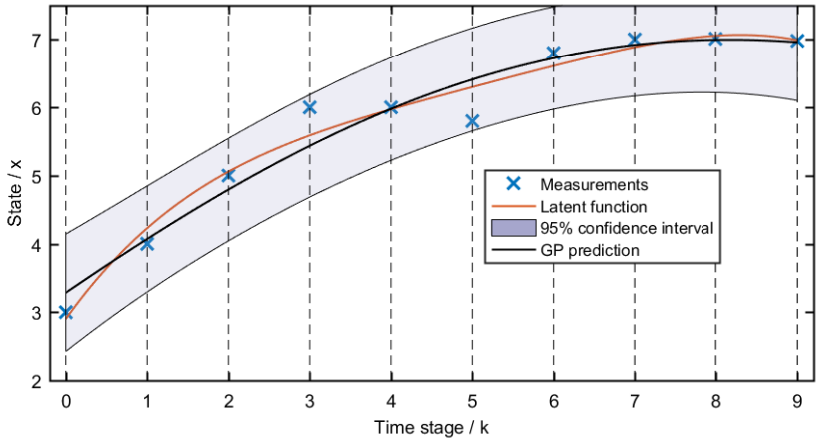


Figure 7.2: Illustration of a latent function of a times series modeled by a GP through a finite number of measurements. The confidence region predicted by the GP is also shown.

Commonly, GPs are employed for multi-input, single-output problems as was introduced in Section 7.2. An effective extension proposed in this research to multi-outputs is to use a separate, independent GP for each output [220], where independence means that the outputs are assumed to be uncorrelated. The latent function in Equation 7.25, is therefore given by:

$$\mathbf{f}(\mathbf{x}_a) = [f^{(1)}(\mathbf{x}_a), \ldots, f^{(n)}(\mathbf{x}_a)]^T \tag{7.26}$$

where each component $f^{(i)}(\mathbf{x}_a))$ is modeled separately by a GP and $\mathbf{x}_a$ is an arbitrary input.

Given that essentially the same process is carried out n-times with different output data, a superscript (i) was added to refer to the separate GPs for each output dimension. We can therefore write that $f^{(i)}(\mathbf{x}_a)$ is distributed as a GP as follows,

the same as Section 7.2:

$$f^{(i)}(\mathbf{x}_a) \sim GP(m^{(i)}(\mathbf{x}_a), k^{(i)}(\mathbf{x}_a, \mathbf{x}'_a)) \tag{7.27}$$

where $\mathbf{x}'_a$ is an arbitrary input, $m^{(i)}(\mathbf{x}_a)$ and $k^{(i)}(\mathbf{x}_a, \mathbf{x}'_a)$ are separate mean and covariance functions for each component $f^{(i)}(\mathbf{x}_a)$ with different parameter values.

As shown in Equation 7.24, $\Delta_x^{(i)}$ is a noisy observation to $f^{(i)}(\mathbf{x}_a)$ perturbed by additive Gaussian noise:

$$\Delta_x^{(i)} = f^{(i)}(\mathbf{x}_a) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, {\sigma_\epsilon^{(i)}}^2) \tag{7.28}$$

Due to the additive property of Gaussian distributions, the observation $\Delta_x^{(i)}$ of $f^{(i)}(\mathbf{x}_a)$ also follows a GP with the same mean, but larger covariance, see Equation 7.10:

$$\Delta_x^{(i)} \sim GP(m^{(i)}(\mathbf{x}_a), k^{(i)}(\mathbf{x}_a, \mathbf{x}'_a) + \sigma_\epsilon^2 \delta(\mathbf{x}_a, \mathbf{x}'_a)) \tag{7.29}$$

Without loss of generality we consider the prior mean function to be zero for each GP, $m^{(i)}(\mathbf{x}_a) := 0$. We propose to use the squared-exponential (SE) covariance function for all the GPs, which is a frequently applied stationary covariance function [200]. The SE covariance function can then be stated as follows for each GP [220]:

$$k^{(i)}(\mathbf{x}_a, \mathbf{x}'_a) = {\alpha^{(i)}}^2 \exp\left(-\frac{1}{2}(\mathbf{x}_a - \mathbf{x}'_a)^T \mathbf{\Lambda}^{(i)} (\mathbf{x}_a - \mathbf{x}'_a)\right) \tag{7.30}$$

where each SE function is parameterized with different parameter values to model the separate GPs indicated by i, $\mathbf{\Lambda}^{(i)} = \text{diag}([{\lambda_1^{(i)}}^{-2}, \ldots, {\lambda_n^{(i)}}^{-2}])$ and ${\alpha^{(i)}}^2$ is the signal variance.

The hyperparameters that define the GPs are given in Equations 7.28 and 7.30 and are given by the vectors $\mathbf{\Theta^{(i)}} = [\log(\lambda_1^{(i)}), \ldots, \log(\lambda_n^{(i)}), \log(\alpha^{(i)}), \log(\sigma_\epsilon^{(i)})]^T$. The GPs for one-step ahead predictions are obtained by using the data defined in Equations 7.22 and 7.23. For each output dimension in $\mathbf{Y}$, *i.e.* for each row in $\mathbf{Y}$, a separate GP needs to be trained (fitted) . In particular, given N training points, n independent GPs are trained with the same input data $\mathbf{X}$ and different response data $\mathbf{y}^{(i)} = \mathbf{Y}_i^T$, where $\mathbf{y}^{(i)} \in \mathbb{R}^N$ is the transpose of the i^{th} row of $\mathbf{Y}$, $\mathbf{Y}_i^T$. The following steps need to be carried out for all $i = 1, \ldots, n$ independent GPs and are basically implemented based on the steps outlined in Section 7.2.

7.3.5 Gaussian process posterior

The prior GP of $f^{(i)}(\mathbf{x}_a)$ and the observation $\Delta_x^{(i)}$ can be expressed as:

$$f^{(i)}(\mathbf{x}_a) \sim GP(0, k^{(i)}(\mathbf{x}_a, \mathbf{x}'_a)), \quad \Delta_x^{(i)} \sim GP(0, k^{(i)}(\mathbf{x}_a, \mathbf{x}'_a) + {\sigma_\epsilon^{(i)}}^2 \delta(\mathbf{x}_a, \mathbf{x}'_a)) \quad (7.31)$$

To build the posterior distribution of these functions knowledge of the observations needs to be incorporated. This is accomplished by considering the joint distribution of observations $\mathbf{y}^{(i)}$ and the response at an arbitrary input $\mathbf{x}_a$, $f^{(i)}(\mathbf{x}_a)$, of the latent function. We assume the input $\mathbf{x}_a$ to be deterministic. This is the same process we used for the derivation in Section 7.2 and can be denoted as follows:

$$\begin{bmatrix} f^{(i)}(\mathbf{x}_a) \\ \mathbf{y}^{(i)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \Sigma_f^{(i)} & \mathbf{\Sigma}_{f,\mathbf{y}}^{(i)} \\ \mathbf{\Sigma}_{\mathbf{y},f}^{(i)} & \mathbf{\Sigma}_{\mathbf{y}}^{(i)} \end{bmatrix} \right) \quad (7.32)$$

where $\Sigma_f^{(i)} = k^{(i)}(\mathbf{x}_a, \mathbf{x}_a)$, $\mathbf{\Sigma}_{f,\mathbf{y}}^{(i)} = [k^{(i)}(\mathbf{x}_a, \mathbf{x}_1), \ldots, k^{(i)}(\mathbf{x}_a, \mathbf{x}_N)]$, $\mathbf{\Sigma}_{\mathbf{y},f}^{(i)} = \mathbf{\Sigma}_{f,\mathbf{y}}^{(i)T}$ and $\mathbf{\Sigma}_{\mathbf{y}}^{(i)} \in \mathbb{R}^{N \times N}$ is the covariance matrix of the data whose entries are given by $\Sigma_{\mathbf{y}jk}^{(i)} = k^{(i)}(\mathbf{x}_{aj}, \mathbf{x}_{ak}) + {\sigma_\epsilon^{(i)}}^2 \delta(\mathbf{x}_{aj}, \mathbf{x}_{ak})$, where $\mathbf{x}_{aj}$ refers to the vector of the j^{th} column of $\mathbf{X}$.

By conditioning $f^{(i)}(\mathbf{x}_a)$ on the observations according to the joint Gaussian distribution given in Equation 7.32 and using the identity given in Equation 7.6, we obtain the posterior predictive distribution of $f^{(i)}(\mathbf{x}_a)$ [220]:

$$f^{(i)}(\mathbf{x}_a)|\mathbf{y}^{(i)} \sim \mathcal{N}(m_f^{(i)}(\mathbf{x}_a), {\sigma_f^{(i)}}^2(\mathbf{x}_a)) \quad (7.33)$$

$$m_f^{(i)}(\mathbf{x}_a) = \mathbf{\Sigma}_{f,\mathbf{y}}^{(i)} (\mathbf{\Sigma}_{\mathbf{y}}^{(i)})^{-1} \mathbf{y}^{(i)} \quad (7.34)$$

$${\sigma_f^{(i)}}^2(\mathbf{x}_a) = \Sigma_f - \mathbf{\Sigma}_{f,\mathbf{y}}^{(i)} (\mathbf{\Sigma}_{\mathbf{y}}^{(i)})^{-1} \mathbf{\Sigma}_{\mathbf{y},f}^{(i)} \quad (7.35)$$

where the mean $m_f^{(i)}$ refers to the best-estimate of the latent function value, while the variance ${\sigma_f^{(i)}}^2$ is a measure of the uncertainty of this prediction.

The predictive distribution of the observation $\Delta_x^{(i)}$ at the test-input is given by the same expressions, except that the term ${\sigma_\epsilon^{(i)}}^2$ needs to be added to the right hand side of Equation 7.35 [84]:

$$\Delta_x^{(i)}|\mathbf{y}^{(i)} \sim \mathcal{N}(m_f^{(i)}(\mathbf{x}_a), {\sigma_f^{(i)}}^2(\mathbf{x}_a) + \sigma_\epsilon^{2(i)}) \quad (7.36)$$

7.3.6 Hyperparameter training

The first step for GP regression is to determine hyperparameter values using MAP, since these are generally unknown *a priori*. This again has to be carried out for every GP separately. In this work, we assume independent Gaussian distributions on the hyperparameters:

$$\Theta_j^{(i)} \sim \mathcal{N}(\mu_{\Theta_j}^{(i)}, {\sigma_{\Theta_j}^{(i)}}^2) \tag{7.37}$$

where $\mu_{\Theta_j}^{(i)}$ is the mean and ${\sigma_{\Theta_j}^{(i)}}^2$ the variance of the prior Gaussian distribution for the hyperparameter Θ_j, which are specified in Table 7.2 for the lutein case study.

Following from Equation 7.32, the log-likelihood of the posterior density of the hyperparameters can be stated as follows [220]:

$$\begin{aligned}\mathcal{L}^{(i)}(\boldsymbol{\Theta}) = &-\frac{1}{2}\log(|\boldsymbol{\Sigma}_{\mathbf{y}}^{(i)}|) - \frac{1}{2}\mathbf{y}^{(i)T}(\boldsymbol{\Sigma}_{\mathbf{y}}^{(i)})^{-1}\mathbf{y}^{(i)} - \frac{N}{2}\log(2\pi) + \\ &\sum_j(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log({\sigma_{\Theta_j}^{(i)}}^2) - \frac{1}{2{\sigma_{\Theta_j}^{(i)}}^2}(\Theta_j - \mu_{\Theta_j}^{(i)})^2)\end{aligned} \tag{7.38}$$

The MAP estimates of $\boldsymbol{\Theta}$ are then given by:

$$\boldsymbol{\Theta}_{MAP}^{(i)} \in \arg\max_{\boldsymbol{\Theta}} \mathcal{L}^{(i)}(\boldsymbol{\Theta}) \tag{7.39}$$

We now have separate optimal hyperparameter vectors $\boldsymbol{\Theta}_{MAP}^{(i)}$ for each output $\mathbf{y}^{(i)}$. We will refer to $k^{(i)}(\mathbf{x}_a, \mathbf{x}'_a)$ as the covariance function with hyperparameter values according to $\boldsymbol{\Theta}_{MAP}^{(i)}$ and hyperparameters with superscript (i) as optimal values from $\boldsymbol{\Theta}_{MAP}^{(i)}$ in the subsequent sections.

7.3.7 One-step ahead predictions

Next with the hyperparameters determined previously, predictions can be made one-step ahead through the GP framework, using the predictive distribution given in 7.33-7.36. The functions $f^{(i)}(\mathbf{x_a})$ were defined to be able to predict the difference vector at each time-stage, which can be used for one-step ahead predictions as follows [71]:

$$\mathbf{x}(t)|\mathbf{x}(t-1) \sim \mathcal{N}(\mathbf{m}_{\mathbf{x}}(t), \boldsymbol{\Sigma}_{\mathbf{x}}(t)) \tag{7.40}$$

$$\mathbf{m}_{\mathbf{x}}(t) = \mathbf{m}_{\mathbf{x}}(t-1) + \mathbf{C}\mathbf{m}_{\mathbf{f}}(\mathbf{x}_a(t-1)) - \mathbf{d} \tag{7.41}$$

$$\boldsymbol{\Sigma}_{\mathbf{x}}(t) = \mathbf{C}\boldsymbol{\Sigma}_{\mathbf{f}}(\mathbf{x}_a(t-1))\mathbf{C}^T \tag{7.42}$$

$$\mathbf{x}_a(t-1) \sim \mathcal{N}(\mathbf{A}^{-1}([\mathbf{m_x}^T(t-1), \mathbf{m_u}^T(t-1)]^T + \mathbf{b}), \\ \mathbf{A}^{-1}(\text{diag}(\mathbf{\Sigma_x}(t-1), \mathbf{\Sigma_u}(t-1))\mathbf{A}^{-T}) \tag{7.43}$$

where $\mathbf{m_f}(\mathbf{x}_a(t-1)) = [m_f^{(1)}(\mathbf{x}_a(t-1)), \ldots, m_f^{(n)}(\mathbf{x}_a(t-1))]^T$ is a stacked vector of independent predictions and the corresponding diagonal matrix of the variances at $\mathbf{x}(t-1)$ is $\mathbf{\Sigma_f}(\mathbf{x}_a(t-1)) = \text{diag}([\sigma_f^{(1)^2}(\mathbf{x}_a(t-1)), \ldots, \sigma_f^{(n)^2}(\mathbf{x}_a(t-1))])$. The equations for $m_f^{(i)}$ and $\sigma_f^{(i)^2}$ can be found in Equations 7.34 and 7.35. Let us emphasize that $\mathbf{m_f}(\mathbf{x}_a(t-1))$ refers to a difference and hence it needs to be added to $\mathbf{m_x}(t-1)$ to calculate the mean of the state at the following time step $\mathbf{m_x}(t)$. The definitions of $\mathbf{A}, \mathbf{b}, \mathbf{C}$ and $\mathbf{d}$ can be found in Equations 7.22 and 7.23, and are used to transform the predictions of the GP to obtain predictions of the true states denoted by $\mathbf{x}$, while $\mathbf{x}_a$ needs to be transformed to match the transformation of the input data.

Let $\mathbf{y_x}$ be the observation of $\mathbf{x}(t)$, then $\mathbf{y_x}$ has the same mean as $\mathbf{x}(t)$, but a larger covariance:

$$\mathbf{y_x}(t)|\mathbf{x}(t-1) \sim \mathcal{N}(\mathbf{m_x}(t), \mathbf{\Sigma_x}(t) + \mathbf{\Sigma_\epsilon}) \tag{7.44}$$

where $\mathbf{y_x}(t)$ has the same distribution as $\mathbf{x}(t)$ with the difference that the measurement noise $\mathbf{\Sigma_\epsilon}$ needs to be added to the covariance matrix.

7.3.8 Multi-step ahead prediction

Using the one-step ahead predictions from the GPs we wish to make multi-step ahead predictions by repeatedly applying Equations 7.41, 7.42 and 7.43. It is, however, important to emphasize that the input to the GP is now a normally distributed random variable, while in GP regression the input is generally assumed to be deterministic. In the one-step ahead predictions in 7.2 the input was essentially deterministic, since we conditioned on it.

In other words, the propagation of uncertainty for a multi-step prediction is demonstrated here. In particular, we assume a joint Gaussian distribution on the test input $\mathbf{x}_a$, $p(\mathbf{x}_a) = \mathcal{N}(\mathbf{m}_{\mathbf{x}_a}, \mathbf{\Sigma}_{\mathbf{x}_a})$. Obtaining the predictive distribution $p(\mathbf{f}(\mathbf{x}_a)|\mathbf{Y}, \mathbf{m}_{\mathbf{x}_a}, \mathbf{\Sigma}_{\mathbf{x}_a})$ of $\mathbf{f}(\mathbf{x}_a)$ at the test input $\mathbf{x}_a$ is now obtained by integrating out $\mathbf{x}_a$, which is however analytically intractable, so that an approximation method is needed [71]. We assume that the predictive distribution of $\mathbf{f}(\mathbf{x}_a)|\mathbf{Y}, \mathbf{m}_{\mathbf{x}_a}, \mathbf{\Sigma}_{\mathbf{x}_a}$ is Gaussian, such that the distribution is fully specified by its mean and covariance. For the SE covariance function in Equation 7.30 exact moment matching is possible, *i.e.* the predictive distribution $p(\mathbf{f}(\mathbf{x}_a)|\mathbf{Y}, \mathbf{m}_{\mathbf{x}_a}, \mathbf{\Sigma}_{\mathbf{x}_a})$ is approximated by a Gaussian which has the same mean and covariance as the true distribution [73]. In the multivariate case the predictive mean vector $\mathbf{m_f}(\mathbf{x}_a)$ for an uncertain input

$\mathbf{x}_a$ is given by Equation 7.45. The target dimensions in general co-vary such that the covariance matrix $\boldsymbol{\Sigma}_\mathbf{f}(\mathbf{x}_a)$ is not diagonal anymore. The covariances on the diagonal can be found using Equation 7.47, while the cross-covariances can be determined using Equation 7.48. The expressions for the mean and covariance of $\mathbf{f}(\mathbf{x}_a)|\mathbf{Y}, \mathbf{m}_{\mathbf{x}_a}, \boldsymbol{\Sigma}_{\mathbf{x}_a})$ are then given by equations involving quantities of all GPs [71]:

$$\mathbf{f}(\mathbf{x}_a)|\mathbf{Y}, \mathbf{m}_{\mathbf{x}_a}, \boldsymbol{\Sigma}_{\mathbf{x}_a} \sim \mathcal{N}(\mathbf{m}_\mathbf{f}(\mathbf{x}_a), \boldsymbol{\Sigma}_\mathbf{f}(\mathbf{x}_a)) \tag{7.45}$$

$$\mathbf{m}_\mathbf{f}(\mathbf{x}_a) = \mathbf{q}^{(i)T}\boldsymbol{\beta}^{(i)} \tag{7.46}$$

$$\Sigma_\mathbf{f}^{(ii)}(\mathbf{x}_a) = {\alpha^{(i)}}^2 + \boldsymbol{\beta}^{(i)T}\mathbf{Q}^{(ii)}\boldsymbol{\beta}^{(i)} - \mathrm{tr}((\boldsymbol{\Sigma}_\mathbf{y}^{(i)})^{-1}\mathbf{Q}^{(ii)}) - m_f^{(i)}(\mathbf{x}_a)^2 \tag{7.47}$$

$$\Sigma_\mathbf{f}^{(ij)}(\mathbf{x}_a) = \boldsymbol{\beta}^{(i)T}\mathbf{Q}^{(ij)}\boldsymbol{\beta}^{(j)} - m_f^{(i)}(\mathbf{x}_a)m_f^{(j)}(\mathbf{x}_a) \tag{7.48}$$

where $\boldsymbol{\beta}^{(i)} = (\boldsymbol{\Sigma}_\mathbf{y}{}^{(i)})^{-1}\mathbf{y}^{(i)}$, $\mathbf{m}_\mathbf{f}(\mathbf{x}_a) = [m_f^{(1)}(\mathbf{x}_a), \ldots, m_f^{(n)}(\mathbf{x}_a)]^T$, $q_p^{(i)} = {\alpha^{(i)}}^2|\boldsymbol{\Sigma}_{\mathbf{x}_a}\boldsymbol{\Lambda}^{(i)} + \mathbf{I}|^{-1/2}\exp(-\frac{1}{2}(\mathbf{m}_{\mathbf{x}_a} - \mathbf{x}_{ap})^T(\boldsymbol{\Sigma}_{\mathbf{x}_a} + (\boldsymbol{\Lambda}^{(i)})^{-1})^{-1}(\mathbf{m}_{\mathbf{x}_a} - \mathbf{x}_{ap}))$ and

$$\boldsymbol{\Sigma}_\mathbf{f}(\mathbf{x}_a) = \begin{bmatrix} \Sigma_\mathbf{f}^{(11)}(\mathbf{x}_a) & \ldots & \Sigma_\mathbf{f}^{(1n)}(\mathbf{x}_a) \\ \vdots & \ddots & \vdots \\ \Sigma_\mathbf{f}^{(n1)}(\mathbf{x}_a) & \ldots & \Sigma_\mathbf{f}^{(nn)}(\mathbf{x}_a) \end{bmatrix}$$

$$Q_{pq}^{(ij)} = k^{(i)}(\mathbf{x}_{ap}, \mathbf{m}_{\mathbf{x}_a})k^{(j)}(\mathbf{x}_{aq}, \mathbf{m}_{\mathbf{x}_a})|\mathbf{R}|^{-1/2}\times\exp\left(\frac{1}{2}(\boldsymbol{\nu} - \mathbf{m}_{\mathbf{x}_a})^T\mathbf{R}^{-1}\boldsymbol{\Sigma}_{\mathbf{x}_a}(\boldsymbol{\nu} - \mathbf{m}_{\mathbf{x}_a})\right) \tag{7.49}$$

where $\mathbf{R} = \boldsymbol{\Sigma}_{\mathbf{x}_a}(\boldsymbol{\Lambda}^{(i)} + \boldsymbol{\Lambda}^{(j)}) + \mathbf{I}$ and $\boldsymbol{\nu} = \boldsymbol{\Lambda}^{(i)}(\mathbf{x}_{ap} - \mathbf{m}_{\mathbf{x}_a}) + \boldsymbol{\Lambda}^{(j)}(\mathbf{x}_{aq} - \mathbf{m}_{\mathbf{x}_a})$. The superscripts i and j refer to the various quantities with respect to the i^{th} and j^{th} Gaussian process, respectively, of the n independent Gaussian processes trained. The vector $\mathbf{x}_{ap}$ is the p^{th} training input contained in $\mathbf{X}$, *i.e.* the p^{th} column of $\mathbf{X}$.

We are now able to make multivariate, multi-step ahead predictions by recursively applying Equations 7.45 to 7.49 together with the following equations to propagate the state through the predicted $\boldsymbol{\Delta}_\mathbf{x}(t)$ [71]:

$$\mathbf{x}(t) \sim \mathcal{N}(\mathbf{m}_\mathbf{x}(t), \boldsymbol{\Sigma}_\mathbf{x}(t)) \tag{7.50}$$

$$\mathbf{m}_\mathbf{x}(t) = \mathbf{m}_\mathbf{x}(t-1) + \mathbf{C}\mathbf{m}_\mathbf{f}(\mathbf{x}_a(t-1)) - \mathbf{d} \tag{7.51}$$

$$\boldsymbol{\Sigma}_\mathbf{x}(t) = \boldsymbol{\Sigma}_\mathbf{x}(t-1) + \mathbf{C}\boldsymbol{\Sigma}_\mathbf{f}(\mathbf{x}_a(t-1))\mathbf{C}^T + 2\mathrm{cov}(\boldsymbol{\Delta}_\mathbf{x}(t-1), \mathbf{u}(t-1)) \tag{7.52}$$

$$\mathbf{x}_a(t-1) \sim \mathcal{N}(\mathbf{A}^{-1}([\mathbf{m}_\mathbf{x}{}^T(t-1), \mathbf{m}_\mathbf{u}{}^T(t-1)]^T + \mathbf{b}), \tag{7.53}$$

$$\mathbf{A}^{-1}(\text{diag}(\mathbf{\Sigma}_{\mathbf{x}}(t-1), \mathbf{\Sigma}_{\mathbf{u}}(t-1))\mathbf{A}^{-T})$$

where $\text{cov}(\mathbf{\Delta}_{\mathbf{x}}(t-1), \mathbf{u}(t-1))$ is 0 and $\mathbf{\Sigma}_{\mathbf{u}} = [0]_{m\times m}$ for deterministic control inputs $\mathbf{u}(t-1)$. For the case when one is interested in a feedback control law, such that the input is given as some function of the current state, $\mathbf{u}(t-1) = \kappa(\mathbf{x}(t-1))$, please refer to [72]. $\mathbf{m}_{\mathbf{x}}(t)$ is the best estimate of the state at k with corresponding covariance $\mathbf{\Sigma}_{\mathbf{x}}(t)$.

Let $\mathbf{y}_{\mathbf{x}}(t)$ be the observations of $\mathbf{x}(t)$, then $\mathbf{y}_{\mathbf{x}}(t)$ has the same mean as $\mathbf{x}(t)$, but a larger covariance:

$$\mathbf{y}_{\mathbf{x}}(t) \sim \mathcal{N}(\mathbf{m}_{\mathbf{x}}(t), \mathbf{\Sigma}_{\mathbf{x}}(t) + \mathbf{\Sigma}_{\boldsymbol{\epsilon}}) \tag{7.54}$$

where $\mathbf{y}_{\mathbf{x}}(t)$ has the same distribution as $\mathbf{x}(t)$ with the difference that the measurement noise $\mathbf{\Sigma}_{\boldsymbol{\epsilon}}$ needs to be added to the covariance matrix.

The initial state $\mathbf{x}(0)$ and covariance matrix $\mathbf{\Sigma}_{\mathbf{x}}(0)$ need to be given from which the state can be then propagated to an arbitrary time horizon recursively. The initial state is generally known, while the covariance matrix can be obtained by error propagation. For the lutein case study these were stated in 7.2.

7.4 Artificial neural network

In this section a brief introduction is given to artificial neural networks (ANNs). Currently, ANNs are used as the standard black-box modeling tool to simulate both traditional chemical engineering processes and emerging biological systems. In order to demonstrate the outstanding characteristics of GPs, in this research ANNs are considered as the benchmark to investigate the simulation and prediction capabilities of GPs for bioprocess systems engineering. In particular, a state-of-the-art ANN construction strategy was recently developed to simulate microalgal lutein production in [77]. Thus, this ANN model will be used to compare against the current constructed GPs.

In general, an ANN is a system of nodes or 'neurons', based on graph theory, organized in layers which are bound together by a series of mono-directional connections and are meant to represent biological learning and computation. These nodes accept inputs and generate outputs which are then either returned or used as inputs to another layer of neurons [98, 121, 269]. The source and destination of the connections depend on the structure of the type of network chosen. In specific, the ANN presented in the recent study was built up based upon a type called multi-layer perceptron [77], where connections only point to the next layer, as in the feed-forward case. A schematic of a multilayer ANN is illustrated in Fig. 7.3. Other architectures have been developed over years, however, they are not within the scope of this study.

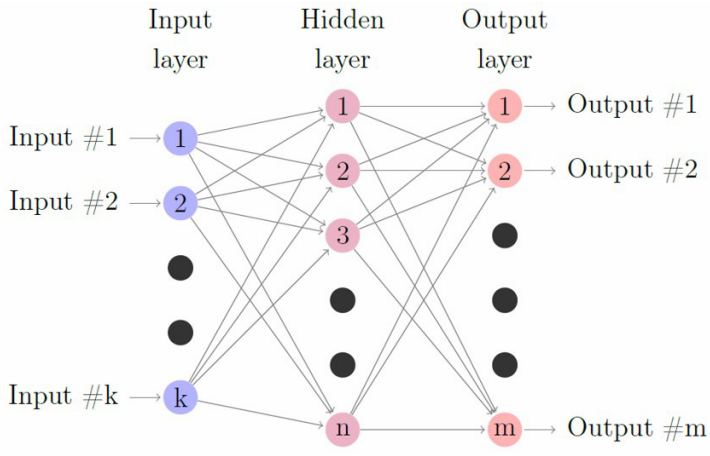


Figure 7.3: Schematic of an ANN with a single hidden layer, k inputs and m outputs

Within the last decade, there has been a push toward the use of ANNs in chemical engineering [115], where they have found use as estimators and as part of simulation of processes [204, 192, 90, 180]. As an example, ANNs have been employed for modeling and prediction in traditional processes such as in distillation, fuel production and in the design of fuel cells [196, 90, 15]. They have been also extensively used to simulate difficult processes such as the influence of parameters in catalyst preparation through experimental data [107], or even to extract such rules from existing literature [197]. More importantly, the modeling of different useful processes involving microorganisms have been successfully tackled by ANNs [254, 189]. Furthermore, there has been newly reported research focusing on applying ANNs to identify optimal operating conditions for the production of microalgae biorenewables, such as the work published in [179] and [81].

Specific to the previous study where ANNs were applied to simulate microalgae biomass growth and lutein production [77], a two hidden layer ANN consisting of 15 nodes in each hidden layer, 5 inputs (biomass concentration, lutein production, light intensity, nitrate concentration, nitrate inflow rate), and 3 outputs (change of biomass concentration, lutein production, nitrate concentration) was constructed, together with another one-layer ANN comprising the same inputs and outputs but 20 nodes in the hidden layer. These optimal network structures were determined through the cutting-edge hyperparameter selection framework, and the ANNs were demonstrated to be of high accuracy and predictive capability. Thus, they are selected in this study for comparison. The hyperparameter selection algo-

rithm is the following:

Algorithm 7.1: ANN hyperparameter selection

Initialization:

Define the set of possible hidden layers Ω and possible neurons per hidden layer Λ. In this research these sets were set to $\Omega = \{1, 2, 3\}$, $\Lambda = \{3, 5, 10, 15, 20, 25\}$.

Define possible neural network structures as $\Upsilon = \Omega \times \Lambda$, where $\times$ denotes the Cartesian product.

Define the set of time-series Γ. In this study Γ was defined by all the experimental time-series.

1. For ANN_k (neural network structure) in Υ
   (a) For i in Γ: Select this time-series i as the test-set and group the rest as the cross-validation data-set Ψ
   (b) Initialize regularization penalty λ to a small value; in this study $\lambda = 0.001$
      i. For each time-series j in Ψ: Select this time-series j as the cross-validation set and group the rest as the training-set Θ
         A. Train ANN_k on Θ
         B. Compute training and cross-validation errors
         C. Increase regularization penalty λ
         D. If training error has stopped its sharp decrease and cross-validation error increases continue to step (ii). Else, return to (A).
      ii. Use ANN_k to predict i (test-set) and compute the error
   (c) Compile test-set errors for ANN_k
2. Compare test-set errors for all ANN structures, and determine the optimal structure for 1 and 2 hidden layer ANNs

Parameter λ is a penalty imposed to the size of the weights of the ANN to avoid overfitting. In this implementation, the term divides the weight values, hence it starts at a small value and increases gradually throughout the algorithm. Step 1,b,i,D allows to compute the best regularization penalty parameter to reduce overfitting.

7.5 Results and discussion

7.5.1 Comparison between Gaussian process and artificial neural network

The performance of the techniques was compared by leaving out the data set for a single experiment and in turn predicting the trajectory of this experiment using the GP methodology outlined in Section 7.2 (multi-step ahead prediction). For the ANN the same cross-validation procedure was implemented for networks with 1 and 2 hidden layers. This approach was applied to all experiments, thus in total there are predictions for all 7 experiments. More importantly, to verify the predictive capability of the GP for complex biological systems, the comparison between GP and ANNs in the current study is executed through an offline framework where only initial operating conditions (and nitrate inflow rate) are provided, and the models have to predict the entire dynamic behavior of the process.

The results are shown in Figures 7.4-7.7, on which the predictions from the GP, ANN with 1 hidden layer and 2 Hidden layers were plotted for each of the 7 experiments. In addition, error bars were added to the GP predictions showing the 99% confidence regions of the latent functions given by the GPs. The confidence regions presented are based on the experimental observations as given by Equation 7.44. This is the correct confidence region to show, since we are trying to see in how far the experimental data perturbed by measurement noise is contained within the confidence region, while the true underlying function is unknown. It is necessary to recall that ANNs are not able to estimate the confidence region of their predicted results.

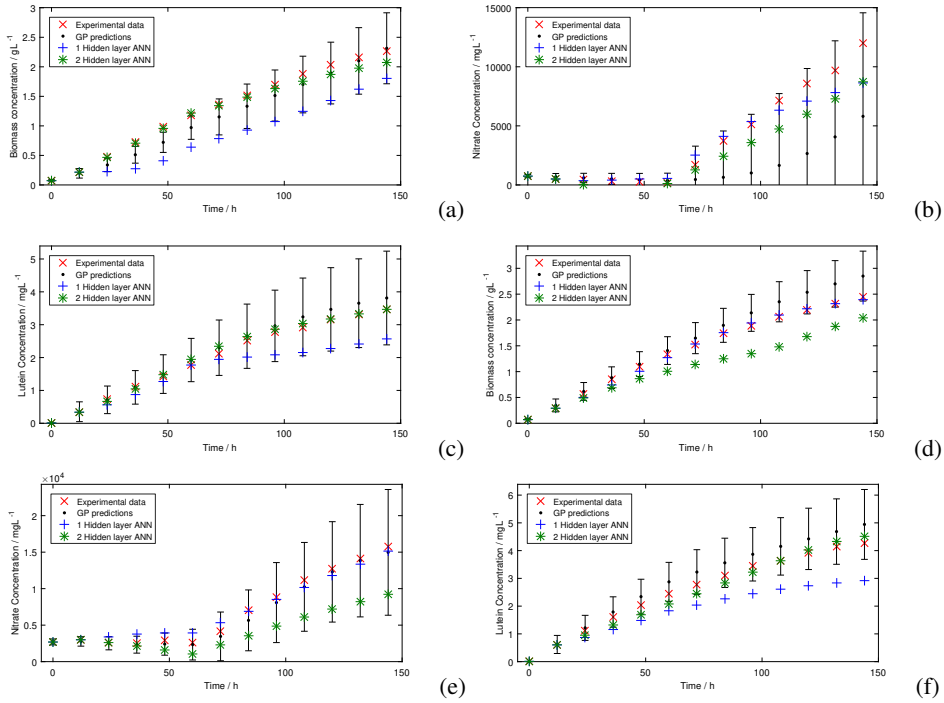


Figure 7.4: Cross-validation for data set 1, where (a) is the dynamic performance for biomass concentration, (b) for nitrate concentration and (c) for lutein concentration; and for cross-validation for data set 2, (d) is the dynamic performance for biomass concentration, (e) for nitrate concentration and (f) for lutein concentration

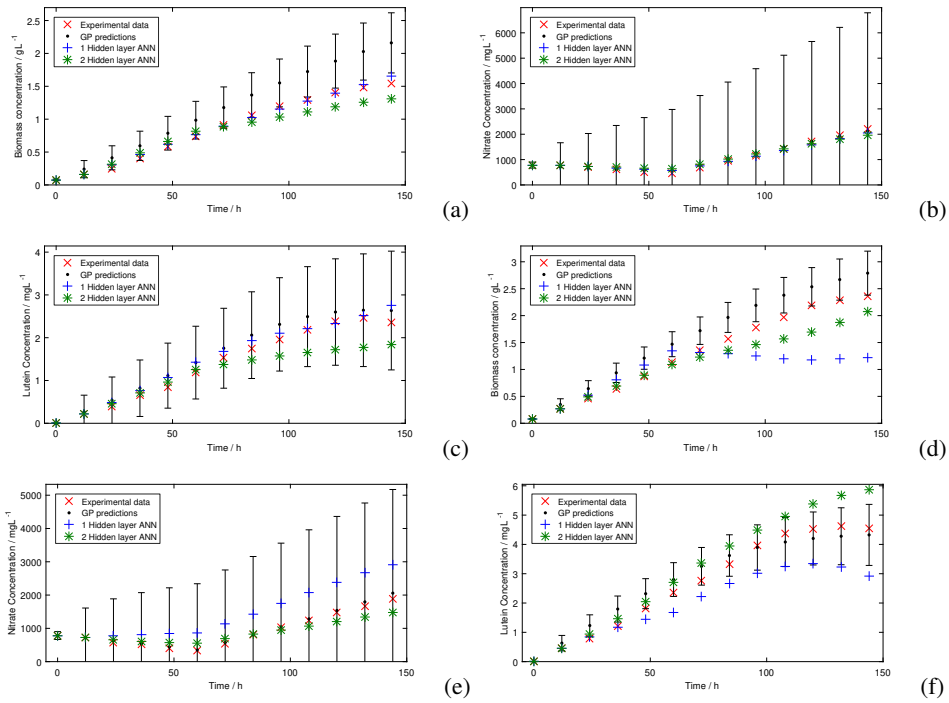


Figure 7.5: Cross-validation for data set 3, where (a) is the dynamic performance for biomass concentration, (b) for nitrate concentration and (c) for lutein concentration; and for cross-validation for data set 4, (d) is the dynamic performance for biomass concentration, (e) for nitrate concentration and (f) for lutein concentration

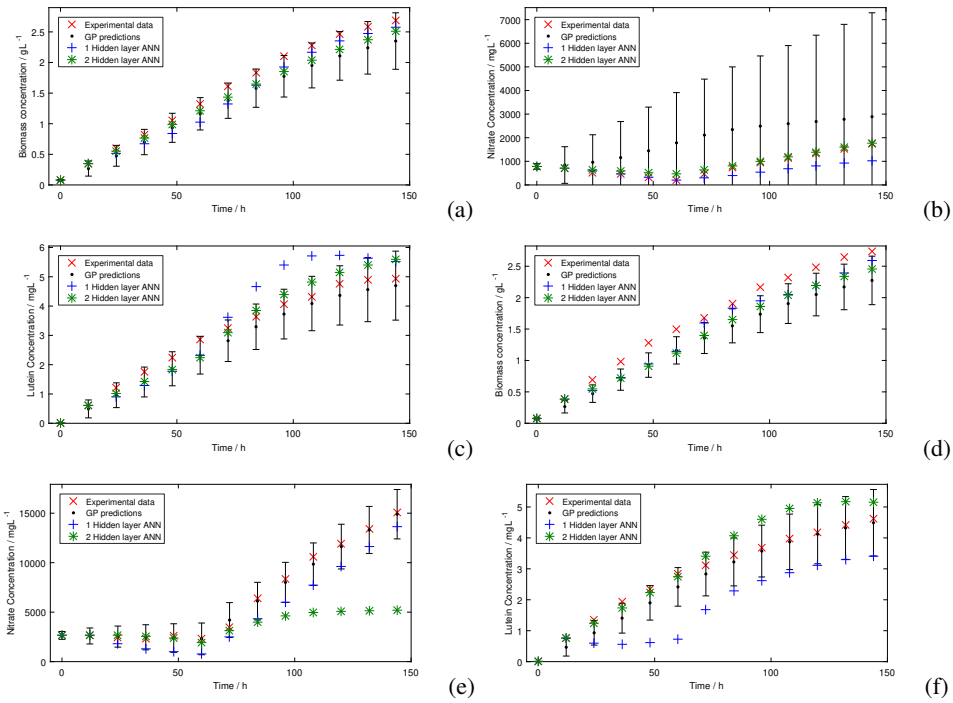


Figure 7.6: Cross-validation for data set 5, where (a) is the dynamic performance for biomass concentration, (b) for nitrate concentration and (c) for lutein concentration; and for cross-validation for data set 6, (d) is the dynamic performance for biomass concentration, (e) for nitrate concentration and (f) for lutein concentration

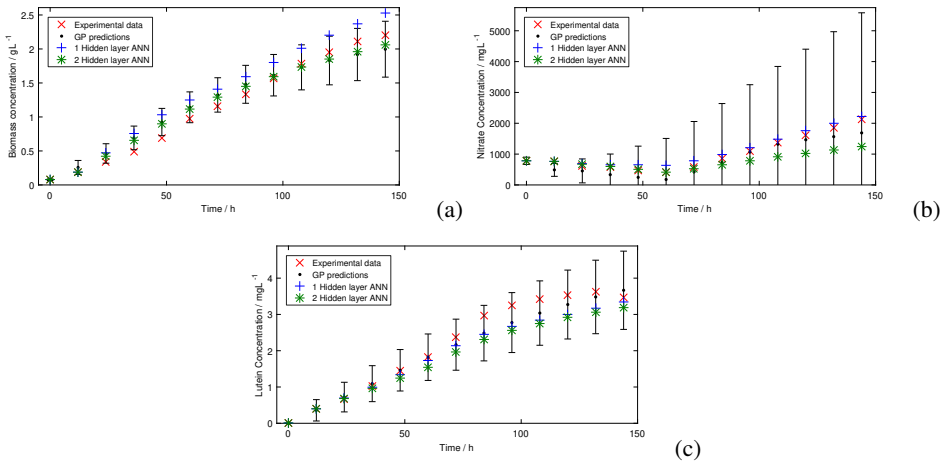


Figure 7.7: Cross-validation for data set 7, where (a) is the dynamic performance for biomass concentration, (b) for nitrate concentration and (c) for lutein concentration

From Figures 7.4-7.7 it can be appreciated that all models show good predictions of the results given the limited data available, although it is difficult to determine the prediction superiority of one method over the other. Each method can be seen to be better at predicting certain datasets, while none is always superior.

From the GP predictions it can be seen that the experimental data is rarely outside the confidence interval provided, this being an encouraging result for future research. However, it can be also seen that the error propagation is relatively large in the case of the nitrate concentration. This is mainly attributed to the fact that the measurement noise is taken to be the same for both high and low nitrate concentrations. For low nitrate concentrations, in fact, percentage-wise the measurement noise is similar to that of the high nitrate concentration. Nonetheless, in an actual implementation, states would be monitored online and real-time datasets would be updated to GPs constantly. As a result, the prediction uncertainty would be contained within a narrower region.

However, in Fig. 7.5 (d) and Fig. 7.6 (d) the data points indeed fall slightly outside the confidence regions. This does indicates two limitations of the GP framework proposed in this paper. The first one is that errors may not be entirely reliable, since the error propagation involves approximating the true distribution using only the mean and variance, which may lead to too low noise if the true distribution is either particular skewed or multi-modal. This could be alleviated in future work by using particle-based approaches instead, which in the limit approximates the true distribution exactly [100]. In addition, the hyperparameters were set using

optimization, which ignores the uncertainty of the parameter values themselves. A more accurate, but expensive solution to this problem is to integrate the hyperparameters out instead [220], which could also be tested in future work.

Finally, through a design of experiment framework, it is possible to improve the accuracy of the GP models. One efficient way to conduct this is to design experiments in areas in which the variance of the GP is high, since this shows regions that have high sparsity of data-points. Furthermore, if we are interested in finding optimal operating conditions, it is sensible to try to learn the model more accurately in areas that are promising. This has been used to great success in the global optimization community by sequentially designing experiments that trade off exploring unknown regions and exploiting regions in which good operating conditions have already been observed [226]. This shows again an advantage of GPs over ANNs due to the availability of an uncertainty measure.

In addition to the visual comparison given in Figures 7.4-7.7, the mean square error (MSE) over each time series was calculated for each machine learning algorithm, *i.e.* the difference of the prediction from the measurements was squared and averaged over each time series. The values of the MSE for the biomass concentration, nitrate concentration and lutein concentration can be found in Tables 7.3, 7.4 and 7.5 respectively. The smallest value (best performing algorithm) among the three models is highlighted in bold.

Table 7.3: Comparison of MSE for GP, 1 Hidden-layer ANN (1HL-ANN), 2 Hidden-layer ANN (2HL-ANN) from Exp 1-7 for **biomass concentration** (g^2L^{-2}) . The best performing algorithm in terms of MSE is highlighted in bold.

Exp	GP	1HL-ANN	2HL-ANN
1	0.028	0.270	**0.010**
2	0.052	**0.011**	**0.011**
3	0.130	**0.002**	0.021
4	0.121	0.381	**0.068**
5	0.086	0.030	**0.028**
6	0.133	**0.057**	0.368
7	0.024	0.062	**0.015**

Table 7.4: Comparison of MSE for GP, 1 Hidden-layer ANN (1HL-ANN), 2 Hidden-layer ANN (2HL-ANN) from Exp 1-7 for **nitrate concentration** (g^2L^{-2}) . The best performing algorithm in terms of MSE is highlighted in bold.

Exp	GP	1HL-ANN	2HL-ANN
1	13.290	**1.552**	2.812
2	**0.745**	4.919	26.448
3	**0.006**	0.008	0.016
4	**0.002**	0.454	0.420
5	0.125	0.142	**0.015**
6	**0.226**	0.951	10.626
7	0.062	**0.019**	0.167

Table 7.5: Comparison of MSE for GP, 1 Hidden-layer ANN (1HL-ANN), 2 Hidden-layer ANN (2HL-ANN) from Exp 1-7 for **lutein concentration** (mg^2L^{-2}) . The best performing algorithm in terms of MSE is highlighted in bold.

Exp	GP	1HL-ANN	2HL-ANN
1	0.045	0.324	**0.011**
2	**0.175**	0.967	0.658
3	0.059	**0.031**	0.144
4	**0.145**	0.793	0.432
5	**0.092**	0.631	0.162
6	**0.090**	1.337	0.227
7	**0.064**	0.127	0.203

From Tables 7.3, 7.4 and 7.5, it is concluded that the GP shows a comparable performance to the ANN (either 1 or 2 layers). For example, it can be seen that GP attains the best prediction result on 5 out of the 7 experiments and comes second for the remaining 2 experiments when predicting lutein concentration. Similarly, it possesses the best prediction on 4 out of the 7 experiments and comes second twice when estimating the trajectory of nitrate concentration in the current study. Even though the best prediction for biomass concentration is always an ANN, the current constructed GP still provides a comparable result (second best prediction on 4 experiments) for the majority of the experimental tests. This clearly indicates the great predictive capability of GPs and promising potential for bioprocess systems engineering applications.

Moreover, significant attention should be paid on the fact that the current used ANNs were constructed based on advanced methodologies through which the optimal structure of ANNs were identified and their predictive capability is maximized

[77]. However, as GPs have never been applied to describe and understand the behavior of complex biological systems, specific strategies capable of identifying the optimal structure of GPs are not available yet. Therefore, the comparable predictive capability and performance of the current GPs against the optimal ANNs strongly suggests the potential of GPs on bioprocess modeling and optimization.

The most important contribution of the GP is that a confidence region is simultaneously estimated during process prediction, and it is found that experimental measurements in almost all the 7 experiments fall within this region. Such a region is essential for sensitive bioprocess optimization and for the implementation of robust optimization strategies (*e.g.* worst-case scenario optimization), as the safety of a bioprocess is in general given higher priority than the process yield. Furthermore, the GP never gave catastrophically unreliable trajectory predictions. This conclusion further emphasizes that GPs can not only provide an accurate prediction for long-term biosystems, but also contribute a reliable estimation for dynamic bioprocess design, modeling and control.

7.5.2 Dynamic optimization with stochastic constraints

One of the main advantages of GP regression over more common regression methods, such as ANNs, is that it gives us a measure of prediction uncertainty. In this section, we show how this measure can be used in optimization to ensure that the optimal solution remains in the validity range of the model. The objective of the optimization is to find operating conditions to yield the maximum lutein concentration by the end of the process at the 144^{th} hour, *i.e.* with a time horizon of length $N = 12$. The operating conditions are given by control actions of light intensity and the nitrate inflow rate chosen at each time stage. In addition, the mean of the initial concentrations of biomass and nitrate was also varied.

The model adopted in this section is determined by using all the data from the 7 experiments given in Table 7.1 and following the procedure in Section 7.2. The control actions are taken to be deterministic. The first constraint given in brackets in Equation 7.55 is the stochastic constraint that limits the variance of lutein at the final stage to be below 0.025 mg L^{-1}, and hence the lutein concentration to lie in a confidence region of 95% with ± 0.025 mg L^{-1}. This measure was chosen, as it directly targets the relevant uncertainty of the measure to be optimized, while still considering all other uncertainties since these are iteratively used as noisy input to obtain the final prediction. The overall optimization problem is given below:

Dynamic optimization problem using GP model

$$\max_{\mathbf{U},\, \mathbf{m_x}(0)} \quad \mathbb{E}\left[C_{L_i}(N)\right] = m_x^{(3)}\left(\mathbf{x}_a(N)\right) \tag{7.55a}$$

subject to:

$$\Sigma_x^{(3,3)}(N) \leq 0.025 \tag{7.55b}$$

$$\mathbf{\Sigma_x}(0) = \text{diag}\left(\left[\left(0.05 m_x^{(1)}(0)\right)^2, \left(0.05 m_x^{(2)}(0)\right)^2, \left(0.05 m_x^{(3)}(0)\right)^2\right]\right) \tag{7.55c}$$

$$[0,0]^T \leq \mathbf{u}(k) \leq [600, 1.5]^T \quad \forall k \in \{0, ..., N-1\} \tag{7.55d}$$

$$[0,0,0]^T \leq \mathbf{m_x}(0) \leq [0.5, 6000, 0]^T \tag{7.55e}$$

$$(7.48) - (7.51) \quad \forall k \in \{1, \ldots, N\} \tag{7.55f}$$

where $\mathbf{U} = [\mathbf{u}(0), \ldots, \mathbf{u}(N-1)]$ is a matrix of control inputs at each time interval, $\Sigma_x^{(3,3)}(t)$ is the 3rd diagonal element of the state matrix $\mathbf{\Sigma_x}(t)$ corresponding to the variance of lutein concentration at time stage k and $m_x^{(i)}(t)$ is the i$^{\text{th}}$ dimension of the vector $\mathbf{m_x}(t)$ corresponding to the expected value of the respective concentrations at time stage k.

The optimization in Equation 7.55 was conducted using the fmincon function in Matlab with 20 multistart initial points chosen according to a *maximin* Latin hypercube. Choosing initial starting points according to a Latin hypercube scaled to the upper and lower bounds of the decision variables has been shown to yield good optimization results in [222]. With stochastic constraints for example the optimization converged to either of six solutions with relatively small variations. The optimal solution yielded a value of lutein of 4.94mg L^{-1}. The optimal solution was observed twice in the optimization procedure. The mean of the solutions obtained was 4.4mg L^{-1} with a standard deviation of 0.33mg L^{-1}. The optimal trajectories are shown in Figures 7.8 and 7.9, where two optimization results are shown, one with the stochastic constraints and another without.

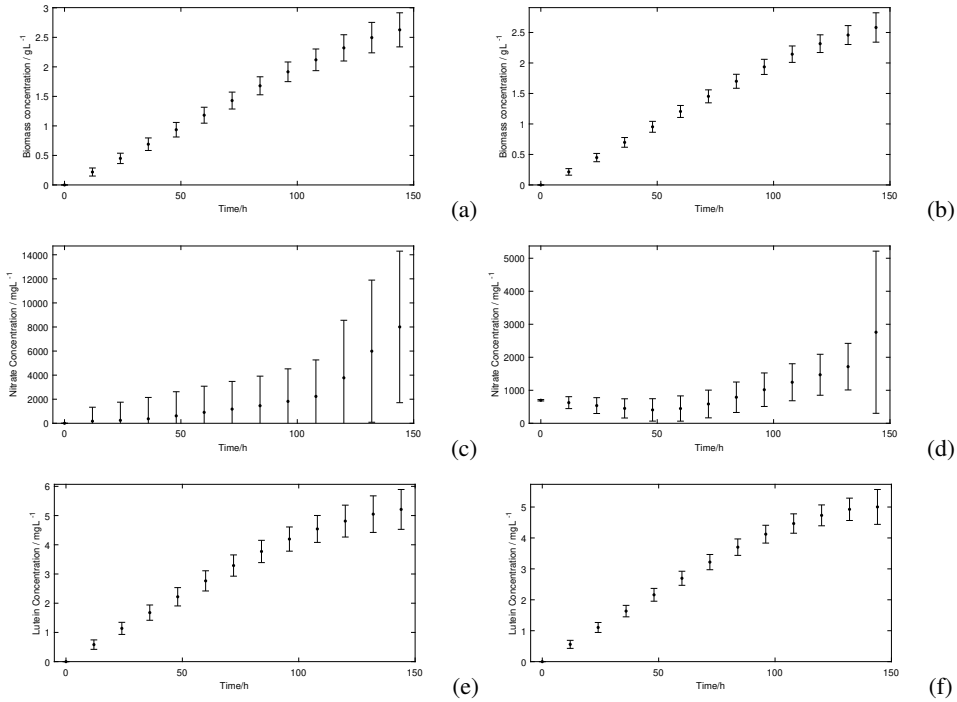


Figure 7.8: Results of dynamic optimization for lutein production. (a), (c), (e): Optimal trajectory of concentrations of biomass, nitrate, and lutein without stochastic constrain, respectively; (b), (d), (f): Optimal trajectory of concentrations of biomass, nitrate, and lutein with stochastic constraint, respectively.

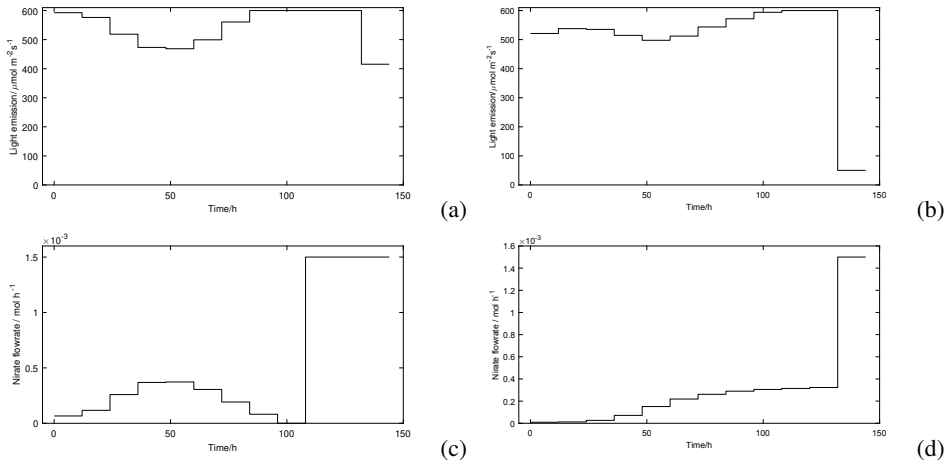


Figure 7.9: Results of the optimal control scheme for lutein production. (a), (c): Optimal control input of light emission and nitrate inflow rate without stochastic constraint, respectively; (b), (d): Optimal control input of light emission and nitrate inflow rate with stochastic constraint, respectively.

From the figures, it is seen that both optimization scenarios yield similar results since the predicted optimal operating conditions lie on the boundary of the model, *i.e.* the optimum is close to the conditions in the second training experiment (Exp2), given in Table 7.1. Comparing the two scenarios, although the optimization without stochastic constraints yields a slightly higher final lutein concentration (5.10 mg L^{-1}), the case with stochastic constraints (4.95 mg $^{-1}$), can be seen to show a lower uncertainty on all the state trajectories, in particular when predicting nitrate concentrations. This suggests that in order to reduce the uncertainty of process optimization and guarantee the safety of underlying biosystems, it is necessary to embed stochastic constraints into the optimization framework. It is also worth noting that the optimal result with stochastic constraints is closer to the second training experiment than without it. This was executed to minimize the uncertainty of the model, *i.e.* as shown in Figure 7.1, where uncertainty is substantially higher away from the measurements. Such a result means that experimental trajectories near optimal solutions can be highly valuable. Furthermore, in general, the model uncertainty is relatively high, suggesting that more data should be used to further enhance the optimization results.

7.6 Conclusions

Overall, a new methodology was introduced to construct a dynamic model for biorenewable synthesis and process optimization by using Gaussian process re-

gression. By comparing against ANNs, the high predictive capability and simultaneous uncertainty measure of GPs show an outstanding capacity to simulate and optimize complex biological processes, particularly in cases where the lack of experimental data becomes a severe challenge for the construction of kinetic models. Furthermore, a distinctive feature of GPs is the simultaneous estimation of model uncertainty alongside the real-time optimisation framework, which is difficult to be achieved by other techniques.

In particular, the provision of a confidence region from GPs has the potential to significantly facilitate their application in process scale-up and real-time optimal control for both traditional bioprocesses such as fermentation and newly proposed algae based photo-production systems, since the precise prediction and control action decision-making throughout the entire process is indispensable to guarantee the safety and productivity of these systems. An important issue to note, is that, compared to traditional empirical and phenomenological models, data-driven models are more susceptible to the amount, quality and range of the data used, this is of paramount importance and should be carefully considered before building such models.

Acknowledgement

This project has received funding from the EPSRC project (EP/P016650/1,P65332).

Chapter 8

Nonlinear model predictive control with explicit back-offs for Gaussian process state space models

This chapter is based on **Paper F**: E. Bradford, L. Imsland, and E. A. del Rio-Chanona. Nonlinear model predictive control with explicit back-offs for Gaussian process state space models. In *58th Conference on decision and control (CDC)*, pages 4747–4754. IEEE, 2019.

Summary

Nonlinear model predictive control (NMPC) is an efficient control approach for multivariate nonlinear dynamic systems with process constraints. NMPC does however require a plant model to be available. A powerful tool to identify such a model is given by Gaussian process (GP) regression. Due to data sparsity this model may have considerable uncertainty though, which can lead to worse control performance and constraint violations. A major advantage of GPs in this context is its probabilistic nature, which allows to account for plant-model mismatch. In this paper we propose to sample possible plant models according to the GP and calculate explicit back-offs for constraint tightening using closed-loop simulations offline. These then in turn guarantee satisfaction of chance constraints online despite the uncertainty present. Important advantages of the proposed method over existing approaches include the cheap online computational time and the consid-

eration of closed-loop behaviour to prevent open-loop growth of uncertainties. In addition we show how the method can account for updating the GP plant model using available online measurements. The proposed algorithm is illustrated on a batch reactor case study.

8.1 Introduction

Model predictive control (MPC) is an advanced control method that has found a wide range of applications in industry. The success of MPC can be largely attributed to its ability to deal with multivariate plants and process constraints [165]. Linear MPC theory is relatively mature and well-established in practice. Many systems however display strong nonlinear behaviour motivating the use of nonlinear MPC (NMPC) [6]. NMPC is being progressively more utilized due to the advent of improved non-convex optimization algorithms [27], for example in chemical engineering [28].

An important requirement for NMPC is the availability of an accurate plant model. The development of an adequate model has been cited to take up to 80% of the MPC commissioning effort [244]. NMPC algorithms exploit numerous different models, commonly developed by first principles [188]. These are however often too complex and in addition frequently accompanied by high development costs. Alternatively, black-box dynamic models can be used instead, such as support vector machines [267], neural network models [211], or Gaussian processes (GP) [143].

GPs are an interpolation technique developed by [150] that were popularized by the machine learning community [220]. GP predictions take the form of Gaussian distributions. The mean of this distribution can be interpreted as a deterministic prediction, while the variance of the distribution can be seen as a corresponding measure of uncertainty. In particular, an appropriate measure of uncertainty is difficult to obtain by nonlinear parametric models [143]. In control this uncertainty measure can be exploited for efficiently learning a dynamic model by exploring unknown regions or obtaining robustness by avoiding regions that have an uncertainty that is too high. GPs have found various applications in control, such as reinforcement learning [71], designing probabilistic robust linear controllers [23], or for adaptive control [65]. In particular, GPs have been applied in NMPC to notable success as approximate plant models.

The use of GPs for NMPC was first proposed in [183], in which a GP model is updated online for reference tracking without constraints. In [143] the GP is instead identified offline and used online. The variance is constrained to prevent the NMPC from steering the dynamic system into regions with high uncertainty.

In [139] GPs are updated online to overcome unmodelled periodic errors, while in [167] the GP is used to update the dynamic model online after a fault has occurred. GPs have also been shown as a useful tool to approximate the mean and variance required in stochastic NMPC [36]. While generally these and other works show the feasibility of GP-based NMPC, there is a lack of efficient methods to account for this uncertainty. Model uncertainty can lead to worse performance and feasibility issues of MPC algorithms, which has led to the development of robust MPC [22] and stochastic MPC [175] approaches.

Most works on GP-based NMPC do consider this uncertainty, however the vast majority of proposed algorithms use stochastic uncertainty propagation to accomplish this, see for example [143, 113, 60]. A recent overview of different stochastic uncertainty propagation approaches can be found in [114]. These approaches have some considerable disadvantages. Firstly, there are no known methods to exactly propagate stochastic uncertainties through the GP model, such that only approximate methods exist usually based on linearization or moment-matching. Secondly, the computational time is often increased significantly due to the stochastic propagation itself. Lastly, most works consider open-loop propagation of uncertainties, which can be prohibitively conservative due to open-loop growth of uncertainties.

Recently there have been some works that do consider other robust control methods. In [147] a robust GP-based NMPC algorithm is developed for learning by propagating ellipsoidal sets using linearization, that provides closed-loop stability guarantees. This approach may however suffer from increased computational times, since the ellipsoidal sets need to be propagated online. In [169] an alternative procedure is proposed, which establishes closed-loop stability by bounding the one-step ahead error, however determining the required parameters seems to be non-trivial. Lastly, in [239] a robust control approach is proposed for linear systems, in which the GP is used to represent unmodelled non-linearities. The method robustly stabilizes the linear system despite the unmodelled non-linearities, which may however not have a solution if these uncertainties are too large in magnitude.

In this paper we propose a new approach to account for the uncertainty of GP state space models for NMPC for finite-horizon control problems. The method exploits recent results using so-called explicit back-offs, which can be used to account for stochastic uncertainties to design the NMPC [146, 206]. These rely on generating Monte Carlo (MC) closed-loop simulations of possible plant models. The back-offs are then used to tighten the constraints of the NMPC to obtain probabilistic constraint satisfaction despite the stochastic uncertainties present. Further, to generate the required MC samples of the GPs we employ results from [68, 249], where it is shown how to obtain exact samples of GPs. There are several

advantages of the proposed method. Firstly, the back-offs are determined using closed-loop simulations, such that the problem of open-loop uncertainty growth is avoided. Further, the required computations are carried-out offline, such that the online computational time is nearly unaffected. In addition, the back-offs are designed based on the empirical cumulative distribution function (cdf), which considers the true underlying distribution [242]. Lastly, due to the independence of the samples some probabilistic guarantees can be given. An extended journal paper of this approach can be found in [42].

The paper is structured as follows. In section 8.2 the problem to be solved is outlined. In section 8.3 we give an overview of GPs for our purposes. Section 8.4 then outlines the solution approach using GPs. Section 8.5 describes the semi-batch reactor case study, for which the results and discussions are given in section 8.6. Subsequently, section 8.7 concludes the paper.

8.2 Problem definition

In this paper we consider a nonlinear discrete-time system:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \tag{8.1}$$

where t denotes the discrete time, $\mathbf{x} \in \mathbb{R}^{n_x}$ denotes the states, $\mathbf{u} \in \mathbb{R}^{n_u}$ represents the control inputs, and $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ denotes the corresponding nonlinear dynamics.

It is assumed that the full state is measurable with a noisy output measurement of the next state given by:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \boldsymbol{\nu} \tag{8.2}$$

where $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\nu}})$ is independent Gaussian distributed measurement noise with zero mean and a corresponding covariance function $\boldsymbol{\Sigma}_{\boldsymbol{\nu}} = \mathrm{diag}(\sigma^2_{\nu_1}, \ldots, \sigma^2_{\nu_{n_\mathbf{x}}})$.

Let $\mathbf{z} = (\mathbf{x}, \mathbf{u})$ for convenience with joint dimension $n_\mathbf{z} = n_\mathbf{x} + n_\mathbf{u}$. We assume we are given N noisy function evaluations of $\mathbf{f}(\mathbf{z})$ in Eq.(8.1) according to Eq.(8.2) denoted as $\mathbf{Y}$ with corresponding input data $\mathbf{Z}$:

$$\mathbf{Z} := [\mathbf{z}_1, \ldots, \mathbf{z}_N]^\mathsf{T} \in \mathbb{R}^{N \times (n_x + n_u)} \tag{8.3}$$

$$\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^\mathsf{T} \in \mathbb{R}^{N \times n_x} \tag{8.4}$$

We aim to minimize a finite-horizon cost function:

$$V_T(\mathbf{x}_0, \mathbf{U}) = \sum_{t=0}^{T-1} \ell(\mathbf{x}_t, \mathbf{u}_t) + \ell_f(\mathbf{x}_T) \tag{8.5}$$

where $T \in \mathcal{N}$ is the time horizon, $\mathbf{U} = [\mathbf{u}_0, \ldots, \mathbf{u}_{T-1}]^\mathsf{T}$ is a collection of control inputs, $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is the stage cost, and $\ell_f : \mathbb{R}^{n_x} \to \mathbb{R}$ is the terminal cost.

We assume that the control inputs are subject to hard-constraints, while the states are subject to a joint chance constraint, which can be stated as:

$$\mathbf{u}_t \in \mathbb{U}_t \qquad \forall t \in \{0, \ldots, T-1\} \tag{8.6}$$

$$\mathbb{P}\left\{\bigcap_{t=0}^{T} \{\mathbf{x}_t \in \mathbb{X}_t\}\right\} \geq 1 - \epsilon \tag{8.7}$$

where $\mathbb{X}_t$ are defined as nonlinear constraint sets
$\mathbb{X}_t = \{\mathbf{x} \in \mathbb{R}^{n_x} \mid g_j^{(t)}(\mathbf{x}) \leq 0, j = 1, \ldots, n_g\}$

The joint chance constraints are formulated such that the joint event of all $\mathbf{x}_t$ fulfilling the nonlinear constraint sets $\mathbb{X}_t$ is greater than $1 - \epsilon$. It should be noted that the uncertainty in this problem arises from the fact that we do not know $\mathbf{f}(\mathbf{x}, \mathbf{u})$ and are instead given noisy observations of $\mathbf{f}(\mathbf{x}, \mathbf{u})$. We aim to solve this OCP by utilizing GPs to model the unknown dynamics and use the GP approximation to obtain probabilistic guarantees for the closed-loop system.

8.3 Gaussian processes

8.3.1 Gaussian process regression

In this section we give an introduction to GP regression. For a more general overview refer to [220]. GP regression aims to describe an unknown function $f : \mathbb{R}^{n_z} \to \mathbb{R}$ using noisy observations y:

$$y = f(\mathbf{z}) + \nu \tag{8.8}$$

where $\mathbf{z} \in \mathbb{R}^{n_z}$ is the argument of $f()$ and $\nu \sim \mathcal{N}(0, \sigma_\nu^2)$ is Gaussian distributed measurement noise with zero mean and variance σ_ν^2.

GPs consider a distribution over functions, and they can be seen as a generalization of multivariate Gaussian distributions, which can be expressed as:

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)) \tag{8.9}$$

where the mean function $m(\cdot)$ can be interpreted as the "average" shape of the function, while the covariance function $k(\cdot, \cdot)$ accounts for correlations between function values.

The prior GP is defined by the choice of mean function and covariance function. In this study we apply a zero mean function and the squared-exponential (SE)

covariance function [220] [1]:

$$m(\mathbf{z}) := 0 \tag{8.10}$$

$$k(\mathbf{z}, \mathbf{z}') := \alpha^2 \exp\left(-\frac{1}{2}(\mathbf{z} - \mathbf{z}')^\mathsf{T} \boldsymbol{\Lambda}^{-2} (\mathbf{z} - \mathbf{z}')\right) \tag{8.11}$$

where $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^{n_z}$ are arbitrary input vectors, α^2 is the covariance magnitude, and $\boldsymbol{\Lambda}^{-2} := \mathrm{diag}(\lambda_1^{-2}, \ldots, \lambda_{n_z}^{-2})$ is a scaling matrix.

Maximum likelihood estimation is commonly applied to infer the unknown hyperparameters $\boldsymbol{\Psi} := [\alpha, \lambda_1, \ldots, \lambda_{n_z}, \sigma_\nu]^\mathsf{T}$, including σ_ν in case the measurement noise variance is also unknown. Consider N noisy function evaluations, denoted by $\mathbf{Y} := [y_1, \ldots, y_N]^\mathsf{T} \in \mathbb{R}^N$, with corresponding inputs collected in the matrix $\mathbf{Z} := [\mathbf{z}_1, \ldots, \mathbf{z}_N] \in \mathbb{R}^{n_z \times N}$. The log-likelihood of the observed data, ignoring constant terms, is given by:

$$\mathcal{L}(\boldsymbol{\Psi}) := -\frac{1}{2}\log(\det(\mathbf{K})) - \frac{1}{2}\mathbf{Y}^\mathsf{T}\mathbf{K}^{-1}\mathbf{Y} \tag{8.12}$$

with $K_{ij} := k(\mathbf{z}_i, \mathbf{z}_j) + \sigma_\nu^2 \delta_{ij}$ for each pair $(i, j) \in \{1, \ldots, N\}^2$ and the Kronecker delta function δ_{ij}.

The posterior distribution of $f(\mathbf{z})$ at an arbitrary input $\mathbf{z}$, given the input-output data $(\mathbf{Z}, \mathbf{Y})$ and the maximum-likelihood estimates of $\boldsymbol{\Psi}$, follows the Gaussian distribution:

$$f(\mathbf{z})|\mathbf{Z}, \mathbf{Y} \sim \mathcal{N}(\mu_f(\mathbf{z}; \mathbf{Z}, \mathbf{Y}), \sigma_f^2(\mathbf{z}; \mathbf{Z}, \mathbf{Y})) \tag{8.13}$$

with

$$\mu_f(\mathbf{z}; \mathbf{Z}, \mathbf{Y}) := \mathbf{k}(\mathbf{z})\mathbf{K}^{-1}\mathbf{Y} \tag{8.14}$$

$$\sigma_f^2(\mathbf{z}; \mathbf{Z}, \mathbf{Y}) := \alpha^2 - \mathbf{k}(\mathbf{z})\mathbf{K}^{-1}\mathbf{k}(\mathbf{z})^\mathsf{T} \tag{8.15}$$

and $\mathbf{k}(\mathbf{z}) := [k(\mathbf{z}, \mathbf{z}_1) \ \cdots \ k(\mathbf{z}, \mathbf{z}_N)]$. The mean function $\mu_f(\mathbf{z}; \mathbf{Z}, \mathbf{Y})$ in this context is the prediction made by the GP at $\mathbf{z}$, while the variance $\sigma_f^2(\mathbf{z}; \mathbf{Z}, \mathbf{Y})$ provides a measure of uncertainty around this predictor.

8.3.2 Gaussian process state space models

In this section we introduce GP state space models. We aim to identify an unknown state space model from input-output data. GP methodology is usually used

[1] The zero-mean assumption can be achieved by normalizing the data. Also the method presented can be used on any chosen covariance function.

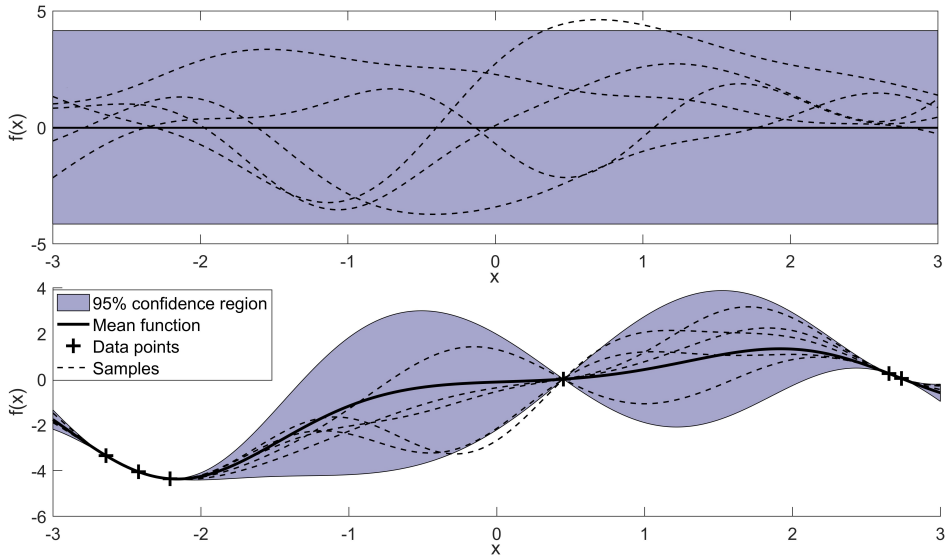


Figure 8.1: Illustration of a GP of a 1-dimensional function perturbed by noise. On the top the prior of the GP is shown, while on the bottom the Gaussian process was fitted to several observations to obtain the posterior. The dashed lines show GP samples.

to model scalar functions with vector inputs, while for the case of vector functions it is common to build a separate, independent GP for each dimension [71]. Let the function in Eq.(8.1) be given by $\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) = [f_1(\mathbf{x}_t, \mathbf{u}_t), \ldots, f_{n_x}(\mathbf{x}_t, \mathbf{u}_t)]^\mathsf{T}$, such that we aim to build a separate GP for each function $f_i(\mathbf{x}_t, \mathbf{u}_t)$. We then build a separate GP according to section 8.3.1 with observations $\mathbf{Y}_i = [y_{i1}, \ldots, y_{iN}]^\mathsf{T}$ and inputs $\mathbf{Z}$ for $i \in \{1, \ldots, n_x\}$, where y_i refers to the i$^{\text{th}}$ dimension of the measurement $\mathbf{y}$ in Eq.(8.2). The posterior Gaussian distribution of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ given the data-set $(\mathbf{Z}, \mathbf{Y})$ in Eq.(8.3) and Eq.(8.4) respectively at an arbitrary input $\mathbf{z} = (\mathbf{x}, \mathbf{u})$ is given by:

$$\mathbf{f}(\mathbf{z})|\mathbf{Z}, \mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}_f(\mathbf{z}; \mathbf{Z}, \mathbf{Y}), \boldsymbol{\Sigma}_f(\mathbf{z}; \mathbf{Z}, \mathbf{Y})) \tag{8.16}$$

with

$$\boldsymbol{\mu}_f(\mathbf{z}; \mathbf{Z}, \mathbf{Y}) = [\mu_f(\mathbf{z}; \mathbf{Z}, \mathbf{Y}_1), \ldots, \mu_f(\mathbf{z}; \mathbf{Z}, \mathbf{Y}_{n_x})]^\mathsf{T} \tag{8.17}$$

$$\boldsymbol{\Sigma}_f(\mathbf{z}; \mathbf{Z}, \mathbf{Y}) = \text{diag}\left(\sigma_f^2(\mathbf{z}; \mathbf{Z}, \mathbf{Y}_1), \ldots, \sigma_f^2(\mathbf{z}; \mathbf{Z}, \mathbf{Y}_{n_x})\right) \tag{8.18}$$

8.3.3 Gaussian process samples

Each sample or realization of a GP in theory yields a deterministic function, however this would require sampling an infinite dimensional stochastic process

and hence there are no known methods to obtain an exact sample from a GP. Instead approximate sampling methods have been used, see for example spectral sampling [46]. Exact samples of GPs can however be obtained in the case that the function needs to be evaluated at only a finite number of points, which is commonly the case for state space models. The exact sampling approach was first proposed in [68] and has been applied in [249] for the optimal design of linear controllers.

Assume we are given a GP state space model as in section 8.3.2 built from an input data-set $\mathbf{Z}$ and corresponding observations $\mathbf{Y}$. We then wish to create a single GP sample from an initial state $\mathbf{x}_0$ for a finite-horizon T given a known control input sequence $\mathbf{U} = [\mathbf{u}_0, \ldots, \mathbf{u}_{T-1}]$. A finite dimensional GP state space model sample over time-horizon T is then represented by a sequence of states. The posterior Gaussian distribution of $\mathbf{x}_1$ is given according to Eq.(8.16) as:

$$\mathbf{x}_1 = \mathbf{f}(\mathbf{z}_0) \sim \mathcal{N}(\boldsymbol{\mu}_f(\mathbf{z}_0; \mathbf{Z}, \mathbf{Y}), \boldsymbol{\Sigma}_f(\mathbf{z}_0; \mathbf{Z}, \mathbf{Y})) \tag{8.19}$$

Now to obtain a sample of $\mathbf{x}_1$, we draw from the Gaussian distribution in Eq.(8.19). Let $\mathbf{x}_1^{(s)}$ refer to this realization of $\mathbf{x}_1$, which is considered a realization of the sampled function. To obtain $\mathbf{x}_2^{(s)}$ we then need to first condition on this realization $\mathbf{x}_1^{(s)}$, since it is part of the sampled function path. The realization $\mathbf{x}_1^{(s)}$ is treated similar to a new training point, however without observation noise (i.e. no σ_v^2 is added to the kernel evaluation $k(\mathbf{z}_0, \mathbf{z}_0)$) and without updating the hyperparameters. If the sampled function revisited the same input, it would lead to the exact same outcome due to the conditioning on a noiseless output.

Now given this new point $\mathbf{x}_1^{(s)}$, we next draw $\mathbf{x}_2^{(s)}$ according to the GP obtained from the updated data-sets. This procedure is recursively repeated until the required time-horizon T has been reached. The overall sampling method is summarized in Algorithm 8.1 below and is illustrated in Fig.8.2. This gives us a single GP sample and hence needs to be repeated multiple times to obtain multiple samples.

8.4 Solution approach

Given the input-output data-sets $\mathbf{Z}$ and $\mathbf{Y}$ we fit a GP state space model, see section 8.3.2. We aim to solve the problem defined in section 8.2 using NMPC based on this GP model. Now the mean model of the GP defines a state trajectory itself, which we will refer to as the *nominal* case. Each sample of the GP defines further deterministic solutions to the GP state space model. Overall each sample of the GP over a finite horizon T is defined by drawing T independent random states as shown in section 8.3.3, which we will refer to as $\boldsymbol{\Delta}$. For convenience we

Algorithm 8.1: Gaussian process trajectory sampling

Input : $\mathbf{z}_0^{(s)} = \mathbf{z}_0$, $\mathbf{U}$, $\mathbf{K}$, $\mathbf{K}^{-1}$, $\mathbf{Z}$, $\mathbf{Y}$, $\mathbf{k}(\mathbf{z})$ T

for *each sampling time* $t = 1, 2, \ldots, T$ **do**

1. Draw $\mathbf{x}_t^{(s)}$ from $\mathcal{N}(\boldsymbol{\mu}_f(\mathbf{z}_{t-1}^{(s)}; \mathbf{Z}, \mathbf{Y}), \boldsymbol{\Sigma}_f(\mathbf{z}_{t-1}^{(s)}; \mathbf{Z}, \mathbf{Y}))$
2. Update $\mathbf{Z} := [\mathbf{Z}^\mathsf{T}, \mathbf{z}_{t-1}^{\mathsf{T}(s)}]^\mathsf{T}$, $\mathbf{Y} := [\mathbf{Y}^\mathsf{T}, \mathbf{x}_t^{\mathsf{T}(s)}]^\mathsf{T}$
3. Update $\mathbf{K} = \begin{bmatrix} \mathbf{K} & \mathbf{k}(\mathbf{z}_{t-1}^{(s)}) \\ \mathbf{k}(\mathbf{z}_{t-1}^{(s)}) & k(\mathbf{z}_{t-1}^{(s)}, \mathbf{z}_{t-1}^{(s)}) \end{bmatrix}$
4. Determine $\mathbf{K}^{-1}$
5. Define $\mathbf{k}(\mathbf{z}) = [\mathbf{k}(\mathbf{z}), k(\mathbf{z}, \mathbf{z}_{t-1}^{(s)})]$

Output : State sequence $\mathbf{x}_1^{(s)}, \ldots, \mathbf{x}_T^{(s)}$

introduce the notation:

$$\mathbf{x}_t = \boldsymbol{\phi}(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x}, \mathbf{U}, \boldsymbol{\Delta}) \tag{8.20}$$

where $\boldsymbol{\phi}(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x}, \mathbf{U}, \boldsymbol{\Delta})$ corresponds to the state at time t, when the initial state is $\mathbf{x}$, the control sequence $\mathbf{U}$ is applied, and the GP realization is given by $\boldsymbol{\Delta}$ given the initial input-output dataset $(\mathbf{Z}, \mathbf{Y})$. Further, by convention let $\boldsymbol{\Delta} = \mathbf{0}$ refer to the *nominal* scenario defined by setting the state space model to the mean function $\boldsymbol{\mu}_f(\mathbf{z}; \mathbf{Y}, \mathbf{Z})$.

8.4.1 Gaussian process model predictive control

In this section we define the NMPC OCP based on the GP *nominal* model given the data-set $(\mathbf{Z}, \mathbf{Y})$. Let the optimization problem be denoted as $P_T(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x})$ for the current state $\mathbf{x}$ at discrete-time t:

$$\begin{aligned} &\underset{\mathbf{U}_{t:T-1}}{\text{minimize}} \quad V_T(\mathbf{x}, \mathbf{U}_{t:T-1}) = \sum_{k=t}^{T-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + \ell_f(\mathbf{x}_T) \\ &\text{subject to} \\ &\mathbf{x}_{k+1} = \boldsymbol{\phi}(k, \mathbf{Y}, \mathbf{Z}; \mathbf{x}, \mathbf{U}_{t:k}, \mathbf{0}) \; \forall k \in \{t, \ldots, T-1\} \\ &\mathbf{x}_{k+1} \in \overline{\mathbb{X}}_{k+1}, \quad \mathbf{u}_k \in \mathbb{U}_k \qquad \forall k \in \{t, \ldots, T-1\} \end{aligned} \tag{8.21}$$

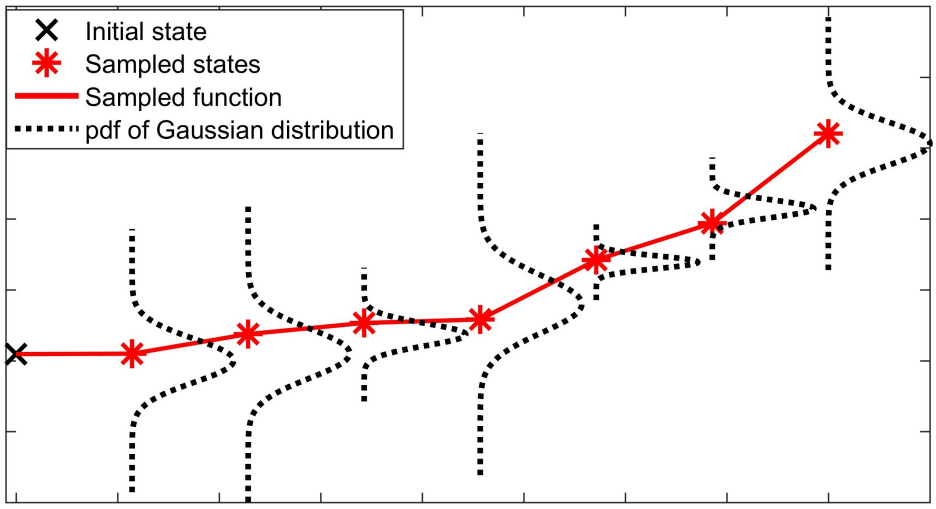


Figure 8.2: Illustration of GP sampling scheme for a 1-dimensional function.

where $\overline{\mathbb{X}}_{k+1}$ is a tightened constraint set denoted by:
$\overline{\mathbb{X}}_{k+1} = \{\mathbf{x} \in \mathbb{R}^{n_x} \mid g_j^{(k)}(\mathbf{x}) + b_j^{(k)} \leq 0\}$, and
$\mathbf{U}_{t:k} = [\mathbf{u}_t, \ldots, \mathbf{u}_k]$.

The variables $b_j^{(k)}$ denote so-called back-offs, which aim to tighten the original constraints $\mathbb{X}_t$ to obtain constraint satisfaction for the real plant model using nominal predictions of the state trajectory as in Eq.(8.21).Let $\mathbf{U}^*_{t:T}(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x}) = [\mathbf{u}^*_t(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x}), \ldots, \mathbf{u}^*_{T-1}(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x})]$ be the optimal control sequence by solving $P_T(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x})$. Only the first of these control actions is applied to the plant at time t. This defines our implicit model predictive control law as $\boldsymbol{\kappa}(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x}) = \mathbf{u}^*_t(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x})$, where the OCP in Eq.(8.21) needs to be re-solved for each new measurement of $\mathbf{x}_t$.

Applying this closed-loop control policy to a GP sample then leads to the following closed-loop response:

$$\mathbf{x}_t^{\text{MPC}}(\boldsymbol{\Delta}, \mathbf{Y}, \mathbf{Z}) = \boldsymbol{\phi}(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x}_0, \mathcal{K}, \boldsymbol{\Delta}) \tag{8.22}$$

where $\mathcal{K} = [\boldsymbol{\kappa}(0, \mathbf{Y}, \mathbf{Z}; \mathbf{x}_0), \ldots, \boldsymbol{\kappa}(t-1, \mathbf{Y}, \mathbf{Z}; \mathbf{x}_{t-1}^{\text{MPC}})]$ is a collection of control actions from the NMPC controller based on observations from the GP plant model given by the realization $\boldsymbol{\Delta}$.

At each time t we are however given a new measurement of $\mathbf{x}_t$ from Eq.(8.2) and we know the previous input, since it is given by $\mathbf{z}_{t-1} = (\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$. Therefore, it may be reasonable to update the mean/nominal GP model of the MPC using this data online. This leads to the following alternative GP NMPC closed-loop

response based on the updated models:

$$\mathbf{x}_t^{\mathrm{MPC,l}}(\boldsymbol{\Delta}, \mathbf{Y}, \mathbf{Z}) = \boldsymbol{\phi}(t, \mathbf{Y}, \mathbf{Z}; \mathbf{x}_0, \mathcal{K}^l, \boldsymbol{\Delta}) \tag{8.23}$$

where the collection of control actions from the GP learning NMPC controller is given as $\mathcal{K}^l = [\boldsymbol{\kappa}(0, \mathbf{Y}, \mathbf{Z}; \mathbf{x}_0), \ldots, \boldsymbol{\kappa}(t-1, \mathbf{Y}^{(t-1)}, \mathbf{Z}^{(t-1)}; \mathbf{x}_{t-1}^{\mathrm{MPC,l}})]$ based on observations from the GP plant model given by GP realization $\boldsymbol{\Delta}$. The datasets $\mathbf{Z}^{(t)}$ and $\mathbf{Y}^{(t)}$ are recursively defined as:

$$\mathbf{Z}^{(t)} = [\mathbf{Z}^{\mathsf{T}(\mathrm{t}-1)}, \mathbf{z}_{t-1}^{\mathsf{T}}]^{\mathsf{T}} \qquad \forall t \in \{1, \ldots, T\} \tag{8.24}$$

$$\mathbf{Y}^{(t)} = [\mathbf{Y}^{\mathsf{T}(\mathrm{t}-1)}, \mathbf{y}_t^{\mathsf{T}}]]^{\mathsf{T}} \qquad \forall t \in \{1, \ldots, T\} \tag{8.25}$$

where $\mathbf{y}_t$ is a measurement obtained from Eq.(8.2).

Note the sampling procedure of the plant model is unchanged, but the learning GP NMPC algorithm is updated based on the most recent measurements. This leads to the mean function being updated in a similar procedure to the GP sampling in Algorithm 8.1, however in this case the noise σ_v^2 is included in the kernel evaluation. The hyperparameters are kept at their nominal value due to the excessive computational cost required to update these. The closed-loop trajectories defined by Eq.(8.22) and Eq.(8.23) are tied to the choice of the tightened constraint set $\overline{\mathbb{X}}$.

8.4.2 Back-off constraints

In this section we outline how to determine the back-off constraints in Eq.(8.21) to obtain probabilistic constraint satisfaction of the real plant as defined in the problem definition in section 8.2 based on the GP description of the plant. The GP provides both a nominal model, but also describes a distribution of many possible plant models based on the initial data-set given. The overall aim is to determine back-off constraints and corresponding tightened constraint sets $\overline{\mathbb{X}}_t$, such that the closed-loop response given by either Eq.(8.22) for GP NMPC without learning or Eq.(8.23) for GP NMPC with learning satisfies the constraint set $\mathbb{X}_t$ with a high probability. Note that the learning GP NMPC has a different closed-loop behaviour from the GP NMPC without learning, such that these yield different back-off values.

We propose to use S independent samples of the GP generated using the procedure in section 8.3.3, which then in turn describe S different possible plant models. The closed-loop response of these is then given by either the GP NMPC without learning:

$$\mathbf{x}_t^{\mathrm{MPC}}(\boldsymbol{\Delta}^{(s)}) = \boldsymbol{\phi}(t; \mathbf{x}_0, \mathcal{K}, \boldsymbol{\Delta}^{(s)}) \quad \forall s \in \{1, \ldots, S\} \tag{8.26}$$

or the GP NMPC with learning:

$$\mathbf{x}_t^{\mathrm{MPC,l}}(\boldsymbol{\Delta}^{(s)}) = \boldsymbol{\phi}(t; \mathbf{x}_0, \mathcal{K}^l, \boldsymbol{\Delta}^{(s)}) \quad \forall s \in \{1, \ldots, S\} \tag{8.27}$$

The aim is now to ensure that $\mathbf{x}_t^{\text{MPC}}(\mathbf{\Delta}^{(s)})$ satisfies $\mathbb{X}_t$ for all t, for all but a few samples to attain a high probability of constraint satisfaction.

It is however very difficult to derive an update rule for the back-off constraints on joint probabilities, i.e. based on the intersection of $\mathbb{X}_k$. Instead, we propose an update rule that is applied point-wise to each nonlinear constraint $g_j^{(t)}(\mathbf{x}_k^{\text{MPC}}(\mathbf{\Delta}^{(s)}))$ following a procedure proposed in [206]. Assume we aim to determine back-off constraints that imply:

$$g_j^{(t)}(\mathbf{x}_k^{\text{MPC}}(\mathbf{0})) + b_j^{(t)} = 0 \implies \mathbb{P}\{g_j^{(t)}(\mathbf{x}_k^{\text{MPC}}(\mathbf{\Delta})) \leq 0\} \geq \delta \tag{8.28}$$

i.e. the back-offs are adjusted such that the satisfaction of the constraints given the nominal model predictions implies satisfaction of the other possible plant models according to the GP distribution with a probability of at least δ.

We then define the empirical cumulative distribution function (ecdf) as:

$$\hat{F}_{g_j^{(t)}} = \frac{1}{S}\sum_{i=1}^{S} \mathbf{1}\{g_j^{(t)}(\mathbf{x}_t^{\text{MPC}}(\mathbf{\Delta}^{(s)})) \leq 0\} \tag{8.29}$$

where $\hat{F}_{g_j}^{(t)}$ is a sample approximation of the chance constraint given in Eq.(8.28) on the RHS.

In [206] it is proposed to iteratively update the back-offs based on Eq.(8.28) using the inverse of the ecdf in Eq.(8.29) [2]. We then iterate over n_b back-off iterations using the approach given in algorithm 8.2.

Algorithm 8.2: Back-off iterative updates

for n_b *back-off iteration* **do**

  for *each sampling time* $t = 1, 2, \ldots, T$ *and constraints* $j = 1, \ldots, n_g^{(t)}$ **do**

    Initialize: Set $b_j^{(t)}$ to some reasonable values

    1. Run S simulations of either Eq.(8.26) without learning or Eq.(8.27) with learning
    2. Update $b_j^{(t)} := \hat{F}^{-1}_{g_j^{(t)}}(\delta) - g_j^{(t)}(\mathbf{x}_k^{MPC}(\mathbf{0}))$

Output : $b_j^{(t)}$

[2] The inverse of ecdf is given by the quantile function.

8.4.3 Probabilistic guarantees

In this section we give some probabilistic guarantees for the problem definition in section 8.2 given fixed back-off values. It should be noted that our problem is posed as a finite-horizon control problem, such that the NMPC implementation has a shrinking horizon. Given S independent GP samples, we can define the following ecdf to approximate the joint probability in Eq.(8.7):

$$\hat{F}_{\mathbb{X}} = \frac{1}{S}\sum_{i=1}^{S} \mathbf{1}\left\{\bigcap_{t=0}^{T}\{\mathbf{x}_t \in \mathbb{X}_t\}\right\} \tag{8.30}$$

where $\hat{F}_{\mathbb{X}}$ is essentially equal to the fraction of trajectories, which violate the joint constraint given in Eq.(8.7).

Theorem 8.4.1 below, which was derived in [242] can be used to obtain probabilistic guarantees based on the ecdf of independent samples. It is therefore important that the sampling of the GP is carried-out using independent MC samples as shown in section 8.3.3.

Theorem 8.4.1. *If $\beta_{cor} = \hat{F}_{\mathbb{X}}$ based on S independent MC samples, then the following lower bound holds on the true probability $\beta = \mathbb{P}\left\{\bigcap_{t=0}^{T}\{\mathbf{x}_t \in \mathbb{X}_t\}\right\}$ with a probability of at least $1-\alpha$:*

$$\beta_{lb} = 1 - betainv\left(\alpha, S+1-\lfloor\beta_{cor}S\rfloor, \lfloor\beta_{cor}S\rfloor\right) \geq \beta \tag{8.31}$$

The operator $\lfloor\rfloor$ denotes rounding towards $-\infty$, and betainv denotes the inverse of the cumulative Beta-distribution.

Feasibility of the original chance constraint in Eq.(8.1) then follows trivially from Theorem 8.4.1 as stated below.

Corollary 8.4.1. *For an unknown plant model that follows the GP distribution identified exactly, such that the uncertainty description of the GP is accurate, and given a $\beta_{lb} \geq 1-\epsilon$ with a probability of $1-\alpha$ that fulfills Theorem 8.4.1, then the chance constraint in Eq.(8.1) holds with a probability no smaller than $1-\alpha$.*

8.4.4 Algorithm

Algorithm 8.3: Back-off GP NMPC

Offline Computations

1. Build GP state-space model from data-sets $\mathbf{Z}, \mathbf{Y}$.
2. Choose initial condition $\mathbf{x}_0$, stage costs ℓ and ℓ_f and constraint sets $\mathbb{X}_t, \mathbb{U}_t \; \forall t \in \{1, \ldots, T\}$
3. Determine explicit back-off constraints using algorithm 8.2 with or without learning.
4. Check probabilistic guarantees as shown in section 8.4.3 to obtain ϵ.

Online Computations

for $t = 0, \ldots, T$ **do**

1. Solve the MPC problem in Eq.(8.21)
2. Apply the first control input of the optimal solution.
3. Measure the state $\mathbf{x}_t$ and update the GP plant model for learning GP NMPC.

8.5 Case study

In this paper we apply the approach to a challenging semi-batch reactor case study adopted from [33], which is an important example of a finite-horizon control problem in chemical engineering. The following series chemical reactions take place in the reactor catalyzed by H_2SO_4:

$$2\mathrm{A} \xrightarrow[(1)]{k_{1\mathrm{A}}} \mathrm{B} \xrightarrow[(2)]{k_{2\mathrm{B}}} 3\mathrm{C}$$

The reactions taking place are all first-order. Chemical reaction (1) is an exothermic reaction, while chemical reaction (2) is endothermic. A cooling jacket is used for temperature control. The control variables are given by the flowrate of pure reactant A entering the reactor and the temperature of the cooling jacket T_0. Overall there are 5 states: concentrations of reactants A, B, and C in mol/L, reactor temperature in K, and the reactor volume in L.

The objective of the case study is to maximize the amount of product *C* at the final time horizon. In addition, there are two path constraints. Firstly, the reactor

temperature needs to be kept below 420K for safety reasons and the volume needs to stay below 800L, which is the capacity of the reactor. The evolution of these states can be described by a differential algebraic equation system, which can be found in [33]. The time horizon T is fixed to 10 with a sampling time of 0.4h giving an overall batch run-time of 4h. The concentrations of A, B and C is initially zero with an initial reactor temperature of 290K and a volume of 100L.

8.6 Results and discussions

In this section the results based on the case study outlined in section 8.5 are presented. Overall 6 different scenarios were run for verification of the approach. We fit an initial GP model using data-sets with 50, 100, and 150 datapoints according to a space-filling Latin hypercube design. Further, each of these was applied to the algorithm presented in section 8.4.4 with online learning and without online learning. The explicit back-offs were determined using 400 GP MC samples for each back-off iteration. The back-offs were adjusted in a total of 5 iterations using algorithm 8.2. Lastly, for the final back-off values the GP NMPC was applied to the real plant represented by the case study equations. The results of these simulations are summarized in Figs.8.3-8.7 and in Table 8.1.

In Fig.8.3 and Fig.8.4 plots are shown of the closed-loop volume and temperature trajectories according to the 400 MC samples of the plant model. The top two graphs of each figure show the trajectories for 50 data points, while the bottom two graphs show it for 100 data points. On the left the graphs show the trajectories considering online learning, while on the right without online learning. We can see that the back-offs are adjusted such that most trajectories are kept below the constraint shown by the red line. Nonetheless, for the trajectories using only 50 data-points many trajectories overshoot the temperature constraint by a significant amount due to the inability of our method to find consistently good back-off values due to the very high uncertainty. For 150 data-points on the other hand the frequency of constraint violation is relatively low. In addition, it can be seen that the learning based method is able to more quickly reach the temperature constraint and also stay closer to it leading to improved performance and less conservative back-offs.

In Fig.8.5 and Fig.8.6 the closed-loop response of the GP NMPC is shown for the "real" plant using the exact case study equations. In the top two graphs of each figure the response is shown using back-offs, while in the bottom figures the response is displayed disregarding back-offs. Further, the left figures show the trajectories without learning, while the right figures utilize the available data. Firstly, it can be seen that disregarding back-offs leads to constraint violations of volume or temperature for all cases due to the mismatch between the "real"

plant and the GP approximation. This in turn is avoided in all cases employing back-offs, except for 50 data points, which overshoots the temperature constraint by a significant amount. This is somewhat expected, since as shown in Fig.8.4 the determined back-offs are inadequate. Lastly, it can be seen that in all 3 cases using back-offs the trajectories using online learning are able to stay closer to the temperature constraint.

Apart from constraint satisfaction, it is also important to look at the performance of the different approaches in terms of the economic objective achieved. This is shown in Fig.8.7, where box-plots are given for the 6 different cases from the objective values obtained at the final back-off iterations. The aim of the NMPC is to maximize the amount of product C. In general one would assume that more data leads to an improved objective value, since then the GP model used is closer to the real plant. We can see that this is mostly true apart for the 50 data points run, which however as mentioned has inadequate back-offs and therefore overshoots the constraints by a large margin. In addition we can see that for both 100 data points and 150 data points learning performs on average better than not learning, which is as expected since it is less conservative and should have a plant model closer to the real plant model.

Lastly, in Table 8.1 we show the probability of constraint violation stated as "Probability", which corresponds to the fraction of the 400 GP NMPC MC trajectories that violated either temperature or volume constraint for the final back-off iteration. "c-Probability" refers to the real guaranteed probability βlb of violation using the theorem outlined in section 8.4.3. We can see a clear trend that the more data we have, the smaller the probability of constraint violation, which is more or less as expected. In particular, for 50 data points without learning we have more MC samples violating the constraints than not, while for 150 data points we are able to guarantee constraint satisfaction of nearly 0.9. In addition, we see that for 50, 100, and 150 data points learning can provide significantly higher probability guarantees than their counter-parts without learning. In addition, computationally times are on-average low in the range of seconds, however they do rapidly increase with the number of data-points.

8.7 Conclusions

In conclusion, a new approach has been proposed for finite-horizon control problems using NMPC in conjunction with GP state space models. The method utilizes the probabilistic nature of GPs to sample deterministic functions of possible plant models. Tightened constraints using explicit back-offs are then determined, such that the closed-loop simulations of these possible plant models is feasible to a high probability. In addition it is shown how probabilistic guarantees

can be derived based on the number of constraint violations from the simulations. It was in addition shown that online learning can be accounted for explicitly in this method, which leads to overall less conservativeness. Lastly, the computational times could be shown to be relatively low, since the constraint tightening is performed offline.

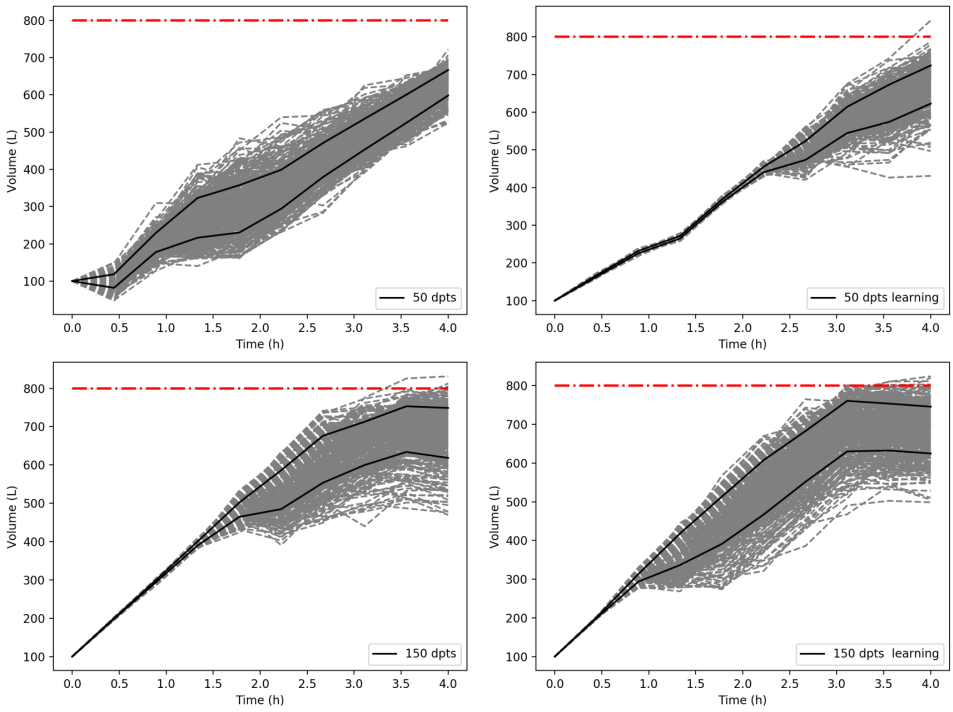


Figure 8.3: Closed-loop trajectories of volume for 400 GP MC samples for the cases 50 and 150 data-points with and without learning.

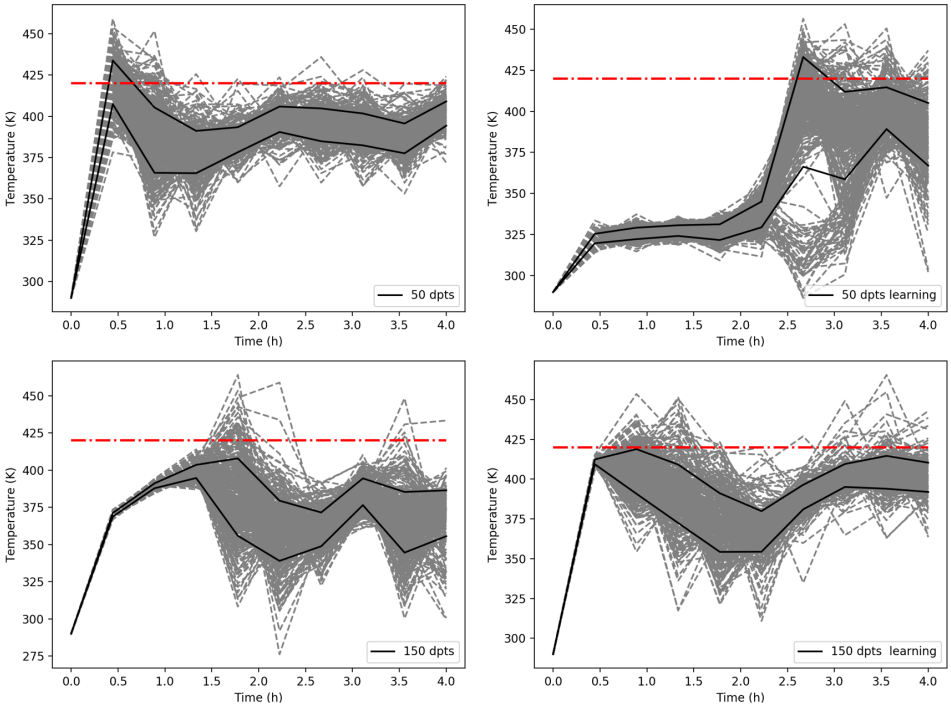


Figure 8.4: Closed-loop trajectories of temperature for 400 GP MC samples for the cases 50 data-points and 150 data-points with and without learning.

Table 8.1: Comparison of closed-loop constraint satisfaction,corrected guaranteed probability, and average OCP/NMPC solution time.

Set-up	**Probability**	**c-Probability**	**OCP time (s)**
50 dpts	0.55	0.57	0.11
50 dpts learning	0.31	0.33	0.19
100 dpts	0.27	0.30	0.52
100 dpts learning	0.21	0.24	0.75
150 dpts	0.16	0.19	1.12
150 dpts learning	0.09	0.11	1.34

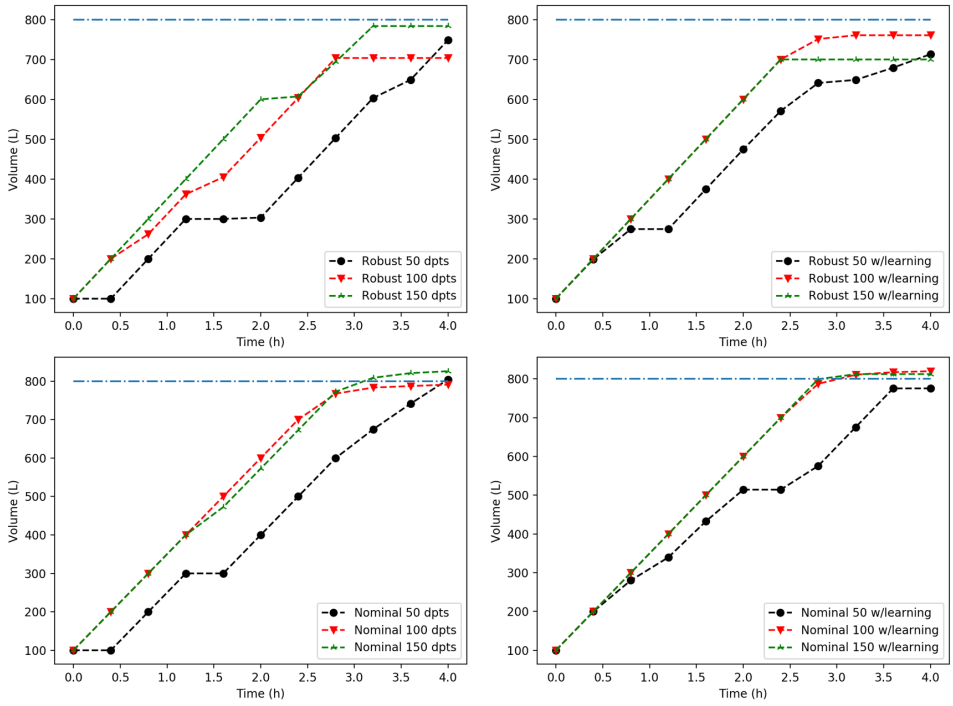


Figure 8.5: Closed-loop trajectories of volume for the "real" plant with tightened constraints at the top and without tightened constraints at the bottom.

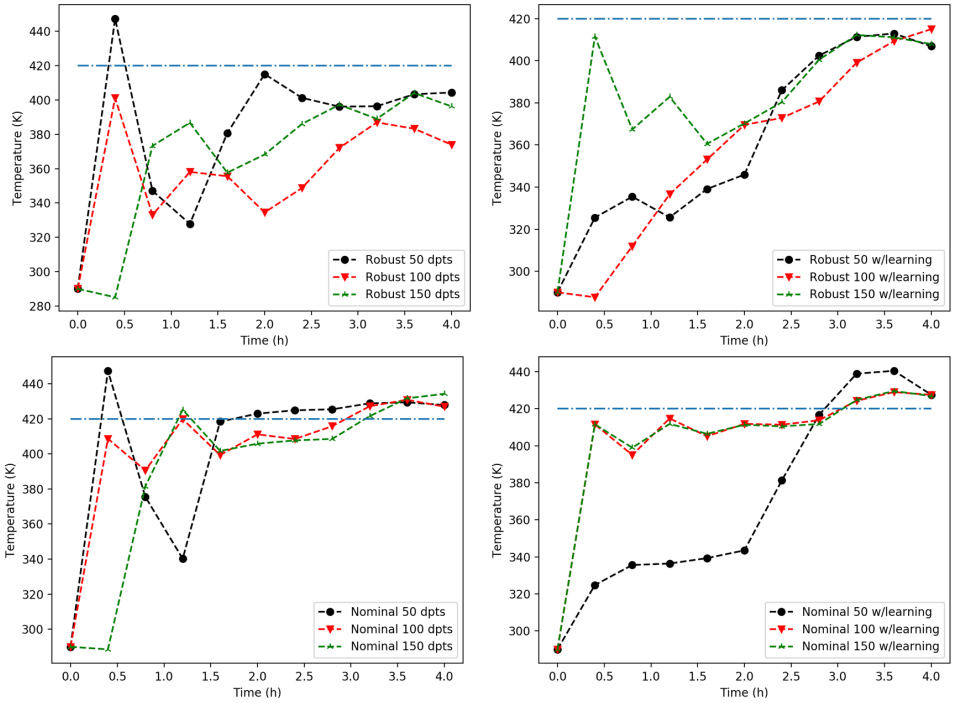


Figure 8.6: Closed-loop trajectories of temperature for the "real" plant with tightened constraints at the top and without tightened constraints at the bottom.

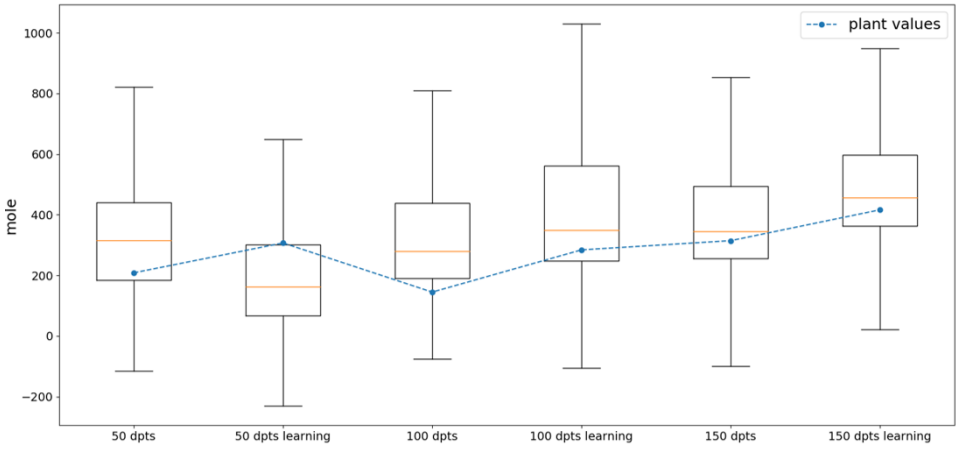


Figure 8.7: Box plot of objective values from 400 MC closed-loop simulations of the final back-off iteration together with the obtained objectives for the "real" plant shown in blue.

Chapter 9

Stochastic data-driven model predictive control using Gaussian processes

This chapter is based on **Paper G**: E. Bradford, L. Imsland, D. Zhang, and E. A. del Rio-Chanona. Stochastic data-driven model predictive control using Gaussian processes. *Computers & Chemical Engineering*, accepted, 2020.

Summary

Nonlinear model predictive control (NMPC) is one of the few control methods that can handle multivariable nonlinear control systems with constraints. Gaussian processes (GPs) present a powerful tool to identify the required plant model and quantify the residual uncertainty of the plant-model mismatch. It is crucial to consider this uncertainty, since it may lead to worse control performance and constraint violations. In this paper we propose a new method to design a GP-based NMPC algorithm for finite horizon control problems. The method generates Monte Carlo samples of the GP offline for constraint tightening using back-offs. The tightened constraints then guarantee the satisfaction of chance constraints online. Advantages of our proposed approach over existing methods include fast online evaluation, consideration of closed-loop behaviour, and the possibility to alleviate conservativeness by considering both online learning and state dependency of the uncertainty. The algorithm is verified on a challenging semi-batch bioprocess case study.

9.1 Introduction

Model predictive control (MPC) describes an advanced control method that has found a wide range of applications in industry. MPC employs an explicit dynamic model of the plant to determine a finite sequence of control actions to take at each sampling time. The main advantage of MPC is its ability to deal with multivariate plants and process constraints explicitly [165]. Linear MPC is relatively mature and well-established in practice, however many systems display strong nonlinear behaviour motivating the use of nonlinear MPC (NMPC) [6]. NMPC is becoming progressively more popular due to the advancement of improved non-convex optimization algorithms [27], in particular in chemical engineering [28].

The performance of MPC is however greatly influenced by the accuracy of the plant model, which is one of the main reasons why MPC is not more widely used in industry [162]. The development of an accurate plant model has been cited to take up to 80% of the MPC commissioning effort [244]. NMPC algorithms exploit various types of models, commonly developed by first principles or based on process mechanisms [188]. Many mechanistic and empirical models are however often too complex to be used online and in addition have often high development costs. Alternatively, black-box identification models can be exploited instead, such as support vector machines [267], fuzzy models [133], neural networks (NNs) [211], or Gaussian processes (GPs) [143]. For example, recently in [266, 265] recurrent NNs are utilised for an extensive NMPC approach with proofs on closed-loop state boundedness and convergence applied to a chemical reactor. In addition, in [264] the approach was extended updating the recurrent NNs online to further improve the effectiveness.

GPs are an interpolation technique developed by [150] that were popularized by the machine learning community [220]. While GPs have been predominantly used to model static nonlinearities, there are several works that apply GPs to model dynamic systems [100, 141, 47]. GP predictions are given by a Gaussian distribution. The mean of this distribution can be viewed as a deterministic prediction, while the variance can be interpreted as a measure of uncertainty for this deterministic prediction. This uncertainty measure is generally difficult to obtain by nonlinear parametric models [143] and may in part explain the relative popularity of GPs. For control applications this uncertainty measure can be utilized to efficiently learn a dynamic model by exploring unknown regions or avoiding regions with too high uncertainty to improve robustness [23]. So far, GPs have been exploited in a multitude of ways in the control community, including reinforcement learning [71], designing robust linear controllers [250], or adaptive control [65]. In particular, GPs have been shown to be an efficient approach to attain approximate

plant models for NMPC.

The use of GPs for NMPC was first proposed in [183], which updates a GP model online for reference tracking without constraints. In [143, 142] the GP is instead identified offline and utilized online, in which the variance is constrained to prevent the NMPC from steering the dynamic system into regions with high uncertainty. A GP plant model is updated online in [139] and in [167] to overcome unmodeled periodic errors or changes to the dynamic system after a fault has occurred respectively. GPs have been shown in [260] to be an efficient means for disturbance forecasting for a linear stochastic MPC approach applied to a drinking water network. GPs have further been applied to approximate the mean and variance required in stochastic NMPC [36]. [103] derived an explicit solution for GP-based NMPC. In [60] a GP dynamic model is employed for the control of an unmanned quadrotor, while in [158] a GP dynamic model is exploited to control a gas-liquid separation process. While these and other works show the feasibility of GP-based MPC, there is a lack of efficient approaches to account for the uncertainty measure provided. Model uncertainty can lead to constraint violations and worse performance. To mitigate the effect of uncertainty on MPC, robust MPC [22] and stochastic MPC [175, 111] methods have been developed.

The majority of works for GP-based MPC indeed consider the uncertainty measure provided, however most proposed algorithms employ stochastic uncertainty propagation to achieve this, for example [143, 142, 113, 60, 103, 260]. [114] give an overview of the various stochastic propagation techniques available. These approaches have some considerable disadvantages, which are:

- No known methods to exactly propagate stochastic uncertainties through GP models. Instead, only approximations are available relying on linearization or statistical moment-matching.
- Increased computational time of GP-based MPC due to the propagation approach itself.
- Most works consider only open-loop propagation of uncertainties, which is often prohibitively conservative due to open-loop growth of uncertainties.

Recently some papers have proposed different robust GP-based MPC algorithms. In [147] a NMPC algorithm is introduced based on propagating ellipsoidal sets using linearization, that provides closed-loop stability guarantees. This approach may however suffer from increased computational times, since the ellipsoidal sets are propagated online. Furthermore, the method may be relatively conservative due to the use of Lipschitz constants. [169] propose the use of a robust MPC approach by bounding the one-step ahead error, while the determination

of the required parameters seems to be relatively difficult. [239] suggest a robust control approach for linear systems, in which the GP is used to represent unmodeled nonlinearities. The approach is shown to stabilize the linear system despite these uncertainties, which however may have no solution if the difference between the linearized system and the actual nonlinear system is too large.

In this paper we extend an algorithm first introduced in [40], for which the following extensions were made:

- Inclusion of uncertainty for the initial state.
- Adding additive disturbance noise to the problem definition.
- Accounting for state dependency on the GP noise.
- Improved algorithm to obtain the required back-offs using root-finding as opposed to the inverse CDF, which has superior convergence and leads to improved satisfaction of the required probability bounds.

The aim of this approach is to take into account the uncertainty given by a GP state space model for a NMPC *finite-horizon* control problem. Due to the issues using stochastic uncertainty propagation for the NMPC formulation as highlight previously, we base the NMPC only on cheap evaluations of the GP. This leads to considerably faster evaluation times with little effect on the performance. The proposed method utilizes explicit back-offs, which were recently proposed in [146, 206] to account for stochastic uncertainties in NMPC. These methods generally rely on generating closed-loop Monte Carlo (MC) samples offline from the plant to attain the required back-off values. To obtain exact MC samples of the GP dynamic models we exploit results from [68, 249]. There are several important advantages of this new method:

- Back-offs are attained using closed-loop simulations, therefore the issue of open-loop growth of uncertainties is avoided.
- Required computations are carried out offline, such that the online computational times are nearly unaffected.
- Independence of samples allows some probabilistic guarantees to be given.
- Explicit consideration of online learning and state dependency of the uncertainty to alleviate conservativeness.

In the proposed algorithm the state dependency of the uncertainty is accounted for by introducing a penalty term on the variance in the objective. This variance

for GPs is a function of the states and hence leads to a trade-off between exploiting the GP model to optimize the objective and avoiding uncertain regions to reduce the spread of the trajectories. The algorithm proposed is aimed at *finite horizon* control problems, for which batch processes are a particularly important example. They are utilized in many different chemical engineering sectors due to their inherent flexibility to deal with variations in feedstock, product specifications, and market demand. Frequent highly nonlinear behaviour and unsteady-state operation of batch processes have led to the increased acceptance for advanced control solutions, such as NMPC [187]. Works on batch process NMPC accounting for uncertainties include an extended and Unscented Kalman filter based algorithms for uncertainty propagation [184, 33], a NMPC algorithm using min-max successive linearization [252], NMPC algorithms that employ PCEs to account for possible parametric uncertainties [177, 37], and multi-stage NMPC [164].

The paper is comprised of the following sections. In Section 9.2 the problem definition is given. In Section 9.3 a general outline of GPs is given including the sampling procedure used. Section 9.4 shows how the GPs can be exploited to solve the defined problem. In Section 9.5 the semi-batch bioprocess case study is described, while in Section 9.6 the results and discussions for this case study are given. Section 9.7 concludes the paper.

Notation

$\mathbb{N}$ and $\mathbb{R}$ represent the sets of natural numbers and real numbers respectively. The variable δ_{ij} denotes the Kronecker delta function, such that:

$$\delta_{ij} := \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

The notation $\mathrm{diag}(a_0, a_1, \ldots, a_n)$ is used to represent the following diagonal matrix:

$$\mathrm{diag}(a_0, a_1, \ldots, a_n) := \begin{bmatrix} a_0 & 0 & \ldots & 0 \\ 0 & a_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & a_n \end{bmatrix}$$

We represent the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Further, $\boldsymbol{\phi} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes that the random variable $\boldsymbol{\phi}$ follows a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

The expected value of a random variable $\boldsymbol{\phi}$ is denoted as:

$$\mathbb{E}[\boldsymbol{\phi}] := \int_{\Omega} \boldsymbol{\phi} dp_{\boldsymbol{\phi}}$$

where $p_{\boldsymbol{\phi}}$ the probability density function of $\boldsymbol{\phi}$ over the sample space Ω.

Further, we define the indicator function and the probability measure of random variable $\boldsymbol{\phi}$ as follows:

$$\mathbf{1}\{C \leq c\} := \begin{cases} 1, & \text{if } C \leq c \\ 0, & \text{otherwise} \end{cases}$$
$$\mathbb{P}\{\boldsymbol{\phi} \in \mathcal{A}\} := \int_{\boldsymbol{\phi} \in \mathcal{A}} \boldsymbol{\phi} dp_{\boldsymbol{\phi}}, \quad \mathbb{P}\{\boldsymbol{\phi} \leq c\} := \mathbb{E}[\mathbf{1}\{\boldsymbol{\phi} \leq c\}]$$

where $\mathcal{A}$ is a set defining an event on $\boldsymbol{\phi}$ and $\mathbb{P}\{\boldsymbol{\phi} \leq c\}$ is the probability that $\boldsymbol{\phi}$ is less than or equal to c.

Lastly, we require the definition of the beta inverse cumulative distribution function (cdf) for a random variable ϕ. This function betainv(P, A, B) returns a value C of ϕ following a beta distribution with parameters A, B that has a probability of P to be less than or equal to C. The definition is as follows:

$$\text{betainv}(P, A, B) \in F_{\phi}^{-1}(P|A, B) = \{C : F_{\phi}(C|A, B) = P)\}$$
$$F_{\phi}(C|A, B) := \frac{1}{\mathcal{B}(A, B)} \int_0^C t^{A-1}(1-t)^{B-1} dt,$$
$$\mathcal{B}(A, B) = \int_0^1 t^{A-1}(1-t)^{B-1} dt$$

9.2 Problem definition

The dynamic system in this paper is given by a discrete-time nonlinear equation system with additive disturbance noise:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\omega}_t, \quad \mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_0}, \boldsymbol{\Sigma}_{\mathbf{x}_0}) \tag{9.1}$$

where t is the discrete time, $\mathbf{x} \in \mathbb{R}^{n_{\mathbf{x}}}$ is the state, $\mathbf{u} \in \mathbb{R}^{n_{\mathbf{u}}}$ are the control inputs, $\mathbf{f} : \mathbb{R}^{n_{\mathbf{x}}} \times \mathbb{R}^{n_{\mathbf{u}}} \rightarrow \mathbb{R}^{n_{\mathbf{x}}}$ are nonlinear equations, and $\boldsymbol{\omega}$ represents Gaussian distributed additive disturbance noise with zero mean and diagonal covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\omega}}$. The initial condition $\mathbf{x}_0$ is assumed to be Gaussian distributed with mean $\boldsymbol{\mu}_{\mathbf{x}_0}$ and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}_0}$.

We assume measurements of the states to be available with additive Gaussian noise expressed as:

$$\mathbf{y} = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \boldsymbol{\nu} \tag{9.2}$$

where $\mathbf{y} \in \mathbb{R}^{n_\mathbf{x}}$ is the measurement of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ perturbed by additive Gaussian noise $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\boldsymbol{\nu})$ with zero mean and
a diagonal covariance matrix $\boldsymbol{\Sigma}_\boldsymbol{\nu} = \operatorname{diag}(\sigma^2_{\nu_1}, \ldots, \sigma^2_{\nu_{n_\mathbf{x}}})$.

The aim of the control problem is to minimize a finite-horizon cost function:

$$V_T(\mathbf{x}_0, \mathbf{U}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \ell(\mathbf{x}_t, \mathbf{u}_t) + \ell_f(\mathbf{x}_T)\right] \tag{9.3}$$

where $T \in \mathbb{N}$ is the time horizon, $\mathbf{U} = [\mathbf{u}_0, \ldots, \mathbf{u}_{T-1}]^\mathsf{T} \in \mathbb{R}^{T \times n_\mathbf{u}}$ is a joint vector of T control inputs, $\ell : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \to \mathbb{R}$ is the stage cost, and $\ell_f : \mathbb{R}^{n_\mathbf{x}} \to \mathbb{R}$ denotes the terminal cost.

The control problem is subject to hard constraints on the inputs:

$$\mathbf{u}_t \in \mathbb{U}_t \qquad \forall t \in \{0, \ldots, T-1\} \tag{9.4}$$

The states are subject to a joint chance constraint that requires the satisfaction of a nonlinear constraint set up to a certain probability, which can be stated as:

$$\mathbb{P}\left\{\bigcap_{t=0}^{T} \{\mathbf{x}_t \in \mathbb{X}_t\}\right\} \geq 1 - \epsilon \tag{9.5a}$$

where $\mathbb{X}_t$ is defined as:

$$\mathbb{X}_t = \{\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}} \mid g_j^{(t)}(\mathbf{x}) \leq 0, j = 1, \ldots, n_g\} \tag{9.5b}$$

The joint chance constraints are formulated such that the joint event over all $t \in \{0, \ldots, T\}$ of all $\mathbf{x}_t$ fulfilling the nonlinear constraint sets $\mathbb{X}_t$ has a probability greater than $1 - \epsilon$.

For convenience we define the tuple $\mathbf{z} = (\mathbf{x}, \mathbf{u}) \in \mathbb{R}^{n_\mathbf{z}}$ with joint dimension $n_\mathbf{z} = n_\mathbf{x} + n_\mathbf{u}$. The dynamic system in Equation 9.1 is assumed to be unknown. Instead, we are only given a *finite number* of noisy measurements according to Equation 9.2. The available data can then be denoted by the following two matrices:

$$\mathbf{Z} = [\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)}]^\mathsf{T} \in \mathbb{R}^{N \times n_\mathbf{z}} \tag{9.6a}$$

$$\mathbf{Y} = [\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)}]^\mathsf{T} \in \mathbb{R}^{N \times n_\mathbf{x}} \tag{9.6b}$$

where $\mathbf{z}^{(i)}$ represents the input of the i-th data point with corresponding noisy observation $\mathbf{y}^{(i)}$, N denotes the overall number of training data points, $\mathbf{Z}$ is a collection of input data, and the corresponding noisy observations are collected in $\mathbf{Y}$.

It should be noted that the uncertainty in this problem arises partially from the uncertain initial condition $\mathbf{x}_0$ and the additive disturbance noise $\boldsymbol{\omega}$. Most of the uncertainty however comes from the fact that we do not know $\mathbf{f}(\mathbf{x}, \mathbf{u})$ and are only given noisy observations of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ instead. To solve this problem we train a GP to approximate $\mathbf{f}(\cdot)$ using the available data in Equation 9.6. The GP methodology is introduced for this purpose in the next section. This GP then represents a distribution over possible functions $\mathbf{f}(\cdot)$ given the available data, which can be exploited to attain stochastic constraint satisfaction of the closed-loop system.

9.3 Gaussian processes

9.3.1 Regression

In this section we introduce the use of GPs to infer a latent function $f : \mathbb{R}^{n_z} \rightarrow \mathbb{R}$ from noisy data. For a more complete overview refer to [220]. Let the noisy observations y of $f(\cdot)$ be given by:

$$y = f(\mathbf{z}) + \nu \tag{9.7}$$

where $\mathbf{z} \in \mathbb{R}^{n_z}$ is the argument of $f(\cdot)$ and y is a perturbed observation of $f(\mathbf{z})$ with additive Gaussian noise $\nu \sim \mathcal{N}(0, \sigma_\nu^2)$ with zero mean and variance σ_ν^2.

GPs can be considered a generalization of multivariate Gaussian distributions to describe a distribution over functions. A GP is fully specified by a mean function and a covariance function. The mean function represents the "average" shape of the function, while the covariance function specifies the covariance between any two function values. We write that a function $f(\cdot)$ is distributed as a GP with mean function $m(\cdot)$ and covariance function $k(\cdot, \cdot)$ as:

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)) \tag{9.8}$$

The prior GP distribution is defined by the *choice* of the mean function and covariance function. In this study we apply a zero mean function and the squared-exponential (SE) covariance function defined as:

$$m(\mathbf{z}) := 0 \tag{9.9}$$

$$k(\mathbf{z}, \mathbf{z}') := \zeta^2 \exp\left(-\frac{1}{2}(\mathbf{z} - \mathbf{z}')^\top \boldsymbol{\Lambda}^{-2}(\mathbf{z} - \mathbf{z}')\right) \tag{9.10}$$

where $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^{n_\mathbf{z}}$ are arbitrary inputs, ζ^2 denotes the covariance magnitude, and $\mathbf{\Lambda}^{-2} \coloneqq \text{diag}(\lambda_1^{-2}, \ldots, \lambda_{n_\mathbf{z}}^{-2})$ is a scaling matrix.

Remark (Prior assumptions)**.** *Note the zero mean assumption can be easily achieved by normalizing the data beforehand. The SE covariance function is smooth and stationary, such that choosing the SE covariance function assumes the latent function $f(\cdot)$ to be smooth and stationary as well. The algorithm presented in this work can be utilised using any covariance function. In the case of highly non-stationary functions it may be necessary to use non-stationary covariance functions [227].*

From the additive property of Gaussian distributions the measurements of $f(\cdot)$ also follow a GP accounting for measurement noise:

$$y \sim \mathcal{GP}(m(\mathbf{z}), k(\mathbf{z}, \mathbf{z}') + \sigma_\nu^2 \delta_{\mathbf{z}\mathbf{z}'}) \tag{9.11}$$

We denote the hyperparameters defining the prior jointly by $\mathbf{\Psi} \coloneqq [\zeta, \lambda_1, \ldots, \lambda_{n_\mathbf{z}}, \sigma_\nu]^\mathsf{T}$, in which the variance σ_ν of the measurement noise is included in case it is unknown. Commonly the hyperparameters are unknown, such that they need to be inferred from the available data using for example maximum likelihood estimation (MLE).

Assume we are given N noisy function evaluations according to Equation 9.7 denoted by $\mathbf{Y} \coloneqq [y^{(1)}, \ldots, y^{(N)}]^\mathsf{T} \in \mathbb{R}^N$ as the result of the inputs given in $\mathbf{Z} = [\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)}]^\mathsf{T} \in \mathbb{R}^{N \times n_\mathbf{z}}$. According to the prior GP assumption, the data follows a multivariate Gaussian distribution:

$$\mathbf{Y} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_\mathbf{Y}) \tag{9.12}$$

where $[\mathbf{\Sigma}_\mathbf{Y}]_{ij} = k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) + \sigma_\nu^2 \delta_{ij}$ for each pair $(i, j) \in \{1, \ldots, N\}^2$.

The log-likelihood of the observations is consequently given by (ignoring constant terms):

$$\mathcal{L}(\mathbf{\Psi}) \coloneqq -\frac{1}{2} \log(\det(\mathbf{\Sigma}_\mathbf{Y})) - \frac{1}{2} \mathbf{Y}^\mathsf{T} \mathbf{\Sigma}_\mathbf{Y}^{-1} \mathbf{Y} \tag{9.13}$$

The MLE estimate of the hyperparameters $\mathbf{\Psi}$ is determined by maximizing Equation 9.13. Once the hyperparameters are known, we need to determine the posterior GP distribution of the latent function $f(\cdot)$. From the prior GP assumption we know that the training data and the value of $f(\cdot)$ at an arbitrary input $\mathbf{z}$ follow a joint multivariate normal distribution:

$$\begin{bmatrix} \mathbf{Y} \\ f(\mathbf{z}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma}_\mathbf{Y} & \mathbf{k}^\mathsf{T}(\mathbf{z}) \\ \mathbf{k}(\mathbf{z}) & k(\mathbf{z}, \mathbf{z}') \end{bmatrix} \right) \tag{9.14}$$

where $\mathbf{k}(\mathbf{z}) \coloneqq [k(\mathbf{z}, \mathbf{z}^{(1)}), \ldots, k(\mathbf{z}, \mathbf{z}^{(N)})]^\mathsf{T}$.

The posterior Gaussian distribution of $f(\mathbf{z})$ given the data $(\mathbf{Z}, \mathbf{Y})$ can then be found by using the conditional distribution rule for multivariate normal distributions based on the joint normal distribution in Equation 9.14, which leads to:

$$f(\mathbf{z})|\mathcal{D} \sim \mathcal{N}(\mu_f(\mathbf{z}; \mathcal{D}), \sigma_f(\mathbf{z}; \mathcal{D})) \tag{9.15a}$$

with

$$\mu_f(\mathbf{z}; \mathcal{D}) \coloneqq \mathbf{k}^\mathsf{T}(\mathbf{z})\boldsymbol{\Sigma}_\mathbf{Y}^{-1}\mathbf{Y} \tag{9.15b}$$

$$\sigma_f^2(\mathbf{z}; \mathcal{D}) \coloneqq \zeta^2 - \mathbf{k}^\mathsf{T}(\mathbf{z})\boldsymbol{\Sigma}_\mathbf{Y}^{-1}\mathbf{k}(\mathbf{z}) \tag{9.15c}$$

where $\mathcal{D} = (\mathbf{Z}, \mathbf{Y})$ denotes the training data available to obtain the posterior Gaussian distribution. The mean function $\mu_f(\mathbf{z}; \mathcal{D})$ in this context is the prediction of the GP at $\mathbf{z}$, while the variance function $\sigma_f^2(\mathbf{z}; \mathcal{D})$ is a measure of uncertainty.

In Figure 9.1 we illustrate a prior GP in the top graph and the posterior GP in the bottom graph.

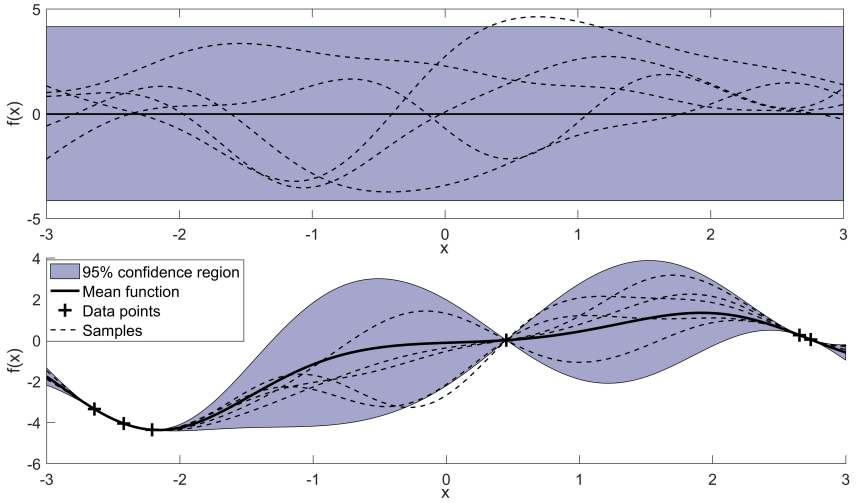


Figure 9.1: Illustration of a GP of a 1-dimensional function perturbed by noise. On the top the prior of the GP is shown, while on the bottom the Gaussian process was fitted to several observations to obtain the posterior.

9.3.2 Recursive update

Often we are given a training dataset $\mathcal{D}$ initially to build a posterior GP and then obtain data points individually afterwards. For example in GP-based MPC

we may build an initial GP offline following the procedure shown in Section 9.3.1, and then also update this model online as new data becomes available. In this work we keep the hyperparameters constant but update the mean function and variance function in Equations 9.15b-9.15c recursively given the new data points. Furthermore, the approach used for MC sampling of the GPs to be introduced in Section 9.3.4 requires recursive updates of this form as well.

Let the new dataset be given by $\mathcal{D}^+ = (\mathcal{D}, (\mathbf{z}^+, y^+))$, where $\mathcal{D}$ is the training data for the initial GP, while $\mathbf{z}^+$ is the new input and y^+ the new corresponding output measurement. Then we will refer to the updated mean function and variance function as:

$$\mu_f^+(\mathbf{z}; \mathcal{D}^+) := \mathbf{k}^{+\mathsf{T}}(\mathbf{z}) \boldsymbol{\Sigma}_{\mathbf{Y}}^{+-1} \mathbf{Y}^+ \tag{9.16a}$$

$$\sigma_f^{2+}(\mathbf{z}; \mathcal{D}^+) := \zeta^2 - \mathbf{k}^{+\mathsf{T}}(\mathbf{z}) \boldsymbol{\Sigma}_{\mathbf{Y}}^{+-1} \mathbf{k}^+(\mathbf{z}) \tag{9.16b}$$

The updated terms in Equations 9.16a-9.16b can be expressed as:

$$\mathbf{k}^+(\mathbf{z}) = [\mathbf{k}^\mathsf{T}(\mathbf{z}), k(\mathbf{z}, \mathbf{z}^+)]^\mathsf{T} \tag{9.17a}$$

$$\mathbf{Y}^+ = [\mathbf{Y}^\mathsf{T}, y^+]^\mathsf{T}, \quad \mathbf{Z}^+ = [\mathbf{Z}^\mathsf{T}, \mathbf{z}^+]^\mathsf{T} \tag{9.17b}$$

$$\boldsymbol{\Sigma}_{\mathbf{Y}}^{+-1} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{Y}} & \mathbf{k}^\mathsf{T}(\mathbf{z}^+) \\ \mathbf{k}(\mathbf{z}^+) & k(\mathbf{z}^+, \mathbf{z}^+)(+\sigma_\nu^2) \end{bmatrix}^{-1} \tag{9.17c}$$

where $\mathbf{k}(\mathbf{z})$, $\mathbf{Z}$ and $\mathbf{Y}$ refer to quantities of the initial GP. The noise term σ_ν^2 in the lower diagonal is shown in brackets, since the new "measurement" y^+ may be noiseless as is the case for GP MC samples. In this case the noise term should not be added to the new diagonal element.

Note the updates $\mathbf{k}^+(\mathbf{z})$, $\mathbf{Z}^+$, and $\mathbf{Y}^+$ are trivial, however the update of the *inverse* covariance matrix $\boldsymbol{\Sigma}_{\mathbf{Y}}^{+-1}$ is more involved. In essence we require the inverse of the previous covariance matrix after adding a horizontal row and a vertical row to the covariance matrix of the initial GP, see Equation 9.17c. For this process there are efficient formula available, one of which is introduced in Section 9.8. These take advantage of the fact that we already know the inverse covariance matrix $\boldsymbol{\Sigma}_{\mathbf{Y}}^{-1}$ of the initial GP. Once the update has been carried out, these terms then define the new initial GP. This update procedure is then repeated for the next measurement.

9.3.3 State space model

In this section we briefly show how the previously introduced GP methodology can be utilized to identify unknown state space models in the form of Equation 9.1 based on the measurements (data) according to Equation 9.2. GPs are commonly applied to model scalar functions with vector inputs as shown in Section 9.3.1. To

extend this to the multi-input, multi-output case as required it is common to build a separate independent GP for each output dimension, see for example [71]. Let the function in Equation 9.1 be given by $\mathbf{f}(\mathbf{x},\mathbf{u}) := \mathbf{f}(\mathbf{z}) := [f_1(\mathbf{z}),\dots,f_{n_\mathbf{x}}(\mathbf{z})]^\mathsf{T}$. We aim to build a separate GP for each function $f_i(\mathbf{z})\,\forall i \in \{1,\dots,n_\mathbf{x}\}$ according to Section 9.3.1. For this purpose we are given observations $\mathbf{Y}_i = [y_i^{(1)},\dots,y_i^{(N)}]^\mathsf{T}\,\forall i \in \{1,\dots,n_\mathbf{x}\}$ and corresponding inputs $\mathbf{Z} = [\mathbf{z}^{(1)},\dots,\mathbf{z}^{(N)}]^\mathsf{T}$, where y_i refers to the i-th dimension of measurements obtained according to Equation 9.2. Let $\mathbf{Y} = [\mathbf{Y}_1,\dots,\mathbf{Y}_{n_\mathbf{x}}]$ correspond to the overall measurements available. The posterior Gaussian distribution of $\mathbf{f}(\cdot)$ at an arbitrary input $\mathbf{z} = (\mathbf{x},\mathbf{u})$ is:

$$\mathbf{f}(\mathbf{z})|\mathcal{D} \sim \mathcal{N}(\boldsymbol{\mu}_f(\mathbf{z};\mathcal{D}),\boldsymbol{\Sigma}_f(\mathbf{z};\mathcal{D})) \tag{9.18a}$$

with

$$\boldsymbol{\mu}_f(\mathbf{z};\mathcal{D}) = [\mu_f(\mathbf{z};\mathcal{D}_1),\dots,\mu_f(\mathbf{z};\mathcal{D}_{n_\mathbf{x}})]^\mathsf{T} \tag{9.18b}$$

$$\boldsymbol{\Sigma}_f(\mathbf{z};\mathcal{D}) = \operatorname{diag}(\sigma_f^2(\mathbf{z};\mathcal{D}_1),\dots,\sigma_f^2(\mathbf{z};\mathcal{D}_{n_\mathbf{x}})) + \boldsymbol{\Sigma}_\boldsymbol{\omega} \tag{9.18c}$$

where $\mu_f(\mathbf{z};\mathcal{D}_i)$ and $\sigma_f^2(\mathbf{z};\mathcal{D}_i)$ are the mean function and variance function built according to Section 9.3.1 with datasets $\mathcal{D}_i = (\mathbf{Z},\mathbf{Y}_i)\,\forall i \in \{1,\dots,n_\mathbf{x}\}$ with $\mathcal{D} = (\mathbf{Z},\mathbf{Y})$.

Remark (Additive disturbance noise)**.** *Note the additive disturbance noise defined in Equation 9.1 is simply added to the posterior covariance matrix due to the additive property of multivariate Gaussian distributions.*

In addition, given an initial GP state space model built with a dataset $\mathcal{D}$ and a new data point $(\mathbf{z}^+,\mathbf{y}^+)$, we can update it recursively utilizing the method introduced in Section 9.3.2:

$$\boldsymbol{\mu}_f^+(\mathbf{z};\mathcal{D}^+) = [\mu_f^+(\mathbf{z};\mathcal{D}_1^+),\dots,\mu_f^+(\mathbf{z};\mathcal{D}_{n_\mathbf{x}}^+)]^\mathsf{T} \tag{9.19a}$$

$$\boldsymbol{\Sigma}_f^+(\mathbf{z};\mathcal{D}^+) = \operatorname{diag}(\sigma_f^{2+}(\mathbf{z};\mathcal{D}_1^+),\dots,\sigma_f^{2+}(\mathbf{z};\mathcal{D}_{n_\mathbf{x}}^+)) + \boldsymbol{\Sigma}_\boldsymbol{\omega} \tag{9.19b}$$

where $\mathcal{D}_i^+ = (\mathcal{D}_i,(\mathbf{z}^+,y_i^+))\,\forall i \in \{1,\dots,n_\mathbf{x}\}$ and $\mathcal{D}^+ = (\mathcal{D},(\mathbf{z}^+,\mathbf{y}^+))$

9.3.4 Monte Carlo sampling

GPs are distribution over functions and hence their realizations yield *deterministic* functions, see for example the GP samples shown in Figure 9.1. In this section we show how to attain independent samples of GP state space models over a *finite* time horizon. Generating a MC sample of a GP would require sampling an infinite dimensional stochastic process, while there is no known method to achieve this. Instead, approximate approaches have been applied such as spectral sampling

[46, 218]. Exact samples of GPs are however possible if the GP MC sample needs to be known at only a finite number of points, which is exactly the situation for state space models over a *finite* time horizon. This technique was first outlined in [68] and has been employed in [249] for the optimal design of linear controllers. We next outline how to obtain an exact sample of a GP state space model over a *finite* time horizon for an arbitrary feedback control policy.

Assume we are given a GP state space model as shown in Section 9.3.3 from the input-output dataset $\mathcal{D} = (\mathbf{Z}, \mathbf{Y})$. The initial condition $\mathbf{x}_0$ is assumed to follow a known Gaussian distribution as defined in Equation 9.1. A GP state space model represents a distribution over possible plant models, for which each realization will lead to a different state sequence. The aim of this section is therefore to show how to obtain a single independent sample of such a state sequence, which can then be repeated to obtain multiple independent MC samples of the GP. Let $\boldsymbol{\mathcal{X}}^{(s)} = [\boldsymbol{\chi}_0^{(s)}, \boldsymbol{\chi}_1^{(s)}, \ldots, \boldsymbol{\chi}_T^{(s)}]^\mathsf{T}$ denote such a state sequence, where s denotes a particular GP realization and $\boldsymbol{\chi}_i^{(s)}$ the realization of the state at discrete time i in the sequence of MC sample s. The control inputs at discrete time i for MC sample s are denoted by $u_i^{(s)}$. The corresponding joint input of $\boldsymbol{\chi}_i^{(s)}$ is represented by $\mathcal{Z}_i^{(s)} = (\boldsymbol{\chi}_i^{(s)}, u_i^{(s)})$. We assume the control inputs to be the result of a feedback control policy, which we represent as $\kappa : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R} \to \mathbb{R}^{n_\mathbf{u}}$. The control actions $u_i^{(s)}$ are then given as follows:

$$u_i^{(s)} = \kappa\left(\boldsymbol{\chi}_i^{(s)}, i\right) \tag{9.20}$$

where i is the current discrete time. Note the control policy depends on the discrete time directly, since it is a finite horizon control policy.

Consequently, the control actions over the finite time horizon T are a function of $\mathcal{X}^{(s)}$ and denoted jointly as $\mathcal{U}^{(s)} = [u_0^{(s)}, \ldots, u_{T-1}^{(s)}]^\mathsf{T} = [\kappa\left(\boldsymbol{\chi}_0^{(s)}, 0\right), \ldots, \kappa\left(\boldsymbol{\chi}_{T-1}^{(s)}, T-1\right)]^\mathsf{T}$. Note these control inputs are different for each MC sample s due to feedback.

We start by sampling the Gaussian distribution of the initial state $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_0}, \boldsymbol{\Sigma}_{\mathbf{x}_0})$ to obtain the realization $\boldsymbol{\chi}_0^{(s)}$. The posterior Gaussian distribution of the next state in the sequence $\mathbf{x}_1$ is subsequently given by the GP of $\mathbf{f}(\cdot)$ as defined in Equation 9.18 dependent on $\boldsymbol{\chi}_0^{(s)}$:

$$\mathbf{x}_1 = \mathbf{f}(\mathcal{Z}_0) \sim \mathcal{N}(\boldsymbol{\mu}_f(\mathcal{Z}_0; \mathcal{D}), \boldsymbol{\Sigma}_f(\mathcal{Z}_0; \mathcal{D})) \tag{9.21}$$

The realization of $\mathbf{x}_1$ is obtained by sampling the above normal distribution, which we will denote as $\boldsymbol{\chi}_1^{(s)}$. To obtain the next state in the sequence $\boldsymbol{\chi}_2^{(s)}$ we need to first condition on $\boldsymbol{\chi}_1^{(s)}$, since this is part of this sampled function path. This

requires to treat $\boldsymbol{\chi}_1^{(s)}$ similarly to a new training point, however without observation noise (i.e. no σ_v^2 is added to the kernel evaluation $k(z_0, z_0)$) and without changing the hyperparameters. Note that if the sampled function were to return to the same input it would lead to the exact same output, since it is conditioned on a noiseless output. This shows that the sampled function is deterministic, since it is the result of sampling. Next we draw $\boldsymbol{\chi}_2^{(s)}$ according to the posterior Gaussian distribution obtained from adding the previously sampled data point to the training dataset $\mathcal{D}$ as a noiseless observation. This sample is then again added to the training dataset as a noiseless observation, from which the GP is updated and the next state is drawn. This process is repeated until the required time horizon T has been reached. In this paper we consider a finite time horizon control problem and hence the GP state space model should for, moderate time horizons, not become too computationally expensive, since at most T new data-points are added. Nonetheless, for large time horizons this could become a problem and approximate sampling approaches, such as spectral sampling should then be considered instead [46, 218].

This sampling approach is summarized in Algorithm 9.1 below and is illustrated in Figure 2. Each GP MC sample is defined by a state and corresponding control action sequence. Note that this gives us a single MC sample and subse-

quently needs to be repeated multiple times to obtain multiple realizations.

Algorithm 9.1: Gaussian process trajectory sampling

Input : $\boldsymbol{\mu}_{\mathbf{x}_0}, \boldsymbol{\Sigma}_{\mathbf{x}_0}, \boldsymbol{\mu}_f(\mathbf{z}; \mathcal{D}), \boldsymbol{\Sigma}_f(\mathbf{z}; \mathcal{D}), \mathcal{D}, T, \kappa(\cdot)$

Initialize: Draw $\boldsymbol{\chi}_0^{(s)}$ from $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_0}, \boldsymbol{\Sigma}_{\mathbf{x}_0})$.

for *each sampling time* $t = 1, 2, \ldots, T$ **do**

1. Determine $u_{t-1}^{(s)} = \kappa(\boldsymbol{\chi}_{t-1}^{(s)}, t-1)$
2. Draw $\boldsymbol{\chi}_t^{(s)}$ from $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_f(Z_{t-1}^{(s)}; \mathcal{D}), \boldsymbol{\Sigma}_f(Z_{t-1}^{(s)}; \mathcal{D}))$, where $Z_{t-1}^{(s)} = (\boldsymbol{\chi}_{t-1}^{(s)}, u_{t-1}^{(s)})$.
3. Define $\mathcal{D}^+ := (\mathcal{D}, (Z_{t-1}^{(s)}, \boldsymbol{\chi}_t^{(s)}))$, where $\boldsymbol{\chi}_t^{(s)}$ can be viewed as noiseless "measurements".
4. Update the dataset $\mathcal{D} := ([\mathbf{Z}^\mathsf{T}, Z_{t-1}^{(s)\mathsf{T}}]^\mathsf{T}, [\mathbf{Y}, \boldsymbol{\chi}_t^{(s)\mathsf{T}}]^\mathsf{T})$.
5. Recursively update the GP mean function $\boldsymbol{\mu}_f(\mathbf{z}; \mathcal{D}) := \boldsymbol{\mu}_f^+(\mathbf{z}; \mathcal{D}^+)$ using Equation 9.19a.
6. Recursively update the GP covariance function $\boldsymbol{\Sigma}_f(\mathbf{z}; \mathcal{D}) := \boldsymbol{\Sigma}_f^+(\mathbf{z}; \mathcal{D}^+)$ using Equation 9.19b.

Output : State sequence $\boldsymbol{\mathcal{X}}^{(s)} = [\boldsymbol{\chi}_0^{(s)}, \boldsymbol{\chi}_1^{(s)}, \ldots, \boldsymbol{\chi}_T^{(s)}]^\mathsf{T}$ and control sequence $\boldsymbol{\mathcal{U}}^{(s)} = [u_0^{(s)}, \ldots, u_{T-1}^{(s)}]^\mathsf{T}$

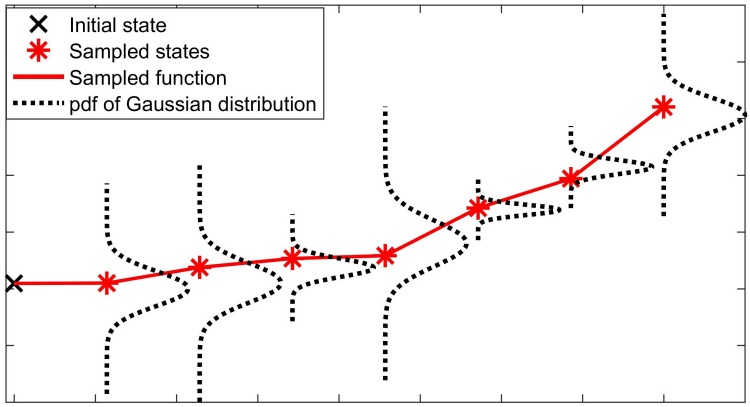


Figure 9.2: Illustration of GP sampling scheme for a 1 dimensional function.

Lastly, we define a *nominal* trajectory by setting all samples in Algorithm 9.1 to their mean values. Let $\overline{\boldsymbol{\mathcal{X}}} = [\overline{\boldsymbol{\chi}}_0, \overline{\boldsymbol{\chi}}_1, \ldots, \overline{\boldsymbol{\chi}}_T]^\mathsf{T}$ refer to this *nominal* state sequence and $\overline{\boldsymbol{\mathcal{U}}} = [\overline{u}_0, \ldots, \overline{u}_{T-1}]^\mathsf{T}$ to the corresponding *nominal* control sequence, where $\overline{\boldsymbol{\chi}}_i$ and $\overline{u}_i$ are the values of the states and control inputs of the *nominal* trajectory at discrete time i respectively. Therefore by definition, $\overline{\boldsymbol{\chi}}_0 = \boldsymbol{\mu}_{\mathbf{x}_0}$, $\overline{u}_{t-1} = \kappa(\overline{\boldsymbol{\chi}}_{t-1}, t-1)$, and $\overline{\boldsymbol{\chi}}_t = \boldsymbol{\mu}_f(\overline{z}_{t-1}; \mathcal{D})$, where $\overline{z}_{t-1} = (\overline{\boldsymbol{\chi}}_{t-1}, \overline{u}_{t-1})$. Note updating $\boldsymbol{\mu}_f(\cdot; \mathcal{D})$ with mean values has no effect.

In Section 9.4 a NMPC formulation is introduced, which uses the mean function of the GP as the prediction model. The control actions from this NMPC algorithm then define the control policy in Equation 9.20, while the MC samples represent possible plant responses. This is then exploited to tune the GP NMPC algorithm to attain the desired behaviour.

9.4 Solution approach

From the input-output dataset $\mathcal{D} = (\mathbf{Z}, \mathbf{Y})$ defined in Equation 9.6, we fit a GP state space model as outlined in Section 9.3.3. The aim is to solve the problem defined in Section 9.2 based on this GP state space model. In this context the GP represents a distribution over possible plant models for the process given the available dataset. In Section 9.3.4 we have shown how to create a sample of this plant model over a finite time horizon T, which each lead to different state sequences and corresponding control sequences based on a control policy. In this paper we aim to design a NMPC algorithm based on the GP that acts as this control policy. The MC samples are utilized to tune the NMPC formulation by adjusting so-called back-offs to tighten the constraints and attain the closed-loop probabilistic constraint satisfaction. We next state the GP NMPC formulation, which is based on the tightened constraint set and predictions from the mean function $\boldsymbol{\mu}_f(\cdot; \mathcal{D})$ and covariance function $\boldsymbol{\Sigma}_f(\cdot; \mathcal{D})$.

9.4.1 Finite-horizon Gaussian process model predictive control formulation

In this section we define the NMPC OCP based on the GP *nominal* model given the dataset $\mathcal{D} = (\mathbf{Z}, \mathbf{Y})$. For the GP NMPC formulation the initial state $\mathbf{x}$ at each sampling time is assumed to be measured or estimated and propagated forward in time exploiting the GP mean function. The predicted states are then used to optimize the objective subject to the tightened constraints. Let the corresponding optimization problem be denoted as $P_T\left(\boldsymbol{\mu}_f(\cdot; \mathcal{D}), \boldsymbol{\Sigma}_f(\cdot; \mathcal{D}); \mathbf{x}, t\right)$ for the current *known* state $\mathbf{x}$ at discrete time t based on the mean function $\boldsymbol{\mu}_f(\cdot; \mathcal{D})$ and covariance

function $\boldsymbol{\Sigma}_f(\cdot;\mathcal{D})$:

$$
\begin{aligned}
&\underset{\hat{\mathbf{U}}_{t:T-1}}{\text{minimize}} \quad \hat{V}_T(\mathbf{x},t,\hat{\mathbf{U}}_{t:T-1}) = \sum_{k=t+1}^{T-1} \left[\ell(\hat{\mathbf{x}}_k,\hat{\mathbf{u}}_k) + \eta_k \operatorname{tr}\left(\boldsymbol{\Sigma}_f(\hat{\mathbf{x}}_k;\mathcal{D})\right)\right] + \ell_f(\hat{\mathbf{x}}_T) \\
&\text{subject to:} \\
&\hat{\mathbf{x}}_{k+1} = \boldsymbol{\mu}_f(\hat{\mathbf{z}}_k;\mathcal{D}), \quad \hat{\mathbf{z}}_k = (\hat{\mathbf{x}}_k,\hat{\mathbf{u}}_k) \quad \forall k \in \{t,\dots,T-1\} \\
&\hat{\mathbf{x}}_{k+1} \in \overline{\mathbb{X}}_{k+1}, \quad \hat{\mathbf{u}}_k \in \mathbb{U}_k \quad \forall k \in \{t,\dots,T-1\} \\
&\hat{\mathbf{x}}_t = \mathbf{x}
\end{aligned}
\tag{9.22}
$$

where $\hat{\mathbf{x}}$, $\hat{\mathbf{u}}$, and $\hat{V}_T(\cdot)$ refers to the states, control inputs, and control objective of the MPC formulation, $\hat{\mathbf{U}}_{t:T-1} = [\hat{\mathbf{u}}_t,\dots,\hat{\mathbf{u}}_{T-1}]^\mathsf{T}$, η_k are weighting factors to penalize uncertainty, and $\overline{\mathbb{X}}_k$ is a tightened constraint set denoted by: $\overline{\mathbb{X}}_k = \{\mathbf{x} \in \mathbb{R}^{n_x} \mid g_j^{(k)}(\mathbf{x}) + b_j^{(k)} \leq 0,\ j = 1,\dots,n_g\}$. The variables $b_j^{(k)}$ represent so-called back-offs, which tighten the original constraints $\mathbb{X}_t$ defined in Equation 9.5.

Remark (Objective in expectation)**.** *It should be noted that the above objective in Equation 9.22 does not exactly optimize the objective in Equation 9.3, since it is difficult to obtain the expectation of a nonlinear function. Approximations of this can be found in [114], however these generally are considerably more expensive and often only lead to marginally improved performance.*

Remark (Scaling for state dependency factors)**.** *Note we have opted for scalar scaling factors η_k to account for state dependency. For this to work reliably it is therefore necessary to normalize the data to ensure that all data has approximately the same magnitude, for example normalizing the data to have zero mean and unit variance.*

The NMPC algorithm solves $P_T\left(\boldsymbol{\mu}_f(\cdot;\mathcal{D}),\boldsymbol{\Sigma}_f(\cdot;\mathcal{D});\mathbf{x}_t,t\right)$ at each sampling time t given the current state $\mathbf{x}_t$ to obtain an optimal control sequence:

$$
\begin{aligned}
&\hat{\mathbf{U}}^*_{t:T-1}\left(\boldsymbol{\mu}_f(\cdot;\mathcal{D}),\boldsymbol{\Sigma}_f(\cdot;\mathcal{D});\mathbf{x}_t,t\right) = \\
&[\hat{\mathbf{u}}^*_t\left(\boldsymbol{\mu}_f(\cdot;\mathcal{D}),\boldsymbol{\Sigma}_f(\cdot;\mathcal{D});\mathbf{x}_t,t\right),\dots,\hat{\mathbf{u}}^*_{T-1}\left(\boldsymbol{\mu}_f(\cdot;\mathcal{D}),\boldsymbol{\Sigma}_f(\cdot;\mathcal{D});\mathbf{x}_t,t\right)]^\mathsf{T}
\end{aligned}
\tag{9.23}
$$

Only the first optimal control action is applied to the plant at time t before the same optimization problem is solved at time $t+1$ with a new state measurement $\mathbf{x}_{t+1}$. This procedure implicitly defines the following feedback control law, which needs to be repeatedly solved for each new measurement $\mathbf{x}_t$:

$$
\kappa(\boldsymbol{\mu}_f(\cdot;\mathcal{D}),\boldsymbol{\Sigma}_f(\cdot;\mathcal{D});\mathbf{x}_t,t) = \hat{\mathbf{u}}^*_t\left(\boldsymbol{\mu}_f(\cdot;\mathcal{D}),\boldsymbol{\Sigma}_f(\cdot;\mathcal{D});\mathbf{x}_t,t\right) \tag{9.24}
$$

It is explicitly denoted that the control actions depend on the GP model used. There are several important variations of the GP NMPC control policy. Firstly, it may seem reasonable to update the mean and covariance function using the previous state measurement and corresponding input by applying the recursive update rules introduced in Section 9.3.2. We will refer to this as *learning*. This may however lead to a more expensive and less reliable NMPC algorithm.

Secondly, the algorithm may want to avoid regions in which there is great uncertainty due to sparsity of data. This can be achieved by assigning some of the η_k with non-zero values to penalize the algorithm moving into these regions with high variance. This explicitly takes advantage of the state dependency of the noise covariance function and is hence referred as *state dependent*. It should be noted that evaluation of the covariance function is computationally expensive. It was determined that setting only η_{t+1} to a non-zero value is often sufficient due to the continued feedback update, i.e. penalizing only the variance for the one-step ahead prediction at time t. These variations can be summarized as follows:

- *Learning*: Update the mean and variance function of the GP using the previous state measurement and the known corresponding input.
- *State dependent*: Set some η_k not equal to zero, which will lead to the NMPC algorithm trying to find a path that has less variance and hence exploiting the state dependent nature of the uncertainty.
- *Non learning*: Keep the mean and variance function the same throughout the run.
- *Non state dependent*: Set all η_k to zero and hence ignoring the possible state dependency of the uncertainty.

Remark (Full state feedback)**.** *Note in the control algorithm we have assumed full state feedback, i.e. it is assumed that the full state can be measured without noise. This assumption can be dropped if required by introducing a suitable observer and introduced in the closed-loop simulations to account for this additional uncertainty.*

9.4.2 Probabilistic guarantees

In this section we illustrate how to obtain probabilistic guarantees for the joint chance constraint introduced in Section 9.2 in Equation 9.5 based on independent samples of the GP plant model. For convenience we define a single-variate random variable $C(\cdot)$ representing the satisfaction of the joint chance constraint [69]:

$$C(\mathbf{X}) = \inf_{(j,t)\in\{1,\dots,n_g\}\times\{0,\dots,T\}} g_j^{(t)}(\mathbf{x}_t) \tag{9.25a}$$

$$\mathbb{P}\{C(\mathbf{X}) \leq 0\} = \mathbb{P}\left\{\bigcap_{t=0}^{T}\{\mathbf{x}_t \in \mathbb{X}_t\}\right\} \tag{9.25b}$$

where $\mathbf{X} = [\mathbf{x}_0, \ldots, \mathbf{x}_T]^\mathsf{T}$ defines a state sequence, and $\mathbb{X}_t = \{\mathbf{x} \in \mathbb{R}^{n_\mathrm{x}} \mid g_j^{(t)}(\mathbf{x}) \leq 0, j = 1, \ldots, n_g\}$.

The probability in Equation 9.25 is intractable, however a good nonparametric approximation is often achieved utilizing the so-called empirical cumulative distribution function (ecdf). We define the cdf to be approximated as follows:

$$F_{C(\mathbf{X})}(c) = \mathbb{P}\{C(\mathbf{X}) \leq c\} \tag{9.26}$$

Assuming we are given S independent and identically distributed MC samples of $\mathbf{X}$ and hence of $C(\mathbf{X})$, the ecdf estimate of the true cdf in Equation 9.26 is given by:

$$F_{C(\mathbf{X})}(c) \approx \hat{F}_{C(\mathbf{X})}(c) = \frac{1}{S}\sum_{s=1}^{S} \mathbf{1}\{C(\boldsymbol{\mathcal{X}}^{(s)}) \leq c\} \tag{9.27}$$

where $\boldsymbol{\mathcal{X}}^{(s)}$ is the s-th MC sample and $\hat{F}_{C(\mathbf{X})}(c)$ is the ecdf approximation of the true cdf $F_{C(\mathbf{X})}(c)$.

The quality of the approximation in Equation 9.27 strongly depends on the number of samples used and it is therefore desirable to quantify the residual uncertainty of the sample approximation. This problem has been studied to a great extent in the statistics literature [67]. In addition, there are several works applying these results for chance constrained optimization, see for example [5]. The main result applied in this study is given below in Theorem 9.4.1.

Theorem 9.4.1 (Confidence interval for empirical cumulative distribution function)**.** *Assume we are given a value of the ecdf, $\hat{\beta} = \hat{F}_{C(\mathbf{X})}(c)$, as defined in Equation 9.27 based on S independent samples of $C(\mathbf{X})$, then the true value of the cdf, $\beta = F_{C(\mathbf{X})}(c)$, as defined in Equation 9.26 has the following lower and upper confidence bounds:*

$$\mathbb{P}\left\{\beta \geq \hat{\beta}_{lb}\right\} \geq 1 - \alpha, \qquad \hat{\beta}_{lb} = \mathrm{betainv}\left(\alpha, S + 1 - S\hat{\beta}, S\hat{\beta}\right), \tag{9.28a}$$

$$\mathbb{P}\left\{\beta \leq \hat{\beta}_{ub}\right\} \geq 1 - \alpha, \qquad \hat{\beta}_{ub} = \mathrm{betainv}\left(1 - \alpha, S + 1 - S\hat{\beta}, S\hat{\beta}\right). \tag{9.28b}$$

Proof. The proof uses standard results in statistics and can be found in [67, 242]. The proof relies on the following observations. Firstly, $\mathbf{1}\{C(\boldsymbol{\mathcal{X}}^{(s)}) \leq c\}$ for a

fixed value of c describes a Bernoulli random variable, in which either $C(\boldsymbol{\mathcal{X}}^{(s)})$ exceeds c and takes the value 0 or otherwise takes the value 1 with probability $F_{C(\mathbf{X})}(c)$. Secondly, the ecdf describes the number of *successes* of S realizations of these Bernoulli random variables divided by the total number of samples and hence follows a binomial distribution, i.e. $\hat{F}_{C(\mathbf{X})}(c) \sim \frac{1}{N_S}\text{Bin}(S, F_{C(\mathbf{X})}(c))$. The confidence bound for the ecdf can consequently be determined from the Binomial cdf. This method was first introduced by [67] as "*exact confidence intervals*". Due to the close relationship between beta distributions and binomial distributions, a simplified expression can be obtained using beta distributions instead, which leads to the theorem shown [242]. ■

In other words the probability of β exceeding the value $\hat{\beta}_{ub}$ has a probability of α and the probability of β being less than or equal to $\hat{\beta}_{lb}$ has also a probability of α. In particular, $\hat{\beta}_{lb}$ for small α represents a conservative lower bound on the true probability β. An illustration of the confidence bound for the ecdf is shown in Figure 9.3. In general more samples will lead to a tighter confidence bound as expected. The theorem provides a lower bound $\hat{\beta}_{lb}$ that accounts for the statistical error due to the finite sample estimate made, i.e. it gives us a conservative value that is less than or equal to the true probability of feasibility with a confidence level of $1 - \alpha$.

We assume we are given S *independent* samples of the trajectory $\mathbf{X}$ generated according to Section 9.3.4. Let the approximate ecdf be given by $\hat{\beta} = \hat{F}_{C(\mathbf{X})}(0)$ according to Equation 9.27, and $\hat{\beta}_{lb}$ the corresponding lower bound according to Theorem 9.4.1 with confidence level $1 - \alpha$. From this the following Corollary follows:

Corollary 9.4.1 (Feasibility probability)**.** *Assuming the GP representation of the plant model to be a correct description of the uncertainty of the system and given a value of the ecdf $\hat{\beta} = \hat{F}_{C(\mathbf{X})}(0)$, as defined in Equation 9.27 based on S independent samples and a corresponding lower bound $\hat{\beta}_{lb} \geq 1 - \epsilon$ with a confidence level of $1 - \alpha$, then the original chance constraint in Equation 9.5 holds true with a probability of at least $1 - \alpha$.*

Proof. The GP MC sample described in Section 9.3.4 is exact and therefore each sample of the GP plant state space model leads to independent state trajectories $\mathbf{X}$ according to the GP distribution. From S such samples a valid lower bound $\hat{\beta}_{lb}$ to the true cdf value β can be determined from Theorem 9.4.1 with a confidence level of $1 - \alpha$. If $\hat{\beta}_{lb}$ is greater than or equal to $1 - \epsilon$, then the following probabilistic bound holds on the true cdf value β according to Theorem 9.4.1: $\mathbb{P}\left\{\beta \geq \hat{\beta}_{lb} \geq 1 - \epsilon\right\} \geq 1 - \alpha$, which in other words means that

$\beta = \mathbb{P}\{C(\mathbf{X} \leq 0)\} \geq 1 - \epsilon$ with a probability of at least $1 - \alpha$. ■

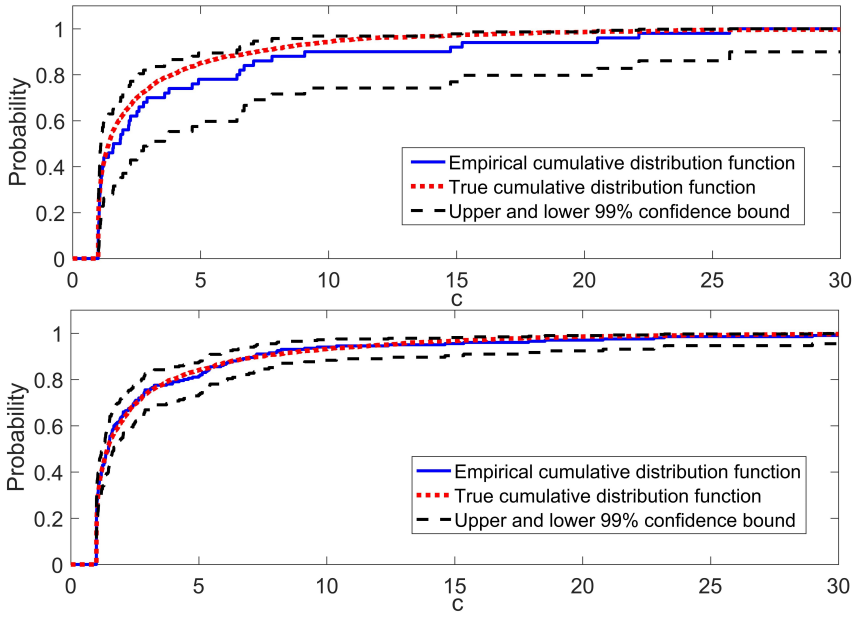


Figure 9.3: Illustration of the cdf confidence bound of $\alpha = 0.01$ for sample sizes of $S = 50$ (top) and $S = 200$ (bottom).

9.4.3 Determining back-off constraints

In this section we describe how to determine the required back-off values to tighten the constraints of the GP NMPC algorithm in Equation 9.22. The aim is to choose these values to obtain probabilistic guarantees for the state constraints defined in Equation 9.5 despite not knowing the exact dynamics. The GP provides a nominal model for the GP NMPC formulation in Equation 9.22 using the mean function and a distribution of possible plant models given the initial dataset in Equation 9.6. This distribution is exploited to attain different possible plant model realizations to simulate the closed-loop response of the GP NMPC algorithm and then uses the response values to tighten the constraints, such that the original constraint set is satisfied with a high probability according to Equation 9.5.

We have shown how to obtain the closed-loop trajectory of a control policy according to the realization of a plant model GP distribution in Section 9.3.4. To this we apply the GP NMPC control policy defined in Equation 9.24. We propose to utilize S independent MC samples of the GP distribution generated according to Section 9.3.4, which then in turn describe S different possible plant models with

corresponding state and control trajectories. The goal now is to adjust the back-offs, such that the S different state trajectories adhere the original constraint set for all but a few samples to attain the required probability of constraint satisfaction. Let $\boldsymbol{\mathcal{X}}^{(s)} = [\boldsymbol{\chi}_0^{(s)}, \ldots, \boldsymbol{\chi}_T^{(s)}]^\mathsf{T}$ refer to the state trajectory of sample s.

The update rule for adjusting the back-offs is based on two steps: First, we define an approximate constraint set, which is then adjusted by a constant factor to obtain the required constraint satisfaction using the ecdf of the joint chance constraint in Equation 9.30. The approximate constraint set in essence needs to reflect the difference in constraint values for the nominal model of the MPC and the realizations of the GP plant model. We first set all the back-off values to zero and run S MC samples of the GP plant model. As defined in Section 9.3.4, let $\overline{\boldsymbol{\chi}}_t$ refer to the states according to the *nominal* trajectory with the back-offs set to zero as well. Now assume we aim to obtain back-off values that imply the satisfaction of the following individual chance constraints:

$$g_j^{(t)}(\overline{\boldsymbol{\chi}}_t) + b_j^{(t)} = 0 \implies \mathbb{P}\left\{g_j^{(t)}(\boldsymbol{\chi}_t) \leq 0\right\} \geq 1 - \delta \tag{9.29}$$

where δ is a tuning parameter and should be set to a reasonably low value.

The rule in other words aims to find approximate back-offs for the nominal predictions $\overline{\boldsymbol{\chi}}_t$ utilized in the MPC formulation in Equation 9.22, such that the chance constraints holds for any possible GP plant model MC sample with a probability of $1 - \delta$. The parameter δ in this case is a tuning parameter to obtain the initial back-off values. Note the considered individual chance constraint in Equation 9.29 is only there to obtain the *initial* constraint set and is unrelated to the joint chance constraint in Equation 9.5, which we fulfil by adjusting this approximate constraint set using a root-finding algorithm.

To accomplish this we define the following ecdf based on the S MC samples available:

$$\hat{F}_{g_j^{(t)}}(0) = \frac{1}{S} \sum_{s=1}^{S} \mathbf{1}\{g_j^{(t)}(\boldsymbol{\chi}_t^{(s)}) \leq 0\} \tag{9.30}$$

where $\hat{F}_{g_j^{(t)}}(0)$ is a sample approximation of the chance constraint given in Equation (9.29) on the RHS.

In [206] it is proposed to employ the inverse ecdf to approximately fulfill the requirement given in Equation (9.29) using the S MC samples available. The back-offs can then be stated as follows:

$$\tilde{b}_j^{(t)} = \hat{F}^{-1}_{g_j^{(t)}}(1 - \delta) - g_j^{(t)}(\overline{\boldsymbol{\chi}}_t) \quad \forall (j,t) \in \{1, \ldots, n_g^{(t)}\} \times \{1, \ldots, T\} \tag{9.31}$$

where $\hat{F}^{-1}_{g_j^{(t)}}$ is the inverse of the ecdf defined in Equation 9.30 and $\tilde{b}_j^{(t)}$ refers to these initial back-off values. Note the inverse of an ecdf is given by the quantile function of the discrete S values with probability δ. This gives us the initial back-off values as required for the first step. Note that both the nominal trajectory $\overline{\boldsymbol{\chi}}_t$ and the GP realizations depend on the back-off values, which is however ignored since we are only interested in obtaining some reasonable initial values. In the next step these back-off values are further adjusted using a constant *back-off* factor γ. The new back-offs are then defined as:

$$b_j^{(t)} = \gamma \tilde{b}_j^{(t)} \quad \forall (j,t) \in \{1,\dots,n_g^{(t)}\} \times \{1,\dots,T\} \tag{9.32}$$

We aim to change γ until the lower bound of the ecdf $\hat{\beta}_{lb}$ as defined in the previous section for the joint chance constraint is equal to $1-\epsilon$ in Equation (9.5). This is a root finding problem, in which γ is adjusted until $\hat{\beta}_{lb}$ reaches the required value:

$$h(\gamma) = \hat{\beta}_{lb}(\gamma) - (1-\epsilon) \tag{9.33}$$

where the aim is to determine a value of γ, such that $h(\gamma)$ is approximately zero.

To attain the required γ we use the so-called *bisection method* [20]. This method determines the root of a function in an interval a_γ and b_γ, where $h(a_\gamma)$ and $h(b_\gamma)$ have opposite signs. In our case this is relatively easy. Setting the value of γ too low returns generally a negative value of $h(\gamma)$ due to the constraint violations using low back-offs, while setting it too high leads to positive values leading to a too conservative solution. Note we generally set the initial a_γ to zero since this corresponds to the S MC samples used to determine $\tilde{b}_j^{(t)}$. The *bisection method* consists of repeatably bisecting the interval, in which the root is contained. The overall algorithm to determine the required back-offs in n_b back-off iterations is summarized below as Algorithm 9.2. The output of the algorithm are the required back-offs with the corresponding lower bound on the probability of satisfying the state chance constraint. Note for *learning* = *true* the mean and covariance function of the GP are recursively updated utilizing the same procedure as for the update of the GP plant model MC sample.

Remark (Conservativeness of chance constraint)**.** *Note to adjust the back-offs we use the ecdf, which does account for the true shape of the underlying probability distribution. This avoids the problem that is often faced in stochastic optimization utilizing Chebyshev's inequality to robustly approximate chance constraints, which is often excessively conservative [207].*

Algorithm 9.2: Back-off iterative updates

Input : $\boldsymbol{\mu}_{\mathbf{x}_0}, \boldsymbol{\Sigma}_{\mathbf{x}_0}, \boldsymbol{\mu}_f(\mathbf{z};\mathcal{D}), \boldsymbol{\Sigma}_f(\mathbf{z};\mathcal{D}), \mathcal{D}, T, V_T(\mathbf{x},t,\hat{\mathbf{U}}_{t:T-1}), \mathbb{X}_t, \mathbb{U}_t, \epsilon, \alpha, \delta$, learning, S, n_b

Initialize : Set all $b_j^{(t)} = 0$ and δ to some reasonable value, set $a_\gamma = 0$ and b_γ to some reasonably high value, such that $b_\gamma - (1-\epsilon)$ has a positive sign. Define $\mathcal{D}_0 = \mathcal{D}$ as the initial dataset.

for n_b *back-off iterations* **do**

  if $n_b > 0$ **then**

    $c_\gamma := (a_\gamma + b_\gamma)/2$

    $b_j^{(t)} := c_\gamma \tilde{b}_j^{(t)} \quad (j,t) \in \{1,\dots,n_g^{(t)}\} \times \{1,\dots,T\}$

  for *each MC sample* $s = 1,2,\dots,S$ **do**

    $\mathcal{D} := \mathcal{D}_0$

    Draw $\boldsymbol{\chi}_0^{(s)}$ from $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_0}, \boldsymbol{\Sigma}_{\mathbf{x}_0})$

    for *each sampling time* $t = 1,2,\dots,T$ **do**

      if *learning = true* **then**

        1. Determine $u_{t-1}^{(s)} = \kappa(\boldsymbol{\mu}_f(\cdot;\mathcal{D}), \boldsymbol{\Sigma}_f(\cdot;\mathcal{D}); \mathbf{x}_t, t)$.

      else

        1. Determine $u_{t-1}^{(s)} = \kappa(\boldsymbol{\mu}_f(\cdot;\mathcal{D}_0), \boldsymbol{\Sigma}_f(\cdot;\mathcal{D}_0); \mathbf{x}_t, t)$.

      2. Draw $\boldsymbol{\chi}_t^{(s)}$ from $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_f(Z_{t-1}^{(s)};\mathcal{D}), \boldsymbol{\Sigma}_f(Z_{t-1}^{(s)};\mathcal{D}))$, where $Z_{t-1}^{(s)} = (\boldsymbol{\chi}_{t-1}^{(s)}, u_{t-1}^{(s)})$.

      3. Define $\mathcal{D}^+ := (\mathcal{D}, (Z_{t-1}^{(s)}, \boldsymbol{\chi}_t^{(s)}))$, where $\boldsymbol{\chi}_t^{(s)}$ can be viewed as noiseless "measurements".

      4. Update the dataset $\mathcal{D} := ([\mathbf{Z}^\mathsf{T}, Z_{t-1}^{(s)\mathsf{T}}]^\mathsf{T}, [\mathbf{Y}, \boldsymbol{\chi}_t^{(s)\mathsf{T}}]^\mathsf{T})$.

      5. Recursively update the GP mean function $\boldsymbol{\mu}_f(\mathbf{z};\mathcal{D}) := \boldsymbol{\mu}_f^+(\mathbf{z};\mathcal{D}^+)$ using Equation 9.19a.

      6. Recursively update the GP covariance function $\boldsymbol{\Sigma}_f(\mathbf{z};\mathcal{D}) := \boldsymbol{\Sigma}_f^+(\mathbf{z};\mathcal{D}^+)$ using Equation 9.19b.

      7. Define $\boldsymbol{\mathcal{X}}^{(s)} = [\boldsymbol{\chi}_0^{(s)}, \boldsymbol{\chi}_1^{(s)}, \dots, \boldsymbol{\chi}_T^{(s)}]^\mathsf{T}$ and $\boldsymbol{\mathcal{U}}^{(s)} = [u_0^{(s)}, \dots, u_{T-1}^{(s)}]^\mathsf{T}$.

  $\hat{\beta} := \hat{F}_{C(\boldsymbol{\mathcal{X}}^{(s)})}(0) = \frac{1}{S}\sum_{s=1}^{S} \mathbf{1}\{C(\boldsymbol{\mathcal{X}}^{(s)}) \le 0\}$

  $\hat{\beta}_{lb} := 1 - \text{betainv}\left(\alpha, S+1-S\hat{\beta}, S\hat{\beta}\right)$

  if $nb = 0$ **then**

    Let $\tilde{b}_j^{(t)} = \hat{F}^{-1}_{g_j^{(t)}}(\delta) - g_j^{(t)}(\overline{\boldsymbol{\chi}}_t) \quad \forall (j,t) \in \{1,\dots,n_g^{(t)}\} \times \{1,\dots,T\}$

    $\hat{\beta}_{lb}^{a_\gamma} := \hat{\beta}_{lb} - (1-\epsilon)$

  else

    $\hat{\beta}_{lb}^{c_\gamma} := \hat{\beta}_{lb} - (1-\epsilon)$

    if $sign(\hat{\beta}_{lb}^{c_\gamma}) = sign(\hat{\beta}_{lb}^{a_\gamma})$ **then**

      $a_\gamma := c_\gamma$

      $\hat{\beta}_{lb}^{a_\gamma} := \hat{\beta}_{lb}^{c_\gamma}$

    else

      $b_\gamma := c_\gamma$

Output : $b_j^{(t)} \quad \forall (j,t) \in \{1,\dots,n_g^{(t)}\} \times \{1,\dots,T\}, \hat{\beta}_{lb}$

9.4.4 Algorithm

The overall algorithm proposed in this paper is summarized in this section. Firstly, the problem to be solved needs to be defined as outlined in Section 9.2. Thereafter, it needs to be decided if the GP NMPC should *learn* online or exploit the state dependency of the uncertainty as shown in Section 9.4. Once the GP NMPC has been formulated the back-offs are determined by running closed-loop simulations of the defined problem as shown in Section 9.4.3. Lastly, these back-offs then give us the tightened constraint set required for the GP NMPC *online*. This GP NMPC is then run online solving the problem initially outlined. An overall summary can be found in Algorithm 9.3.

Algorithm 9.3: Back-off GP NMPC

Offline Computations

1. Build GP state-space model from data-set $\mathcal{D} = (\mathbf{Z}, \mathbf{Y})$ and additive disturbance $\mathbf{\Sigma}_{\boldsymbol{\omega}}$.
2. Choose time horizon T, initial condition mean $\boldsymbol{\mu}_{\mathbf{x}_0}$ and covariance $\mathbf{\Sigma}_{\mathbf{x}_0}$, stage costs ℓ and ℓ_f, state dependent factor η_t, constraint sets $\mathbb{X}_t, \mathbb{U}_t$ $\forall t \in \{1, \ldots, T\}$, chance constraint probability ϵ, ecdf confidence α, tuning parameter δ, decide if *learning* should be carried out, the number of back-off iterations n_b and the number of Monte Carlo simulations S to estimate the back-offs.
3. Determine explicit back-off constraints using Algorithm 9.2.
4. Check final probabilistic value $\hat{\beta}_{lb}$ from Algorithm 9.2 if it is close enough to ϵ.

Online Computations

for $t = 0, \ldots, T-1$ **do**

1. Solve the MPC problem in Equation 9.22 with the tightened constraint set from the *Offline Computations*.
2. Apply the first control input of the optimal solution to the real plant.
3. Measure the state $\mathbf{x}_t$ and update the GP plant model for learning GP NMPC.

Note the back-offs could be also updated online making use of the new initial conditions and updated prediction model in the case of "learning", which would

lead to overall less conservativeness [206]. This would however require to carry-out the *offline* calculations online, which is expensive, and not computationally desirable.

9.5 Case study

The case study utilized in this paper deals with the photo-production of phycocyanin synthesized by cyanobacterium *Arthrospira platensis*. Phycocyanin is a high-value bioproduct and its biological function is to enhance the photosynthetic efficiency of cyanobacteria and red algae. It has been considered as a valuable compound because of its applications as a natural colorant to replace other toxic synthetic pigments in both food and cosmetic production. Furthermore, it has shown great promise for the pharmaceutical industry because of its unique antioxidant, neuroprotective, and anti-inflammatory properties. Using a simplified dynamic model we verify our GP NMPC algorithm by operating this process using a limited dataset. The GP NMPC problem is formulated with an economic objective aiming to directly maximize the bioproduct concentration of the final batch subject to two path constraints and one terminal constraint.

9.5.1 Semi-batch bioreactor model

The simplified dynamic system consists of three ODEs describing the evolution of the concentration of biomass, nitrate, and bioproduct. The dynamic model is based on the Monod kinetics, which describes microorganism growth in nutrient sufficient cultures, where intracellular nutrient concentration is kept constant because of the rapid replenishment. We assume a fixed volume fed-batch. Control inputs are given by the light intensity (I) in μmol.m^{-2}.s^{-1} and nitrate inflow rate (F_N) in mg.L^{-1}.h^{-1} [76]. To capture the effects of light intensity on microalgae growth and bioproduction (photolimitation, photosaturation, and photoinhibition phenomena) the Aiba model is used [2]. The balance equations are given as follows:

$$\frac{dC_X}{dt} = u_m \cdot \frac{I}{I + k_s + \frac{I^2}{k_i}} \cdot C_X \cdot \frac{C_N}{C_N + K_N} - u_d \cdot C_X, \qquad C_X(0) = C_{X0} \tag{9.34a}$$

$$\frac{dC_N}{dt} = -Y_{\frac{N}{X}} \cdot u_m \cdot \frac{I}{I + k_s + \frac{I^2}{k_i}} \cdot C_X \cdot \frac{C_N}{C_N + K_N} + F_N, \quad C_N(0) = C_{N0} \tag{9.34b}$$

$$\frac{dC_{q_c}}{dt} = k_m \cdot \frac{I}{I + k_{sq} + \frac{I^2}{k_{iq}}} \cdot C_X - \frac{k_d C_{q_c}}{C_N + K_{Np}}, \qquad C_{q_c}(0) = C_{q_c\,0} \tag{9.34c}$$

where C_X is the biomass concentration in g/L, C_N is the nitrate concentration in mg/L, and C_{q_c} is the phycocyanin (bioproduct) concentration in the culture in mg/L. The corresponding state vector and control vector are given by $\mathbf{x} = [C_X, C_N, C_{q_c}]^\mathsf{T}$ and $\mathbf{u} = [I, F_N]^\mathsf{T}$ respectively. The initial condition is denoted as $\mathbf{x}_0 = [C_{X0}, C_{N0}, C_{q_c\,0}]^\mathsf{T}$. The missing parameter values can be found in Table 9.1.

Table 9.1: Parameter values for ordinary differential equation system in Equation 9.34.

Parameter	Value	Units
u_m	0.0572	h^{-1}
u_d	0.0	h^{-1}
K_N	393.1	$mg.L^{-1}$
$Y_{\frac{N}{X}}$	504.5	$mg.g^{-1}$
k_m	0.00016	$mg.g^{-1}.h^{-1}$
k_d	0.281	h^{-1}
k_s	178.9	$\mu mol.m^{-2}.s^{-1}$
k_i	447.1	$\mu mol.m^{-2}.s^{-1}$
k_{sq}	23.51	$\mu mol.m^{-2}.s^{-1}$
k_{iq}	800.0	$\mu mol.m^{-2}.s^{-1}$
K_{Np}	16.89	$mg.L^{-1}$

9.5.2 Problem set-up

The time horizon T was set to 12 with an overall batch time of 240h, and consequently the sampling time is 20h. Based on the dynamic system in Equation 9.34 we define the objective and the constraints according to the general problem definition in Section 9.2. The measurement noise matrix $\mathbf{\Sigma_v}$ and disturbance noise matrix $\mathbf{\Sigma_\omega}$ were set to:

$$\mathbf{\Sigma_v} = \mathrm{diag}(4\times10^{-4}, 0.1, 1\times10^{-8}), \quad \mathbf{\Sigma_\omega} = \mathrm{diag}(4\times10^{-4}, 0.1, 1\times10^{-8}) \tag{9.35}$$

The mean $\boldsymbol{\mu}_{\mathbf{x}_0}$ and covariance $\mathbf{\Sigma}_{\mathbf{x}_0}$ of the initial condition are given by:

$$\boldsymbol{\mu}_{\mathbf{x}_0} = [1., 150, 0.]^\mathsf{T}, \quad \mathbf{\Sigma}_{\mathbf{x}_0} = \mathrm{diag}(1\times10^{-3}, 22.5, 0) \tag{9.36}$$

The control algorithm aims to maximize the amount of bioproduct produced C_{q_c} with a penalty on the change of control actions. The corresponding stage and

terminal cost can be stated as follows:

$$\ell(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{\Delta}_{\mathbf{u}_t}^{\top} \mathbf{R} \mathbf{\Delta}_{\mathbf{u}_t} \tag{9.37a}$$

$$\ell_f(\mathbf{x}_T) = -C_{q_c T} \tag{9.37b}$$

where $\mathbf{\Delta}_{\mathbf{u}_t} = \mathbf{u}_t - \mathbf{u}_{t-1}$ and $\mathbf{R} = \mathrm{diag}(3.125 \times 10^{-8}, 3.125 \times 10^{-6})$. The overall objective is then defined by Equation 9.3.

For the case of state dependency we set all η_i to zero except η_0, see Equation 9.22. The value of η_0 was set to 15. Note that for these factors to work properly it is important to normalize the data as we did in this case study.

There are two path constraints in the problem. The amount of nitrate is constrained to remain below 800 mg/L, while the ratio of bioproduct to biomass may not exceed 11.0 mg/g for high density biomass cultivation. These constraints can be stated as:

$$g_1^{(t)}(\mathbf{x}_t) = C_{N t} - 800 \leq 0 \qquad \forall t \in \{0, \ldots, T\} \tag{9.38a}$$

$$g_2^{(t)}(\mathbf{x}_t) = C_{q_c t} - 0.011 C_{X t} \leq 0 \qquad \forall t \in \{0, \ldots, T\} \tag{9.38b}$$

Lastly, there is a terminal constraint on nitrate to reach a final concentration of below 150 mg/L:

$$g_3^{(T)}(\mathbf{x}_T) = C_{NT} - 150 \leq 0, \quad g_3^{(t)}(\mathbf{x}_T) = 0 \; \forall t \in \{0, \ldots, T-1\} \tag{9.39}$$

The maximum probability for violating the joint chance constraint was set to $\epsilon = 0.1$. The control inputs light intensity and nitrate inflow rate are constrained as follows:

$$120 \leq I_t \leq 400 \qquad \forall t \in \{0, \ldots, T\} \tag{9.40a}$$

$$0 \leq F_{N t} \leq 40 \qquad \forall t \in \{0, \ldots, T\} \tag{9.40b}$$

For the back-off iterations we employed $S = 1000$ MC iterations with the initial back-offs computed according to Equation 9.31 with $\delta = 0.1$ and $\alpha = 0.01$. The maximum number of back-off iterations for the bisection algorithm was set to $n_b = 16$.

9.5.3 Implementation details and initial dataset generation

The optimization problem for the GP NMPC in Equation 9.22 is solved using Casadi [9] to obtain the gradients of the problem using automatic differentiation

in conjunction with IPOPT [258]. The "real" plant model was simulated using IDAS [116]. In the next section, different variations of the proposed algorithm are presented, for which two different type of datasets were collected. For the first type of dataset we designed the entire input data matrix $\mathbf{Z}$ according to a Sobol sequence [238] in the range $\mathbf{z}_i \in [0,20]\times[50,800]\times[0,0.18]\times[120,400]\times[0,40]$. The ranges were chosen for the data to cover the expected operating region. The corresponding outputs $\mathbf{Y}$ were then obtained from the IDAS simulation of the system perturbed by Gaussian noise as defined in the problem setup. In the second approach only the control inputs were set according to the Sobol sequence in the range $\mathbf{u}_i \in [120,400]\times[0,40]$ and the corresponding states $\mathbf{Y}$ were obtained from the trajectories of the "real" system perturbed by noise using samples of the initial condition and the time horizon as defined in the problem setup based on these control inputs. The system was simulated in "open-loop" using these control actions, i.e. without any feedback controller present. For both datasets the input data $\mathbf{Z}$ and output data $\mathbf{Y}$ are normalized to zero mean and a standard deviation of one. The reason we use two different types of datasets is to highlight the advantages of accounting for state dependency in two of the algorithm variations. In the first dataset the data is relatively evenly distributed and hence considering the state dependency of the uncertainty to avoid regions with high data sparsity has essentially no effect, while in the second approach there are clearly defined trajectories that can be followed by accounting for state dependency.

9.6 Results and discussions

In this section we present and discuss the results from the case study described in the previous section. For comparison purposes we compare six different variations of the proposed GP NMPC approach, which are as follows:

- GP NMPC 50, 60, 100: GP NMPC approach without learning and without taking into account state dependency for dataset sizes of 50, 60, and 100 points using the first type of dataset.
- GP NMPC learning 50: GP NMPC approach with learning and without state dependency for a dataset size of 50 points, which will be compared to the above case of 50 data points without learning. The first type dataset is utilized.
- GP NMPC SD/NSD 50: GP NMPC approach with and without accounting for the state dependency for a dataset size of 50 points employing the second type of dataset.

In addition, we compare the approaches to a nominal NMPC algorithm based

on the GP model to show the importance of employing back-offs to prevent constraint violations, i.e. we run the GP NMPC on the "real" plant model, while setting the back-offs to zero. The results of the outlined runs are summarized in Figures 9.6-9.12, and in Tables 9.2-9.3. In Figures 9.6-9.7 we show the evolution of the back-off factor and the probability of constraint satisfaction $\hat{\beta}_{lb}$ over the 16 back-off iterations from Algorithm 9.2. The next two Figures 9.4-9.5 show the 1000 MC trajectories of the constraints with a line to highlight the nominal prediction of the GP NMPC. Next the GP NMPC was applied to the "real" plant with back-offs from the final iteration and without back-offs referred to as *nominal* as shown in Figures 9.8-9.9. Figure 9.10 shows the probability density function of the objective values obtained from the "real" plant, where in the figure larger objective values correspond to better objective values. Figure 9.11 shows representative control trajectories for GP NMPC 50, 60, 100 compared with the optimal trajectory obtained from solving the OCP of the "real" plant ignoring uncertainties. Figure 9.12 shows the back-off values for the nitrate constraints g_1 and g_2 for GP NMPC 50 and GP NMPC 50 learning. Lastly, Table 9.2 shows the mean values for the back-offs averaged over time for the final back-off iteration, while Table 9.3 shows the attained probability of satisfaction $\hat{\beta}_{lb}$ together with the average computational times for solving a single GP NMPC optimization problem. We can draw the following conclusions from these results:

- Figures 9.6-9.7 and Table 9.2 show that apart from GP NMPC 50 the other variations reach the required $\hat{\beta}_{lb}$ and hence successfully converge to a reasonable back-off factor. For these, as expected, a low back-off value leads to too low $\hat{\beta}_{lb}$ values near zero, while too high back-off values lead to too high $\hat{\beta}_{lb}$ values. The value of $\hat{\beta}_{lb}$ does vary by ± 0.01 even on convergence, which is due to the randomness of the MC samples. Nonetheless, since $\hat{\beta}_{lb}$ is a sample robust value, it is high enough if at least 0.9 is reached once over the 16 iterations, which is the case for all of them. Note that GP NMPC 50 does not converge, since even without back-offs the NMPC remains feasible for all MC trajectories and hence the bisection procedure fails. The performance of GP NMPC 50 is however also by far the worst. This is due to insufficient amounts of data in crucial areas for the control problem.
- From GP NMPC 50, 60 to 100 the objective values steadily increase and hence improve with increased number of data points as shown in Figure 9.10. This is as expected, since more data points should lead to a more accurate GP plant model and hence more optimal control actions. Lastly, a more accurate GP plant model should also require less conservative back-offs. For the constraint g_2 the mean of the back-off values steadily decreases from 0.022 mg/L for GP NMPC 50 to 0.003 for GP NMPC 100 as shown in

Table 9.2. For the constraints g_1/g_3 on the other hand the mean of the back-offs decreases dramatically from GP NMPC 50 with a value of 250mg/L to a value of 34.2mg/L for GP NMPC 60, while slightly increasing again for GP NMPC 100 to 38.8 mg/L. This is further illustrated in Figure 9.4, for which the spread of the trajectories decreases steadily from GP NMPC 50, 60 to 100. All in all, larger datasets lead to improved solutions.

- The learning approach GP NMPC 50 learning leads to a reasonable solution with an objective value that is on average higher and therefore an improvement over GP NMPC 60 as can be seen in Figure 9.10. Further, the sharper peak of the objective value suggests a more reliable performance. In contrast, GP NMPC 50 without learning is unable to determine a good solution and therefore has an objective value that is considerably worse than the remaining scenarios with an objective value that is on average over 30% lower. GP NMPC 50 is also unable to reach a $\hat{\beta}_{lb}$ value of 0.9 and has instead a much higher value as shown in Table 9.3, but at the expense of performance. This is highlighted in Figure 9.11 in which the control trajectory of GP NMPC 50 decreases the nitrate flowrate early to satisfy the terminal constraint, which leads to a sub optimal solution. This is believed to be due to the high uncertainty of the $g_1/g_2/g_3$ constraint trajectories of GP NMPC 50, which can be seen by the large spread of the trajectories in Figure 9.4. GP NMPC 50 learning on the other hand has a spread of the constraints $g_1/g_2/g_3$ that is significantly less than GP NMPC 50. This is further highlighted by the considerably higher back-off values of GP NMPC 50 compared to GP NMPC 50 learning as can be seen in Table 9.2, where the g_1/g_3 back-off values are nearly 400% larger, while the g_2 back-off values are over 300% larger. In conclusion, accounting for online learning has lead to a significantly better solution, although it should be noted that at larger datasets the effect is nearly negligible.

- GP NMPC 50 can be seen to be less erratic than GP NMPC 50 learning in Figure 9.4, which is however expected. GP NMPC 50 uses the same prediction model throughout and hence the control inputs are only influenced by changes in the initial condition. For GP NMPC 50 learning on the other hand the prediction model changes at each sampling point and hence this leads to more irregular behaviour, which can be seen by the increased oscillations. This then leads to overall improved control actions and performance due to exploiting the new data available, as can be seen in Figure 9.10.

- It can be seen in Figure 9.5 that GP NMPC NSD 50 has a larger spread of trajectories than GP NMPC SD 50. This is as expected, since GP NMPC

SD 50 aims directly in its objective to minimize uncertainty. Consequently, the mean of the back-off values of constraints g_1/g_3 and g_2 in Table 9.2 are over 50% larger and over 100% greater than those for GP NMPC SD 50, respectively. Nonetheless, the attained average objective of GP NMPC SD 50 is marginally lower and hence worse as can be seen in Figure 9.10. This is expected, since accounting for state dependency reduces conservativeness, but may also lead to a sub optimal solution due to conflicting objectives.

- In Table 9.3 the average computational times of a single GP NMPC evaluation are shown, which range from 135ms to 48ms. Overall, it can be seen that by far the largest computational times are attributed to GP NMPC 100, which is reasonable since the complexity of GP plant models grows exponentially with the number of data points. GP NMPC 50 and GP NMPC 50 learning can be seen however to have higher computational times, which is due to a more complex optimization problem from the reduced amount of data. This is further highlighted by GP NMPC 50 SD and GP NMPC 50 NSD attaining the lowest average computational times due to the second type of dataset leading to easier optimization solutions. In Table 9.3 we can further see that the computational times required for a single back-off iteration is for all variations nearly solely determined by the GP NMPC evaluation time.

- Lastly, in Figures 9.8-9.9 trajectories of the constraints are shown by applying the GP NMPC variants to the "real" plant. It can be seen that the *nominal* variations of GP NMPC 50, 60, 100, and GP NMPC 50 learning with back-offs set to zero violate the nitrate constraint g_1 to remain below 800mg/L by a substantial amount of up to 50mg/L for GP NMPC 50 learning and GP NMPC 60. With back-offs on the other hand the approaches remain feasible throughout the run, which illustrates the importance of employing back-offs. GP NMPC NSD 50 *nominal* can be also seen to violate the nitrate constraint g_1 by 50mg/L, while GP NMPC SD 50 *nominal* does not violate this constraint. This is likely due to GP NMPC SD 50 *nominal* following a feasible trajectory in the dataset. GP NMPC NSD 50 with back-offs remains feasible. Overall, it can be seen that back-offs are important to achieve feasibility given the presence of plant-model mismatch.

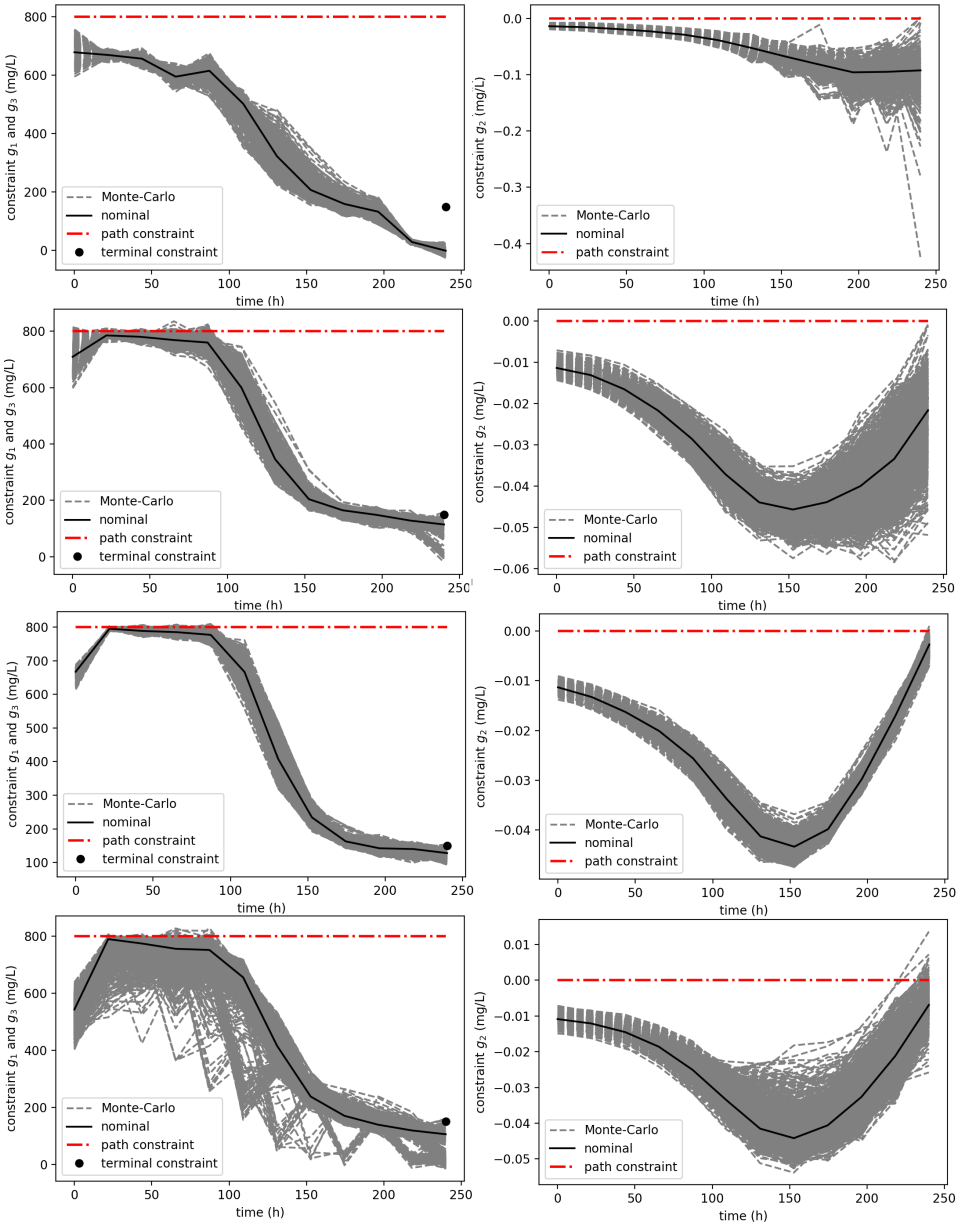


Figure 9.4: The 1000 MC trajectories at the final back-off iteration of the nitrate concentration constraints g_1 and g_3 (LHS) and the ratio of bioproduct to biomass constraint g_2 (RHS) for from top to bottom GP NMPC 50, 60, 100 and GP NMPC 50 learning.

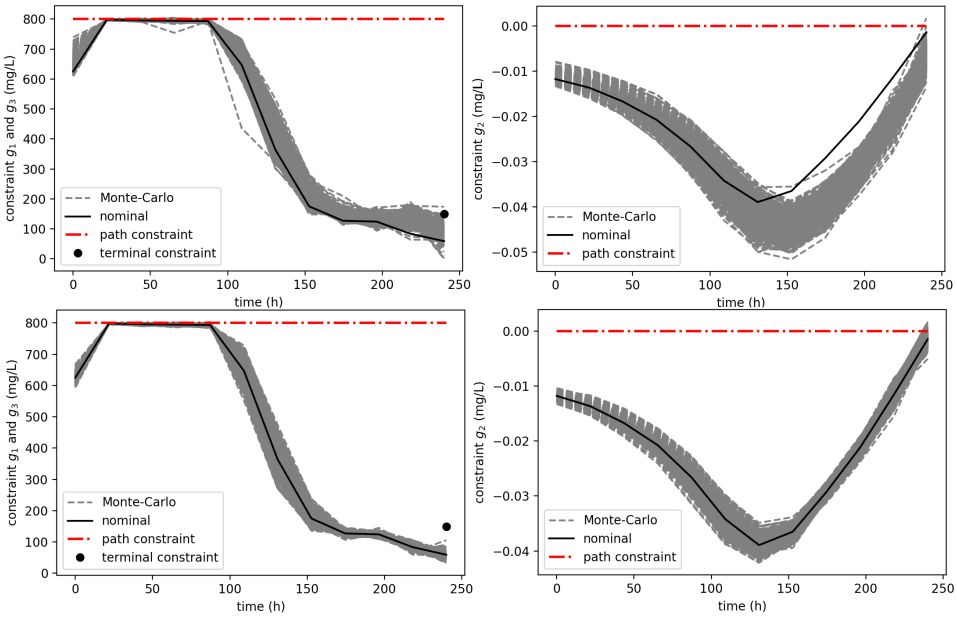


Figure 9.5: The 1000 MC trajectories at the final back-off iteration of the nitrate concentration constraints g_1 and g_3 (LHS) and the ratio of bioproduct to biomass constraint g_2 (RHS) for GP NMPC NSD 50 (top) and GP NMPC SD 50 (bottom).

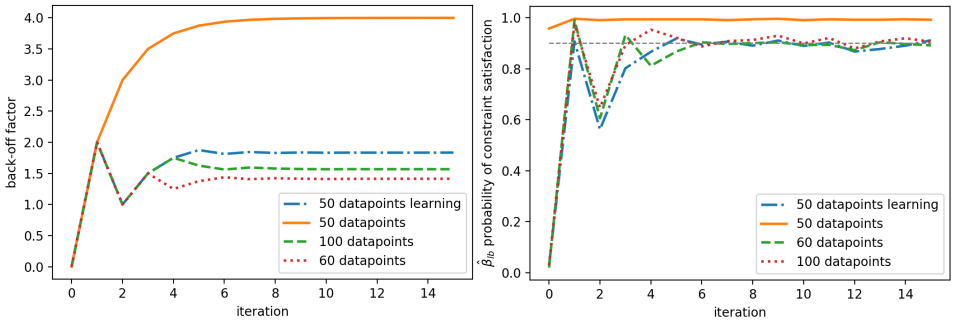


Figure 9.6: Plots of evolution of the back-off factor and the probability of constraint satisfaction $\hat{\beta}_{lb}$ over the 16 back-off iterations for GP NMPC 50, 60, 100, and GP NMPC learning 50.

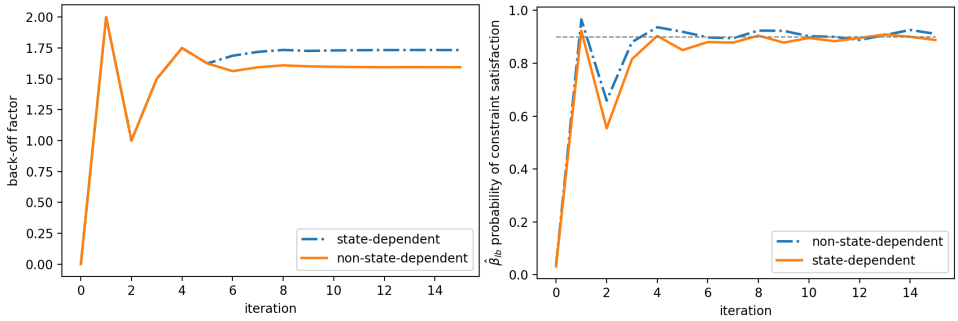


Figure 9.7: Plots of evolution of the back-off factor and the probability of constraint satisfaction $\hat{\beta}_{lb}$ over the 16 back-off iterations for GP NMPC SD 50 and GP NMPC NSD 50.

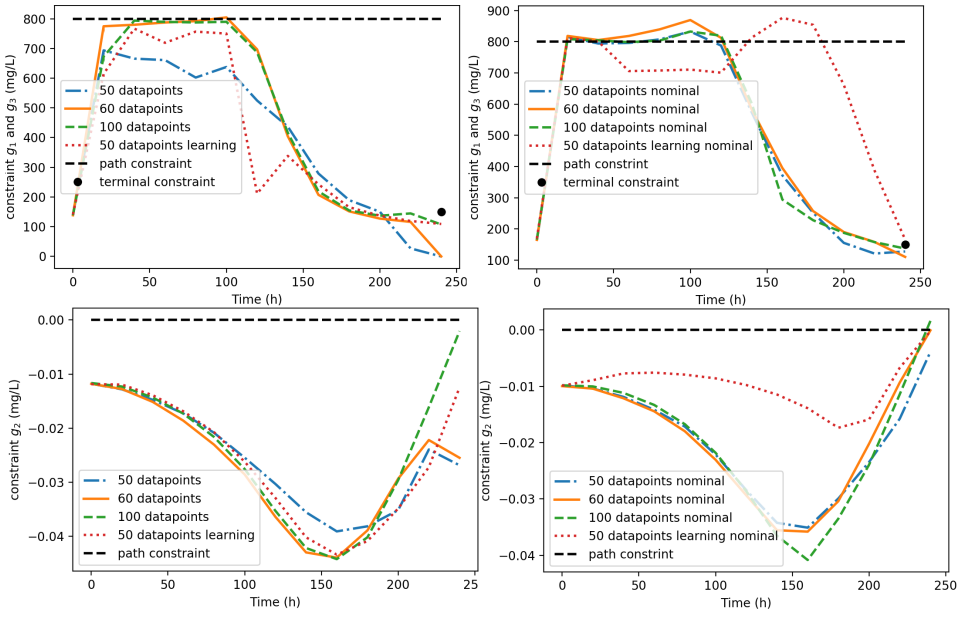


Figure 9.8: Trajectories of the nitrate concentration constraints g_1 and g_3 (top) and the ratio of bioproduct to biomass constraint g_2 (bottom) for the GP NMPC 50, 60, 100, and GP NMPC 50 learning applied to the "real" plant model with the final tightened constraint set on the LHS and with no back-off constraints on the RHS referred to as *nominal*.

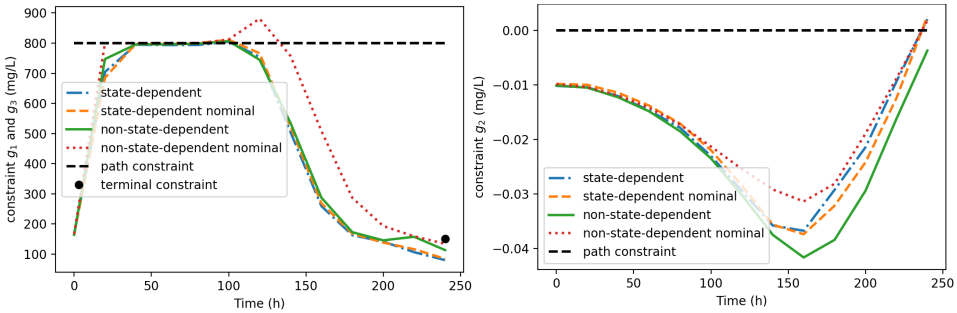


Figure 9.9: Trajectories of the nitrate concentration constraints g_1 and g_3 (LHS) and the ratio of bioproduct to biomass constraint g_2 (RHS) for the GP NMPC 50, 60, 100, and GP NMPC 50 learning applied to the "real" plant model with the final tightened constraint set and with no back-off constraints referred to as *nominal*.

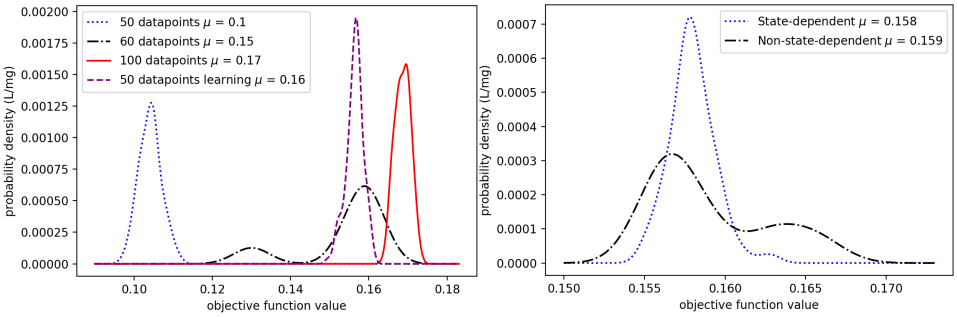


Figure 9.10: Probability density function for the "real" plant objective values for all variations of the GP NMPC algorithm.

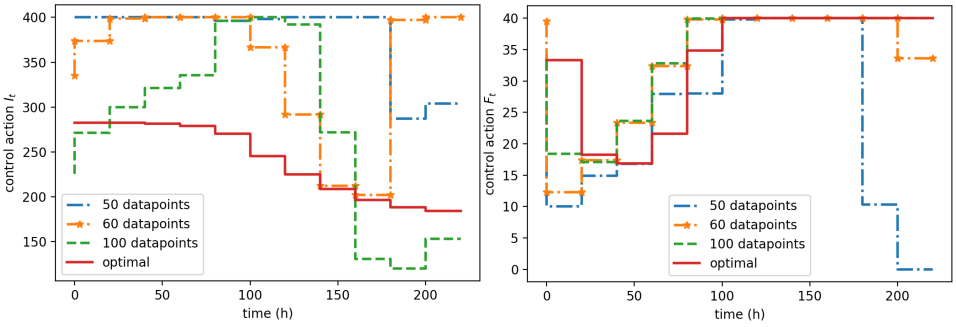


Figure 9.11: Example control trajectories for the light intensity on the LHS and the nitrate flow rate on the RHS based on GP NMPC 50, 60, 100. The red line represents the optimal control trajectories ignoring the noise present in the process.

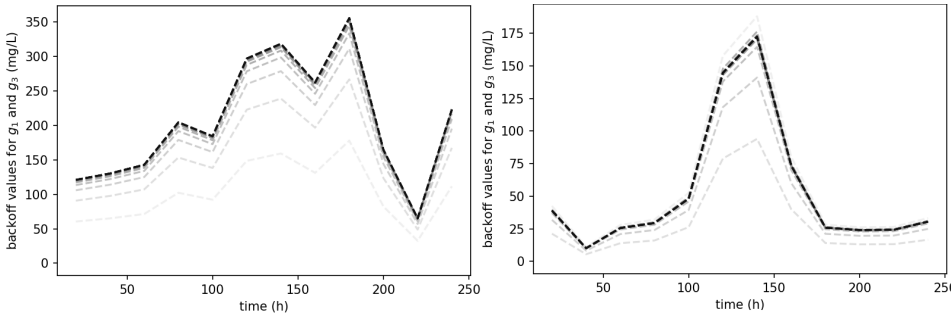


Figure 9.12: Example back-off values for the nitrate concentration constraints g_1 and g_3 of GP NMPC 50 (LHS) and GP NMPC 50 learning (RHS). The lines are plotted over the 16 back-off iterations, which are faded out towards earlier iterations.

Table 9.2: The mean of the back-off values for the nitrate concentration constraints g_1/g_3 and the ratio of bioproduct to biomass constraint g_2 from the final back-off iteration.

Algorithm variation	Mean back-off g_1/g_3 (mg/L)	Mean back-off g_2 (mg/L)
GP NMPC 50	205.8	0.022
GP NMPC 60	34.2	0.008
GP NMPC 100	38.8	0.003
GP NMPC learning 50	54.0	0.007
GP NMPC 50 SD	23.3	0.002
GP NMPC 50 NSD	36.2	0.004

Table 9.3: Lower bound on the probability of satisfying the joint constraint $\hat{\beta}_{lb}$, average computational times to solve a single optimal control problem (OCP) for the GP NMPC, and the average computational time required to complete one back-off iteration.

Algorithm variation	Probability $\hat{\beta}_{lb}$	OCP time (ms)	Back-off iteration time (s)
GP NMPC 50	0.99	65	782
GP NMPC 60	0.89	54	753
GP NMPC 100	0.91	135	1626
GP NMPC learning 50	0.91	69	825
GP NMPC 50 SD	0.89	48	574
GP NMPC 50 NSD	0.91	49	584

9.7 Conclusions

In conclusion, a new approach is proposed for finite-horizon control problems using NMPC in conjunction with GP state space models. The method utilizes

the probabilistic nature of GPs to sample deterministic functions of possible plant models. Tightened constraints using explicit back-offs are then determined, such that the closed-loop simulations of these possible plant models are feasible to a high probability. In addition, it is shown how probabilistic guarantees can be derived based on the number of constraint violations from the simulations. Furthermore, it is shown that online learning and state dependency of the uncertainty can be taken into account explicitly in this method, which leads to overall less conservativeness. Moreover, the computational times are shown to be relatively low, since constraint tightening is performed offline. Finally, through the comprehensive semi-batch bioprocess case study, the efficiency and potential of this method for the optimisation of complex stochastic systems (e.g. biological processes) is well demonstrated.

9.8 Recursive inverse matrix update

In this paper we are often concerned with inverting matrices as shown in Section 9.3.2 in Equation 9.17c:

$$\boldsymbol{\Sigma}_{\mathbf{Y}}^{+-1} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{Y}} & \mathbf{k}^{\mathsf{T}}(\mathbf{z}^{+}) \\ \mathbf{k}(\mathbf{z}^{+}) & k(\mathbf{z}^{+},\mathbf{z}^{+})(+\sigma_{\nu}^{2}) \end{bmatrix}^{-1} \tag{9.41}$$

This is an recursive update formula and hence $\boldsymbol{\Sigma}_{\mathbf{Y}}^{-1}$ is known *a priori*. In this Section we introduce a recursive formula to exploit this fact to obtain $\boldsymbol{\Sigma}_{\mathbf{Y}}^{+-1}$ in a cheaper fashion taken from [241] adjusted to our case. The following quantities need to be computed to obtain $\boldsymbol{\Sigma}_{\mathbf{Y}}^{+-1}$:

$$\mathrm{I} = \mathbf{k}^{\mathsf{T}}(\mathbf{z}^{+})\boldsymbol{\Sigma}_{\mathbf{Y}}^{-1} \tag{9.42a}$$

$$\mathrm{II} = \mathbf{k}^{\mathsf{T}}(\mathbf{z}^{+})\mathrm{I}^{\mathsf{T}} \tag{9.42b}$$

$$\mathbf{C}_{12} = \mathrm{I}^{\mathsf{T}} \times \mathrm{II} \tag{9.42c}$$

$$C_{11} = \boldsymbol{\Sigma}_{\mathbf{Y}}^{-1} - \mathrm{I}^{\mathsf{T}} \times \mathbf{C}_{12}^{\mathsf{T}} \tag{9.42d}$$

$$C_{22} = -\left(\mathrm{II} - k(\mathbf{z}^{+},\mathbf{z}^{+})(+\sigma_{\nu}^{2})\right)^{-1} \tag{9.42e}$$

The inverted matrix is then given by:

$$\boldsymbol{\Sigma}_{\mathbf{Y}}^{+-1} = \begin{bmatrix} C_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{12}^{\mathsf{T}} & C_{22} \end{bmatrix} \tag{9.43}$$

Chapter 10

Hybrid Gaussian process modelling applied to economic stochastic model predictive control of batch processes

This chapter is based on **Paper H**: E. Bradford, L. Imsland, M. Reble, and E. A. del Rio-Chanona. Hybrid Gaussian process modelling applied to economic stochastic model predictive control of batch processes. In *Progress on Economic and Distributed Model Predictive Control and Applications*. Springer, submitted, 2020.

Summary

Nonlinear model predictive control (NMPC) is an efficient approach for the control of nonlinear multivariable dynamic systems with constraints, which however requires an accurate plant model. Plant models can often be determined from first principles, parts of the model are however difficult to derive using physical laws alone. In this paper a hybrid Gaussian process (GP) first principles modeling scheme is proposed to overcome this issue, which exploits GPs to model the parts of the dynamic system that are difficult to describe using first principles. GPs not only give accurate predictions, but also quantify the residual uncertainty of this model. It is vital to account for this uncertainty in the control algorithm, to prevent constraint violations and performance deterioration. Monte Carlo samples of the GPs are generated offline to tighten constraints of the NMPC to ensure joint probabilistic constraint satisfaction online. Advantages of our method include fast

online evaluation times, possibility to account for online learning alleviating conservativeness, and exploiting the flexibility of GPs and the data efficiency of first principle models. The algorithm is verified on a case study involving a challenging semi-batch bioreactor.

10.1 Introduction

Model Predictive Control (MPC) was developed in the late seventies and has progressed significantly since then. Model predictive control is the only advanced control methodology, which has made a significant impact on industrial control engineering [165]. MPC is especially useful to deal with multivariable control problems and important process constraints . Many processes are highly nonlinear and may be operated along state trajectories, which motivates the use of nonlinear MPC (NMPC). In particular NMPC applications based on first principles models are becoming increasingly popular due to the advent of improved optimization methods and the availability of more models [27]. In this paper we focus on *finite horizon* control problems, for which chemical batch processes are a particularly important example. These are employed in many different chemical sectors due to their inherent flexibility. Previous works for batch processes include NMPC based on the extended and unscented Kalman filter [33, 187], polynomial chaos expansions [37, 177], and multi-stage NMPC [164].

A major limitation of NMPC in practice is the requirement for an accurate dynamic plant model, which has been cited to take up to 80% of the MPC commissioning effort [244]. The required dynamic model for NMPC is often derived from first principles taking advantage of the available prior knowledge of the process [188]. While this can be an efficient modeling approach, often parts of the model are notoriously difficult to represent using physical laws. In addition, modeling certain phenomena may require excessive amounts of computational time. Hybrid modeling approaches have therefore been proposed, which combine first principles modeling with data-driven regression methods. For example in chemical engineering hybrid models have been developed to capture chemical reaction kinetics [216, 247], the complex mechanics of catalyst deactivation [14], or for the correction of first principles models using available measurements [26, 112]. Most hybrid modeling applications focused on using neural networks (NNs). In this paper we propose to use Gaussian processes (GPs) instead [220] due to their ability to not only provide predictions, but also provide a measure of uncertainty for these predictions difficult to obtain by other nonlinear modeling approaches [141]. GPs compute predictive Gaussian distributions as output estimates, whose mean serves as the prediction and the variance as a measure of uncertainty. GPs for regression were first introduced by [200]. It has been shown in [219] that GPs achieve

comparable predictive accuracy as other modeling approaches like NNs. It is important to account for the GP measure of uncertainty to avoid constraint violations and performance deterioration. To consider uncertainty for NMPC formulations explicitly robust MPC [53] and stochastic MPC [88] approaches have been developed. Previous works on using GPs for hybrid modeling mainly focused on linear ordinary differential equation systems of first- and second order that can be solved exactly, see for example [7, 153, 228].

GP-based MPC was first proposed in [183], in which the GP is recursively updated for reference tracking. In [142, 143] it is proposed to identify the GP offline and apply it online for NMPC instead. The variance therein is constrained to avoid the NMPC steering into regions of high uncertainty. Furthermore, GPs have been used to overcome deviations between the approximate plant model and the real plant model [139, 167]. GPs may also act as an efficient surrogate to estimate the mean and variance required for stochastic NMPC [36]. Applications of GP-based MPC include the control of an unmanned quadrotor [60], the control of a gas-liquid separation process [158], and the steering of miniature cars [113]. While these works show the feasibility of GP-based MPC, most formulations use stochastic uncertainty propagation to account for the uncertainty measure provided by the GP, e.g. [60, 113, 142, 143]. An overview of these approaches can be found in [114]. Major limitations of stochastic propagation are open-loop uncertainty growth and significantly increased computational times. Recently, several papers have proposed alternative techniques to consider the GP uncertainty measure. [147] propagate ellipsoidal sets using linearization and accounting for the linearization error by employing Lipschitz constants, which is however relatively conservative. [169] use a robust MPC approach by bounding the one-step ahead error from the GP, while [239] suggest a robust control approach for linear systems to account for unmodeled nonlinearities. This approach may however be infeasible if the deviation between the nonlinear system and linear system is too large.

In this paper we extend a method first proposed in [40, 42] to the hybrid modeling case. The approach determines explicit back-offs to tighten constraints offline using closed-loop Monte Carlo (MC) simulations for *finite horizon* control problems. These in turn guarantee the satisfaction of probabilistic constraints online. There are several advantages of this approach including avoidance of closed-loop uncertainty growth, fast online computational times, and probabilistic guarantees on constraint satisfaction. In addition, sampled GPs lead to deterministic models that can be easily handled in a hybrid modeling framework. In contrast, obtaining statistical moments for stochastic uncertainty propagation for hybrid models is difficult.

The paper is comprised of the following sections. In Section 9.2 the problem definition is given. Thereafter, in Section 8.4 we outline the solution approach. Section 8.5 outlines the semi-batch bioprocess case study to be solved, while in Section 10.5 results and discussions for this case study are presented. Section 10.6 concludes the paper.

10.2 Problem definition

The dynamic system in this paper is assumed to be given by a discrete time nonlinear equation system with additive disturbance noise and an unknown function $\mathbf{q}(\cdot, \mathbf{u}_k)$:

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}(\cdot, \mathbf{u}_k)) + \boldsymbol{\omega}_k, \quad \mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_0}, \boldsymbol{\Sigma}_{\mathbf{x}_0}), \tag{10.1}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_\mathrm{x}}$ represent the states, $\mathbf{u}_k$ denotes the control inputs, $\mathbf{q} : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}^{n_\mathrm{q}}$ are unknown nonlinear functions, and $\mathbf{F} : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \times \mathbb{R}^{n_\mathrm{q}} \to \mathbb{R}^{n_\mathrm{x}}$ are known nonlinear functions. The initial condition $\mathbf{x}_0$ is assumed to follow a Gaussian distribution with mean $\boldsymbol{\mu}_{\mathbf{x}_0}$ and covariance $\boldsymbol{\Sigma}_{\mathbf{x}_0}$. Additive disturbance noise is denoted by $\boldsymbol{\omega}_k$, which is assumed to follow a Gaussian distribution with zero mean and covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\omega}}$, $\boldsymbol{\omega}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}})$.

Note most first principles models are given in continuous-time, which has important implications on the unknown function $\mathbf{q}(\cdot, \mathbf{u}_k)$. For example, this model needs to be well-identified not only at discrete times. Let $\delta_t = t_k - t_{k-1}$ be a constant sampling time at which measurements are taken. The corresponding continuous-time model to $\mathbf{F}(\cdot)$ is represented by $\mathbf{f}(\cdot)$ assuming a zero hold:

$$\mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{q}(\cdot, \mathbf{u}_k)) = \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x}(t), \mathbf{u}_k, \mathbf{q}(\mathbf{x}(t), \mathbf{u}_k))\mathrm{d}t + \mathbf{x}_k, \tag{10.2}$$

where $\mathbf{x}_k = \mathbf{x}(t_k)$ is the value of the state at discrete time k.

In general $\mathbf{q}(\cdot)$ may be composed of $n_\mathbf{q}$ separate scalar functions:

$$\mathbf{q}(\mathbf{x}, \mathbf{u}) = [q_1(\mathbf{q}_1^{in}(\mathbf{x}, \mathbf{u})), \ldots, q_{n_\mathbf{q}}(\mathbf{q}_{n_\mathbf{q}}^{in}(\mathbf{x}, \mathbf{u}))]^\mathsf{T} \tag{10.3}$$

with n_q separate input functions $\mathbf{q}_i^{in} : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}^{n_{\mathbf{q}_i^{in}}}$ for $i = 1, \ldots, n_\mathbf{q}$. Note these input functions are assumed to be known, since commonly the unknown function denotes an unmodeled physical process, for which the inputs are known a priori. The input dimension $n_{\mathbf{q}_i^{in}}$ is usually much lower than the dimension of states and control inputs combined, and therefore modeling these components can be considerably more data-efficient than determining the full state space model from data instead.

The variable $\boldsymbol{\omega}_k$ represents additive disturbance noise with zero mean and a covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\omega}}$. The measurement at discrete time $t = t_k$ can be expressed as follows:

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \boldsymbol{\nu}_k, \tag{10.4}$$

where $\mathbf{y}_k$ is the corresponding measurement, $\mathbf{H} \in \mathbb{R}^{n_\mathrm{y} \times n_\mathrm{x}}$ is the linear observation model, and $\boldsymbol{\nu}_k$ denotes additive measurement noise with zero mean and a covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\nu}}$.

The aim of the control problem is to minimize a finite-horizon cost function:

$$V_T(\mathbf{x}_0, \mathbf{U}) = \mathbb{E}\left[\sum_{k=0}^{T-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + \ell_f(\mathbf{x}_T)\right], \tag{10.5}$$

where $T \in \mathbb{N}$ is the time horizon, $\mathbf{U} = [\mathbf{u}_0, \ldots, \mathbf{u}_{T-1}]^\mathsf{T} \in \mathbb{R}^{T \times n_\mathrm{u}}$ is a joint matrix over all control inputs for time horizon T, $\ell : \mathbb{R}^{n_\mathrm{x}} \times \mathbb{R}^{n_\mathrm{u}} \to \mathbb{R}$ represent the stage costs, and $\ell_f : \mathbb{R}^{n_\mathrm{x}} \to \mathbb{R}$ is the terminal cost.

The control inputs are subject to hard constraints:

$$\mathbf{u}_k \in \mathbb{U} \quad \forall k \in \{0, \ldots, T-1\}. \tag{10.6}$$

The states are subject to the satisfaction of a joint nonlinear chance constraint over the time horizon T, which can be stated as:

$$\mathbb{P}\left\{\bigcap_{k=0}^{T} \{\mathbf{x}_k \in \mathbb{X}_k\}\right\} \geq 1 - \epsilon, \tag{10.7a}$$

where $\mathbb{X}_t$ is defined as:

$$\mathbb{X}_k = \{\mathbf{x} \in \mathbb{R}^{n_\mathrm{x}} \mid g_j^{(k)}(\mathbf{x}) \leq 0, j = 1, \ldots, n_g\}. \tag{10.7b}$$

The state constraint requires the joint event of all $\mathbf{x}_k$ for all $k \in \{0, \ldots, T\}$ fulfilling the nonlinear constraint sets $\mathbb{X}_k$ to have a probability greater than $1 - \epsilon$.

It is assumed that $\mathbf{f}(\cdot)$ and $\mathbf{q}_i^{in}$ for $i = 1, \ldots, n_\mathbf{q}$ are known, while $\mathbf{q}(\cdot)$ is unknown and needs to be identified from data. We assume we are given N noisy measurements according to Equation 10.4, which are represented by the following two matrices:

$$\mathbf{Z} = [\mathbf{z}_k^{(1)}, \ldots, \mathbf{z}_k^{(N)}]^\mathsf{T} \in \mathbb{R}^{N \times n_\mathbf{z}}, \tag{10.8a}$$

$$\mathbf{Y} = [\mathbf{y}_{k+1}^{(1)}, \ldots, \mathbf{y}_{k+1}^{(N)}]^\mathsf{T} \in \mathbb{R}^{N \times n_\mathrm{y}}, \tag{10.8b}$$

where $\mathbf{z}_k^{(i)} = (\mathbf{x}_k^{(i)}, \mathbf{u}_k^{(i)})$ is a tuple of $\mathbf{x}_k^{(i)}$ and $\mathbf{u}_k^{(i)}$, which are the i-th input of the data at discrete time k with corresponding noisy measurements given by $\mathbf{y}_{k+1}^{(i)}$ at discrete time $k+1$. The matrix $\mathbf{Z}$ is a collection of input data with the corresponding noisy observations collected in $\mathbf{Y}$.

The noise in this problem arises in part from the additive disturbance noise $\boldsymbol{\omega}$ and from the noisy initial condition $\mathbf{x}_0$. The more important source of noise however originates from the unknown function $\mathbf{q}(\cdot)$, which is identified from only *finite* amount of data. To solve this problem we train GPs to approximate $\mathbf{q}(\cdot)$ from the data in Equation 10.8. In the next section we first introduce GPs to model the function $\mathbf{q}(\cdot)$, which then also represent the residual uncertainty of $\mathbf{q}(\cdot)$. This uncertainty representation is thereafter exploited to obtain the required stochastic constraint satisfaction of the closed-loop system.

10.3 Solution approach

10.3.1 Gaussian process hybrid model training

In this section we introduce GPs to obtain a probabilistic model description for $\mathbf{q}(\cdot)$. GPs are an example of flexible non-parametric models. The main appeal of GPs is the fact that it requires little prior process knowledge and quantifies the residual model uncertainty. A GP describes a distribution over functions and can be viewed as a generalization of multivariate Gaussian distributions, and hence can be utilized as a prior probability distribution on unknown functions in a Bayesian framework. Formally, a GP is a collection of random variables of which any finite subset follows a Gaussian distribution. For more information on GPs refer to [220, 263]. Hybrid modeling in this paper refers to the combination of first principles modeling and data-driven modeling.

We use a separate GP for each component $q_i(\mathbf{q}_i^{in}(\mathbf{x}, \mathbf{u}))$ for $i = 1, \ldots, n_{\mathbf{q}}$, which is standard practice in the GP community to handle multivariate outputs [71]. Let i refer to the GP of function i of $\mathbf{q}$. Assume $q_i(\cdot)$ is distributed as a GP with mean function $m_i(\cdot)$ and covariance function $k_i(\cdot, \cdot)$, which fully specifies the GP prior:

$$q_i(\cdot) \sim GP(m_i(\cdot), k_i(\cdot, \cdot)). \tag{10.9}$$

The *choice* of the mean and covariance function define the GP prior. In this study we use a zero mean function and the squared-exponential (SE) covariance function:

$$m_i(\mathbf{q}_i^{in}) := 0, \tag{10.10a}$$

$$k_i(\mathbf{q}_i^{in}, \mathbf{q}_i^{'in}) := \zeta_i^2 \exp\left(-\frac{1}{2}(\mathbf{z} - \mathbf{z}')^\top \boldsymbol{\Lambda}_i (\mathbf{z} - \mathbf{z}')\right), \tag{10.10b}$$

where $\mathbf{q}_i^{in}, \mathbf{q}_i^{'in} \in \mathbb{R}^{n_z}$ are arbitrary inputs, ζ_i^2 denotes the covariance magnitude, and $\mathbf{\Lambda}_i := \text{diag}(\lambda_1, \ldots, \lambda_{n_z})$ is a scaling matrix.

Remark (Prior assumptions)**.** *Zero mean can be realized by normalizing the data. Choosing the SE covariance function assumes the function to be modeled $q_i(\cdot)$ is smooth and stationary.*

Now assume we are given N values of $q_i(\cdot)$, which we jointly denote as:

$$\mathbf{Q}_i = [q_i^{(1)}, \ldots, q_i^{(N)}]^\mathsf{T}. \tag{10.11}$$

Assume these correspond to their $q(\cdot)$ values at the inputs defined in $\mathbf{Z}$ in Equation 10.8. The corresponding input response matrices to $\mathbf{Z}$ are then given by:

$$\mathbf{Q}_i^{in} = [\mathbf{q}_i^{in}(\mathbf{z}_k^{(1)}), \ldots, \mathbf{q}_i^{in}(\mathbf{z}_k^{(N)})]^\mathsf{T}. \tag{10.12}$$

According to the GP prior the data vectors $\mathbf{Q}_i$ follow the multivariate normal distribution:

$$\mathbf{Q}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathbf{Q}_i}), \tag{10.13}$$

where $[\mathbf{\Sigma}_{\mathbf{Q}_i}]_{lm} = k_i(\mathbf{q}_i^{in}(\mathbf{z}_k^{(l)}), \mathbf{q}_i^{in}(\mathbf{z}_k^{(m)})) + \sigma_{\nu i}^2 \delta_{l,m}$ for each pair $(l, m) \in \{1, \ldots, N\}^2$. In essence this places a likelihood on the training dataset based on the continuity and smoothness assumptions made by the choice of the covariance function. The characteristic length-scales and hyperparameters introduced are jointly denoted by $\mathbf{\Psi}_i = [\lambda_1, \ldots, \lambda_{n_{\mathbf{q}_i^{in}}}, \zeta_i^2, \sigma_{\nu i}^2]^\mathsf{T}$.

Given a value of $\mathbf{Q}_i$ we can further determine a likelihood for values not part of $\mathbf{Q}_i$ using conditioning. Let $\hat{\mathbf{Q}}_i$ represent $\hat{N}$ such values at the inputs:

$$\hat{\mathbf{Q}}_i^{in} = [\mathbf{q}_i^{in}(\hat{\mathbf{z}}_k^{(1)}), \ldots, \mathbf{q}_i^{in}(\hat{\mathbf{z}}_k^{(\hat{N})})]^\mathsf{T}. \tag{10.14}$$

From the prior GP assumption $\mathbf{Q}_i$ and $\hat{\mathbf{Q}}_i$ follow a joint Gaussian distribution:

$$\begin{bmatrix} \mathbf{Q}_i \\ \hat{\mathbf{Q}}_i \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma}_{\mathbf{Q}_i} & \mathbf{\Sigma}_{\hat{\mathbf{Q}}_i, \mathbf{Q}_i}^\mathsf{T} \\ \mathbf{\Sigma}_{\hat{\mathbf{Q}}_i, \mathbf{Q}_i} & \mathbf{\Sigma}_{\hat{\mathbf{Q}}_i} \end{bmatrix} \right), \tag{10.15}$$

where $[\mathbf{\Sigma}_{\hat{\mathbf{Q}}_i}]_{lm} = k(\mathbf{q}_i^{in}(\hat{\mathbf{z}}_k^{(l)}), \mathbf{q}_i^{in}(\hat{\mathbf{z}}_k^{(m)})) + \sigma_{\nu i}^2 \delta_{l,m}$ for each pair $(l, m) \in \{1, \ldots, \hat{N}\}^2$ and $[\mathbf{\Sigma}_{\hat{\mathbf{Q}}_i, \mathbf{Q}_i}]_{lm} = k(\mathbf{q}_i^{in}(\hat{\mathbf{z}}_k^{(l)}), \mathbf{q}_i^{in}(\mathbf{z}_k^{(m)}))$ for each pair $(l, m) \in \{1, \ldots, N\} \times \{1, \ldots, \hat{N}\}$ [220].

The likelihood of $\hat{\mathbf{Q}}_i$ conditioned on $\mathbf{Q}_i$ is given by:

$$\hat{\mathbf{Q}}_i \sim \mathcal{N}(\boldsymbol{\mu}_{\hat{\mathbf{Q}}_i}|\mathbf{Q}_i, \boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_i}|\mathbf{Q}_i), \tag{10.16}$$

where $\boldsymbol{\mu}_{\hat{\mathbf{Q}}_i}|\mathbf{Q}_i := \boldsymbol{\Sigma}^{\top}_{\hat{\mathbf{Q}}_i,\mathbf{Q}_i}\boldsymbol{\Sigma}^{-1}_{\hat{\mathbf{Q}}_i}\mathbf{Q}_i$ and $\boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_i}|\mathbf{Q}_i = \boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_i} - \boldsymbol{\Sigma}^{\top}_{\hat{\mathbf{Q}}_i,\mathbf{Q}_i}\boldsymbol{\Sigma}^{-1}_{\mathbf{Q}_i}\boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_i,\mathbf{Q}_i}$.

So far the treatment of GPs has been relatively standard, however we are unable to observe $\mathbf{Q}_i$ and $\hat{\mathbf{Q}}_i$ directly. This problem is a common occurrence for latent state space models, for which MCMC sampling [97] or maximum a posteriori (MAP) [140] has been applied. In this paper we apply MAP to obtain the required vectors $\mathbf{Q}_i$, for which we require the following likelihood based on Equation 10.7 and Equation 10.4:

$$\mathbf{y}_{k+1} \sim \mathcal{N}(\mathbf{H}\mathbf{x}_{k+1}, \boldsymbol{\Sigma}_{\mathbf{v}} + \mathbf{H}\boldsymbol{\Sigma}_{\omega}\mathbf{H}^{\top}), \tag{10.17a}$$

where $\mathbf{x}_{k+1} = \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x}(t), \mathbf{u}_k, \mathbf{q}(\mathbf{x}(t), \mathbf{u}_k))dt + \mathbf{x}_k$ is dependent on the dynamics and crucially on the unknown function $\mathbf{q}(\cdot)$.

Let $\mathbf{Q}$, $\hat{\mathbf{Q}}$, and $\boldsymbol{\Psi}$ refer to the joint $\mathbf{Q}_i$, $\hat{\mathbf{Q}}_i$ and $\boldsymbol{\Psi}_i$ respectively, i.e.

$$\mathbf{Q} = [\mathbf{Q}_1, \ldots, \mathbf{Q}_{n_\mathrm{q}}], \quad \hat{\mathbf{Q}} = [\hat{\mathbf{Q}}_1, \ldots, \hat{\mathbf{Q}}_{n_\mathrm{q}}], \quad \boldsymbol{\Psi} = [\boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_{n_\mathrm{q}}]. \tag{10.18}$$

Based on the different likelihoods we can now write down the likelihood equation for the data:

$$\begin{aligned} p(\mathbf{Y}|\mathbf{Q}, \hat{\mathbf{Q}}, \boldsymbol{\Psi}, \mathbf{Z}) \propto \quad & p(\mathbf{Y}|\mathbf{Q}, \hat{\mathbf{Q}}, \mathbf{Z})p(\hat{\mathbf{Q}}|\mathbf{Q}, \boldsymbol{\Psi}, \mathbf{Z}) \\ & \times p(\mathbf{Q}|\boldsymbol{\Psi}, \mathbf{Z})p(\mathbf{Q})p(\hat{\mathbf{Q}})p(\boldsymbol{\Psi}), \end{aligned} \tag{10.19a}$$

where the different likelihoods are given as follows:

$$p(\mathbf{Y}|\hat{\mathbf{Q}}, \mathbf{Z}) = \prod_{j=1}^{N} \mathcal{N}(\mathbf{y}_{k+1}^{(j)}; \hat{\mathbf{F}}(\mathbf{x}_k^{(j)}, \mathbf{u}_k^{(j)}, \hat{\mathbf{Q}}), \boldsymbol{\Sigma}_{\mathbf{v}} + \mathbf{H}\boldsymbol{\Sigma}_{\omega}\mathbf{H}^{\top}), \tag{10.19b}$$

$$p(\hat{\mathbf{Q}}|\mathbf{Q}, \boldsymbol{\Psi}, \mathbf{Z}) = \prod_{i=1}^{n_\mathrm{q}} \mathcal{N}(\hat{\mathbf{Q}}_i; \boldsymbol{\mu}_{\hat{\mathbf{Q}}_i}|\mathbf{Q}_i, \boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_i}|\mathbf{Q}_i), \tag{10.19c}$$

$$p(\mathbf{Q}|\boldsymbol{\Psi}, \mathbf{Z}) = \prod_{i=1}^{n_\mathrm{q}} \mathcal{N}(\mathbf{Q}_i; \mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{Q}_i}), \tag{10.19d}$$

where $\hat{\mathbf{F}}(\mathbf{x}_k, \mathbf{u}_k, \hat{\mathbf{Q}})$ refers to a discretized state-space model, for which $\hat{\mathbf{Q}}$ represents the values of $\mathbf{q}(\cdot)$ at the discretizaton points. The likelihoods stated above can be understood as follows: $p(\mathbf{Y}|\mathbf{Q}, \hat{\mathbf{Q}}, \mathbf{Z})$ is the likelihood of the observed data given $\hat{\mathbf{Q}}, \mathbf{Z}$, $p(\hat{\mathbf{Q}}|\mathbf{Q}, \boldsymbol{\Psi}, \mathbf{Z})$ is the likelihood of $\hat{\mathbf{Q}}$ given $\mathbf{Q}, \boldsymbol{\Psi}, \mathbf{Z}$, and lastly $p(\mathbf{Q}|\boldsymbol{\Psi}, \mathbf{Z})$ refers to the likelihood of $\mathbf{Q}$ given $\boldsymbol{\Psi}, \mathbf{Z}$.

Example 10.3.1 (Example discretization for MAP)**.** $\hat{\mathbf{F}}(\mathbf{x}_k, \mathbf{u}_k, \hat{\mathbf{Q}})$ *can in general represent any valid discretization rule. Assume for example we apply the trapezium rule for discretization, then we obtain the following relation for the known input* $\mathbf{z}_k^{(j)} = (\mathbf{x}_k^{(j)}, \mathbf{u}_k^{(j)})$*:*

$$\mathbf{x}_{k+1}^{(j)} = \mathbf{x}_k^{(j)} + 0.5\delta_t \left(\mathbf{f}(\hat{\mathbf{x}}_1^{(j)}, \mathbf{u}_k^{(j)}, \hat{\mathbf{q}}_1^{(j)}) + \mathbf{f}(\hat{\mathbf{x}}_2^{(j)}, \mathbf{u}_k^{(j)}, \hat{\mathbf{q}}_2^{(j)}) \right) \tag{10.20}$$

where $\hat{\mathbf{x}}_i$ *and* $\hat{\mathbf{q}}_1^{(j)}$ *refer to the ith state and q-value of the discretization rule and* $\hat{\mathbf{F}}(\mathbf{x}_k, \mathbf{u}_k, \hat{\mathbf{Q}}) = \mathbf{x}_{k+1}^{(j)}$*. These states* $\hat{\mathbf{x}}_1^{(j)} = \mathbf{x}_k^{(j)}$ *and* $\hat{\mathbf{x}}_2^{(j)} = \mathbf{x}_{k+1}^{(j)}$*, however note in general that the initial- and end-point may not be part of the discretization points. Let the number of discretization points required per interval be given by* d_s*, such that for the trapezium rule above* $d_s = 2$*. The corresponding matrices required for the MAP likelihood are given by* $\hat{\mathbf{Q}} = [\hat{\mathbf{q}}_1^{(1)}, \ldots, \hat{\mathbf{q}}_{d_s}^{(1)}, \ldots, \hat{\mathbf{q}}_1^{(N)}, \ldots, \hat{\mathbf{q}}_{d_s}^{(N)}]^\mathsf{T} \in \mathbb{R}^{(d_s N) \times n_\mathbf{q}}$ *and* $\hat{\mathbf{Z}} = [\hat{\mathbf{z}}_1^{(1)}, \ldots, \hat{\mathbf{z}}_{d_s}^{(1)}, \ldots, \hat{\mathbf{z}}_1^{(N)}, \ldots, \hat{\mathbf{z}}_{d_s}^{(N)}]^\mathsf{T} \in \mathbb{R}^{(d_s N) \times n_\mathbf{q}}$*, where* $\hat{\mathbf{z}}_i^{(j)} = (\hat{\mathbf{x}}_i^{(j)}, \mathbf{u}_k^{(j)})$*. For implicit integration rules like the one above either a Newton solver needs to be employed or the unknown values* $\hat{\mathbf{x}}_2^{(j)}$ *are added to the optimization variables with Equation 10.20 as additional equality constraints for each training data-point.*

The remaining likelihoods $p(\mathbf{Q})$, $p(\hat{\mathbf{Q}})$, and $p(\mathbf{\Psi})$ are prior distributions of $\mathbf{Q}$, $\hat{\mathbf{Q}}$, and $\mathbf{\Psi}$ respectively. These are a helpful tool to avoid overfitting and can be used to easily integrate prior knowledge into the optimization problem, e.g. knowledge on the approximate magnitude of $\mathbf{Q}$. Refer to [140] for examples on how priors can be used to incorporate prior knowledge on latent variables, such as $\mathbf{Q}$.

The required values for $\mathbf{Q}$ and $\mathbf{\Psi}$ are then found by minimizing the negative log-likelihood of Equation 10.19a:

$$(\mathbf{Q}^*, \mathbf{\Psi}^*, \hat{\mathbf{Q}}^*) \in \underset{\mathbf{Q}, \mathbf{\Psi}, \hat{\mathbf{Q}}}{\operatorname{argmin}} \mathcal{L}(\mathbf{Q}, \mathbf{\Psi}) = -\log p(\mathbf{Y} | \mathbf{Q}, \hat{\mathbf{Q}}, \mathbf{\Psi}, \mathbf{Z}), \tag{10.21}$$

where $\mathbf{Q}^*$, $\mathbf{\Psi}^*$, $\hat{\mathbf{Q}}^*$ are the required maximum a posteriori (MAP) estimates.

In the following sections we assume that the GP has been fitted in this way such that we have MAP values $\mathbf{Q}^*$ and $\mathbf{\Psi}^*$. The predictive distribution of $\mathbf{q}(\cdot)$ at an arbitrary input $\mathbf{z} = (\mathbf{x}, \mathbf{u})$ given the dataset $\mathcal{D} = (\mathbf{Z}, \mathbf{Q}^*)$ is as follows:

$$\mathbf{q}(\mathbf{z}) | \mathcal{D} \sim \mathcal{N}(\boldsymbol{\mu}_q(\mathbf{z}; \mathcal{D}), \mathbf{\Sigma}_q(\mathbf{z}; \mathcal{D})) \tag{10.22a}$$

with

$$\boldsymbol{\mu}_q(\mathbf{z}; \mathcal{D}) = [\mathbf{k}_1^\mathsf{T} \mathbf{\Sigma}_{\mathbf{Q}_1}^{-1} \mathbf{Q}_1^*, \ldots, \mathbf{k}_{n_\mathbf{q}}^\mathsf{T} \mathbf{\Sigma}_{\mathbf{Q}_{n_\mathbf{q}}}^{-1}, \mathbf{Q}_{n_\mathbf{q}}^*]^\mathsf{T} \tag{10.22b}$$

$$\boldsymbol{\Sigma}_q(\mathbf{z};\mathcal{D}) = \tag{10.22c}$$
$$\operatorname{diag}\left(\zeta_1^{2*} + \sigma_{\nu 1}^{2*} - \mathbf{k}_1^{\mathsf{T}}\boldsymbol{\Sigma}_{\mathbf{Q}_1}^{-1}\mathbf{k}_1, \ldots, \zeta_{n_\mathrm{q}}^{2*} + \sigma_{\nu n_\mathrm{q}}^{2*} - \mathbf{k}_{n_\mathrm{q}}^{\mathsf{T}}\boldsymbol{\Sigma}_{\mathbf{Q}_{n_\mathrm{q}}}^{-1}\mathbf{k}_{n_\mathrm{q}}\right),$$

where $\mathbf{k}_i = [k_i(\mathbf{q}_i^{in}(\mathbf{z}), \mathbf{q}_i^{in}(\mathbf{z}_k^{(1)})), \ldots, k_i(\mathbf{q}_i^{in}(\mathbf{z}), \mathbf{q}_i^{in}(\mathbf{z}_k^{(N)}))]^{\mathsf{T}}$. In Figure 10.1 we illustrate a prior GP in the top graph and the posterior GP in the bottom graph.

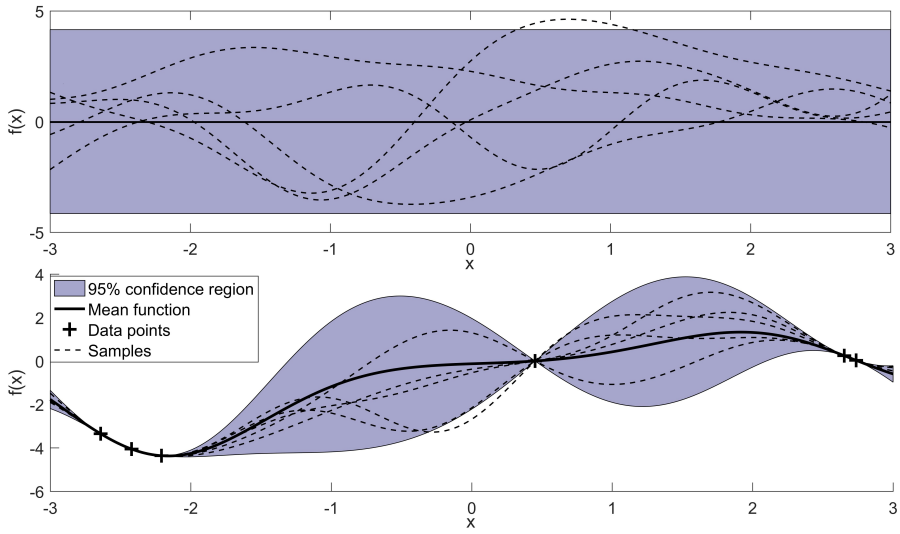


Figure 10.1: Illustration of a GP of a 1-dimensional function perturbed by noise. On the top the prior of the GP is shown, while on the bottom the Gaussian process was fitted to several data points to obtain the posterior.

10.3.2 Hybrid Gaussian process model predictive control formulation

In this section we define the NMPC optimal control problem (OCP) based on the GP hybrid *nominal* model fitted in the previous section, where the *nominal* model refers to the mean function in Equation 10.22. The initial state for the GP hybrid NMPC formulation is assumed to be measured or estimated, and propagated forward using Equation 10.1. The predicted states are exploited to optimize the objective subject to tightened constraints. Let the corresponding optimization problem be denoted as $P_T\left(\boldsymbol{\mu}_q(\cdot;\mathcal{D}); \mathbf{x}, k\right)$ for the current *known* state $\mathbf{x}$ at discrete

time k based on the mean function $\boldsymbol{\mu}_q(\cdot;\mathcal{D})$:

$$\begin{aligned}
&\underset{\hat{\mathbf{U}}_{k:T-1}}{\text{minimize}} \quad \hat{V}_T(\mathbf{x}, k, \hat{\mathbf{U}}_{k:T-1}) = \sum_{j=k+1}^{T-1} \ell(\hat{\mathbf{x}}_j, \hat{\mathbf{u}}_j) + \ell_f(\hat{\mathbf{x}}_T) \\
&\text{subject to:} \\
&\hat{\mathbf{x}}_{j+1} = \hat{\mathbf{F}}(\hat{\mathbf{x}}_j, \hat{\mathbf{u}}_j, \boldsymbol{\mu}_q(\hat{\mathbf{z}}(t);\mathcal{D})), \quad \hat{\mathbf{z}}(t) = (\hat{\mathbf{x}}(t), \hat{\mathbf{u}}_j) \quad \forall j \in \{k, \dots, T-1\} \\
&\hat{\mathbf{x}}_{j+1} \in \overline{\mathbb{X}}_{j+1}, \quad \hat{\mathbf{u}}_j \in \mathbb{U} \quad \forall j \in \{k, \dots, T-1\} \\
&\hat{\mathbf{x}}_k = \mathbf{x},
\end{aligned} \tag{10.23}$$

where $\hat{\mathbf{x}}$, $\hat{\mathbf{u}}$, and $\hat{V}_T(\cdot)$ refer to the states, control inputs, and control objective of the MPC formulation, $\hat{\mathbf{U}}_{k:T-1} = [\hat{\mathbf{u}}_k, \dots, \hat{\mathbf{u}}_{T-1}]^\mathsf{T}$, and $\overline{\mathbb{X}}_k$ is a tightened constraint set denoted by: $\overline{\mathbb{X}}_k = \{\mathbf{x} \in \mathbb{R}^{n_x} \mid g_i^{(k)}(\mathbf{x}) + b_i^{(k)} \leq 0,\ i = 1, \dots, n_g\}$. The variables $b_i^{(k)}$ represent so-called back-offs, which tighten the original constraints $\mathbb{X}_k$ defined in Equation 10.7.

Remark (Objective in expectation)**.** *Note the objective above in Equation 10.23 aims to determine the optimal trajectory for the nominal and not the expectation of the objective as defined in Equation 10.5, since it is computationally expensive to obtain the expectation of a nonlinear function [114]. Further, the difference between the expectation and the nominal system is commonly marginal.*

The NMPC algorithm solves $P_T\left(\boldsymbol{\mu}_q(\cdot;\mathcal{D}); \mathbf{x}_k, k\right)$ at each sampling time t_k given the current state $\mathbf{x}_k$ to obtain an optimal control sequence:

$$\hat{\mathbf{U}}^*_{k:T-1}\left(\boldsymbol{\mu}_q(\cdot;\mathcal{D}); \mathbf{x}_k, k\right) = [\hat{\mathbf{u}}^*_k\left(\boldsymbol{\mu}_q(\cdot;\mathcal{D}); \mathbf{x}_k, k\right), \dots, \hat{\mathbf{u}}^*_{T-1}\left(\boldsymbol{\mu}_q(\cdot;\mathcal{D}); \mathbf{x}_k, k\right)]^\mathsf{T}. \tag{10.24}$$

Only the first optimal control action is applied to the plant at time t_k before the same optimization problem is solved at time t_{k+1} with a new state measurement $\mathbf{x}_{k+1}$. This procedure implicitly defines the following feedback control law, which needs to be repeatedly solved for each new measurement $\mathbf{x}_k$:

$$\kappa(\boldsymbol{\mu}_q(\cdot;\mathcal{D}); \mathbf{x}_k, k) = \hat{\mathbf{u}}^*_k\left(\boldsymbol{\mu}_q(\cdot;\mathcal{D}); \mathbf{x}_k, k\right). \tag{10.25}$$

It is explicitly denoted that the control actions depend on the GP hybrid model used.

Remark (Full state feedback)**.** *Note in the control algorithm we have assumed full state feedback, i.e. it is assumed that the full state can be measured without noise. This assumption can be dropped if required by introducing a suitable observer and introduced in the closed-loop simulations to account for this additional uncertainty.*

10.3.3 Closed-loop Monte Carlo sample

In Equation 10.25 the control policy is stated, which is obtained by repeatedly solving the optimization problem in Equation 10.23 with updated initial conditions. GPs are distribution over functions and hence a GP sample describes a *deterministic* function. An example of this can be seen in Figure 10.1, in which several GP samples are depicted. In this section we outline how MC samples of GPs can be obtained for a *finite time horizon*, each describing separate state trajectories according to Equation 10.1. These are then exploited in the next section to tighten the constraints defined in the previous section. In general exact GP realizations cannot be obtained by any known approach, since generating such a sample would require sampling an infinite dimensional stochastic process. Instead, approximate approaches have been applied, such as spectral sampling [46]. Exact samples of GP are however possible if the GP only needs to be evaluated at a *finite number* of points. This is for example the case for discrete time GP state space models as proposed in [68, 249]. We next outline this technique and show how this can be extended to the continuous-time case for hybrid GP models, in which discretization is applied.

Assume we are given a state space model defined as in Section 10.2 in Equation 10.1, and a fitted GP model for $\mathbf{q}(\cdot)$ determined from Section 10.3.1. The predictive distribution based on available data $\mathcal{D} = (\mathbf{Z}, \mathbf{Y})$ is then given by Equation 10.22. The aim here is to show how to obtain a sample of the state sequence, which can be repeated multiple times to obtain multiple possible state sequences. The initial condition $\mathbf{x}_0$ follows a known Gaussian distribution as defined in Equation 10.1. Let the state sequence be given by:

$$\boldsymbol{\mathcal{X}}^{(s)} = [\boldsymbol{\chi}_0^{(s)}, \ldots, \boldsymbol{\chi}_T^{(s)}]^\mathsf{T}, \tag{10.26}$$

where $\boldsymbol{\chi}_k^{(s)}$ represents the state sequence of a GP realization s and $\boldsymbol{\chi}_k^{(s)}$ the state of this realization at time $t = t_k$.

Further, let the corresponding control actions at time $t = t_k$ be denoted by $u_k^{(s)}$. The control actions are assumed to be the result of the GP nominal NMPC feedback policy defined in Equation 10.25 and hence can be stated as:

$$u_k^{(s)} = \kappa(\boldsymbol{\mu}_q(\cdot; \mathcal{D}); \boldsymbol{\chi}_k^{(s)}, k). \tag{10.27}$$

We denote the control actions over the time horizon T jointly as:

$$\boldsymbol{\mathcal{U}}^{(s)} = [u_0^{(s)}, \ldots, u_{T-1}^{(s)}]^\mathsf{T} = [\kappa(\boldsymbol{\mu}_q(\cdot; \mathcal{D}); \boldsymbol{\chi}_0^{(s)}, 0), \ldots, \kappa(\boldsymbol{\mu}_q(\cdot; \mathcal{D}); \boldsymbol{\chi}_{T-1}^{(s)}, T-1)]^\mathsf{T}, \tag{10.28}$$

which are different for each MC sample s due to feedback.

To obtain a sample of a state sequence we first need to sample the initial state $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}_0}, \boldsymbol{\Sigma}_{\boldsymbol{x}_0})$ to attain the realization $\boldsymbol{\chi}_0^{(s)}$. Thereafter, the next state is given by Equation 10.2, which is dependent on the fitted GP of $\mathbf{q}(\cdot)$. An exact approach to obtain an independent sample of a GP is as follows. Any time the GP needs to be evaluated at a certain point, the response at this point $\mathbf{q}(\cdot)$ is sampled according to the predictive distribution in Equation 10.22. This sampled point is then part of the sampled function path, and hence the GP needs to be conditioned on it. This necessitates treating this point as a noiseless pseudo *training* point without changing the hyperparameters. Note if the sample path were to return to the same evaluation point, it would then lead to the same output due to this conditioning procedure. Consequently, the sampled function is deterministic as expected.

Furthermore, we also need to sample $\boldsymbol{\omega}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}})$ for each k. We refer to these realizations as $w_k^{(s)}$. We assume Equation 10.2 has been adequately discretized, such that the GP of $\mathbf{q}(\cdot)$ needs to be evaluated at only a finite number of points. The state sequence for realization s can then be given as follows:

$$\boldsymbol{\chi}_{k+1}^{(s)} = \hat{\mathbf{F}}(\boldsymbol{\chi}_k^{(s)}, u_k^{(s)}, \boldsymbol{Q}_k^{(s)}) + w_k^{(s)} \quad \forall k \in \{1, \dots, T\}, \tag{10.29}$$

where $\boldsymbol{Q}_k^{(s)}$ are discretization points sampled from the GP following the procedure outlined above and $\hat{\mathbf{F}}(\cdot)$ represents the discretized version of Equation 10.2.

Example 10.3.2. *We give an example here of the procedure above exploiting the trapezium rule for* $\hat{\mathbf{F}}(\boldsymbol{\chi}_k^{(s)}, u_k^{(s)}, \boldsymbol{Q}_k^{(s)})$. *Note the covariance matrix and dataset of the GPs are updated recursively. Assume we are at time k for MC sample s, and the covariance matrix of* $q_i(\cdot)$ *is given by* $\boldsymbol{\Sigma}_{\mathbf{Q}_{ik}}^{(s)}$ *with the updated data set* $\mathcal{D}_k^{(s)} = (\mathbf{Z}_k^{(s)}, \mathbf{Q}_k^{*(s)})$, *where* $\mathbf{Q}_k^{*(s)} = [\mathbf{Q}_{1k}^{*(s)}, \dots, \mathbf{Q}_{n_\mathbf{q}k}^{*(s)}]$ *and* $\mathbf{Z}_k^{(s)} = [\mathbf{z}^{(s1)}, \dots, \mathbf{z}^{(sN^k)}]^\mathsf{T}$ *as in Section 10.3.1. The dataset size* $N^k = N + (k-1) \times d_s$, *since at each time step k,* d_s *discretization points are added to the dataset.*

Let the number of discretization points per time interval be given by d_s, *for the trapezium rule* $d_s = 2$. *The discretization points then follow the distribution:*

$$\hat{\mathbf{Q}}_{ik}^{(s)} \in \mathbb{R}^{d_s} \sim \mathcal{N}(\boldsymbol{\mu}_{\hat{\mathbf{Q}}_{ik}^{(s)}|\mathbf{Q}_{ik}^{(s)}}, \boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_{ik}^{(s)}|\mathbf{Q}_{ik}^{(s)}}), \tag{10.30}$$

where $\boldsymbol{\mu}_{\hat{\mathbf{Q}}_{ik}^{(s)}|\mathbf{Q}_{ik}^{(s)}} := \boldsymbol{\Sigma}^\mathsf{T}_{\hat{\mathbf{Q}}_{ik}^{(s)}, \mathbf{Q}_{ik}^{(s)}} \boldsymbol{\Sigma}^{-1}_{\hat{\mathbf{Q}}_{ik}^{(s)}} \mathbf{Q}_{ik}^{(s)}$ *and* $\boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_{ik}^{(s)}|\mathbf{Q}_{ik}^{(s)}} = \boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_{ik}^{(s)}} - \boldsymbol{\Sigma}^\mathsf{T}_{\hat{\mathbf{Q}}_{ik}^{(s)}, \mathbf{Q}_{ik}^{(s)}} \boldsymbol{\Sigma}_{\mathbf{Q}_{ik}^{(s)}} \boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_{ik}^{(s)}, \mathbf{Q}_{ik}^{(s)}}$, $[\boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_{ik}^{(s)}}]_{lm} = k(\mathbf{q}_i^{in}(\hat{\mathbf{z}}_k^{(sl)}), \mathbf{q}_i^{in}(\hat{\mathbf{z}}_k^{(sm)}))$ *for each pair* $(l, m) \in \{1, \dots, d_s\}^2$ *and* $[\boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_{ik}^{(s)}, \mathbf{Q}_{ik}^{(s)}}]_{lm} = k(\mathbf{q}_i^{in}(\hat{\mathbf{z}}_k^{(sl)}), \mathbf{q}_i^{in}(\mathbf{z}_k^{(sm)}))$ *for each pair* $(l, m) \in \{1, \dots, N^k\} \times \{1, \dots, d_s\}$.

Firstly, we sample d_s independent standard normally distributed $\boldsymbol{\xi}_i \in \mathbb{R}^{n_\mathbf{q}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for each GP i. The sampled discretization points can then be expressed by:

$$\boldsymbol{Q}_{ik}^{(s)} = \boldsymbol{\mu}_{\hat{\mathbf{Q}}_{ik}^{(s)}|\mathbf{Q}_{ik}^{(s)}} + \boldsymbol{\xi}_i \cdot \boldsymbol{\Sigma}^{\frac{1}{2}}_{\hat{\mathbf{Q}}_{ik}^{(s)}|\mathbf{Q}_{ik}^{(s)}}, \tag{10.31}$$

where $\boldsymbol{Q}_k^{(s)} \in \mathbb{R}^{d_s} \sim \mathcal{N}(\boldsymbol{\mu}_{\hat{\mathbf{Q}}_{ik}^{(s)}|\mathbf{Q}_{ik}^{(s)}}, \boldsymbol{\Sigma}_{\hat{\mathbf{Q}}_{ik}^{(s)}|\mathbf{Q}_{ik}^{(s)}})$.

Once $\boldsymbol{Q}_k^{(s)}$ has been sampled we arrive at the next state $k+1$ for the MC sample as follows:

$$\boldsymbol{\chi}_{k+1}^{(s)} = \boldsymbol{\chi}_k^{(s)} + 0.5\delta_t \left(\mathbf{f}(\hat{\chi}_1^{(j)}, \mathbf{u}_k^{(j)}, \boldsymbol{Q}_{1k}^{(s)}) + \mathbf{f}(\hat{\chi}_2^{(j)}, \mathbf{u}_k^{(j)}, \boldsymbol{Q}_{2k}^{(s)}) \right), \tag{10.32}$$

where $\hat{\chi}_1^{(j)} = \boldsymbol{\chi}_k^{(s)}$ and $\hat{\chi}_2^{(j)} = \boldsymbol{\chi}_{k+1}^{(s)}$, since for the trapezium rule the discretization points coincide with the initial and the end-points, which is not true for other discretization rules. The value of the inputs for the discretization points are consequently given by $\hat{\mathbf{z}}_k^{(sl)} = (\hat{\chi}_l^{(s)}, u_k^{(s)})$. For implicit integration rules like the one above a Newton solver needs to be employed using Equations 10.31 and 10.32. Equation 10.31 is required due to the dependency of $\hat{\mathbf{z}}_k^{(sl)}$ on $\hat{\chi}_l^{(j)}$.

Lastly, the data matrices for the particular MC sample need to be updated as follows:

$$\mathbf{Z}_{k+1}^{(s)} = [\mathbf{Z}_k^{(s)}, \hat{\mathbf{z}}_k^{(s1)}, \ldots, \hat{\mathbf{z}}_k^{(sd_s)}]^\mathsf{T}, \tag{10.33a}$$

$$\mathbf{Q}_{k+1}^{*(s)} = [\mathbf{Q}_k^{*(s)\mathsf{T}}, \boldsymbol{Q}_k^{(s)\mathsf{T}}]^\mathsf{T}, \tag{10.33b}$$

$$[\boldsymbol{\Sigma}_{\mathbf{Q}_{ik}}]_{lm} = k_i(\mathbf{q}_i^{in}(\mathbf{z}_k^{(sl)}), \mathbf{q}_i^{in}(\mathbf{z}_k^{(sm)})) + \sigma_{vi}^2 \delta_{l,m} \quad \forall (l,m) \in \{1, \ldots, N^{k+1}\}^2. \tag{10.33c}$$

Repeating this procedure multiple times then gives us multiple MC samples of the state sequence $\boldsymbol{\mathcal{X}}^{(s)}$. The aim then is to use the information obtained from these sequences to iteratively tighten the constraints for the GP NMPC problem in Equation 10.23 to obtain the probabilistic constraint satisfaction required from the initial problem definition in Section 10.2.

10.3.4 Probabilistic constraint tightening

This section outlines how to systemically tighten the constraints based on MC samples using the procedure outlined in the previous chapter. Firstly define the function $C(\cdot)$, which is a single-variate random variable that represents the satisfaction of the joint chance constraints:

$$C(\mathbf{X}) = \inf_{(j,k) \in \{1,\ldots,n_g\} \times \{0,\ldots,T\}} g_j^{(k)}(\mathbf{x}_k), \tag{10.34a}$$

$$F_{C(\mathbf{X})} = \mathbb{P}\{C(\mathbf{X}) \leq 0\} = \mathbb{P}\left\{\bigcap_{k=0}^{T}\{\mathbf{x}_k \in \mathbb{X}_k\}\right\}, \tag{10.34b}$$

where $\mathbf{X} = [\mathbf{x}_0, \ldots, \mathbf{x}_T]^\mathsf{T}$ defines a state sequence, and $\mathbb{X}_k = \{\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}} \mid g_j^{(k)}(\mathbf{x}) \leq 0, j = 1, \ldots, n_g\}$.

The evaluation of the probability in Equation 10.34 is generally intractable, and instead a non-parametric sample approximation is applied, known as the *empirical cumulative distribution function* (ecdf). Assuming we are given S MC samples of the state trajectory $\mathbf{X}$ and hence of $C(\mathbf{X})$, the ecdf estimate of the probability in Equation 10.34 can be defined as follows:

$$F_{C(\mathbf{X})} \approx \hat{F}_{C(\mathbf{X})} = \frac{1}{S}\sum_{s=1}^{S} \mathbf{1}\{C(\boldsymbol{\mathcal{X}}^{(s)}) \leq 0\}, \tag{10.35}$$

where $\boldsymbol{\mathcal{X}}^{(s)}$ is the s-th MC sample and $\hat{F}_{C(\mathbf{X})}$ is the ecdf approximation of the true probability $F_{C(\mathbf{X})}$.

The accuracy of the ecdf in Equation 10.35 significantly depends on the number of samples used and it is therefore paramount to account for the residual uncertainty of this sample approximation. This problem has been previously studied in statistics, for which the following probabilistic lower bound has been proposed known as "*exact confidence bound*" [67]:

Theorem 10.3.1 (Confidence interval for empirical cumulative distribution function)**.** *Assume we are given a value of the ecdf, $\hat{\beta} = \hat{F}_{C(\mathbf{X})}$, as defined in Equation 10.35 based on S independent samples of $C(\mathbf{X})$, then the true value of the cdf, $\beta = F_{C(\mathbf{X})}$, as defined in Equation 10.34 has the following lower confidence bounds:*

$$\mathbb{P}\left\{\beta \geq \hat{\beta}_{lb}\right\} \geq 1 - \alpha, \qquad \hat{\beta}_{lb} = \text{betainv}\left(\alpha, S + 1 - S\hat{\beta}, S\hat{\beta}\right), \tag{10.36}$$

Proof. The proof uses standard results in statistics and can be found in [67, 242]. ∎

In other words the probability that the probability defined in Equation 10.34, β, exceeds $\hat{\beta}_{lb}$ is greater than $1 - \alpha$. Consequently, for small α $\hat{\beta}_{lb}$ can be seen as a conservative lower bound of the true probability β accounting for the statistical error introduced through the *finite* sample approximation. Based on the definition of $C(\mathbf{X})$ and the availability of S closed-loop MC simulations of the state sequence $\mathbf{X}$, assuming we are given a value for $\hat{\beta}_{lb}$ according to Equation 10.36 with a confidence level of $1 - \alpha$, then the following Corollary holds:

Corollary 10.3.1 (Feasibility probability)**.** *Assuming the stochastic system in Equation 10.1 is a correct description of the uncertainty of the system including the fitted GP and ignoring possible inaccuracies due to discretization errors, and given a value of the lower bound* $\hat{\beta}_{lb} \geq 1-\epsilon$ *defined in Equation 10.36 with a confidence level of* $1-\alpha$*, then the original chance constraint in Equation 10.7 holds true with a probability of at least* $1-\alpha$*.*

Proof. The realizations of possible state sequences described in Section 10.3.3 are exact within an arbitrary small discretization error and therefore these S independent state trajectories $\boldsymbol{\mathcal{X}}$ provide a valid lower bound $\hat{\beta}_{lb}$ from Equation 10.36 to the true cdf value β. If $\hat{\beta}_{lb}$ is greater than or equal to $1-\epsilon$, then the following probabilistic bound holds on the true cdf value β according to Theorem 10.3.1: $\mathbb{P}\left\{\beta \geq \hat{\beta}_{lb} \geq 1-\epsilon\right\} \geq 1-\alpha$, which in other words means that $\beta = \mathbb{P}\{C(\mathbf{X} \leq 0)\} \leq 1-\epsilon$ with a probability of at least $1-\alpha$. ∎

Now assume we want to determine back-off values for the nominal GP NMPC algorithm in Equation 10.23, such that β_{lb} is equal to $1-\epsilon$ for a chosen confidence level $1-\alpha$. This in turn guarantees the satisfaction of the original chance constraint with a probability of at least $1-\alpha$. The update rule to accomplish this has two steps: Firstly an approximate constraint set is defined and secondly this set is iteratively adjusted. The approximate constraint set should reflect the difference of the constraint values for the state sequence of the nominal MPC model and the constraint values of possible state sequence realizations of the *real* system in Equation 10.1. The back-offs are first set to zero and S MC samples are run according to Section 10.3.3. Now assume we aim to obtain back-off values that imply satisfaction of individual chance constraints as follows to attain an approximate initial constraint set:

$$g_j^{(k)}(\overline{\boldsymbol{\chi}}_k) + b_j^{(k)} = 0 \implies \mathbb{P}\left\{g_j^{(k)}(\boldsymbol{\chi}_k) \leq 0\right\} \geq 1-\delta, \tag{10.37}$$

where δ is a tuning parameter and should be set to a reasonably low value and $\overline{\boldsymbol{\chi}}_k$ refers to states according to the *nominal* trajectory as defined in Section 10.3.3.

It is proposed in [206] to exploit the inverse ecdf to fulfill the requirement given in Equation (10.37) using the S MC samples available. The back-offs can then be stated as:

$$\tilde{b}_j^{(k)} = \hat{F}^{-1}_{g_j^{(k)}}(1-\delta) - g_j^{(k)}(\overline{\boldsymbol{\chi}}_k) \quad \forall (j,k) \in \{1,\ldots,n_g^{(k)}\} \times \{1,\ldots,T\}, \tag{10.38}$$

where $\hat{F}^{-1}_{g_j^{(t)}}$ denotes the inverse of the ecdf given in Equation 10.35 and $\tilde{b}_j^{(t)}$ refers to these initial back-off values. The inverse of an ecdf can be determined by the

quantile values of the S constraint values from the MC samples with cut-off probability $1 - \delta$.

This first step gives us an initial constraint set that depends on the difference between the nominal prediction $\overline{\boldsymbol{\chi}}_k$ as used in the MPC and possible state sequences according to the MC simulations. The parameter δ in this case is only a tuning parameter to obtain the initial back-off values.

In the next step these back-off values are modified using a *back-off factor* γ:

$$b_j^{(k)} = \gamma \tilde{b}_j^{(k)} \quad \forall (j,k) \in \{1,\ldots,n_g^{(k)}\} \times \{1,\ldots,T\}. \tag{10.39}$$

A value of γ is sought for which the lower bound β_{lb} is equal to $1 - \epsilon$ to obtain the required chance constraint satisfaction in Equation 10.7, which can be formulated as a root finding problem:

$$h(\gamma) = \hat{\beta}_{lb}(\gamma) - (1 - \epsilon), \tag{10.40}$$

where the aim is to determine a value of γ, such that $h(\gamma)$ is approximately zero. $\hat{\beta}_{lb}(\gamma)$ refers to the implicit dependence of $\hat{\beta}_{lb}$ on the S MC simulations resulting from the tightened constraints of the nominal GP NMPC algorithm according to Equation 10.39.

In other words the back-off values of the NMPC are adjusted until they return the required chance constraint satisfaction in Equation 10.7. To drive $h(\gamma)$ to zero we employ the *bisection technique* [20], which seeks the root of a function in an interval a_γ and b_γ, such that $h(a_\gamma)$ and $h(b_\gamma)$ have opposite signs. It is expected that a too high value of the back-off factor leads to a highly conservative solution with a positive sign of $h(\cdot)$, while a low value of the back-off factor often results in negative values of $h(\cdot)$. In our algorithm the initial a_γ is set to zero to evaluate $\tilde{b}_j^{(k)}$ in the first step. The *bisection method* repeatedly bisects the interval, in which the root is contained. The output of the algorithm are the required back-offs in n_b back-off iterations. The overall procedure to attain the back-offs is given in

Algorithm 10.1.

Algorithm 10.1: Back-off iterative updates

Input : $\boldsymbol{\mu}_{\mathbf{x}_0}$, $\boldsymbol{\Sigma}_{\mathbf{x}_0}$, $\boldsymbol{\mu}_q(\mathbf{z};\mathcal{D})$, $\boldsymbol{\Sigma}_q(\mathbf{z};\mathcal{D})$, $\mathcal{D}$, T, $V_T(\mathbf{x},k,\hat{\mathbf{U}}_{k:T-1})$, $\mathbb{X}_k$, $\mathbb{U}_k$, ϵ, α, δ, learning, S, n_b

Initialize: Set all $b_j^{(k)} = 0$ and δ to some reasonable value, set $a_\gamma = 0$ and b_γ to some reasonably high value, such that $b_\gamma - (1-\epsilon)$ has a positive sign.

for n_b *back-off iterations* **do**

  if $n_b > 0$ **then**

    $c_\gamma := (a_\gamma + b_\gamma)/2$

    $b_j^{(t)} := c_\gamma \tilde{b}_j^{(t)} \quad (j,t) \in \{1,\ldots,n_g^{(t)}\} \times \{1,\ldots,T\}$

  Define GP NMPC in Equation 10.23 with back-offs $b_j^{(t)}$

  Run S MC simulations to obtain $\boldsymbol{\mathcal{X}}^{(s)}$ using the GP NMPC policy with updated back-offs

  $\hat{\beta} := \hat{F}_{C(\boldsymbol{\mathcal{X}}^{(s)})} = \frac{1}{S}\sum_{s=1}^{S} \mathbf{1}\{C(\boldsymbol{\mathcal{X}}^{(s)}) \leq 0\}$

  $\hat{\beta}_{lb} := \text{betainv}\left(\alpha, S+1-S\hat{\beta}, S\hat{\beta}\right)$

  if $nb = 0$ **then**

    $\tilde{b}_j^{(t)} = \hat{F}^{-1}_{g_j^{(t)}}(\delta) - g_j^{(t)}(\overline{\boldsymbol{\chi}}_t) \; \forall (j,t) \in \{1,\ldots,n_g^{(t)}\} \times \{1,\ldots,T\}$

    $\hat{\beta}_{lb}^{a_\gamma} := \hat{\beta}_{lb} - (1-\epsilon)$

  else

    $\hat{\beta}_{lb}^{c_\gamma} := \hat{\beta}_{lb} - (1-\epsilon)$

    if $sign(\hat{\beta}_{lb}^{c_\gamma}) = sign(\hat{\beta}_{lb}^{a_\gamma})$ **then**

      $a_\gamma := c_\gamma$

      $\hat{\beta}_{lb}^{a_\gamma} := \hat{\beta}_{lb}^{c_\gamma}$

    else

      $b_\gamma := c_\gamma$

Output : $b_j^{(t)} \quad \forall (j,t) \in \{1,\ldots,n_g^{(t)}\} \times \{1,\ldots,T\}, \hat{\beta}_{lb}$

10.3.5 Algorithm

A summary of the overall algorithm proposed in this paper is given in this section. As first step the problem needs to be specified following the problem definition in Section 10.2. From the available data the GP hybrid model needs to be trained as outlined in Section 10.3.1. Thereafter, the back-offs are determined *offline* iteratively following Algorithm 10.1. These back-offs then define the tightened constraint set for the GP NMPC feedback policy, which is run *online* to solve

the problem initially outlined. An overall summary can be found in Algorithm 10.2.

Algorithm 10.2: Back-off GP NMPC

Offline Computations

1. Build GP hybrid model from data-set $\mathcal{D} = (\mathbf{Z}, \mathbf{Y})$ as shown in Section 10.3.1.
2. Choose time horizon T, initial condition mean $\boldsymbol{\mu}_{\mathbf{x}_0}$ and covariance $\boldsymbol{\Sigma}_{\mathbf{x}_0}$, measurement covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\nu}}$, disturbance covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\omega}}$, stage costs ℓ and ℓ_f, constraint sets $\mathbb{X}_k, \mathbb{U}_k \; \forall k \in \{1, \ldots, T\}$, chance constraint probability ϵ, ecdf confidence α, tuning parameter δ, the number of back-off iterations n_b, and the number of Monte Carlo simulations S to estimate the back-offs.
3. Determine explicit back-off constraints using Algorithm 10.1.
4. Check final probabilistic value $\hat{\beta}_{lb}$ from Algorithm 10.1 if it is close enough to ϵ.

Online Computations

for $k = 0, \ldots, T-1$ **do**

1. Solve the MPC problem in Equation 10.23 with the tightened constraint set from the *Offline Computations*.
2. Apply the first control input of the optimal solution to the real plant.
3. Measure the current state $\mathbf{x}_k$.

10.4 Case study

The case study is based on a semi-batch reaction for the production of fatty acid methyl ester (FAME) from microalgae, which is considered a promising renewable feedstock to meet the growing global energy demand. FAME is the final product of this process, which can be employed as biodiesel [82]. We exploit a simplified dynamic model to verify the hybrid GP NMPC algorithm proposed in this paper. The GP NMPC has an economic objective, which is to maximize the FAME (biodiesel) concentration for the final batch product subject to two path constraints and one terminal constraint.

10.4.1 Semi-batch bioreactor model

The simplified dynamic system consists of four ODEs describing the evolution of the concentration of biomass, nitrate, nitrogen quota, and FAME. We assume a fixed volume fed-batch reactor. The balance equations can be stated as follows [82]:

$$\begin{aligned}
\frac{dC_X}{dt} &= 2\mu_m(I_0, C_X)\left(1 - \frac{k_q}{q}\right)\left(\frac{N}{N + K_N}\right)C_X - \mu_d C_X,\ C_X(0) = C_{X0} \\
\frac{dC_N}{dt} &= -\mu_N\left(\frac{C_N}{C_N + K_N}\right)C_X + F_N, \quad C_N(0) = C_{N0} \\
\frac{dq}{dt} &= \mu_N\left(\frac{C_N}{C_N + K_N}\right) - \mu_m(I_0, C_X)\left(1 - \frac{k_q}{q}\right)q, \quad q(0) = q_0 \\
\frac{dFA}{dt} &= \mu_m(I_0, C_X)(\theta' q - \epsilon' FA)\left(1 - \frac{k_q}{q}\right) \\
&\quad - \gamma'\mu_N\left(\frac{C_N}{C_N + K_N}\right)C_X, \quad FA(0) = FA_0,
\end{aligned} \tag{10.41}$$

where C_X is the concentration of biomass in gL^{-1}, C_N is the nitrate concentration in mgL^{-1}, q is the dimensionless intracellular nitrogen content (nitrogen quota), and FA is the concentration of FAME (biodiesel) in gL^{-1}. Control inputs are given by the incident light intensity (I_0) in $\mu\mathrm{mol.m}^{-2}.\mathrm{s}^{-1}$ and nitrate inflow rate (F_N) in $\mathrm{mg.L}^{-1}.\mathrm{h}^{-1}$. The state vector is hence given by $\mathbf{x} = [C_X, C_N, q, FA]^\mathsf{T}$ and the input vector by $\mathbf{u} = [I_0, F_N]^\mathsf{T}$. The corresponding initial state vector is given by $\mathbf{x}_0 = [C_{X0}, C_{N0}, q_0, FA_0]^\mathsf{T}$. The remaining parameters can be found in Table 10.1 taken in part from [82].

Table 10.1: Parameter values for ordinary differential equation system in Equation 10.41.

Parameter	Value	Units
μ_M	0.359	h^{-1}
μ_d	0.004	h^{-1}
k_q	1.963	$mg.g^{-1}$
μ_N	2.692	$mg.g^{-1}.h^{-1}$
K_N	0.8	$mg.L^{-1}$
k_s	91.2	$\mu mol.m^{-2}.s^{-1}$
k_i	100.0	$\mu mol.m^{-2}.s^{-1}$
α'	196.4	$L.mg^{-1}.m^{-1}$
θ'	6.691	-
γ'	7.53×10^3	-
ϵ'	0.01	-
τ'	1.376	-
δ'	9.904	-
ϕ'	16.89	-
β'	0.0	m^{-1}
L	0.0044	m

The function $\mu_m(I_0, C_X)$ describes the complex effects of light intensity on the biomass growth, which we assume to be unknown in this study. This helps simplify the model significantly, since these effects are dependent on the distance from the light source and hence would lead to a partial differential equation (PDE) model if modeled by first principles. The actual function can be given as follows to obtain values to train the hybrid GP:

$$\mu_m(I_0, C_X) = \frac{\mu_M}{L} \int_{z=0}^{L} \left(\frac{I(z, I_0, C_X)}{I(z, I_0, C_X) + k_s + \frac{I(z, I_0, C_X)^2}{k_i}} \right) dz, \tag{10.42}$$

where $I(z, I_0, C_X) = I_0 \exp\left(-(\alpha' C_X + \beta')z\right)$, z is the distance from the light source in m, and L is the reactor width in m.

10.4.2 Problem set-up

The problem has a time horizon $T = 12$ with a batch time of 480h, and hence a sampling time of 40h. Next we state the objective and constraint functions according to the general problem definition in Section 10.2 based on the dynamic system in Equation 10.41.

Measurement noise covariance matrix $\mathbf{\Sigma_\nu}$ and disturbance noise matrix $\mathbf{\Sigma_\omega}$ are

defined as:

$$\boldsymbol{\Sigma}_{\mathbf{v}} = 10^{-4} \times \operatorname{diag}\left(2.5^2, 800^2, 500^2, 3000^2\right), \tag{10.43a}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\omega}} = 10^{-4} \times \operatorname{diag}\left(0.1^2, 200^2, 10^2, 100^2\right). \tag{10.43b}$$

The mean and covariance of the initial condition are set to:

$$\boldsymbol{\mu}_{\mathbf{x}_0} = [0.4, 0, 150, 0]^{\mathsf{T}}, \boldsymbol{\Sigma}_{\mathbf{x}_0} = 10^{-3} \times \operatorname{diag}(0.2^2, 0, 100^2, 0). \tag{10.44}$$

The aim of the control problem is to maximize the amount of biodiesel in the final batch with a penalty on the change of control actions. The corresponding stage and terminal costs can be given as:

$$\ell(\mathbf{x}_t, \mathbf{u}_t) = \boldsymbol{\Delta}_{\mathbf{u}_t}^{\mathsf{T}} \mathbf{R} \boldsymbol{\Delta}_{\mathbf{u}_t}, \quad \ell_f(\mathbf{x}_T) = -FA_T. \tag{10.45}$$

where $\boldsymbol{\Delta}_{\mathbf{u}_t} = \mathbf{u}_t - \mathbf{u}_{t-1}$ and $\mathbf{R} = 5 \times 10^{-3} \times \operatorname{diag}(1/400^2, 1/40^2)$. The objective is then defined by Equation 10.5.

There are two path constraints. Firstly, the nitrate is constrained to be below 800mg/L. Secondly, the ratio of nitrogen quota q to biomass may not exceed 0.011 for high density biomass cultivation. These are then defined as:

$$g_1^{(t)} = C_{Nt} - 800 \leq 0 \qquad \forall t \in \{0, \ldots, T\}, \tag{10.46a}$$

$$g_2^{(t)} = q_t - 0.011 C_{Xt} \leq 0 \qquad \forall t \in \{0, \ldots, T\}. \tag{10.46b}$$

Further, the nitrate should reach a concentration below 150mg/L for the final batch. This constraint can be stated as:

$$g_3^{(T)}(\mathbf{x}_T) = C_{NT} - 200 \leq 0,\ g_3^{(t)}(\mathbf{x}_t) = 0\ \forall t \in \{0, \ldots, T-1\}. \tag{10.47}$$

The control inputs light intensity and nitrate inflow rate are subject to the following box constraints:

$$120 \leq I_t \leq 300 \qquad \forall t \in \{0, \ldots, T\}, \tag{10.48a}$$

$$0 \leq F_{Nt} \leq 10 \qquad \forall t \in \{0, \ldots, T\}. \tag{10.48b}$$

The priors were set to the following values:

$$p(\mathbf{Q}) = \mathcal{N}(-6 \times \mathbf{1}, 50 \times \mathbf{I}), \tag{10.49a}$$

$$p(\hat{\mathbf{Q}}) = \mathcal{N}(-6 \times \mathbf{1}, 50 \times \mathbf{I}), \tag{10.49b}$$

$$p(\mathbf{\Psi}) = \mathcal{N}([\mathbf{0}, 5 \times 10^{-3}]^{\mathsf{T}}, \mathrm{diag}(20 \times \mathbf{I}, 1 \times 10^{-6})). \tag{10.49c}$$

Maximum probability of violation was to $\epsilon = 0.1$. To compute the back-offs a total of $S = 1000$ MC iterations are employed for each iteration according to $\delta = 0.05$ and $\alpha = 0.01$. The number of back-off iterations was set to $n_b = 14$.

10.4.3 Implementation and initial dataset generation

The discretization rule used for the MAP fit, for the GP MC sample, and for the GP NMPC formulation exploits direct collocation with 4th order polynomials with the Radau collocation points. The MAP optimization problem and the GP NMPC optimization problem are solved using Casadi [9] to obtain the gradients of the problem using automatic differentiation in conjunction with IPOPT [258]. IDAS [116] is utilized to simulate the real plant. The input dataset $\mathbf{Z}$ was designed using the Sobol sequence [238] for the entire input data in the range $\mathbf{z}_i \in [0,3] \times [0,800] \times [0,600] \times [0,3500] \times [120,300] \times [0,10]$. The ranges were chosen for the data to cover the expected operating region. The outputs $\mathbf{Y}$ were then obtained from the IDAS simulation of the system perturbed by Gaussian noise as defined in the problem setup.

10.5 Results and discussions

Firstly, the accuracy of the proposed hybrid GP model is verified by creating 1000 random datapoints. For these we calculate the absolute prediction error and the absolute error over the standard deviation, which gives an indication on the accuracy of the uncertainty measure provided by the GP. These results are summarized in Figure 10.2. For comparison purposes three cases of the GP NMPC algorithm are compared. Firstly, we run the above case study using 30 datapoints and 50 datapoints. In addition, we compare this with the previously proposed GP NMPC algorithm in [42] that aims to model the dynamic state space equations using GPs using 50 datapoints. Lastly, these three cases are further compared to their *nominal* variations, i.e. setting all back-offs in the formulations to zero. The results of these runs are highlighted in Figures 10.3-10.8 and in Table 10.2. From these results we can draw the following conclusions:

- From Figure 2 we can see in the first graph that the median absolute error decreases significantly going from a dataset size of 30 to 50, which is as expected. Overall the hybrid model predictions seem reasonably well. The GP error measure can be tested by dividing the absolute error by the standard deviation, for which the vast majority of values should be within approximately a range of 0 to 3. A value above 3 has a chance of 0.6% of occurrence

according to the underlying Gaussian distribution. For $N = 30$ we observe no value above 3, while for $N = 50$ we observed 1.1% of values above 3. It can therefore be said that the error measure for $N = 30$ is more conservative, but both seem to show reasonable behavior.

- From Figures 10.3-10.5 it can be seen that the hybrid approaches both lead to generally good solutions, while the non-hybrid approach is unable to deal with the spread of the trajectories for constraint g_2. Further, it can be seen that the uncertainty of GP hybrid 50 is less than GP hybrid 30 from the significantly smaller spread of constraint g_2, which is as expected given the observations from Figure 10.2.

- Figure 10.6 illustrates the better performance of GP hybrid 50 over GP hybrid 30 obtaining a nearly 40% increase in the objective on average. This is due to two reasons: Firstly more data leads to better decisions on average and secondly lower uncertainty means that the GP hybrid 50 is less conservative than GP hybrid 30. Lastly, GP non-hybrid 50 achieves high objective values by violating the second constraint g_2 by a substantial amount.

- Figures 10.7 and 10.8 show that the *nominal* approach ignoring back-offs leads to constraint violations for all GP NMPC variations, while with back-offs the two hybrid approaches remain feasible throughout. GP non-hybrid 50 overshoots the constraint by a huge amount due to the NMPC becoming infeasible for the real plant. Overall, the importance of back-offs is shown to maintain feasibility given the presence of plant-model mismatch for both GP hybrid cases. However, for GP non-hybrid 50, the uncertainty is too large to attain a reasonable solution.

- In Table 10.2 the average computational times are between 78ms and 174ms. It can be seen that the GP hybrid approaches have higher computational times, which is due to the discretization required in the NMPC optimization problem. Overall the computational time of a single NMPC iteration is relatively low, while the offline computational times required to attain the back-offs is relatively high.

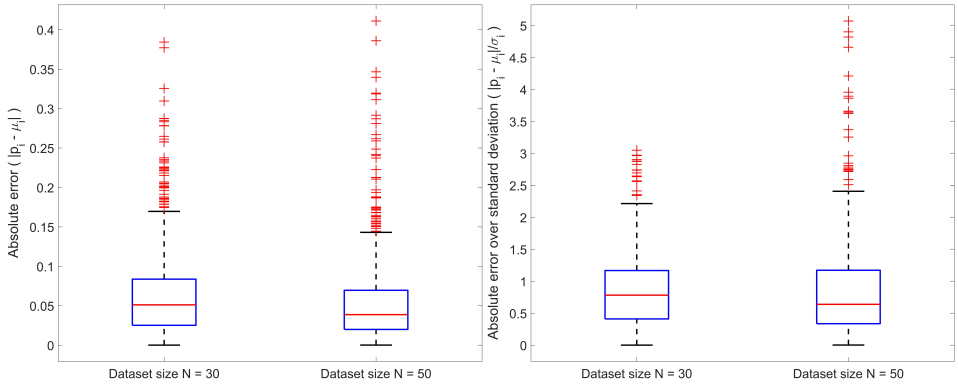


Figure 10.2: GP hybrid model cross-validation for dataset sizes $N = 30$ and $N = 50$ using 1000 randomly generated points in the same range as the training datapoints. The LHS graph shows the box plot of the absolute error, while the RHS graph shows the absolute error over the standard deviation.

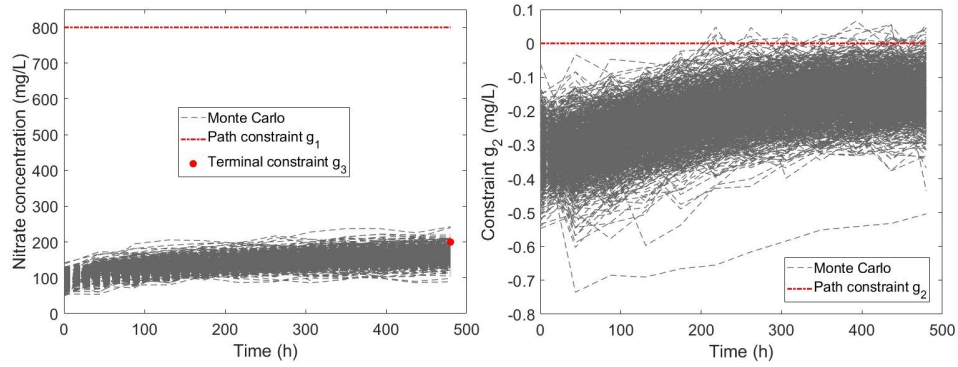


Figure 10.5: The 1000 MC trajectories at the final back-off iteration of the nitrate concentration for the constraints g_1 and g_2 (LHS) and the ratio of bioproduct to biomass constraint g_2 (RHS) for the non-hybrid GP with $N = 50$ modeling the entire state space model.

Table 10.2: Lower bound on the probability of satisfying the joint constraint $\hat{\beta}_{lb}$, average computational times to solve a single OCP for the GP NMPC, and the average computational time required to complete one back-off iteration.

Algorithm variation	Probability $\hat{\beta}_{lb}$	OCP time (ms)	Back-off iteration time (s)
GP hybrid 30	0.89	109	1316
GP hybrid 50	0.91	174	2087
GP non-hybrid 50	0.91	78	824

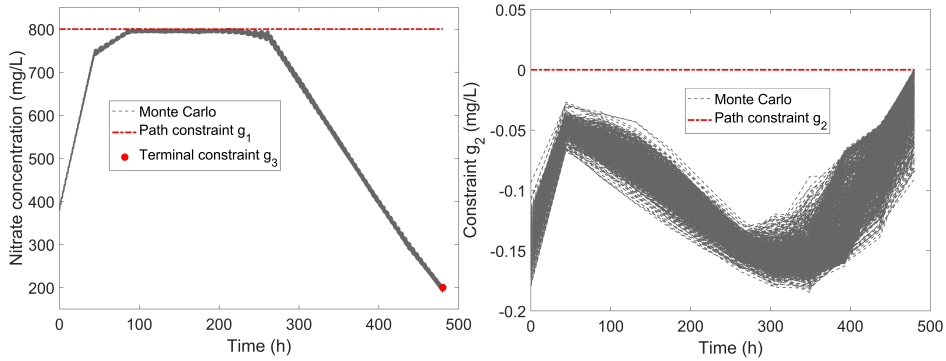


Figure 10.3: The 1000 MC trajectories at the final back-off iteration of the nitrate concentration for the constraints g_1 and g_2 (LHS) and the ratio of bioproduct to biomass constraint g_2 (RHS) for hybrid GP $N = 30$.

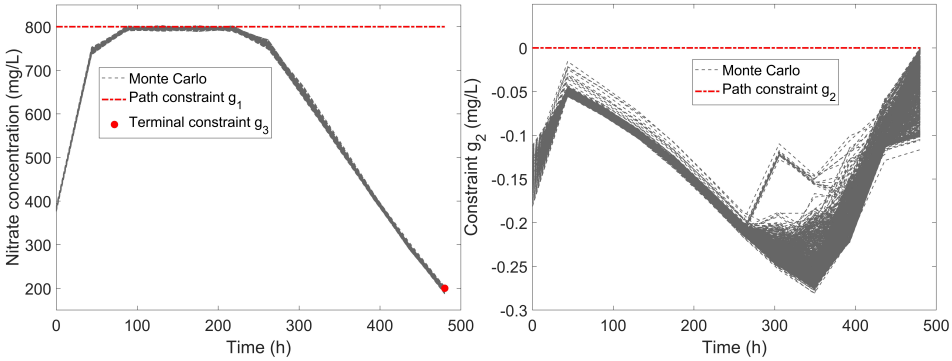


Figure 10.4: The 1000 MC trajectories at the final back-off iteration of the nitrate concentration for the constraints g_1 and g_2 (LHS) and the ratio of bioproduct to biomass constraint g_2 (RHS) for hybrid GP $N = 50$.

10.6 Conclusions

In conclusion, a new approach is proposed to combine first principles derived models with GP regression for NMPC. In addition, it is shown how the probabilistic nature of the GPs can be exploited to sample functions of possible dynamic models. These in turn are used to determine explicit back-offs, such that closed-loop simulations of the sampled models remain feasible to a high probability. It is shown how probabilistic guarantees can be obtained based on the number of constraint violations of the simulations. Online computational times are kept low by carrying out the constraint tightening offline. Lastly, a challenging semi-batch bioreactor case study demonstrates the efficiency and potential for this technique to operate complex dynamic systems.

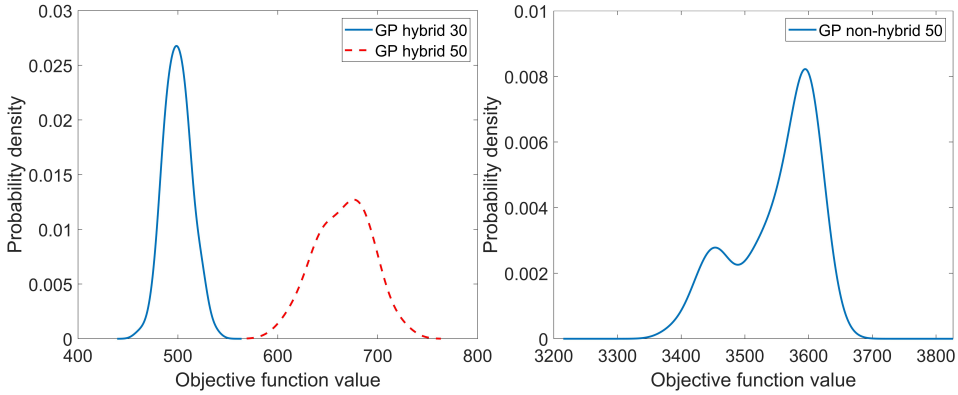


Figure 10.6: Probability density function for the "real" plant objective values for GP hybrid $N = 30$ and $N = 50$ on the LHS, and for the non-hybrid GP with $N = 50$ on the RHS.

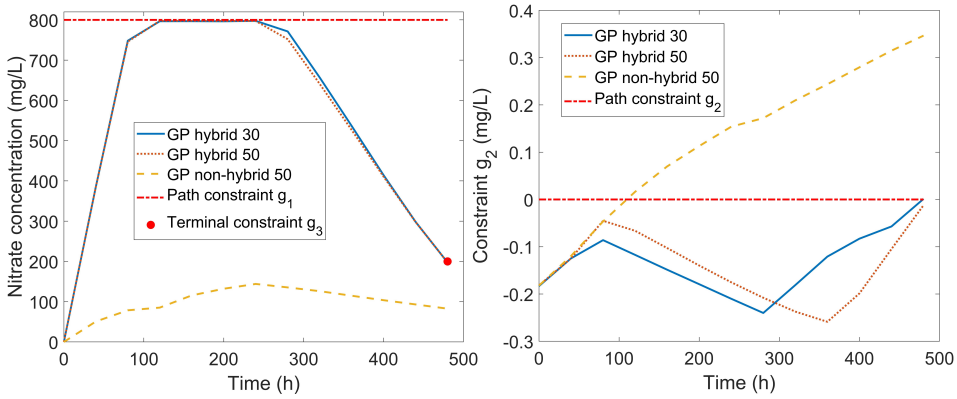


Figure 10.7: 90th percentile trajectory values of the nitrate concentration for constraints g_1 and g_3 (LHS) and the ratio of the bioproduct constraint g_2 (RHS) for all variations applied to the "real" plant with the final tightened constraint set.

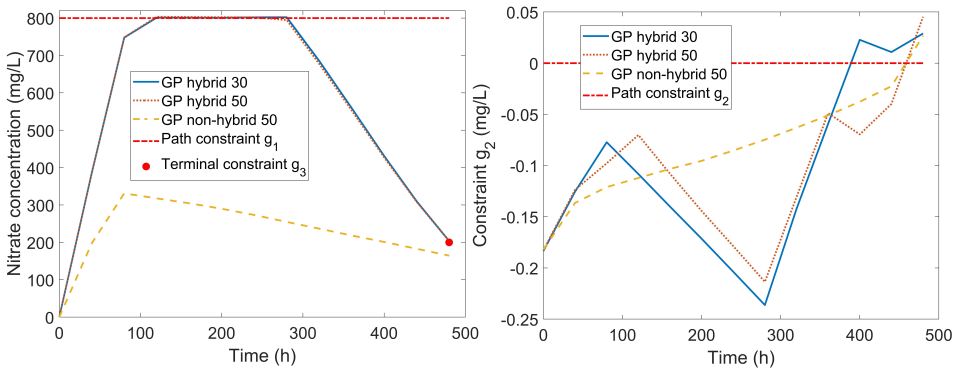


Figure 10.8: 90th percentile trajectory values of the nitrate concentration for constraints g_1 and g_3 (LHS) and the ratio of the bioproduct constraint g_2 (RHS) for all variations applied to the "real" plant with back-off values set.

Part IV

Conclusions and future work

Chapter 11

Concluding remarks

This thesis presents contributions on novel NMPC formulations that explicitly consider stochastic uncertainties for batch processes. The thesis was divided into two parts based on separate assumptions made on the uncertainty. In Part II the availability of a complete first principles model is assumed with uncertainties arising from parametric uncertainties, time varying disturbances, or state estimation errors. In Part III it is instead assumed that a complete first principles is not available and instead the full state space model or parts of it need to be identified from data directly. To accomplish this GP regression is exploited, which quantifies the residual nonparametric uncertainty given the available data. Comments on possible future work are given in Chapter 12.

In Part II firstly a simple algorithm is proposed in Chapter 3 using the Unscented transformation for both the SNMPC formulation and state estimation to account for stochastic uncertainties from Gaussian distributed state estimates, additive disturbances, and uncertain parameters. While the approach does work relatively well, it can be seen from Chapter 6 that the Unscented transformation may struggle significantly to capture the uncertainties of highly nonlinear problems due to its limited number of samples. In addition, it was assumed that at each sampling time the stochastic uncertainties and state estimates follow a Gaussian distribution. In Chapter 4 PCEs are employed instead for both state estimation and formulation of the SNMPC algorithm. In addition, the uncertainties in this approach are assumed to be represented by PCEs, which are able to capture more general probability distributions.

In Chapter 5 it is shown how GPs can be exploited as surrogates for uncertainty propagation similar to PCEs. An advantage of GPs over PCEs is that they not only

account for the uncertainty induced by the uncertain parameters, but also account for the uncertainty of the finite sample approximation of the surrogate itself. It is shown that GP surrogates are able to capture the probability distributions of the constraint and objective functions more accurately than either crude MC or Latin hypercube sampling. Given the overall excellent performance of PCEs and GPs, a novel uncertainty propagation algorithm is proposed in chapter 6 using a combination of both. It is highlighted that the approach is able to capture the shape of the probability density functions better than either GPs or PCEs on their own leading to improved mean and variance estimates. It is further shown to be superior to the Unscented transformation, which leads to relatively poor approximations.

All in all, in Part II it is clearly shown in each chapter that the presented SNMPC algorithms are able to adhere the constraints of semi-batch reactor case studies for a large number of scenarios, while in contrast a nominal NMPC algorithm violates in many scenarios the constraints substantially. This improved robustness however comes at a price of economic performance, such that for example in Chapter 4 batch times were on average 50% longer for the SNMPC algorithm than for the nominal NMPC to ensure constraint satisfaction. Furthermore, it is shown how PCEs can be utilized to represent a more general class of probability distributions and how these can be be updated efficiently using a PCE state estimator. A novel uncertainty propagation algorithm is introduced in Chapter 6 combining PCEs and GPs, which is thoroughly demonstrated to represent probability distributions more accurately than either PCEs or GPs on their own.

Initially in Chapter 7 in Part III we employ GPs to identify a dynamic model for a batch bioreactor from experimental input/output data and compare the results to artificial neural networks (ANNs). The GP is not only utilized to obtain predictions of unknown trajectories, but also the uncertainty measure is propagated to attain confidence intervals for these predictions. From the results it was determined that GPs are able to predict the trajectories of batch bioreactors well with comparable predictive capabilities as ANNs. Furthermore, GPs are able to provide confidence intervals that can be exploited to attain more robust solutions.

Next in Chapters 8 and 9 a novel algorithm is proposed for NMPC of batch reactors from input/output data using GPs for the identification of the required plant model and quantify the residual uncertainty of the plant-model mismatch. To consider this uncertainty Monte Carlo (MC) samples of the GP are generated offline, which in turn are utilized to tighten the constraints to guarantee the satisfaction of chance constraints online. It is shown that the approach is able to account for both online learning and the state dependency of the uncertainty to alleviate conservativeness. Furthermore, online computational times are kept relatively low.

Lastly, in Chapter 10 the approach is extended to hybrid modelling case. Often in chemical engineering a first principles is available, however parts of this model are difficult to derive using physical laws alone. It is therefore proposed to identify this part from input/output data using GPs using maximum a posteriori estimation. It is shown that not only is the identified GP able to give good predictions, but also the confidence interval from the uncertainty measure is reasonable.

In conclusion, in Part III the ability of GPs to dynamically model batch reactors is thoroughly demonstrated in Chapter 7. Subsequently, a new algorithm is proposed in Chapters 8 and 9 that tunes a GP NMPC algorithm by tightening constraints systematically using MC samples of the GP offline. The performance of the algorithm is extensively verified on semi-batch reactor case studies. In particular, it could be shown that accounting for online learning or the state dependency can alleviate conservativenss significantly. Further, it can be seen that it is vital to account for the uncertainty, since otherwise the nominal GP NMPC leads to significant constraint violations. Finally, it is shown in Chapter 10 how the approach can be extended to model only a part of a first principles model, which can be considerably more data efficient.

Chapter 12

Suggestions for future work

Given the results presented the following opportunities for future research are proposed.

12.1 Batch-to-batch learning

In this thesis there has been an emphasis on accounting for and learning the uncertainties for a single batch run using NMPC, however batch processes are characterized by repeating many batch runs. Furthermore, online measurements are often restrictive and therefore the quality of the batch becomes only apparent at the end of a batch run. It is therefore desirable to consider learning uncertainties from batch-to-batch and tuning the SNMPC algorithm accordingly to improve the quality of the batch or reducing the overall batch time. SNMPC presents a unique opportunity in this regard, which can exploit the stochastic information of the uncertainties to trade-off exploration and exploitation, i.e. trade-off running the batch process optimally given the uncertainties or run it sub-optimally to reduce the uncertainties to achieve better batch runs in the future [104].

Iterative learning control has been previously utilized to tune MPC controllers for trajectory tracking exploiting measurements of past batches [156]. While iterative learning control has been shown to perform well, it commonly does not actively learn the uncertainty. Alternatively, reinforcement learning (RL) may also be applied to iteratively update control policies from batch-to-batch [208], which inherently trades off exploration and exploitation. The GP approach in Chapters 8 -10 could be combined with a RL approach based on GPs to successively tune the NMPC algorithm from batch-to-batch [71], i.e. parameters defining the MPC could be adjusted between batches to improve a defined batch control quality cri-

terion [105]. For chapters 3-7 the mean and variance estimates can be used to explicitly trade off exploration and exploitation between batches, where the mean represents exploiting current knowledge, while the variance represents the inherent uncertainty. A similar procedure is utilized in Bayesian optimization [234].

12.2 Hybrid Gaussian process sampling

For the hybrid GP approach presented maximum a posteriori estimation was used to determine the required quantities, i.e. the hyperparameters and the latent function values. While this can work well, it relies heavily on the priors assumed and can lead to overfitting if these are chosen poorly. This is in particular a problem, since then the uncertainties are heavily underestimated, which can lead to large constraint violations. Instead, Markov chain Monte Carlo (MCMC) can be utilized to sample the required hyperparameters and latent function values, which should lead to a better representation of the uncertainty. An example on using MCMC to sample a GP state space model can be found in [97]. In addition, parametric uncertainties of the first principles based model could be considered for both building the hybrid GP and for the closed-loop simulations. These could again be sampled using the same MCMC algorithm.

12.3 Stochastic stability and recursive feasibility

In this thesis only finite horizon control problems have been considered due to the nature of batch processes. Most common control problems are however infinite horizon problems. An extension of the current work could be to add terminal constraint sets and terminal costs functions to obtain guarantees on stochastic asymptotic stability and recursive feasibility. Guaranteeing recursive feasibility of the MPC optimization problem in the case of noise with unbounded support is still open.

Some solution approaches for linear MPC to address recursive feasibility in literature are to use soft constraints, the use of an alternative and feasible optimization problem in case of infeasibility [55], or guarantee recursive feasibility only up to a certain probability [202]. For bounded noise on the other hand a mixture of probabilistic and worst-case constraint tightening can be applied to ensure feasibility [149]. For stochastic MPC commonly mean square stability is proven, which ensures that the mean value of the state converges to zero. In the case of additive disturbances it can be shown that the mean of the state converges to a neighborhood of the steady state condition [161, 56], while for multiplicative the mean of the state can be shown to converge to zero [214, 58]. In general, mean square stability is also applicable to nonlinear systems. The main difficulty for nonlinear systems is the propagation of uncertainty sets and the definition of a worst-case to

establish recursive feasibility and stability. Most results in literature are based on either lipschitz continuity [159] or other bounds on the nonlinear function [255], which could be extended to the stochastic case.

12.4 Less conservative chance constraints

Commonly the use of SMPC over robust MPC is to alleviate uncertainty by allowing for some constraint violations as opposed to not allowing for any. Most approaches based on uncertainty propagation use the mean and variance to state the chance constraints using either Chebyshev's inequality or Chernoff's inequality as has been done in this work. These are however notoriously conservative and often lead to much too high constraint satisfaction probabilities, which is in conflict with the main motivation of applying SMPC in the first place [111]. Instead, the chance constraint may in general be represented by a radius r that needs to be chosen such that $\mu_g + r\sqrt{\sigma_g^2} \leq 0 \implies \mathbb{P}(g \leq 0) \geq 1 - \epsilon$, where μ_g and σ_g^2 are the mean and variance of the constraint g [207]. This could be for example achieved by adjusting the back-off factor r iteratively using closed-loop simulations as in Chapters 8 -10. Alternatively, a non-parametric approximation of the chance constraint could be used instead, such as kernel density estimations (kde) or the empirical cumulative distribution function (ecdf). Work on the use of kde to reformulate chance constraints can be found in [51], while work on the use of the ecdf for PCEs can be found in [242].

12.5 Validation using real experiments

Many chemical engineering systems have relatively large sampling times, which gives a unique opportunity for more advanced algorithms to be employed that have relatively long computational times. Algorithms should ideally be verified using closed-loop implementations.

Part V

Appendices

Appendix A

Expectation constrained stochastic nonlinear model predictive control of a batch bioreactor

This chapter is based on **Paper I**: E. Bradford and L. Imsland. Expectation constrained stochastic nonlinear model predictive control of a batch bioreactor. *Computer Aided Chemical Engineering*, 40:1621–1626, 2017.

Antonio Espuña, Moisès Graells and Luis Puigjaner (Editors), Proceedings of the 27th European Symposium on Computer Aided Process Engineering – ESCAPE 27
October 1st - 5th, 2017, Barcelona, Spain

Expectation constrained stochastic nonlinear model predictive control of a batch bioreactor

Eric Bradford[a*] and Lars Imsland[a]

[a]*Department of Engineering Cybernetics; NTNU; Trondheim, Norway*
eric.bradford@ntnu.no

Abstract

Nonlinear model predictive control is a popular control approach for highly nonlinear and unsteady state processes, which however can fail due to unaccounted uncertainties. This paper proposes to apply a sample-average approach to solve the general stochastic nonlinear model predictive control problem to handle probabilistic uncertainties. Each sample represents a nonlinear simulation, which is expensive. Therefore, variance reduction methods were systematically compared to lower the necessary number of samples. The method was shown to perform well on a semi-batch bioreactor case study compared to a nominal nonlinear model predictive controller. Expectation constraints were employed to deal with state constraints in this case study, which take into account both magnitude and probability of deviations.

Keywords: Randomized MPC, Stochastic programming, Monte Carlo method, Uncertainty, Variance reduction

1. Introduction

Model predictive control (MPC) refers to a class of control methods which explicitly employ a model to solve an open-loop optimal control problem (OCP) over a finite sequence of control actions at each sampling instance. Its main advantages are its ability to deal with constraints and strongly coupled, multi variable plants (Maciejowski, 2002). MPC based on linear models is relatively mature and well-established. Many systems, however, display strong nonlinear behaviour and have stringent performance demands, which necessitate the use of nonlinear model predictive control (NMPC) (Findeisen et al., 2003). In particular, batch processes require NMPC approaches because of unsteady state operation. NMPC further allows the direct optimization of economic criteria, which has attracted significant attention in recent years (Rawlings and Amrit, 2009). Nominal NMPC however is prone to errors due to unaccounted uncertainties. Consequently, there have been significant developments in robust NMPC (RNMPC) to handle uncertainties explicitly. RNMPC assumes uncertainties are deterministic and bounded. The main methods for RNMPC are min-max NMPC and tube-based NMPC. An alternative to RNMPC is stochastic NMPC (SNMPC) in which the uncertainties are given by known probability distributions. In this framework constraints are addressed in a probabilistic sense. Unlike RNMPC, in SNMPC the control of the objectives can be systematically traded-off with an admissible level of constraint violation (Mesbah, 2016). Important methods for SNMPC include polynomial chaos (Mesbah et al., 2014) and an approach based on Markov Chain Monte Carlo (MCMC) (Visintini et al., 2006). In this paper a tractable

approximation to SNMPC is proposed based on Monte Carlo (MC) simulations, known as "sample-average approximation (SAA)" in stochastic programming (Homem-de Mello and Bayraksan, 2014). Variance reduction (VR) techniques are compared and applied to reduce the sample size required. The performance of the MC SNMPC is illustrated on a semi-batch bioreactor case study and compared to the performance of a nominal NMPC.

2. Monte Carlo stochastic nonlinear model predictive control

The dynamics we consider are given by an uncertain, continuous-time nonlinear system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \xi) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ represents the system states, $\mathbf{u} \in \mathbb{R}^{n_u}$ the control inputs, $\xi \in \mathbb{R}^{n_\xi}$ is a random vector representing the uncertain system parameters, and $\mathbf{f}(\cdot)$ a measurable function representing the nonlinear system dynamics.

The random vector ξ consists of independently distributed random variables ξ_i with known probability density functions (pdfs) p_{ξ_i}. The finite horizon OCP for SNMPC can be stated as follows:

$$\begin{aligned}
\underset{\mathbf{u}(t)}{\text{minimize}} \quad & \mathbb{E}(J(\mathbf{x}, \mathbf{u}, \xi)) \\
\text{subject to} \quad & \\
& \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \xi) \\
& g_k(\mathbf{x}, \mathbf{u}) \leq 0 \quad \forall k \in \{0, \ldots, n_g\} \\
& \mathbb{E}(h_k(\mathbf{x}, \mathbf{u}, \xi)) \leq \varepsilon_k \quad \forall k \in \{0, \ldots, n_h\} \\
& \mathbf{u} \in \mathbb{U} \\
& \mathbf{x}(t_0) = \mathbf{x}_0 \\
& t \in [t_0, t_f]
\end{aligned} \tag{2}$$

where $\mathbb{E}(\cdot)$ is the expectation, $\mathbf{u}(t)$ denotes the control policy, g_k represents the hard inequality constraints, h_k denotes the probabilistic inequality constraints, $\mathbb{U} \subset \mathbb{R}^{n_u}$ denotes the set of feasible control inputs, $\mathbf{x}_0$ the initial conditions of the states $\mathbf{x}$, $J(\cdot)$ the probabilistic objective, ε_k allows deviation in expectation of h_k, t the time, t_0 the initial time, and t_f the final time. Chance-constraints are also covered by Eq.(2) by noting that $\mathbb{P}(h_k(\mathbf{x}, \mathbf{u}, \xi) \leq 0) := \mathbb{E}(\mathbb{I}\{h_k(\mathbf{x}, \mathbf{u}, \xi) \leq 0\})$, where $\mathbb{P}(\cdot)$ denotes probability and $\mathbb{I}\{\cdot\}$ represents the indicator function (Homem-de Mello and Bayraksan, 2014).

The resulting problem in Eq.(2) is intractable. Subsequently, we utilise the SAA approach to obtain an approximate solution. Consider a finite sequence of N independent and identically distributed realizations of ξ, given by $\Xi := \{\widetilde{\xi}^{(0)}, \widetilde{\xi}^{(1)}, \ldots, \widetilde{\xi}^{(N-1)}\}$. Then the problem in Eq.(2) is approximately given by Eq.(3). It has been shown by Bastin et al. (2006), that under some smoothness conditions the local optima in Eq.(3) converge almost surely to the local optima in Eq.(2) as $N \to \infty$. One of the main issues of using Eq.(3) is that each sample $\widetilde{\xi}^{(i)}$ depicts a separate nonlinear system simulation, which is expensive.

$$
\begin{aligned}
\underset{\mathbf{u}(t)}{\text{minimize}} \quad & \frac{1}{N}\sum_{i=0}^{N-1} J(\mathbf{x}^{(i)},\mathbf{u},\widetilde{\xi}^{(i)}) \\
\text{subject to} \quad & \\
& \dot{\mathbf{x}}^{(i)} = \mathbf{f}(\mathbf{x}^{(i)},\mathbf{u},\widetilde{\xi}^{(i)}) \quad \forall i \in \{0,\dots,N-1\} \\
& g_k(\mathbf{x}^{(i)},\mathbf{u}) \leq 0 \quad \forall (i,k) \in \{0,\dots,N-1\}\times\{0,\dots,n_g\} \\
& \frac{1}{N}\sum_{i=0}^{N-1} h_k(\mathbf{x}^{(i)},\mathbf{u},\widetilde{\xi}^{(i)}) \leq \varepsilon_k \quad \forall k \in \{0,\dots,n_h\} \\
& \mathbf{u} \in \mathbb{U} \\
& \mathbf{x}^{(i)}(t_0) = \mathbf{x}_0 \\
& t \in [t_0, t_f]
\end{aligned}
\tag{3}
$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^{n_x}$ corresponds to a separate state vector for each realization of the random variable $\widetilde{\xi}^{(i)} \in \Xi$.
The effectiveness of the samples can be notably improved by using VR methods compared to standard MC. Let $\hat{f}_N(\mathbf{x},\mathbf{u}) = \frac{1}{N}\sum_{i=0}^{N-1} F(\mathbf{x},\mathbf{u},\widetilde{\xi}^{(i)})$ denote the sample-average of either objective or expectation constraints, then the variance of $\hat{f}_N(\mathbf{x},\mathbf{u})$ for standard MC is equal to $\sigma_N^2(\mathbf{x},\mathbf{u}) := \mathrm{Var}(F(\mathbf{x},\mathbf{u},\xi))/N$, which is a measure of the expected error. VR techniques reduce the variance of $\hat{f}_N(\mathbf{x},\mathbf{u})$ without increasing the sample size N to aid convergence of the SAA problem. In this report we analysed several of these techniques: Antithetic Variates (AV), Latin Hypercube Sampling (LHS), and different Quasi-Monte Carlo (QMC) approaches. AV is a method which reduces variance by inducing negative correlations between samples $\widetilde{\xi}^{(i)}$, which is effective for monotone functions. LHS creates a partition of the sample space and fixes the number of samples on each component proportional to the probability of that component. QMC methods constitute a deterministic choice of points that are carefully chosen to obtain better uniformity than random sequences (Homem-de Mello and Bayraksan, 2014). Three different QMC approaches were included in the comparison: A rank-1 lattice rule (LR), the Sobol sequence (SOB), and the Halton (HAL) sequence (L'Ecuyer and Lemieux, 2005).

3. Semi-batch bioreactor case study

The case study presented here deals with a semi-batch bioreactor for the production of ethanol. The differential algebraic equation (DAE) system of index-1 can be found in Wang and Sheu (2000). The equations describe the evolution of 4 states x, p, s, and V, which represent the concentrations of cell mass, glucose, and ethanol, and the working volume of the fermenter respectively. The control input is the feeding rate F. In the usual notation we can write $\mathbf{x} = [x,s,p,V]^T$ and $u = F$. Several of the parameters in the original problem were assumed uncertain following a log-normal distribution, given by $\xi = [\mu_m, \nu_m, K_s, s_F, K_p, Y_{p/s}]^T$. The pdfs are summarised in Table 1. The remaining parameter values were kept at their nominal values in Wang and Sheu (2000). The OCP based on this DAE system and sample-averages of ξ is given below in Eq.(4):

$$
\begin{aligned}
\underset{u(t)}{\text{minimize}} \quad & \frac{1}{N}\sum_{i=0}^{N-1} -x_3^{(i)}(t_f)x_4^{(i)}(t_f) \\
\text{subject to} \quad & \\
& \dot{\mathbf{x}}^{(i)} = \mathbf{f}(\mathbf{x}^{(i)}, u, \widetilde{\boldsymbol{\xi}}^{(i)}) \quad \forall i \in \{0,\dots,N-1\} \\
& \frac{1}{2N}\sum_{i=0}^{N-1}\left(\sqrt{(x_2^{(i)}(t)-90)^2+\delta^2}+(x_2^{(i)}(t)-90)\right) \leq \varepsilon \\
& x_4^{(i)}(t_f) - 5 \leq 0 \quad \forall i \in \{0,\dots,N-1\} \\
& u(t) \in [0,1] \\
& \mathbf{x}(0) = \mathbf{x}_0 \\
& t \in [0, t_f]
\end{aligned} \tag{4}
$$

The objective is to maximize the average amount of ethanol at a fixed final time by varying the feed rate. Further, the glucose concentration is constrained to lie below $90g/l$ by the first inequality constraint, which is formulated as an expectation constraint using a smooth approximation for the max operator, $\max(x_2(t) - 90, 0)$. δ was set to 10^{-4}. This constraint requires the "average" deviation of the constraint to lie below ε. The second constraint ensures the volume does not exceed the capacity of $5l$. The final time t_f was set to 6h and the initial conditions $\mathbf{x}_0$ were set to $[0.33, 90, 0.4, 1.5]^T$. The OCP in Eq.(4) was solved by employing direct collocation with Radau quadrature rule and degree 4 polynomials, with 8 control intervals. The inequality constraints were enforced at each sampling point. The resulting NLP problem was solved utilising IPOPT (Wächter and Biegler, 2006) in conjunction with CasADi (Andersson, 2013) in Python. The computational work was carried out on a Dell XPS 15 notebook with a Quad-core 6th Generation Intel i-7 process with up to 3.5 GHZ and 16 GB RAM.

Table 1: pdfs of uncertain parameters, where $\log\mathcal{N}(\mu, \sigma^2)$ denotes the log-normal pdf with mean μ and variance σ^2 of the associated normal distribution

Parameter	pdfs(p_{ξ_i})	Units
μ_m	$\log\mathcal{N}\left(0.855, (0.02)^2\right)$	s^{-1}
ν_m	$\log\mathcal{N}\left(0.855, (0.02)^2\right)$	s^{-1}
K_s	$\log\mathcal{N}\left(0.848, (0.03)^2\right)$	$[\cdot]$
s_F	$\log\mathcal{N}\left(5.010, (0.06)^2\right)$	gl^{-1}
$Y_{p/s}$	$\log\mathcal{N}\left(-0.751, (0.01)^2\right)$	$[\cdot]$
K_p	$\log\mathcal{N}\left(3.330, (0.03)^2\right)$	$[\cdot]$

4. Results and discussion

The VR methods were compared by solving the OCP problem in Eq.(4) 200 times with consistent solver settings, which should converge to the same local solution. The performance of each VR technique is gauged by determining the variance of the objective values attained compared to standard MC. Since the QMC methods are deterministic, it is necessary to randomize these for the error analysis. This was achieved by using the Cranley-Patterson randomization scheme. The VR measure used in this report was σ_{MC}/σ_{VR}, where σ_{MC} is the sample variance of standard MC objective values and σ_{VR} the sample variance of the respective VR method (Koivu, 2005). The results are shown in Table 2. Overall it can be said that VR leads to large improvements over the MC approach lowering variance by factors of up to 20. It can be seen that LHS performs consistently well, while the factors of the QMC-based methods increase as the sample size grows. At a

sample size of $N = 256$ the randomized LR outperforms LHS. For SNMPC, sample sizes are generally below 100, such that it was decided to use LHS for SNMPC. In Figure 1 the difference between the objective values obtained from MC to LHS is highlighted.

Table 2: Variance reduction compared to MC over the 200 optimizations (σ_{MC}/σ_{VR})

Variance reduction	$N = 16$	$N = 32$	$N = 64$	$N = 128$	$N = 256$
Antithetic Variance	4.9	4.1	3.6	3.2	4.3
Latin Hypercube Sampling	8.6	9.6	10.8	16.0	19.4
Randomized Sobol	2.9	3.0	4.5	7.5	15.1
Randomized Halton	2.3	3.4	3.8	6.5	13.0
Randomized Lattice Rule	3.9	5.1	6.9	10.9	23.6

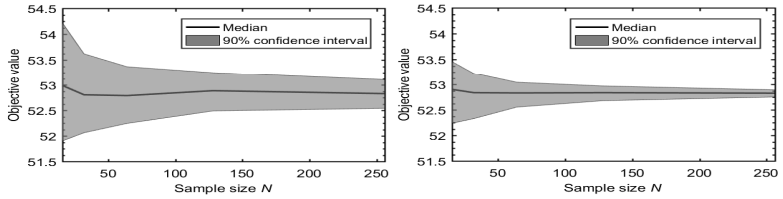


Figure 1: Comparison of objective values of 200 optimizations using MC (left) to LHS (right)

Next the OCP in Eq.(4) was solved using LHS in a receding horizon fashion for 12h simulations of the DAE system. The sample size N was set to 80. This approach was compared to a nominal NMPC implementation by running 200 separate simulations with 200 different parameter values sampled according to the pdfs in Table 1. The "real" DAE system was simulated utilising IDAS (Hindmarsh et al., 2005). The constraint on the glucose concentration shows the disparity of the approaches. In Figure 2 the comparison is shown between nominal NMPC and MC SNMPC with ε set to 0.2 and 0.1. As can be seen the method is able to reduce the deviations substantially compared to nominal NMPC. For $\varepsilon = 0.2$ there are still some rare but substantial deviations, while for $\varepsilon = 0.1$ deviations are rare and small, hence conservativeness can be adjusted by changing ε. Handling state constraints by expectation constraints as opposed to chance constraints is uncommon, however has the advantage that it takes into account both magnitude and probability of deviations. The average computational time required with standard deviations for solving one OCP over 200 separate simulations is shown in Table 3.

Table 3: Mean and standard deviation of OCP computational times

N	mean (s)	std (s)
$1(Nominal)$	0.08	0.01
10	0.28	0.14
20	0.94	0.65
40	6.53	9.70
80	9.17	8.49

5. Conclusion

Overall it has been shown how to obtain a tractable problem using sample-averages from a general SNMPC formulation, which is more efficient with a larger number of uncertainty parameters than polynomial chaos (Xiu and Karniadakis, 2002) and makes use of gradient information unlike the MCMC approach. The method was further improved by applying VR, which reduced the variance substantially. In particular, LHS was selected for the final SNMPC implementation due to its consistently good performance. The final SNMPC algorithm showed promising performance

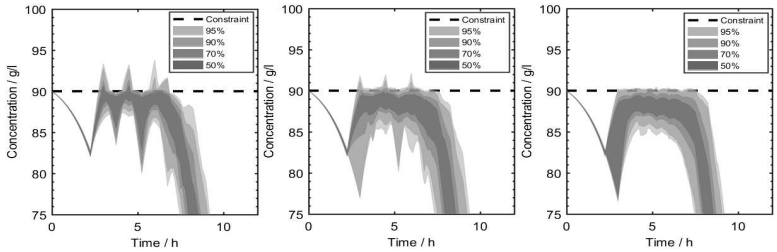


Figure 2: Comparison of glucose concentration trajectories for nominal NMPC (left), MC SNMPC with $\varepsilon = 0.2$ (centre) and MC SNMPC with $\varepsilon = 0.1$ (right)

on the bioreactor case study to handle state constraints using expectations compared to a nominal NMPC implementation. Further, it was shown that the conservativeness of the approach could be readily adjusted.

Acknowledgements:
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 675215.

References

Andersson, J., 2013. A general-purpose software framework for dynamic optimization. Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10.

Bastin, F., Cirillo, C., Toint, P. L., 2006. Convergence theory for nonconvex stochastic programming with an application to mixed logit. Mathematical Programming 108 (2), 207–234.

Findeisen, R., Imsland, L., Allgower, F., Foss, B. A., 2003. State and output feedback nonlinear model predictive control: An overview. European journal of control 9 (2-3), 190–206.

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., Woodward, C. S., 2005. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Transactions on Mathematical Software (TOMS) 31 (3), 363–396.

Homem-de Mello, T., Bayraksan, G., 2014. Monte Carlo sampling-based methods for stochastic optimization. Surveys in Operations Research and Management Science 19 (1), 56–85.

Koivu, M., 2005. Variance reduction in sample approximations of stochastic programs. Mathematical programming 103 (3), 463–485.

L'Ecuyer, P., Lemieux, C., 2005. Recent advances in randomized quasi-Monte Carlo methods. In: Modeling uncertainty. Springer, pp. 419–474.

Maciejowski, J. M., 2002. Predictive control: with constraints. Pearson education.

Mesbah, A., 2016. Stochastic model predictive control: An overview and perspectives for future research.

Mesbah, A., Streif, S., Findeisen, R., Braatz, R. D., 2014. Stochastic nonlinear model predictive control with probabilistic constraints. In: 2014 American Control Conference. IEEE, pp. 2413–2419.

Rawlings, J. B., Amrit, R., 2009. Optimizing process economic performance using model predictive control. In: Nonlinear model predictive control. Springer, pp. 119–138.

Visintini, A. L., Glover, W., Lygeros, J., Maciejowski, J., 2006. Monte Carlo optimization for conflict resolution in air traffic control. IEEE Transactions on Intelligent Transportation Systems 7 (4), 470–482.

Wächter, A., Biegler, L. T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming 106 (1), 25–57.

Wang, F.-S., Sheu, J.-W., 2000. Multiobjective parameter estimation problems of fermentation processes using a high ethanol tolerance yeast. Chemical Engineering Science 55 (18), 3685–3695.

Xiu, D., Karniadakis, G. E., 2002. The Wiener–Askey polynomial chaos for stochastic differential equations. SIAM journal on scientific computing 24 (2), 619–644.

Appendix B

Stochastic NMPC of Batch Processes Using Parametrized Control Policies

This chapter is based on **Paper J**: E. Bradford and L. Imsland. Stochastic NMPC of Batch Processes Using Parameterized Control Policies. *Computer Aided Chemical Engineering*, 44:625–630, 2018.

Mario R. Eden, Marianthi Ierapetritou and Gavin P. Towler (Editors) Proceedings of the 13th International Symposium on Process Systems Engineering – PSE 2018
July 1-5, 2018, San Diego, California, USA

Stochastic NMPC of Batch Processes Using Parametrized Control Policies

Eric Bradford[a*], Lars Imsland[a]

[a]*Department of Engineering Cybernetics, Norwegian University of Science and Technology, O. S. Bragstads plass 2D, 7034 Trondheim, Norway*
eric.bradford@ntnu.no

Abstract

Nonlinear model predictive control (NMPC) is an effective method for optimal operation of batch processes. Most dynamic models however contain significant uncertainties. It is therefore important to take these uncertainties into account in the formulation of the open-loop MPC problem to prevent infeasibilities or worse performance. An issue of such formulations is the disregard of feedback in the predictions, which leads to overly conservative control actions. The introduction of feedback through parametrized control policies is one way to solve this issue. In this work we compare the performance of affine feedback policies against more complex policies given by radial basis function networks. We incorporate these feedback policies into a polynomial chaos based stochastic NMPC algorithm to gauge their efficiency. The parameters of the feedback policies are either determined online by the NMPC algorithm or are pre-computed offline.

Keywords: Nonlinear model-based control, Polynomial chaos, Closed-loop policies, Chemical process control, Uncertain dynamic system

1. Introduction

Batch processes are used in many chemical sectors due to their inherent flexibility. These are operated at unsteady state and are often highly nonlinear, which motivates the application of nonlinear model predictive control (NMPC) (Nagy and Braatz, 2003). Many dynamic models however have limited accuracy due to various uncertainties. This can lead to constraint violations and worse control performance, which can be circumvented by incorporating these uncertainties in the NMPC algorithm (Mesbah, 2016). If we assume the uncertainties to be described by known probability density functions (pdfs), then the inclusion of the uncertainties in the NMPC algorithm leads to stochastic NMPC (SNMPC) formulations. In SNMPC constraints are addressed probabilistically, which allows for the systematic trade-off of constraint violation in probability with the conservativeness of the MPC solution (Mesbah, 2016). SNMPC methods include successive linearization (Cannon et al., 2009), sample-average NMPC (Bradford and Imsland, 2017), unscented sampling NMPC (Bradford and Imsland, 2017) and polynomial chaos expansion (PCE) NMPC (Fagiano and Khammash, 2012). A well-known problem of MPC under uncertainty is the fact that open-loop control actions are exceedingly conservative. To ensure reasonable predictions of the uncertainty, feedback needs to be considered. One way to achieve this is to optimize over parametrized feedback policies (Goulart et al., 2006). For linear stochastic MPC it is common to either evaluate a feedback matrix offline for pre-stabilization (Cannon et al., 2011) or by determining the parameters of the feedback control law online as

decision variables (Hokayem et al., 2012). For SNMPC this problem is often ignored as in the PCE based SNMPC paper by (Mesbah et al., 2014) or heuristics are applied as in Bradford and Imsland (2017). In this paper we focus on parametrized feedback policies for batch processes. For the SNMPC algorithm we use PCE due to its accuracy. We compare two different approaches to determine the parameters of the feedback policies: online optimization in the SNMPC algorithm or pre-computation offline. The first approach allows for revaluation of the feedback policy with knowledge of the new measurements, while the second allows for more complex feedback policies to be used. In addition, we consider three different parametrizations: affine time invariant, affine time varying and lastly radial basis function networks. In particular, radial basis function networks allow us to consider arbitrarily complex parametrizations. The various constellations were compared on a semi batch reactor case study via closed-loop simulations.

2. SNMPC with feedback policies

The aim of SNMPC is to control a discrete-time stochastic nonlinear system:

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

where k represents the discrete time, $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$ are the system states, $\mathbf{u} \in \mathbb{R}^{n_\mathbf{u}}$ are the control inputs, $\boldsymbol{\theta} \in \mathbb{R}^{n_\boldsymbol{\theta}}$ are uncertain parameters and $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta})$ denotes the system dynamics. We assume $\boldsymbol{\theta}$ to be independent Gaussian distributed random variables. The components have known mean μ_i and standard deviation σ_i. We can then formulate a general SNMPC problem with parametric feedback policies as follows:

$$\underset{\boldsymbol{u}_N}{\text{minimize}}\ \mathbb{E}\big(J(N, \mathbf{x}(\mathrm{n}), \boldsymbol{u}_N, \boldsymbol{\theta})\big)$$

subject to:

$$\begin{aligned}
&\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}) && \forall k \in \{0, \dots, N-1\} \\
&\mathbb{P}\big(g_j^{(k)}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}) \le 0\big) \ge 1 - p_j^{(k)} && \forall k \in \{1, \dots, N\} \times \left\{1, \dots, n_g^{(k)}\right\} \\
&\mathbf{u}_k = \boldsymbol{\varphi}\big(\boldsymbol{\nu}_k + \boldsymbol{\kappa}(\mathbf{x}_k, \boldsymbol{\gamma}_k)\big) && \forall k \in \{0, \dots, N-1\} \\
&\mathbf{x}_0 = \mathbf{x}(\mathrm{n})
\end{aligned} \tag{2}$$

where the objective function is the expectation of $J(N, \mathbf{x}(n), \boldsymbol{u}_N, \boldsymbol{\theta})$, $g_j^{(k)}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta})$ are individual chance constraints, $p_j^{(k)} \in (0,1) \subset \mathbb{R}$ is the probability of constraint violation, $\boldsymbol{u}_N = \{\boldsymbol{\nu}_0, \dots, \boldsymbol{\nu}_{N-1}, \boldsymbol{\gamma}_0, \dots, \boldsymbol{\gamma}_{N-1}\}$ is the set of decision variables, $\boldsymbol{\varphi}(\cdot)$ is a saturation function, $\boldsymbol{\kappa}(\cdot)$ is the feedback policy employed, $\boldsymbol{\nu}_k$ are the feed-forward control inputs, $\mathbf{x}(n)$ is the known initial state at time step n and N is the time horizon.

The saturation function in this report was defined to be individual sigmoid functions for each control input to introduce implicit constraints on $\mathbf{u}_k$: $\boldsymbol{\varphi}\big(\boldsymbol{\nu}_k + \boldsymbol{\kappa}(\mathbf{x}_k, \boldsymbol{\gamma}_k)\big) = \left[\frac{u_{max,1} - u_{min,1}}{1+\exp(-[\boldsymbol{\nu}_k + \boldsymbol{\kappa}(\mathbf{x}_k, \boldsymbol{\gamma}_k)]_1)} + u_{min,1}, \dots, \frac{u_{max,n_\mathbf{u}} - u_{min,n_\mathbf{u}}}{1+\exp(-[\boldsymbol{\nu}_k + \boldsymbol{\kappa}(\mathbf{x}_k, \boldsymbol{\gamma}_k)]_{n_\mathbf{u}})} + u_{min,n_\mathbf{u}}\right]^T$, i.e. $u_{min,i} \le u_{k,i} \le u_{max,i}\ \forall i \in \{1, \dots, n_\mathbf{u}\}$.

Three different parametrizations of feedback control laws were considered:

1) Affine time invariant (ATI)

$$\boldsymbol{\kappa}(\mathbf{x}_k, \boldsymbol{\gamma}) = \mathbf{K}\mathbf{x} \tag{3}$$

where $\mathbf{K} = \begin{bmatrix} \gamma_1 & \cdots & \gamma_{n_\mathbf{x}} \\ \vdots & \ddots & \vdots \\ \gamma_{n_\mathbf{u}} & \cdots & \gamma_{n_\mathbf{u} \times n_\mathbf{x}} \end{bmatrix}$, Number of parameters: $n_\mathbf{u} \times n_\mathbf{x}$

2) Affine time varying (ATV)

$$\boldsymbol{\kappa}(\mathbf{x}_k, \boldsymbol{\gamma}_k) = \mathbf{K}_k\mathbf{x} \tag{4}$$

where $\mathbf{K}_k = \begin{bmatrix} \gamma_{k,1} & \cdots & \gamma_{k,n_\mathbf{x}} \\ \vdots & \ddots & \vdots \\ \gamma_{k,n_\mathbf{u}} & \cdots & \gamma_{k,n_\mathbf{u} \times n_\mathbf{x}} \end{bmatrix}$, Number of parameters: $n_\mathbf{u} \times n_\mathbf{x} \times (n_k - 1)$

3) Radial basis function network time invariant (RBFN-n_{RBF})

$$\kappa_i(\mathbf{x}_k, \boldsymbol{\gamma}) = \sum_{j=1}^{n_{RBF}} \omega_j \phi_j(\mathbf{x}_k) \tag{5}$$

$$\phi_j(\mathbf{x}_k) = \exp\left(-0.5(\mathbf{x}_k - \boldsymbol{\mu}_j)^T \operatorname{diag}(\boldsymbol{\lambda}_j)(\mathbf{x}_k - \boldsymbol{\mu}_j)\right) \tag{6}$$

where n_{RBF} is the number of radial basis functions with parameters $[\omega_j, \boldsymbol{\mu}_j, \boldsymbol{\lambda}_j]^T = \boldsymbol{\gamma}_{(j-1)(1+2n_\mathbf{x})+1:j(1+2n_\mathbf{x})}$, Number of parameters: $n_\mathbf{u} \times n_{RBF}(1 + 2n_\mathbf{x})$

The open-loop case is given by $\boldsymbol{\kappa}(\cdot) = \mathbf{0}$. The nonlinear radial basis function network parametrization was taken from Deisenroth and Rasmussen (2011) and allows for arbitrarily complex feedback policies by adjusting the number of radial basis functions.

3. PCE based SNMPC

The problem defined in Eq.(2) is intractable, since it involves probability and expectations on nonlinear functions of $\boldsymbol{\theta}$. PCE can be used to solve this efficiently. Assume we are given a nonlinear transformation of independent standard normal variables $\zeta(\mathbf{z})$ with finite second order moments. According to PCE theory $\zeta(\mathbf{z})$ can be approximated by a truncated orthogonal polynomial basis (Owen et al., 2017):

$$\zeta(\mathbf{z}) \approx \sum_{0 \le |\boldsymbol{\alpha}| \le p} a_{\boldsymbol{\alpha}} He_{\boldsymbol{\alpha}}(\mathbf{z}) = \mathbf{a}^T \boldsymbol{\Phi}(\mathbf{z}) \tag{7}$$

where $\mathbf{z} \in \mathbb{R}^{n_\boldsymbol{\theta}}$ is a vector of standard normal variables with components $z_i \sim \mathcal{N}(0,1)$, $a_{\boldsymbol{\alpha}}$ are unknown expansion coefficients, $|\boldsymbol{\alpha}| = \sum_{j=1}^{n_\boldsymbol{\theta}} \alpha_j$, p is the order of truncation, $\mathbf{a}$ is a vector of coefficients $a_{\boldsymbol{\alpha}}$ and $\boldsymbol{\Phi}(\mathbf{z})$ contains $L = \frac{(n_\boldsymbol{\theta}+p)!}{n_\boldsymbol{\theta}! p!}$ polynomial elements $He_{\boldsymbol{\alpha}}(\mathbf{z})$ of the truncated expansion. $He_{\boldsymbol{\alpha}}(\mathbf{z})$ are known multivariate Hermite polynomials defined by a tensor product of univariate Hermite polynomials: $He_{\boldsymbol{\alpha}}(\mathbf{z}) = He_{\alpha_1}^{(1)}(z_1) \times \ldots \times He_{\alpha_{n_\boldsymbol{\theta}}}^{(n_\boldsymbol{\theta})}(z_{n_\boldsymbol{\theta}})$, where $He_j^{(i)}$ is the univariate Hermite polynomial of order j in terms of the i[th] component of $\mathbf{z}$. Univariate Hermite polynomial can be found in look-up tables.

We now need to find values for the coefficients $a_{\boldsymbol{\alpha}}$ to obtain a good approximation to $\zeta(\mathbf{z})$, which is done by using observations of $\zeta(\mathbf{z})$ at different values of $\mathbf{z}$. In this report we used a Sobol quasi random design to determine the sample points and transform it to follow a Gaussian distribution by using the inverse cumulative normal distribution function. We will refer to this design as $\mathcal{Z} = \{\mathbf{z}^{(0)}, \ldots, \mathbf{z}^{(n_s-1)}\}$, which contains n_s sampling points. We then obtain a response for each sample, which gives us $\mathbf{Y} = \left[\zeta(\mathbf{z}^{(0)}), \ldots, \zeta(\mathbf{z}^{(n_s-1)})\right]^T$. The coefficients can be found by least-squares estimation:

$$\hat{\mathbf{a}} = (\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\mathbf{Y} \tag{8}$$

where $\Psi_{ij} = He_{\boldsymbol{\alpha}_j}\left(\mathbf{z}^{(i)}\right)\ i = 0, \dots, n_s - 1, j = 1, \dots, L$ is a data matrix containing the polynomial terms in Eq.(7) and $He_{\boldsymbol{\alpha}_j}$ is the j[th] Hermite polynomial in the series expansion.

The variance and mean of $\zeta(\mathbf{z})$ are approximately given by (Mesbah et al., 2014):

$$\mathbb{E}\left(\zeta(\mathbf{z})\right) \approx \hat{a}_1, \quad \mathrm{Var}\left(\zeta(\mathbf{z})\right) \approx \sum_{j=2}^{L} \hat{a}_j^2 \mathbb{E}\left(He_{\boldsymbol{\alpha}_j}^{\ 2}(\mathbf{z})\right) \tag{9}$$

Using PCEs the problem in Eq.(2) can be simplified as follows:

$$\underset{\boldsymbol{u}_N}{\text{minimize}}\ \mathbb{E}\left(\xi_J\right)$$

subject to:

$$\mathbf{x}_{k+1}^{(i)} = \mathbf{f}\left(\mathbf{x}_k^{(i)}, \mathbf{u}_k, \boldsymbol{\theta}^{(i)}\right) \quad \forall (k,i)\ \in \{0, \dots, N-1\} \times \{0, \dots, n_s - 1\} \tag{10}$$

$$\sqrt{\mathrm{Var}\left(\zeta_{g_{jk}}\right)\left(1 - p_j^{(k)}\right)/p_j^{(k)}} + \mathbb{E}\left(\zeta_{g_{jk}}\right) \le 0 \quad \forall k\ \in \{1, \dots, N\} \times \left\{1, \dots, n_g^{(k)}\right\}$$

$$\hat{\mathbf{a}}_J = (\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\mathbf{Y}_J, \qquad \hat{\mathbf{a}}_{g_{jk}} = (\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T\mathbf{Y}_{g_{jk}}$$

$$\mathbb{E}\left(\xi_J\right) = \hat{a}_{J,1}, \qquad \mathbb{E}\left(\zeta_{g_{jk}}\right) = \hat{a}_{g_{jk},1}$$

$$\mathrm{Var}\left(\zeta_{g_{jk}}\right) = \sum_{m=2}^{L} \hat{a}_{g_{jk},m}^2 \mathbb{E}\left(He_{\boldsymbol{\alpha}_m}^{\ 2}(\mathbf{z})\right)$$

$$\mathbf{u}_k^{(i)} = \boldsymbol{\varphi}\left(\boldsymbol{\nu}_k + \boldsymbol{\kappa}\left(\mathbf{x}_k^{(i)}, \boldsymbol{\gamma}_k\right)\right) \qquad \forall k\ \in \{0, \dots, N-1\}$$

$$\mathbf{x}_0^{(i)} = \mathbf{x}(\mathrm{n}) \qquad \forall i\ \in \{0, \dots, n_s - 1\}$$

where $\theta_l^{(i)} = \sigma_l z_l^{(i)} + \mu_l$, $\mathbf{x}^{(i)}$ represents the state vector of scenario i with uncertain parameter $\boldsymbol{\theta}^{(i)}$, $\mathbf{Y}_J = \left[J\left(N, \mathbf{x}(\mathrm{n}), \boldsymbol{u}_N, \boldsymbol{\theta}^{(0)}\right), \dots, J\left(N, \mathbf{x}(\mathrm{n}), \boldsymbol{u}_N, \boldsymbol{\theta}^{(n_s-1)}\right)\right]^T$ and $\mathbf{Y}_{g_{jk}} = \left[g_j^{(k)}\left(\mathbf{x}_k^{(i)}, \mathbf{u}_k^{(i)}, \boldsymbol{\theta}^{(0)}\right), \dots, g_j^{(k)}\left(\mathbf{x}_k^{(i)}, \mathbf{u}_k^{(i)}, \boldsymbol{\theta}^{(n_s-1)}\right)\right]$ are data matrices to determine the required coefficients**.** It should be pointed out that while $\mathbf{Y}_J$ and $\mathbf{Y}_{g_{jk}}$ change each iteration of the optimization algorithm**,** the sample design $\mathcal{Z}$ remains constant, such that $(\boldsymbol{\Psi}^T\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^T$ and the $\mathbb{E}\left(He_{\boldsymbol{\alpha}_m}^{\ 2}(\mathbf{z})\right)$ can be calculated offline. The probability constraints were approximated robustly using Chebyshev inequality, for more information see (Mesbah et al., 2014).

4. Case study

The case study involves an isothermal semi batch reactor with a second-order exothermic reaction taking place; $\mathrm{A} + \mathrm{B} \rightarrow \mathrm{C}$. The exact differential equation for the case study can be found in Lucia and Paulen (2014). The control input is the feed rate u and the state vector is $\mathbf{x} = [C_A, C_B, C_C, V]^T$, where C_i are the concentrations of species i and V is the reactor volume. The flow rate u can be adjusted between 0 and 0.4. The initial conditions were set to $[C_{A0}, C_{B0}, C_{C0}, V_0] = [3,0,0,0.7]$ and the temperature is kept at 70°C. The semi batch reactor is controlled by a shrinking horizon implementation of Eq.(10). The objective of the SNMPC problem was set to maximize the average amount of C at the end of the batch, while keeping the adiabatic temperature below 85°C to

prevent uncontrollable behaviour and the concentration of B at the final time below $0.5\text{mol}\text{dm}^{-3}$. The probability of constraint violation for both was set to 0.05. The resulting OCP was solved employing direct collocation with 10 control intervals and a final time of 1h utilizing CasADi in Python (Andersson et al., 2012).
To gauge the performance of the various feedback policy methods, we ran 100 simulations based on a Sobol design for 5 different constellations with 4 computed online: Open-loop without feedback, ATI, ATV and RBFN with a single radial basis function referred to as RBFN-1. A more complex RBFN was computed offline with 10 radial basis functions referred to as RBFN-10. In Fig.1 a box-plot is shown of the attained amount of product C and hence the higher the amount of C, the better the closed-loop performance. In Tab.1 the corresponding average computational times are shown with the number of decision variables of the optimization problem. We can see that the open-loop approach has the lowest median and the fastest computational time, which is expected since it is the most conservative and has the least number of decision variables. ATI and RBFN-1 perform slightly better than open-loop implementation but, due their simplicity, they are worse than the others. In particular, RBFN-1 is a poor choice, since it leads to the same performance as ATI with larger computational times. ATV on the other hand has the highest median production rate, yet it requires also the highest computational time due to its large number of decision variables. RBFN-10 is only marginally worse, however has a computational time that is significantly lower than the other feedback policy approaches. It does however take twice as long as the open-loop approach.

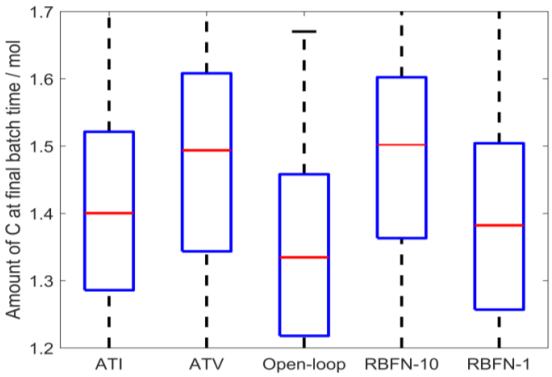


Figure 1 Box-plot of amount of product for 100 Sobol design points using different feedback policy approaches

Table 1 Comparison between number of decision variables evaluated online and average evaluation time of a single NMPC control step of the feedback policies

Feedback policy	**No. decision variables**	**Evaluation time (s)**
ATI	14	5.9
ATV	46	11.6
Open-loop	10	0.87
RBFN-10	10	1.87
RBFN-1	19	9.58

5. Conclusions

In conclusion, we investigated different approaches to introduce feedback into a SNMPC algorithm. In general, feedback always improves the performance of the otherwise open-loop algorithm. For the feedback policies evaluated online it could be shown that ATI and RBFN-1 were outperformed by the more complex ATV, however ATV carries the largest computational time due to the number of decision variables it

introduces. On the other hand, the pre-computed RBFN-10 managed to perform only marginally worse while being significantly faster in terms of computational time. We therefore conclude that the use of pre-computed complex feedback control parametrizations is an interesting approach to obtain good closed-loop performance without introducing a too heavy computational load.

6. Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska- Curie grant agreement No 675215.

References

J. Andersson, J. Åkesson, and M. Diehl, 2012, "CasADi: A symbolic package for automatic differentiation and optimal control, Recent advances in algorithmic differentiation", Lecture Notes in Computational Science and Engineering, Volume 87, p. 297-307.

E. Bradford, and L. Imsland, 2017, "Stochastic Nonlinear Model Predictive Control with State Estimation by Incorporation of the Unscented Kalman Filter".

E. Bradford, and L. Imsland, 2017, "Expectation constrained stochastic nonlinear model predictive control of a batch bioreactor", Computer Aided Chemical Engineering, Volume 40, p. 1621-1626.

M. Cannon, B. Kouvaritakis, S. V. Rakovic, and Q. Cheng, 2011, "Stochastic tubes in model predictive control with probabilistic constraints", IEEE Transactions on Automatic Control, Volume 56, Issue 1, p. 194-200.

M. Cannon, D. Ng, and B. Kouvaritakis, 2009, "Successive linearization NMPC for a class of stochastic nonlinear systems", Nonlinear Model Predictive Control, Lecture Notes in Control and Information Sciences, Volume 384, p. 249-262.

M. Deisenroth, and C. E. Rasmussen, 2011, "PILCO: A model-based and data-efficient approach to policy search", *in* Proceedings of the 28th International Conference on machine learning (ICML-11), p. 465-472.

L. Fagiano, and M. Khammash, 2012, "Nonlinear stochastic model predictive control via regularized polynomial chaos expansions", *in* Proceedings of the 51st IEEE Conference on Decision and Control (CDC), p. 142-147.

P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski, 2006, "Optimization over state feedback policies for robust control with constraints", Automatica, Volume 42, Issue 4, p. 523-533.

P. Hokayem, E. Cinquemani, D. Chatterjee, F. Ramponi, and J. Lygeros, 2012, "Stochastic receding horizon control with output feedback and bounded controls", Automatica, Volume 48, Issue 1, p. 77-88.

S. Lucia, and R. Paulen, 2014, "Robust nonlinear model predictive control with reduction of uncertainty via robust optimal experiment design", IFAC Proceedings Volumes, Volume 47, Issue 3, p. 1904-1909.

A. Mesbah, 2016, "Stochastic model predictive control: An overview and perspectives for future research", IEEE Control Systems, Volume 36, Issue 6, p. 30-44.

A. Mesbah, S. Streif, R. Findeisen, and R. D. Braatz, 2014, "Stochastic nonlinear model predictive control with probabilistic constraints", *in* Proceedings of the IEEE 2014 American Control Conference, p. 2413-2419.

Z. K. Nagy, and R. D. Braatz, 2003, "Robust nonlinear model predictive control of batch processes", AIChE Journal, Volume 49, Issue 7, p. 1776-1786.

N. Owen, P. Challenor, P. Menon, and S. Bennani, 2017, "Comparison of surrogate-based uncertainty quantification methods for computationally expensive simulators", SIAM/ASA Journal on Uncertainty Quantification, Volume 5, Issue 1, p. 403-435.

Appendix C

Stochastic nonlinear model predictive control of a batch fermentation process

This chapter is based on **Paper K**: E. Bradford and L. Imsland. Stochastic nonlinear model predictive control of a batch fermentation process. *Computer Aided Chemical Engineering*, 46:1237–1242, 2019.

Anton A. Kiss, Edwin Zondervan, Richard Lakerveld, Leyla Özkan (Eds.)
Proceedings of the 29th European Symposium on Computer Aided Process Engineering
June 16th to 19th, 2019, Eindhoven, The Netherlands.

Stochastic nonlinear model predictive control of a batch fermentation process

Eric Bradford[a*] and Lars Imsland[a]

[a]*Engineering Cybernetics; NTNU; O. S. Bragstads plass 2D, Trondheim 7034, Norway*
eric.bradford@ntnu.no

Abstract

Nonlinear model predictive control (NMPC) is an attractive control approach to regulate batch processes reliant on an accurate dynamic model. Most dynamic models however are affected by significant uncertainties, which may lead to worse control performance and infeasibilities, considering the tendency of NMPC to drive the system to its constraints. This paper proposes a novel NMPC framework to mitigate this issue by explicitly taking into account time-invariant stochastic uncertainties. Parametric uncertainties are assumed to be given by so-called polynomial chaos expansions (PCE), which constitutes a flexible approach to depict arbitrary probability distributions. It is assumed that at each sampling time only noisy output measurements are available. The proposed procedure uses a sparse Gauss-Hermite sampling rule to formulate an efficient scenario-based NMPC algorithm based on the PCE, while a stochastic nonlinear filter is employed to update the PCE given the available measurements. The framework is shown to be effective on a challenging semi-batch fermentation process simulation case study.

Keywords: Chemical process control, Polynomial chaos, Nonlinear filters, Model-based control

1. Introduction

Batch processes are commonly used in many chemical sectors, including pharmaceuticals, bulk chemicals and biotechnology. Batch processes are operated at unsteady state and are highly nonlinear, which motivates the use of nonlinear model predictive control (NMPC). The performance of the NMPC algorithm depends strongly on the accuracy of the dynamic model used and inherent uncertainties may lead to constraint violations and worse control actions. If we assume these uncertainties to be given by known probability distributions, stochastic NMPC (SNMPC) methods can be used (Mesbah, 2016). The main difficulty in SNMPC lies in propagating stochastic uncertainties through nonlinear system models. Several SNMPC algorithms have been proposed using different methods to propagate stochastic uncertainties:

- Unscented transformation sampling (Bradford and Imsland, 2018a)
- Polynomial chaos expansions (Fagiano and Khammash, 2012)
- Markov Chain Monte Carlo (Maciejowski et al., 2007)
- Gaussian processes (Bradford and Imsland, 2018b)
- Quasi Monte Carlo methods (Bradford and Imsland, 2017)
- Particle filters (Sehr and Bitmead, 2017)

Most work in SNMPC assumes full state feedback, which is uncommon for real processes. Instead, the measurements made at each sampling time are both noisy and incomplete. In this paper we therefore propose to use a nonlinear filter to update the stochastic uncertainties at each sampling time. The uncertainties are represented by so-called polynomial chaos expansions (PCE), which allow for complex probability distributions to be given by polynomials of simpler stochastic variables. In addition, we suggest to efficiently formulate the SNMPC problem using a sparse Gauss-Hermite quadrature rule. The framework is verified on a fermentation case study.

2. Problem formulation

We aim to control a discrete-time nonlinear equation system with stochastic uncertainties:

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}(\boldsymbol{\xi})), \qquad \mathbf{x}(0) = \mathbf{x}_0(\boldsymbol{\theta}(\boldsymbol{\xi})) \tag{1}$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \boldsymbol{\theta}(\boldsymbol{\xi})) + \boldsymbol{\nu} \tag{2}$$

where k is the discrete time, $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$ are the system states, $\mathbf{u} \in \mathbb{R}^{n_\mathbf{u}}$ denote the control inputs, $\boldsymbol{\theta}(\boldsymbol{\xi}) \in \mathbb{R}^{n_\boldsymbol{\theta}}$ are time-invariant uncertainties, $\boldsymbol{\xi} \in \mathbb{R}^{n_\boldsymbol{\theta}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ describe standard normally distributed random variables parametrizing the PCE of $\boldsymbol{\theta}$, $\mathbf{f} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_\boldsymbol{\theta}} \to \mathbb{R}^{n_\mathbf{x}}$ represents the nonlinear dynamic system, $\mathbf{y} \in \mathbb{R}^{n_\mathbf{y}}$ denote the measurements, $\mathbf{h} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\boldsymbol{\theta}}$ are the output equations and $\boldsymbol{\nu} \in \mathbb{R}^{n_\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\boldsymbol{\nu})$ is the measurement noise assumed to be zero mean multivariate normally distributed with known covariance matrix $\boldsymbol{\Sigma}_\boldsymbol{\nu}$. The initial condition $\mathbf{x}(0)$ may also be uncertain and hence is a function of $\boldsymbol{\theta}(\boldsymbol{\xi})$.

The time-invariant uncertainty described by $\boldsymbol{\theta}$ are assumed to be given by a known truncated PCE:

$$\theta_i(\boldsymbol{\xi}) = \sum_{0 \le |\boldsymbol{\alpha}| \le m} a_j^{(i)} \phi_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi}) = \mathbf{a}_i^T \boldsymbol{\phi}(\boldsymbol{\xi}) \tag{3}$$

where each component $\theta_i(\boldsymbol{\xi})$ is given by an individual polynomial series with multivariate polynomials $\phi_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi})$ with coefficients $a_j^{(i)}$. We summarise the coefficients as $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_{n_\boldsymbol{\theta}}]$. The multivariate polynomials are given by products of univariate polynomials $\phi_{\boldsymbol{\alpha}_j} = \prod_{i=1}^{n_\boldsymbol{\xi}} \phi_{\alpha_{j_i}}(\xi_i)$ with $\phi_{\alpha_{j_i}}(\xi_i)$ being univariate polynomials of ξ_i of degree α_{j_i}. The vector $\boldsymbol{\phi}(\cdot) = [\phi_1(\cdot), \ldots, \phi_L(\cdot)]^T$ contains the multivariate polynomials of the expansions, m denotes the order of truncation and $|\boldsymbol{\alpha}| = \sum_{i=1}^{n_\boldsymbol{\xi}} \alpha_i$. Each polynomial series consists of $L = \frac{(n_\boldsymbol{\xi}+m)!}{n_\boldsymbol{\xi}!m!}$ terms and $\mathbf{a}_i \in \mathbb{R}^L$ represents a vector of coefficients of these terms. The univariate polynomials are given by Hermite polynomials:

$$\phi_j(\xi_i) = (-1)^j \exp\left(\frac{1}{2}\xi_i^2\right) \frac{d^j}{d\xi_i^j} \exp\left(-\frac{1}{2}\xi_i^2\right) \tag{4}$$

3. Gauss-Hermite nonlinear model predictive control

Once the uncertainties are defined as PCE as shown in section 2, we aim to exploit this information together with the dynamic equation system in Eq.(1) to formulate an optimal control problem (OCP) to be solved iteratively. To achieve this we use Gauss-Hermite quadrature rules to create several realizations of $\boldsymbol{\theta}$ from its PCE representation to formulate a scenario-based MPC problem. The Gauss-Hermite quadrature rules can be seen to give an approximation to the integral:

$$\mathbb{E}[f(\boldsymbol{\xi})] = \int_{-\infty}^{\infty} f(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi}, \qquad p(\boldsymbol{\xi}) = \prod_i \exp(-\xi_i^2)/\sqrt{\pi} \tag{5}$$

where $\boldsymbol{\xi}$ is a standard normally distributed random variable and $\mathbb{E}[\cdot]$ is the expectation operator.

The Gauss-Hermite quadrature rules accomplish this by creating deterministic samples of $\boldsymbol{\xi}$ with corresponding weights. The approximations of expectation and variances of a function $f(\boldsymbol{\xi})$ are:

$$\mu_f = \mathbb{E}[f(\boldsymbol{\xi})] \approx \sum_{q=1}^{N_q} w_q f(\boldsymbol{\xi}_q), \quad \sigma_f^2 = \mathbb{E}\left[\left(f(\boldsymbol{\xi}) - \mu_f\right)^2\right] \approx \sum_{q=1}^{N_q} w_q \left(f(\boldsymbol{\xi}_q) - \mu_f\right)^2 \tag{6}$$

where $\boldsymbol{\xi}_q$ and w_q are given by the quadrature rule with overall N_q points. The Gauss-Hermite rule used in this work was taken from Jia et al. (2012), which is a sparse Gauss-Hermite quadrature rule. These require less samples than the full Gauss-Hermite rules for the same order of accuracy.

Using Chebyshev's inequality probability constraints can be robustly reformulated involving only the mean and variance of the random variable. Let γ be a generic random variable, then

$$\mathbb{P}(\gamma \leq 0) \geq 1-\varepsilon \implies \kappa_\varepsilon \sigma_\gamma + \hat{\gamma} \leq 0, \quad \kappa_\varepsilon = \sqrt{(1-\varepsilon)/\varepsilon} \tag{7}$$

where $\varepsilon \in (0,1) \subset \mathbb{R}$ is the probability that γ exceeds 0, $\hat{\gamma}$ and σ_γ^2 are the mean and variance of γ respectively.

The optimal control problem to be solved can subsequently be stated as follows using the Gauss-Hermite quadrature rule with N_q points, the dynamic system in Eq.(1), the PCE representation of $\theta(\cdot)$ and the initial condition at time t given as a function of $\theta(\cdot)$ as $\mathbf{x}_t(\theta(\cdot))$:

$$\begin{aligned}
\underset{\mathbf{U}}{\text{minimize}} \quad & \sum_{q=1}^{N_q} w_q J(\mathbf{x}^{(q)}(0), \mathbf{U}, \xi_q) \\
\text{subject to} \quad & \\
& \mathbf{x}^{(q)}(k+1) = \mathbf{f}(\mathbf{x}^{(q)}(k), \mathbf{u}(k), \theta(\xi_q)) && \forall (k,q) \in \mathbb{N}_k \times \mathbb{N}_q \\
& \mu_{g_{jk}} + \kappa_\varepsilon \sigma_{g_{jk}} \leq 0, \quad \kappa_\varepsilon = \sqrt{(1-\varepsilon)/\varepsilon} && \forall (j,k) \in \mathbb{N}_g^{(k)} \times \mathbb{N}_{k+1} \\
& \mu_{g_{jk}} = \sum_{q=1}^{N_q} w_q g_j^{(k)}(\mathbf{x}^{(q)}(k), \theta(\xi_q)) && \forall (j,k) \in \mathbb{N}_g^{(k)} \times \mathbb{N}_{k+1} \\
& \sigma_{g_{jk}} = \sum_{q=1}^{N_q} w_q \left(g_j^{(k)}(\mathbf{x}^{(q)}(k), \theta(\xi_q)) - \mu_{g_{jk}} \right)^2 && \forall (j,k) \in \mathbb{N}_g^{(k)} \times \mathbb{N}_{k+1} \\
& \mathbf{u}(k) \in \mathbb{U} && \forall k \in \mathbb{N}_k \\
& \mathbf{x}^{(q)}(0) = \mathbf{x}_t(\theta(\xi_q)) && \forall q \in \mathbb{N}_q
\end{aligned} \tag{8}$$

where $\mathbb{N}_k = \{0, \ldots, N-1\}$, $\mathbb{N}_{k+1} = \{1, \ldots, N\}$, $\mathbb{N}_g = \{1, \ldots, n_g^{(k)}\}$, $\mathbb{N}_q = \{1, \ldots, N_q\}$, N is the time horizon, $\mathbf{U} = \{\mathbf{u}(0), \ldots, \mathbf{u}(N-1)\}$, the objective is given by the expectation of a nonlinear function $J(\mathbf{x}_q(0), \mathbf{U}, \xi_q)$ approximated by the Gauss-Hermite rule, $\mathbb{U}$ represents the constraints on $\mathbf{u}(k)$ and $\mathbf{x}^{(q)}$ represents the state vector for each sampling point q. The chance constraints are approximated by Chebyshev's inequality given in Eq.(7) as nonlinear functions $g_j^{(k)}(\mathbf{x}^{(q)}(k), \theta(\xi_q))$ constrained robustly to be less than 0 with a probability of ε.

4. Polynomial chaos expansion filter

The PCE filter updates θ given the noisy measurements available from Eq.(2) and was first proposed in Madankan et al. (2013). It has further been applied for linear stochastic MPC in Mühlpfordt et al. (2016). Let $D_t = \{\mathbf{y}(1), \ldots, \mathbf{y}(t)\}$ be the measurements collected up to time t. Bayes's rule can be employed to update θ recursively using the previous PCE of θ:

$$p(\theta|D_t) = \frac{p(\mathbf{y}(t)|\theta) p(\theta|D_{t-1})}{p(\mathbf{y}(t)|D_{t-1})} \tag{9}$$

where $p(\theta|D_{t-1})$ is the prior distribution of θ given observations up to time $t-1$, $p(\mathbf{y}(t)|\theta)$ is the likelihood $\mathbf{y}(t)$ is observed given θ at time t. We defined $p(\mathbf{y}(t)|\theta) = \mathcal{N}(\mathbf{h}(\mathbf{x}(t), \theta), \mathbf{\Sigma_v})$ as standard normal distribution in Eq.(2) with mean $\mathbf{h}(\mathbf{x}(t), \theta)$ and covariance $\mathbf{\Sigma_v}$.

If we take both sides of Eq.(9) times $\prod_{j=1}^{n_\theta} \theta_j^{r_j}$ and integrate over both sides we obtain:

$$M_{\mathbf{r}}^+ = \frac{\int \prod_{j=1}^{n_\theta} \theta_j^{r_j} p(\mathbf{y}(t)|\theta) p(\theta|D_{t-1}) d\theta}{p(\mathbf{y}(t)|D_{t-1})} \tag{10}$$

where $M_{\mathbf{r}}^+ = \int \prod_{j=1}^{n_\theta} \theta_j^{r_j} p(\theta|D_t) d\theta$ and let $k = \sum_{j=1}^{n_\theta} r_j$. Now $M_{\mathbf{r}}^+$ refers to the various k-th order moments with respect to the updated distribution of θ, $p(\theta|D_t)$.

In our case the distribution of $\theta(\xi)$ depends on its PCE coefficients $\mathbf{A}$ and hence these need to be updated, see Eq.(3). We can approximate the distribution of $p(\theta(\xi)|D_{t-1})$ using sampling, since it is assumed that ξ follows a standard normal distribution. Let $\theta_t(\xi)$ denote the PCE of θ at time t with coefficients $\mathbf{A}_t$. Applying a sample estimate to Eq.(10) with a sample design of size N_s given by $\{\xi_1,\dots,\xi_{N_s}\}$, where $\xi_i \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ we obtain the following:

$$M_{\mathbf{r}}^{(s)+} = \frac{1}{\alpha N_s}\sum_{i=1}^{N_s}\prod_{j=1}^{n_\theta}(\theta_{t-1}(\xi_i))_j^{r_j}\, p(\mathbf{y}(t)|\theta_{t-1}(\xi_i)) \tag{11}$$

where $M_{\mathbf{r}}^{(s)+}$ is the sample approximation of the RHS of Eq.(10) of the posterior moments and $\alpha = \frac{1}{N_s}\sum_{i=1}^{N_s} p(\mathbf{y}(t)|\theta(\xi_i))$. The sample design was created using Latin hypercube sampling and the inverse transform of the standard normal distribution.

To update the coefficients $\mathbf{A}_{t-1}$ of θ_{t-1} to $\mathbf{A}_t$ to represent θ_t we use moment matching. It is possible to determine closed-form expressions of moments of PCE expansions as given in Eq.(3) in terms of their coefficient, see Dutta and Bhattacharya (2010). The difference between these and the sample estimate of the posterior moments is then minimized to update the coefficients:

$$\hat{\mathbf{A}}_t = \arg\min_{\mathbf{A}_t}\sum_{k\le m}||M_{\mathbf{r}}^{+}(\mathbf{A}_t) - M_{\mathbf{r}}^{(s)+}||_2^2 \tag{12}$$

where $k = \sum_{j=1}^{n_\theta} r_j$ was defined as the order of the moments and hence m defines the total order of moments we want to match. $M_{\mathbf{r}}^{+}(\mathbf{A}_t)$ denotes the moments of the PCE expansion as a function of the coefficients $\mathbf{A}_t$ and $\hat{\mathbf{A}}_t$ are the updated coefficients to match the posterior moments.

5. Case study

The overall framework is summarised in Algorithm 1 for receding horizon SNMPC. First it is initialized by specifying the problem, including initial coefficients of the PCE expansion of θ.

Algorithm 1: Output feedback SNMPC

Input : $\hat{\mathbf{A}}_0, \mathbf{\Sigma_v}, \mathbf{f}(\cdot), \mathbf{h}(\cdot), \mathbf{x}_0(\theta)$

for *each sampling time* $t = 0,1,2,\dots$ **do**

1. Determine $\mathbf{x}_t(\theta_t(\xi))$ using $\mathbf{f}(\cdot,\cdot,\theta_t(\xi))$ recursively from an updated initial condition $\mathbf{x}(0) = \mathbf{x}_0(\theta_t(\xi))$.
2. Solve SNMPC problem with $\theta_t(\xi)$ and $\mathbf{x}_t(\theta_t(\xi))$ and obtain optimal control actions.
3. Apply first part of the control actions to the plant.
4. Measure $\mathbf{y}(t+1)$.
5. Apply the PCE filter to update $\theta_t(\xi)$ to $\theta_{t+1}(\xi)$ by determining the coefficients $\hat{\mathbf{A}}_{t+1}$.

end

The case study aims to control a fermentation bioreactor using Algorithm 1. Fermentation is an important process in the biochemical and pharmaceutical industries. Uncertainties are often considerable and disregarding these may lead to inadequate performance. In this paper we consider a semi-batch bioreactor with the inlet substrate flowrate as the control variable. The two variables that are considered uncertain are the inlet concentration of the substrate $C_{S,in}$ and the kinetic parameter $\mu_{\max}$. The dynamic model was taken from Petersen and Jørgensen (2014) and describes the fermentation of a single cell protein using Methylococcus Capsulatus:

$$\dot{V} = F \tag{13a}$$

$$\dot{C_X} = -FC_X/V + \exp\left[\mu_{\max}(\xi)\right]\frac{C_S}{K_S + C_S + C_S^2/K_I} \tag{13b}$$

$$\dot{C_S} = -F(C_S - \exp\left[C_{S,in}(\xi)\right]/V - \gamma_s \exp\left[\mu_{\max}(\xi)\right]\frac{C_S}{K_S + C_S + C_S^2/K_I} \tag{13c}$$

where V is the volume of the reactor in m^3, F is the feed rate of substrate in m^3/h, C_X and C_S are the concentrations of biomass and substrate respectively in kg/m^3. The uncertain parameters are assumed to be given as a PCE in terms of standard normally distributed variables $\boldsymbol{\xi}$, which were log-transformed to ensure positiveness. This defines $\mathbf{f}(\cdot)$ in Eq.(1) with $\mathbf{x} = [V, C_X, C_S]^T$, $u = F$ and $\theta(\boldsymbol{\xi}) = [\mu_{\max}(\boldsymbol{\xi}), C_{S,in}(\boldsymbol{\xi})]^T$. The parameter values are $\gamma_S = 1.777$, $K_S = 0.021 kg/m^3$ and $K_I = 0.38 kg/m^3$. The corresponding output equation is given as follows:

$$\mathbf{y} = \mathbf{K}\mathbf{x} + \boldsymbol{\nu}, \quad \mathbf{K} = \mathrm{diag}\left([1,0,1]\right), \quad \boldsymbol{\nu} \sim \mathcal{N}\left(\mathbf{0}, \mathrm{diag}\left([3\times10^{-4}, 3\times10^{-4}]\right)\right) \tag{14}$$

The respective initial PCE expansions are given as, which defines $\hat{\mathbf{A}}_0$ and $\theta_0(\boldsymbol{\xi})$:

$$\mu_{\max}(\boldsymbol{\xi}) = \log(0.37) + 0.04\xi_1 + 0.02(\xi_1^2 - 1) + 0.0067(\xi_1^3 - 3\xi_1) \tag{15}$$

$$C_{S,in}(\boldsymbol{\xi}) = \log(1.00) + 0.04\xi_2 + 0.02(\xi_2^2 - 1) + 0.0067(\xi_2^3 - 3\xi_2) \tag{16}$$

The objective was set to minimize the batch time with a chance constraint to produce a minimum concentration of biomass of $10 kg/m^3$ with a probability of 0.05 with 10 control intervals in a shrinking horizon implementation. The input F was constrained between $0 kg/h$ and $10 kg/h$.

6. Results, discussion and conclusions

Firstly the simulation was run by setting the parameters of the plant model to $[\mu_{\max}, C_{S,in}] = [0.41, 1.05]$. The results of this are shown in Fig. 1. Next the initial parameter PCE of $\mu_{\max}$ and $C_{S,in}$ given in Eq.(15) and Eq.(16) were sampled 100 times randomly and used to simulate the "true" system according to Eq.(13), for which the results are given in Fig. 2. In each case Algorithm 1 was then used to control these systems in closed-loop to verify the performance given the objective and constraints outlined in the previous section. $N_s = 200$ samples were used for the PCE filter, see section 4.

In Fig. 1 it can be seen that given the measurement available at $t = 1$ the initial distribution at $t = 0$ moves towards the correct value. With several more updates it then converges to a sharp distribution at $t = 10$ due to the relatively low measurement noise. The last row highlights the working of the algorithm: The lower left plot shows that less and less time is necessary to reach the required biomass concentration due to the better estimates available of the uncertain parameters, i.e. it becomes possible to reduce the sampling times as it becomes less and less conservative. In the second graph we see that the biomass reaches nearly exactly 10 kg/m^3 at the final time with the control inputs shown in the last graph.

In Fig. 2 we see that the batch times of the Monte Carlo simulations vary significantly with batch times ranging from 70h to 140h, with most batch times around 110h. In the last two graphs it can be seen that the required biomass of 10 kg/m^3 was reached in most simulations despite the uncertainties present, however in about 7% of the scenarios this was not achieved. This is due to the parameter update being overconfident from the limited number of samples used. In particular, it can be seen that two scenarios do not reach even 4 kg/m^3, which happens if the parameters converge to the wrong value due to the limited number of samples used.

In conclusion a novel framework has been proposed by employing PCE to describe parametric uncertainties and exploiting this uncertainty description in a sparse Gauss-Hermite MPC formulation taking into account these uncertainties efficiently. Noisy measurements were used at each sampling time to update the PCE of the uncertain parameters and reduce the inherent uncertainties present significantly. It could be shown that the algorithm is able to achieve the required biomass in 93% of the scenarios, however the parameter update may be overconfident using 200 samples.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 675215.

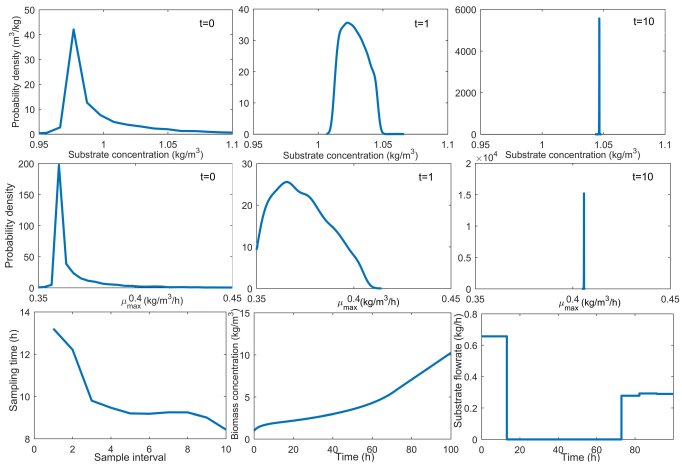


Figure 1: Run of Algorithm 1 with $[\mu_{\max}, C_{S,in}] = [0.41, 1.05]$. The first two graph rows show the evolution of the marginal distribution of the inlet substrate concentration and $\mu_{\max}$ at $t = 0$, $t = 1$ and $t = 10$. The last graph row shows the trajectories of the sampling time, biomass concentration and substrate flowrate.

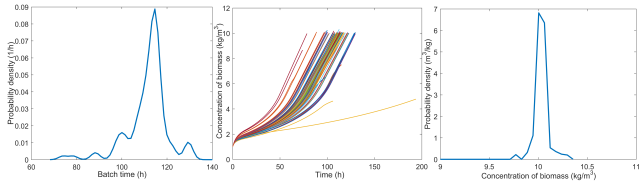


Figure 2: From left to right: Probability density function of batch times, biomass concentration trajectories and probability density function of final biomass concentrations

References

E. Bradford, L. Imsland, 2017. Expectation constrained stochastic nonlinear model predictive control of a batch bioreactor. Computer Aided Chemical Engineering 40, 1621–1626.

E. Bradford, L. Imsland, 2018a. Economic Stochastic Model Predictive Control Using the Unscented Kalman Filter. IFAC-PapersOnLine 51 (18), 417–422.

E. Bradford, L. Imsland, 2018b. Stochastic Nonlinear Model Predictive Control Using Gaussian Processes. In: 2018 European Control Conference (ECC). IEEE, pp. 1027–1034.

P. Dutta, R. Bhattacharya, 2010. Nonlinear estimation with polynomial chaos and higher order moment updates. In: American Control Conference (ACC), 2010. IEEE, pp. 3142–3147.

L. Fagiano, M. Khammash, 2012. Nonlinear stochastic model predictive control via regularized polynomial chaos expansions. In: 51st IEEE Conference on Decision and Control (CDC). IEEE, pp. 142–147.

B. Jia, M. Xin, Y. Cheng, 2012. Sparse-grid quadrature nonlinear filtering. Automatica 48 (2), 327–341.

J. M. Maciejowski, A. L. Visintini, J. Lygeros, 2007. NMPC for complex stochastic systems using a Markov chain Monte Carlo approach. In: Assessment and Future Directions of Nonlinear Model Predictive Control. Springer, pp. 269–281.

R. Madankan, P. Singla, T. Singh, P. D. Scott, 2013. Polynomial-chaos-based Bayesian approach for state and parameter estimations. Journal of Guidance, Control, and Dynamics 36 (4), 1058–1074.

A. Mesbah, 2016. Stochastic model predictive control: An overview and perspectives for future research.

T. Mühlpfordt, J. A. Paulson, R. D. Braatz, R. Findeisen, 2016. Output feedback model predictive control with probabilistic uncertainties for linear systems. In: American Control Conference (ACC), 2016. IEEE, pp. 2035–2040.

L. N. Petersen, J. B. Jørgensen, 2014. Real-time economic optimization for a fermentation process using Model Predictive Control. In: 13th European Control Conference (ECC). IEEE, pp. 1831–1836.

M. A. Sehr, R. R. Bitmead, 2017. Particle model predictive control: Tractable stochastic nonlinear output-feedback MPC. IFAC-PapersOnLine 50 (1), 15361–15366.

Appendix D

Economic stochastic nonlinear model predictive control of a semi-batch polymerization reaction

This chapter is based on **Paper L**: E. Bradford, M. Reble, A. Bouaswaig, and L. Imsland. Economic stochastic nonlinear model predictive control of a semi-batch polymerization reaction. *IFAC-PapersOnLine*, 52(1):667–672, 2019.

Economic stochastic nonlinear model predictive control of a semi-batch polymerization reaction

Eric Bradford * **Marcus Reble** ** **Ala Bouaswaig** ** **Lars Imsland** *

* *Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway (e-mail: eric.bradford,lars.imsland@ntnu.no).*
** *BASF SE, 67056 Ludwigshafen, Germany (e-mail: marcus.reble,ala.bouaswaig@basf.com).*

Abstract: Batch processes are ubiquitous in the chemical industry and difficult to control, such that nonlinear model predictive control is one of the few promising control techniques. Many chemical process models however are affected by various uncertainties, which can lower the performance and lead to constraint violations. In this paper we propose a framework for output feedback stochastic nonlinear model predictive control (SNMPC) to consider the uncertainties explicitly, which are assumed to follow known probability distributions. Polynomial chaos expansions are employed both for the formulation of the SNMPC algorithm and a nonlinear filter for the estimation of the uncertain parameters online given noisy measurements. The effectiveness of the proposed SNMPC scheme was verified on an extensive case study involving the production of the polymer polypropylene glycol in a semi-batch reactor.

Keywords: Model-based control, Uncertain dynamic systems, Stochastic parameters, Nonlinear filters, Chemical process control, Output feedback, Stochastic programming, Polynomial chaos expansions

1. INTRODUCTION

Batch processes play a vital role for the production of high value products, which make-up an important portion of the revenue in the chemical industry. Batch processes are used due to their inherent flexibility to produce multiple products and deal with variations in feedstock, product specifications, and market demand (Nagy and Braatz, 2003). Most batch process control methods have focused on tracking recipes with empirical models as in iterative learning control (Lee and Lee, 2007). Due to the high competitiveness however, there is an increasing acceptance for advanced control methods.

Model predictive control (MPC) was developed in the late seventies and refers to a class of advanced control methods that make explicit use of a dynamic model. Most applications of MPC have been limited to linear MPC (LMPC), for which plants are assumed to be weakly nonlinear. Batch processes are often highly nonlinear and operated at unsteady state. In this case nonlinear MPC (NMPC) is the method of choice (Nagy et al., 2007). In addition, NMPC can be used directly to minimize economic costs, which is becoming increasingly of interest, known as economic MPC (EMPC) (Rawlings and Amrit, 2009). NMPC applications based on first principle models are becoming more popular due to advent of improved optimization algorithms (Biegler, 2010).

Most dynamic models involve significant uncertainties, which need to be taken into account explicitly to avoid infeasibilities and performance deterioration, including parametric uncertainties, external disturbances and state estimation errors. In EMPC the system is often driven close to the constraints, however relatively little attention has been paid to handling uncertainty in EMPC with few exceptions reported (Lucia et al., 2014).

Uncertainty in MPC can be either assumed to lie in a bounded set or to be stochastic with known probability density functions (pdf), which leads to robust or stochastic MPC formulations respectively. Robust NMPC methods include min-max NMPC and tube-based NMPC, which have been both extended to handle economic objectives in Bayer et al. (2016) and in Bayer et al. (2014) respectively. Alternatively, several algorithms for stochastic NMPC (SNMPC) have been proposed. A simple solution to SNMPC is given by the successive linearization of the nonlinear system and the subsequent application of stochastic LMPC approaches, such as stochastic tube-based MPC (Cannon et al., 2009). Unscented transformations have been used in Bradford and Imsland (2017b, 2018a) to propagate uncertainties in a SNMPC application. Both linearization and unscented transformation are computationally cheap, but are only applicable to moderately nonlinear systems. A sampling average approach was used in Bradford and Imsland (2017a) with variance reduction to reduce the required number of samples, while in Maciejowski et al. (2007) Markov Chain Monte Carlo is applied. Both procedures can approximate the SNMPC problem arbitrarily well with increasing sample size but the required number of samples quickly becomes intractable. If it is assumed that the stochastic uncertainties can only take a discrete set of realizations, then multi-stage NMPC formulations have been proposed (Lucia et al., 2013; Patrinos et al., 2014). In particular, this methodology has been extensively applied to EMPC problems (Lucia et al., 2014; Sopasakis et al., 2017). While nearly all SNMPC algorithms consider full state feedback, there are several exceptions. The Unscented transformation SNMPC used in Bradford and Imsland (2017b, 2018a) are based on feedback from the Unscented Kalman filter, in Homer and Mhaskar (2018) a Lyapunov based algorithm using FokkerPlanck equation for uncertainty propagation is combined with a probabilist high-gain observer for output feedback and lastly in Sehr and Bitmead (2017) the particle filter is used for both state estimation and uncertainty propagation.

Most work in SNMPC has been concerned with the application of polynomial chaos expansions (PCE) first proposed in Fagiano and Khammash (2012). PCE have been shown to be significantly more efficient at propagating uncertainties than Monte Carlo methods for moderate numbers of uncertain parameters. The method has been applied by Mesbah et al. (2014) to an EMPC problem to obtain the required distribution of crystals of a batch process, which use Chebyshev's inequality to formulate the chance constraints. Alternatively, Streif et al. (2014) uses a sample based method instead on the PCE expansion to approximate the chance constraint, which is less conservative but also more expensive. PCE has further been applied in stochastic LMPC to great success for which the coefficients are found using the Galerkin method (Paulson et al., 2014; Lucia et al., 2015). A major disadvantage of PCE is the inherent exponential scaling with the number of uncertain parameters and the difficulty of dealing with time-varying uncertainties. Similarly to PCE it was suggested in Bradford and Imsland (2018b) to use Gaussian processes instead of orthogonal polynomials, which has the advantage that it considers the uncertainty from the approximation itself.

In this paper the PCE NMPC methodology is extended to handle the case of output feedback, i.e. where only noisy measurements of a measured output are available instead of the full state. To accomplish this, similarly as in the case of PCE LMPC (Mühlpfordt et al., 2016), we combine PCE NMPC with a recursive PCE filtering approach. The scheme is verified on a complex case study of a semi-batch polymerization reaction directly minimising the required batch time, while fulfilling several safety and product quality constraints. The paper consists of the following sections. In the next section the problem to be solved is defined and the main algorithm is introduced. In section 3 background on PCE is given, in section 4 a PCE filter is outlined and in section 5 a PCE SNMPC formulation is described. In section 6 a case study is introduced. Section 7 presents the results and discussion of this case study. Lastly, in section 8 conclusions are given.

2. PROBLEM SETUP

In this section we outline the problem to be solved, for which we propose a new framework. Consider a discrete-time nonlinear equation system with stochastic uncertainties:

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}), \quad \mathbf{x}(0) = \mathbf{x}_0(\boldsymbol{\theta}) \tag{1}$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \boldsymbol{\theta}) + \boldsymbol{\nu} \tag{2}$$

where t is the discrete time, $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$ are the system states, $\mathbf{u} \in \mathbb{R}^{n_\mathbf{u}}$ denote the control inputs, $\boldsymbol{\theta} \in \mathbb{R}^{n_\boldsymbol{\theta}}$ are time-invariant uncertainties, $\mathbf{f} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_\boldsymbol{\theta}} \to \mathbb{R}^{n_\mathbf{x}}$ represents the nonlinear dynamic system, $\mathbf{y} \in \mathbb{R}^{n_\mathbf{y}}$ denote the measurements, $\mathbf{h} : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\boldsymbol{\theta}} \to \mathbb{R}^{n_\mathbf{y}}$ are the output equations and $\boldsymbol{\nu} \in \mathbb{R}^{n_\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\boldsymbol{\nu})$ is the measurement noise assumed to follow a zero mean multivariate normal distribution with known covariance matrix $\boldsymbol{\Sigma}_\boldsymbol{\nu}$. The initial condition $\mathbf{x}(0)$ is assumed to be a function of the same uncertain parameters expressed as $\mathbf{x}_0(\boldsymbol{\theta})$.

To express the probability distribution of $\boldsymbol{\theta}$ we use PCEs; for background information on PCEs refer to section 3. Let $\boldsymbol{\theta}_t(\boldsymbol{\xi})$ correspond to the PCE of $\boldsymbol{\theta}$ at time t. It is assumed that we are initially given a PCE of $\boldsymbol{\theta}$ denoted by $\boldsymbol{\theta}_0(\boldsymbol{\xi})$. In general this initial probability distribution will be broad with a large variance to represent the uncertainty in the value of $\boldsymbol{\theta}$. At each sampling time $t+1$ we measure a value of $\mathbf{y}(t+1)$ according to Eq.(2), which is then used to update $\boldsymbol{\theta}_t(\boldsymbol{\xi})$ to $\boldsymbol{\theta}_{t+1}(\boldsymbol{\xi})$ recursively using the PCE filter outlined in section 4 by updating the coefficients of the PCE.

It should be noted that the uncertainty of the current state estimate $\mathbf{x}(t)$ is a consequence of the uncertainty in $\boldsymbol{\theta}$ and can be expressed as a function of it, which we will denote as $\mathbf{x}(t) = \mathbf{x}_t(\boldsymbol{\theta})$. Often an explicit form of $\mathbf{x}_t(\boldsymbol{\theta})$ is not available and instead $\mathbf{x}_t(\boldsymbol{\theta})$ needs to be understood as the simulation forward from $\mathbf{x}(0) = \mathbf{x}_0(\boldsymbol{\theta})$ to $\mathbf{x}(t) = \mathbf{x}_t(\boldsymbol{\theta})$ using Eq.(1).

Given the PCE $\boldsymbol{\theta}_t(\boldsymbol{\xi})$ and the function $\mathbf{x}_t(\boldsymbol{\theta})$ at each discrete time t, we wish to control the dynamic system defined by Eq.(1) subject to chance constraints and a stochastic objective. To accomplish this we solve a probabilistic finite time-horizon optimal control problem repeatedly in MPC fashion at time t:

$$\begin{aligned}
&\underset{\mathbf{u}_N}{\text{minimize}} \quad \mathbb{E}(J(N, \mathbf{x}_t(\cdot), \mathbf{u}_N, \boldsymbol{\theta}_t(\boldsymbol{\xi}))) \\
&\text{subject to} \\
&\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k), \boldsymbol{\theta}_t(\boldsymbol{\xi})) && \forall k \in \mathbb{N}_k \\
&\mathbb{P}(g_j(\mathbf{x}(k), \mathbf{u}(k)) \leq 0) \geq 1 - \epsilon && \forall (k,j) \in \mathbb{N}_{k+1} \times \mathbb{N}_g \\
&\mathbb{P}(g_j^N(\mathbf{x}(N), \mathbf{u}(N)) \leq 0) \geq 1 - \epsilon && \forall j \in \mathbb{N}_g^N \\
&\mathbf{u}(k) \in \mathbb{U}_k && \forall k \in \mathbb{N}_k \\
&\mathbf{x}(0) = \mathbf{x}_t(\boldsymbol{\theta}_t(\boldsymbol{\xi}))
\end{aligned} \tag{3}$$

where $\mathbb{N}_g = \{1, \ldots, n_g\}$, $\mathbb{N}_g^N = \{1, \ldots, n_g^N\}$, $\mathbb{N}_k = \{0, ..., N-1\}$, $\mathbb{N}_{k+1} = \{1, ..., N\}$, the expectation of $J(N, \mathbf{x}_t, \mathbf{u}_N, \boldsymbol{\theta})$ is the objective, N is the time horizon, the probability of the functions $g_j : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_\boldsymbol{\theta}} \to \mathbb{R}$ over all times and $g_j^N : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \times \mathbb{R}^{n_\boldsymbol{\theta}} \to \mathbb{R}$ at the final time exceeding 0 should be less than ϵ, the constraints on the inputs are given by $\mathbb{U}_k \subset \mathbb{R}^{n_\mathbf{u}}$ and lastly $\mathbf{u}_N := \{\mathbf{u}(0), \ldots, \mathbf{u}(N-1)\}$ represents the control inputs.

The problem in Eq.(3) is intractable due to the requirement to propagate stochastic uncertainties through nonlinear transformations and in addition the multivariate integral definition of the chance constraints. Instead, we solve a simplified problem approximating Eq.(3) using PCEs outlined in section 5. Overall the algorithm we propose uses PCE to express the uncertainty $\boldsymbol{\theta}$ described in section 3, exploits this uncertainty description to control the dynamic system in Eq.(1) using PCE SNMPC introduced in section 5 and lastly uses the measurements from Eq.(2) to update the uncertainty description utilising a PCE filter outlined in section 4. The algorithm is summarised below.

Algorithm 1: Output feedback PCE SNMPC

Input : $\mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta})$, $\mathbf{h}(\mathbf{x}, \boldsymbol{\theta})$, $\boldsymbol{\Sigma}_\boldsymbol{\nu}$, $\boldsymbol{\theta}_0(\boldsymbol{\xi})$, $\mathbf{x}_0(\boldsymbol{\theta})$

for *each sampling time* $t = 0, 1, 2, \ldots$ **do**

(1) Solve PCE SNMPC problem with $\boldsymbol{\theta}_t(\boldsymbol{\xi})$ and $\mathbf{x}_t(\boldsymbol{\theta}_t(\boldsymbol{\xi}))$ and obtain optimal control actions

(2) Apply first part of the control actions to the plant

(3) Measure $\mathbf{y}(t+1)$

(4) Apply the PCE filter to update $\boldsymbol{\theta}_t(\boldsymbol{\xi})$ to $\boldsymbol{\theta}_{t+1}(\boldsymbol{\xi})$

(5) Determine $\mathbf{x}_{t+1}(\boldsymbol{\theta}_{t+1})$ using $\mathbf{f}(\cdot, \cdot, \boldsymbol{\theta}_{t+1})$ recursively from an updated initial condition $\mathbf{x}(0) = \mathbf{x}_0(\boldsymbol{\theta}_{t+1})$

end

3. BACKGROUND: PCE

The generalized polynomial chaos expansion (gPCE) scheme will be briefly outlined in this section, for more information refer to Mesbah et al. (2014); Xiu and Karniadakis (2003);

Eldred and Burkardt (2009). A second order process $\theta(\boldsymbol{\xi})$ can be expressed as the following convergent expansion:

$$\theta(\boldsymbol{\xi}) = \sum_{j=0}^{\infty} a_j \phi_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi}) \tag{4}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$ is a n_ξ-dimensional random variable with a specified pdf, a_j denotes expansion coefficients and $\phi_{\boldsymbol{\alpha}_j} = \prod_{i=1}^{n_\xi} \phi_{\alpha_{j_i}}(\xi_i)$ denotes multivariate polynomials with $\phi_{\alpha_{j_i}}(\xi_i)$ being univariate polynomials of ξ_i of degree α_{j_i}.

The univariate polynomials are chosen according to the Askey scheme based on the probability distribution of the corresponding ξ_i to satisfy an orthogonality property, e.g. if ξ_i is a standard Gaussian random variable with zero-mean and unit variance, then $\phi_{\alpha_{j_i}}(\xi_i)$ are chosen as Hermite polynomials. Univariate Hermite polynomials He with degree j in terms of ξ_i are:

$$He_j(\xi_i) = (-1)^j \exp\left(\frac{1}{2}\xi_i^2\right) \frac{d^j}{d\xi_i^j} \exp\left(-\frac{1}{2}\xi_i^2\right) \tag{5}$$

For these orthogonal polynomials we have the useful property:

$$\langle \phi_i, \phi_j \rangle = \int \phi_i(\boldsymbol{\xi}) \phi_j(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi} = \delta_{ij} \langle \phi_i^2 \rangle \tag{6}$$

where δ_{ij} is the Kronecker delta and $p(\boldsymbol{\xi})$ is the pdf of $\boldsymbol{\xi}$.

To approximate $\theta(\boldsymbol{\xi})$ for practical reasons the PCE in Eq.(4) needs to be truncated:

$$\theta(\boldsymbol{\xi}) = \sum_{0 \leq |\boldsymbol{\alpha}| \leq m} a_j \phi_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi}) = \mathbf{a}^T \boldsymbol{\phi}(\boldsymbol{\xi}) \tag{7}$$

where $\boldsymbol{\phi}(\cdot) = [\phi_1(\cdot), \ldots, \phi_L(\cdot)]^T$ contains the multivariate polynomials of the expansion, m denotes the order of truncation and $|\boldsymbol{\alpha}| = \sum_{i=1}^{n_\xi} \alpha_i$. The truncated series consists of $L = \frac{(n_\xi + m)!}{n_\xi! m!}$ terms and $\mathbf{a} \in \mathbb{R}^L$ represents a vector of coefficients of these terms.

Next we need to evaluate the coefficients $\mathbf{a}$, which we accomplish by using the non-intrusive spectral projection approach based on the orthogonality property in Eq. (6):

$$a_j = \frac{\langle \theta(\boldsymbol{\xi}), \phi_j \rangle}{\langle \phi_j^2 \rangle} = \frac{1}{\langle \phi_j^2 \rangle} \int \theta(\boldsymbol{\xi}) \phi_j(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{8}$$

The evaluation of the integral in Eq.(8) can be approximated employing sample-based approaches. Quadrature methods are the most popular due to their significantly improved convergence rates compared to MC approaches for moderate dimensional problems. Quadrature methods take the following form:

$$\int \theta(\boldsymbol{\xi}) \phi_j(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi} \approx \frac{1}{\langle \phi_j^2 \rangle} \sum_{q=1}^{N_q} w_q \theta(\boldsymbol{\xi}^{(q)}) \phi_j(\boldsymbol{\xi}^{(q)}) \tag{9}$$

leading to the following sample estimate of the coefficients:

$$\hat{\mathbf{a}} = \mathbf{w}(\boldsymbol{\Theta})^T \boldsymbol{\Phi}(\boldsymbol{\Xi}) * \langle \boldsymbol{\phi}^2 \rangle^{-1} \tag{10}$$

where $*$ denotes element-wise multiplication, N_q is the total number of quadrature points, $\boldsymbol{\Xi} = [\boldsymbol{\xi}^{(1)}, \ldots, \boldsymbol{\xi}^{(N_q)}]^T \in \mathbb{R}^{N_q \times n_\xi}$ represents the quadrature sample design, $\mathbf{w}(\boldsymbol{\Xi}) = [w_1 \theta(\boldsymbol{\xi}^{(1)}), \ldots, w_{N_q} \theta(\boldsymbol{\xi}^{(N_q)})]^T \in \mathbb{R}^{N_q}$, w_q the quadrature weights, $\langle \boldsymbol{\phi}^2 \rangle^{-1} = [\langle \phi_1^2 \rangle, \ldots, \langle \phi_L^2 \rangle] \in \mathbb{R}^L$, $\boldsymbol{\Phi}(\boldsymbol{\Xi}) = [\boldsymbol{\phi}(\boldsymbol{\xi}^{(1)}), \ldots, \boldsymbol{\phi}(\boldsymbol{\xi}^{(N_q)})]^T \in \mathbb{R}^{N \times L}$ and the response vector is given by $\boldsymbol{\Theta} = [\theta(\boldsymbol{\xi}^{(1)}), \ldots, \theta(\boldsymbol{\xi}^{(N_q)})] \in \mathbb{R}^{N_q}$.

The type of quadrature method is again chosen based on the pdf of $\boldsymbol{\xi}$. For standard Gaussian distributed ξ_i Gauss-Hermite quadrature is chosen. The number of points required depends on the order of accuracy required and the dimension of $\boldsymbol{\xi}$. To integrate polynomials correctly up to degree p, $N_q = (p+1)^{n_\xi}$ points are required. This quickly becomes prohibitive, so we instead use a sparse Gauss-Hermite (sGH) quadrature method in this work proposed in Jia et al. (2012).

Using the coefficient approximation from Eq. (10) we have a representation for the random variable $\theta(\boldsymbol{\xi})$ parametrized by $\boldsymbol{\xi}$. The polynomial chaos expansion may also be used to represent multivariate random variables. Let a multivariate stochastic variable be given by $\boldsymbol{\theta}(\boldsymbol{\xi}) = [\theta_1(\boldsymbol{\xi}), \ldots, \theta_{n_\theta}(\boldsymbol{\xi})]^T \in \mathbb{R}^{n_\theta = n_\xi}$ with coefficients collected in $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_{n_\theta}]$, which is parametrized in terms of standard normal variables $\boldsymbol{\xi}$ with the same dimension. The properties of $\boldsymbol{\theta}(\boldsymbol{\xi})$ are dependent on the coefficients $\mathbf{A}$ of the expansion.

Let each component of $\boldsymbol{\theta}(\boldsymbol{\xi})$ be given by a truncated PCE with the same order of truncation and the same number of terms L, then the moments of $\boldsymbol{\theta}(\boldsymbol{\xi})$ have a closed-form expression in terms of the PCE coefficients. Moments can be defined as:

$$M_{\mathbf{r}}(\mathbf{A}) = \int \prod_{i=1}^{n_\xi} \theta_i^{r_i}(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{11}$$

where $\mathbf{r} \in \mathbb{R}^{n_\xi}$ is a vector defining the moments with $k = \sum_{i=1}^{n_\xi} r_i$ order.

The moments of the PCE expansion with the definition in Eq.(11) are (Dutta and Bhattacharya, 2010):

$$M_{\mathbf{r}}(\mathbf{A}) = \int \prod_{i=1}^{n_\xi} (\mathbf{a}_i^T \boldsymbol{\phi}(\boldsymbol{\xi}))^{r_i} p(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{12}$$

4. PCE FILTER

The state estimation step concerns the update of $\boldsymbol{\theta}_{t-1}(\boldsymbol{\xi})$ to $\boldsymbol{\theta}_t(\boldsymbol{\xi})$ given the noisy measurements available, in which we assume that $\boldsymbol{\xi}$ follows a standard normal distribution. The following outline was taken from Madankan et al. (2013); Mühlpfordt et al. (2016). Let $D_t = \{\mathbf{y}(1) \ldots, \mathbf{y}(t)\}$ be the measurements collected up to time t and $\mathbf{y}(t)$ the most recent measurement. Bayes's rule can be employed to update $\boldsymbol{\theta}(\boldsymbol{\xi})$ recursively as:

$$p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_t) = \frac{p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_{t-1}) p(\mathbf{y}(t)|\boldsymbol{\theta}(\boldsymbol{\xi}), D_{t-1})}{p(\mathbf{y}(t)|D_{t-1})} \tag{13}$$

where $p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_{t-1})$ is the prior distribution of $\boldsymbol{\theta}(\boldsymbol{\xi})$ at time t given all observations up to time $t-1$, $p(\mathbf{y}(t)|\boldsymbol{\theta}(\boldsymbol{\xi}), D_{t-1})$ is the likelihood $\mathbf{y}(t)$ is observed given $\boldsymbol{\theta}(\boldsymbol{\xi})$ at time t, which does not depend on the observations D_{t-1} and D_{t-1} is therefore dropped. We define $p(\mathbf{y}(t)|\boldsymbol{\theta}(\boldsymbol{\xi})) = \mathcal{N}(\mathbf{y}(t)|\mathbf{h}(\mathbf{x}(t), \boldsymbol{\theta}(\boldsymbol{\xi})), \boldsymbol{\Sigma}_{\mathbf{v}})$ as multivariate normal likelihood with mean $\mathbf{h}(\mathbf{x}(t), \boldsymbol{\theta}(\boldsymbol{\xi}))$ and covariance $\boldsymbol{\Sigma}_{\mathbf{v}}$ evaluated at $\mathbf{y}(t)$. The pdf $p(\mathbf{y}(t)|D_{t-1})$ is the total probability of observation $\mathbf{y}(t)$ at time t given by:

$$p(\mathbf{y}(t)|D_{t-1}) = \int p(\mathbf{y}(t)|\boldsymbol{\theta}(\boldsymbol{\xi})) p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_{t-1}) d\boldsymbol{\theta} \tag{14}$$

Calculating Eq. (14) analytically is difficult and we therefore use sampling instead. We know the distribution $p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_{t-1})$, since it is assumed that $\boldsymbol{\xi}$ follows a standard normal distribution. The functions $\boldsymbol{\theta}_{t-1}(\boldsymbol{\xi})$ and $\boldsymbol{\theta}_t(\boldsymbol{\xi})$ are the PCEs corresponding to the pdfs $p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_{t-1})$ and $p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_t)$ respectively. Latin hypercube sampling was applied together with the inverse normal cumulative transformation (Stein, 1987):

$$\alpha = \frac{1}{N_s} \sum_{s=1}^{N_s} p(\mathbf{y}(t)|\boldsymbol{\theta}_{t-1}(\boldsymbol{\xi}^{(s)})) \tag{15}$$

where α is the sample estimate of $p(\mathbf{y}(t)|D_{t-.1})$, N_s is the sample size and $\boldsymbol{\xi}^{(s)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are the sample points.

The prior distribution $p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_{t-1})$ is given by the previous posterior distribution of $\boldsymbol{\theta}$ due to the assumed time-invariance. If we take both sides of Eq.(13) times $\prod_{j=1}^{n_\xi} \theta_j^{r_j}$ and integrate over both sides we obtain the following:

$$M_{\mathbf{r}}^{+} = \frac{\int \prod_{j=1}^{n_\xi} \theta_j^{r_j}(\boldsymbol{\xi}) p(\mathbf{y}(t)|\boldsymbol{\theta}(\boldsymbol{\xi})) p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_{t-1}) d\boldsymbol{\theta}}{p(\mathbf{y}(t)|D_{t-1})} \quad (16)$$

where $M_{\mathbf{r}}^{+} = \int \prod_{j=1}^{n_\xi} \theta_j^{r_j}(\boldsymbol{\xi}) p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_t) d\boldsymbol{\theta}$ and let $k = \sum_{j=1}^{n_\xi} r_j$. Now $M_{\mathbf{r}}^{+}$ refers to the various k-th order moments with respect to the updated distribution of $\boldsymbol{\theta}$, $p(\boldsymbol{\theta}(\boldsymbol{\xi})|D_t)$.

Now using the sample estimate in Eq.(15) and applying a further sample estimate to Eq.(16) we obtain:

$$M_{\mathbf{r}}^{(s)+} = \frac{1}{\alpha N_s} \sum_{s=1}^{N_s} \prod_{j=1}^{n_\xi} \theta_{t-1,j}^{r_j}(\boldsymbol{\xi}^{(s)}) p(\mathbf{y}(t)|\boldsymbol{\theta}_{t-1}(\boldsymbol{\xi}^{(s)})) \quad (17)$$

where $M_{\mathbf{r}}^{(s)+}$ is an approximation of the RHS of Eq.(16).

To update $\boldsymbol{\theta}_{t-1}(\boldsymbol{\xi})$ we match the moments found in Eq.(17) with those of the PCE $\boldsymbol{\theta}_t(\boldsymbol{\xi})$, which are a function of its coefficients as shown in Eq.(12). The PCE is then fitted by solving a nonlinear least-squares optimization problem:

$$\hat{\mathbf{A}}_t = \arg\min_{\mathbf{A}_t} \sum_{k \leq m} ||M_{\mathbf{r}}^{+}(\mathbf{A}_t) - M_{\mathbf{r}}^{(s)+}||_2^2 \quad (18)$$

where $k = \sum_{j=1}^{n_\xi} r_j$ was defined above as the order of the moments and hence m defines the total order of moments we want to match. $M_{\mathbf{r}}^{+}(\mathbf{A}_t)$ is parametrized by $\mathbf{A}_t$ as shown in Eq.(12). The estimated coefficients $\hat{\mathbf{A}}_t$ then define the updated PCE $\boldsymbol{\theta}_t(\boldsymbol{\xi})$ as required for Algorithm 1.

5. PCE SNMPC

In this section we formulate an approximate algorithm to solve the OCP in Eq.(3) using PCEs. We assume the time is t and we are given the PCE $\boldsymbol{\theta}_t(\boldsymbol{\xi})$ accounting for all the data available with the function $\mathbf{x}_t(\boldsymbol{\theta})$ describing the current state in terms of $\boldsymbol{\theta}$. The aim is to control the dynamic system in Eq.(1) given these uncertainty descriptions by reformulating Eq.(3). PCEs in this regard can be used to obtain accurate mean and variance predictions of nonlinear transformations, however estimating the chance constraints remains a difficult problem.

The mean and variance of a PCE expansion in terms of $\boldsymbol{\xi}$ of a 1-dimensional random variable γ with coefficients $\mathbf{a} \in \mathbb{R}^L$ using the definition in Eq.(12) can be expressed as follows:

$$\mathbb{E}(\gamma) \approx a_1 \quad (19)$$

$$\mathrm{Var}(\gamma) \approx \sum_{i=2}^{L} a_i^2 \mathbb{E}(\Phi_i^2(\boldsymbol{\xi})) \quad (20)$$

We use Chebychev's inequality to robustly reformulate the chance constraints in terms of only the mean and variance of the constraint function. Let γ be a generic random variable with a finite variance, then (Mesbah et al., 2014):

$$\kappa_\epsilon \sigma_\gamma + \hat{\gamma} \leq 0, \; \kappa_\epsilon = \sqrt{(1-\epsilon)/\epsilon} \implies \mathbb{P}(\gamma \leq 0) \geq 1 - \epsilon \quad (21)$$

where $\epsilon \in (0,1) \subset \mathbb{R}$ is the probability that γ exceeds 0, $\hat{\gamma}$ and σ_γ^2 are the mean and variance of γ respectively.

Next we use results from section 3. In essence we evaluate the coefficients of a PCE expansion online using a quadrature rule as shown in Eq.(10). The quadrature sample design is given by $\boldsymbol{\Xi} = [\boldsymbol{\xi}^{(1)}, \ldots, \boldsymbol{\xi}^{(N_q)}]^T \in \mathbb{R}^{N_q \times n_\xi}$ with N_q being the number of sample points. Once these are defined the matrices $\boldsymbol{\Phi}(\boldsymbol{\Xi}) = [\boldsymbol{\phi}(\boldsymbol{\xi}^{(1)}), \ldots, \boldsymbol{\phi}(\boldsymbol{\xi}^{(N_q)})]^T$ and $\langle \boldsymbol{\phi}^2 \rangle^{-1}$ can be calculated offline. Each sample in $\boldsymbol{\Xi}$ represents a separate dynamic simulation according to Eq.(1), the data from which is then used according to Eq.(10) to determine the PCE coefficients online. These are then in turn used to estimate the mean and variance from Eqs.(20) and (19) to estimate the objective and chance constraints according to Chebychev's inequality in Eq.(3). It is important to note that the SNMPC algorithm can have a different order of PCE than $\boldsymbol{\theta}(\boldsymbol{\xi})$. The SNMPC algorithm to reformulate Eq. (3) can be stated as:

$$\begin{aligned}
&\underset{\mathbf{u}_N}{\text{minimize}} \quad \hat{a}_1^J \\
&\text{subject to} \\
&\mathbf{x}^{(i)}(k+1) = \mathbf{f}(\mathbf{x}^{(i)}(k), \mathbf{u}(k), \boldsymbol{\theta}_t(\boldsymbol{\xi}^{(i)})) \; \forall (k,i) \in \mathbb{N}_k \times \mathbb{N}_q \\
&\kappa_\epsilon \sum_{i=2}^{L} \left(\hat{a}_i^{g_{jk}}\right)^2 \mathbb{E}(\Phi_i^2(\boldsymbol{\xi})) + \hat{a}_1^{g_{jk}} \leq 0 \qquad \forall (k,j) \in \mathbb{N}_{k+1} \times \mathbb{N}_g \\
&\kappa_\epsilon \sum_{i=2}^{L} \left(\hat{a}_i^{g_j^N}\right)^2 \mathbb{E}(\Phi_i^2(\boldsymbol{\xi})) + \hat{a}_1^{g_j^N} \leq 0 \qquad \forall j \in \times \mathbb{N}_g^N \\
&\hat{\mathbf{a}}^J = \mathbf{w}(\boldsymbol{\Theta}^J)^T \boldsymbol{\Phi}(\boldsymbol{\Xi}) * \langle \boldsymbol{\phi}^2 \rangle^{-1} \\
&\hat{\mathbf{a}}^{g_{jk}} = \mathbf{w}(\boldsymbol{\Theta}^{g_{jk}})^T \boldsymbol{\Phi}(\boldsymbol{\Xi}) * \langle \boldsymbol{\phi}^2 \rangle^{-1} \qquad \forall (k,j) \in \mathbb{N}_{k+1} \times \mathbb{N}_g \\
&\hat{\mathbf{a}}^{g_j^N} = \mathbf{w}(\boldsymbol{\Theta}^{g_j^N})^T \boldsymbol{\Phi}(\boldsymbol{\Xi}) * \langle \boldsymbol{\phi}^2 \rangle^{-1} \qquad \forall j \in \mathbb{N}_g^N \\
&\mathbf{u}(k) \in \mathbb{U}_k \qquad \forall k \in \mathbb{N}_k \\
&\mathbf{x}^{(i)}(0) = \mathbf{x}_t(\boldsymbol{\theta}_t(\boldsymbol{\xi}^{(i)})) \qquad \forall i \in \mathbb{N}_q
\end{aligned} \quad (22)$$

where $\mathbf{x}^{(i)}$ denotes the state for each scenario i, $\mathbf{w}(\boldsymbol{\Theta}) = [w_1 \Theta_1, \ldots, w_{N_q} \Theta_{N_q}]$ with w_i being the quadrature weights and the data matrices $\boldsymbol{\Theta}^J = [J(N, \mathbf{x}_t(\boldsymbol{\theta}_t(\boldsymbol{\xi}^{(1)}), \mathbf{u}_N, \boldsymbol{\theta}_t(\boldsymbol{\xi}^{(1)})), \ldots, J(N, \mathbf{x}_t(\boldsymbol{\theta}_t(\boldsymbol{\xi}^{(N_q)}), \mathbf{u}_N, \boldsymbol{\theta}_t(\boldsymbol{\xi}^{(N_q)}))]$, $\boldsymbol{\Theta}^{g_{jk}} = [g_j(\mathbf{x}^{(1)}(k), \mathbf{u}(k)), \ldots, g_j(\mathbf{x}^{(N_q)}(k), \mathbf{u}(k))]$, $\boldsymbol{\Theta}^{g_j^N} = [g_j^N(\mathbf{x}^{(1)}(N), \mathbf{u}(N)), \ldots, g_j^N(\mathbf{x}^{(N_q)}(N), \mathbf{u}(N))]$ Eq.(22) gives the required control inputs for Algorithm 1.

6. SEMI-BATCH REACTOR CASE STUDY

Algorithm 1 outlined in section 2 is applied to a semi-batch polymerization reactor for the production of polyol from propylene oxide (PO). An extensive model for this process has been presented in Nie et al. (2013a), which has been used in Jung et al. (2015) for NMPC and in Jang et al. (2016) for multi-stage NMPC. A schematic of the process is shown in Fig. 1.

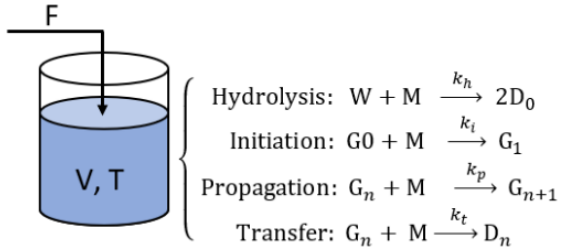


Fig. 1. F is the monomer feedrate, V and T are the volume and temperature of the liquid in the reactor respectively, W is water, M is the monomer, D_n and G_n are the dormant and active product chains with length n respectively.

To reduce the computational times we applied the method of moments (Rivero, 2005; Nie et al., 2013b) to derive differential equations for the average molecular weight. In addition, we disregard the balance equations for the unsaturated proportion of the polymer. Due the importance of temperature control a heat balance was added, in which perfect temperature control was previously assumed. This equation can nonetheless be found in Nie et al. (2013a), where it is used as a constraint. The objective was set to minimize batch time (t_f[s]) by varying the monomer feed rate F[mol/s] and the cooling water temperature T_C[K] to achieve a number average molecular weight ($NAMW$) of 450g/mol and ensure that the amount of the monomer (PO) contained in the reactor does not exceed 120ppm. During this operation the reactor temperature T[K] is constrained to remain below 420K. The chance of constraint violation was set to 0.1. We assume the amount of catalyst (n_C[mol]) and the pre-exponential coefficient of the propagation kinetic constant (A_p[m^3/mol/s]) to be uncertain and given by a PCE. Measurements during the reaction are the pressure (P[bar]) and temperature (T[K]) of the reactor. For discredization orthogonal collocation was employed. The optimization problems for the PCE SNMPC and PCE filter were solved using Casadi (Andersson et al., 2018) in conjunction with IPOPT (Wächter and Biegler, 2006). The control problem to be solved is summarised in Tab. 1. The missing parameter values and dynamic equation system can be found in Nie et al. (2013a).

Table 1. Specifications of control problem

States ($\mathbf{x}$)	m[g], PO[mol], W[mol], $T[K]$, X_0[mol], γ_0[mol], γ_1[mol]
Outputs ($\mathbf{y}$)	P[bar], T[K]
Output noise	$\mathbf{\Sigma_y} = \text{diag}(0.25, 0.01)$
Inputs (u)	F[mol], T_C[K]
Uncertainties	A_p[m^3/mol/s], n_C[mol]
Objective	minimize t_f[s]
Path constraints	$T[K] - 420 \leq 0$
End constraints	$450 - \text{NAMW[g/mol]} \leq 0$, PO[ppm] $- 120 \leq 0$
Probability	$\epsilon = 0.1$
Input constraints	$0 \leq F$[mol/s] ≤ 10, $298.15 \leq T_C$[K]
PCE SNMPC	PCE order = 3, sGH accuracy = 2, sGH manner = 1
PCE filter	Samples = 800, Moments considered = 5, PCE order = 3
Discredization	N = 12, Degree = 5
Initial PCE A_P	$9.05 + 0.25\xi_1 + 0.13(\xi_1 - 1)^2 + 0.04(\xi_1^3 - 3\xi_1)$
Initial PCE n_C	$6.91 + 0.25\xi_2 + 0.13(\xi_2 - 1)^2 + 0.04(\xi_2^3 - 3\xi_2)$
Reactor specs.	V = 17m^3, $UA = 1.5 \times 10^4 W/m^2/K$
Initial cond.	$m(0) = 1.6 \times 10^6 g$, $PO(0) = 10^4$mol, $W(0) = 10^3$mol, $X_0(0) = 0$mol, $T(0) = 378.15$K, $\gamma_0(0) = \gamma_0(1) = 10^4$mol

7. RESULTS AND DISCUSSION

Algorithm 1 outlined in section 2 was verified on the case study defined in the previous section firstly by running the NMPC on a specific realization of θ for plant model, in our case $[A_p, n_C] = [7200\text{m}^3/\text{mol/s}, 1700\text{mol}]$, significantly different from the nominal values $[8504\text{m}^3/\text{mol/s}, 1000\text{mol}]$. The results of this are shown in Fig. 2. Firstly, we can see from the first two row of graphs that the parameters are significantly better approximated at the final time than initially, which leads to a large reduction in uncertainty shown by sharper distribution in both cases. Nonetheless while little uncertainty remains of the value for A_p, n_C has still a high uncertainty with a clear bias towards a lower value. This is due to the influence of the prior, which assumed n_C to be around 1000mol. The next three rows of graphs show the control inputs and trajectories of constraints and objective. We can see that generally the batch time becomes less and less, which has two reasons. Firstly, the uncertainty is reduced at every sampling time making the algorithm less conservative and in addition the estimate of the amount catalyst is corrected upwards, which leads to higher NAMW in less time and higher consumption of monomer. First less monomer is fed, since the reactor starts with high concentrations of monomer to ensure the temperature constraint. Thereafter, the monomer is fed in at a maximum rate to reach the required NAMW in minimum time. Lastly, the monomer feedrate is reduced to 0, since at the final time the ppm needs to be less than 120. In this run the NAMW reaches a value of 451g/mol, while the monomer concentration becomes 36ppm. The relative conservativeness particularly with regards to the amount of monomer is due to the bias of the amount of catalyst to a lower value than the true value. The cooling water temperature is lowest at the beginning when the reaction rate is maximum, while at the end the cooling water temperature is relatively high since the reaction rate is close to zero due to the low monomer concentration.

Next the approach was applied to 100 MC samples of the plant according to the initial PCE representations of the uncertainties given in Tab. 1. We compare it to a NMPC approach, which uses the PCE filter to update the parameters, but ignores the distribution of the uncertain parameters and instead uses the mean value as the current best estimate. The first graph shows that the SNMPC variant is more conservative with on average longer batch times, which is expected since it accounts for the uncertainty. The next two graphs highlight the problem of ignoring the uncertainty on the parameters. For the SNMPC approach all of the scenarios obtain a NAMW larger than 450 and only 4% of the scenarios have a ppm larger than 120. This can be seen by the flat pdfs with nearly all the area of the curve in regions required by the chance constraints. For the NMPC scheme however the distributions are peaked around the required value with only 57% of scenarios reaching the required NAMW and 51% of scenarios exceeding 120ppm. Lastly, temperature control for both approaches is good showing that the uncertainties have little effect on the heat balance.

8. CONCLUSIONS

In conclusion, a novel algorithm for output feedback SNMPC has been proposed by employing PCE for both control and filtering. The SNMPC problem involved both objective and probability constraints based on general nonlinear functions. A challenging semi-batch reactor case study showed that the SNMPC framework is able to regulate the process with plant parameters vastly different from the nominal values. It managed to estimate more accurate parameter values, while still accounting for the remaining uncertainty adhering the constraints. In addition, it was shown that taking into account the uncertainty of the parameters is important even after the updates, since it otherwise leads to more than 50% of constraint violations of the end-point constraints.

ACKNOWLEDGEMENTS

Eric Bradford gratefully appreciates BASF for hosting his placement over the course of the Marie-Curie project and the helpful input and discussions for this work. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 675215.

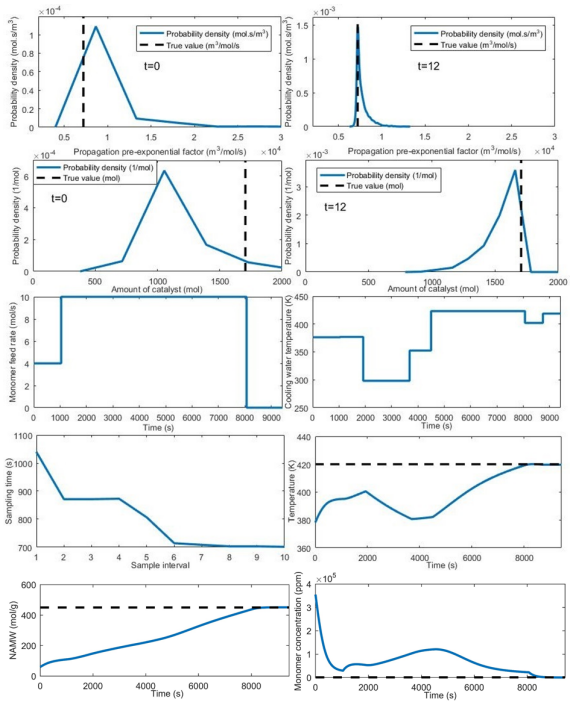


Fig. 2. Probability densities at initial and final time for both uncertainties and state trajectories for a plant model with $[A_p, n_C] = [7200\text{m}^3/\text{mol/s}, 1700\text{mol}]$

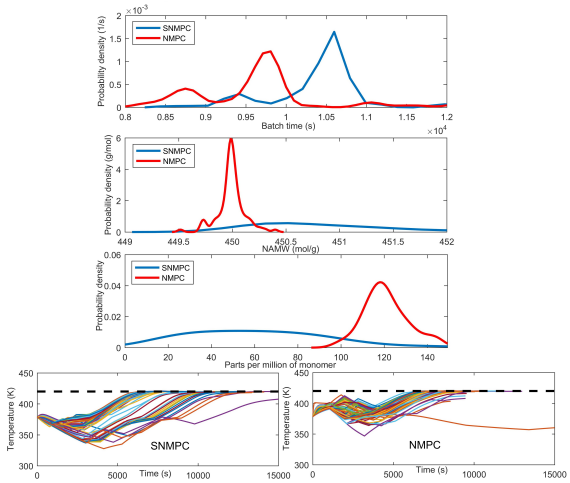


Fig. 3. Probability densities of batch time, NAMW, ppm of monomer at final time and temperature trajectories of NMPC and SNMPC based on 100 MC simulations

REFERENCES

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2018). CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 1–36.

Bayer, F.A., Müller, M.A., and Allgöwer, F. (2014). Tube-based robust economic model predictive control. *Journal of Process Control*, 24(8), 1237–1246.

Bayer, F.A., Müller, M.A., and Allgöwer, F. (2016). Min-max economic model predictive control approaches with guaranteed performance. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, 3210–3215. IEEE.

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

Bradford, E. and Imsland, L. (2017a). Expectation constrained stochastic nonlinear model predictive control of a batch bioreactor. *Computer Aided Chemical Engineering*, 40, 1621–1626.

Bradford, E. and Imsland, L. (2017b). Stochastic Nonlinear Model Predictive Control with State Estimation by Incorporation of the Unscented Kalman Filter. *arXiv preprint arXiv:1709.01201*.

Bradford, E. and Imsland, L. (2018a). Economic Stochastic Model Predictive Control Using the Unscented Kalman Filter. *IFAC-PapersOnLine*, 51(18), 417–422.

Bradford, E. and Imsland, L. (2018b). Stochastic Nonlinear Model Predictive Control Using Gaussian Processes. In *2018 European Control Conference (ECC)*, 1027–1034. IEEE.

Cannon, M., Ng, D., and Kouvaritakis, B. (2009). Successive linearization NMPC for a class of stochastic nonlinear systems. In *Nonlinear Model Predictive Control*, 249–262. Springer.

Dutta, P. and Bhattacharya, R. (2010). Nonlinear estimation with polynomial chaos and higher order moment updates. In *American Control Conference (ACC), 2010*, 3142–3147. IEEE.

Eldred, M. and Burkardt, J. (2009). Comparison of Non-Intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Quantification. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, 976. Aerospace Sciences Meetings.

Fagiano, L. and Khammash, M. (2012). Nonlinear stochastic model predictive control via regularized polynomial chaos expansions. In *51st IEEE Conference on Decision and Control (CDC)*, 142–147. IEEE.

Homer, T. and Mhaskar, P. (2018). Output-Feedback Lyapunov-Based Predictive Control of Stochastic Nonlinear Systems. *IEEE Transactions on Automatic Control*, 63(2), 571–577.

Jang, H., Lee, J.H., and Biegler, L.T. (2016). A robust NMPC scheme for semi-batch polymerization reactors. *IFAC-PapersOnLine*, 49(7), 37–42.

Jia, B., Xin, M., and Cheng, Y. (2012). Sparse-grid quadrature nonlinear filtering. *Automatica*, 48(2), 327–341.

Jung, T.Y., Nie, Y., Lee, J.H., and Biegler, L.T. (2015). Model-based on-line optimization framework for semi-batch polymerization reactors. *IFAC-PapersOnLine*, 48(8), 164–169.

Lee, J.H. and Lee, K.S. (2007). Iterative learning control applied to batch processes: An overview. *Control Engineering Practice*, 15(10), 1306–1318.

Lucia, S., Andersson, J.A.E., Brandt, H., Diehl, M., and Engell, S. (2014). Handling uncertainty in economic nonlinear model predictive control: A comparative case study. *Journal of Process Control*, 24(8), 1247–1259.

Lucia, S., Finkler, T., and Engell, S. (2013). Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9), 1306–1319.

Lucia, S., Zometa, P., Kögel, M., and Findeisen, R. (2015). Efficient stochastic model predictive control based on poly-

nomial chaos expansions for embedded applications. In *54th Annual Conference on Decision and Control (CDC)*, 3006–3012. IEEE.

Maciejowski, J.M., Visintini, A.L., and Lygeros, J. (2007). NMPC for complex stochastic systems using a Markov chain Monte Carlo approach. In *Assessment and Future Directions of Nonlinear Model Predictive Control*, 269–281. Springer.

Madankan, R., Singla, P., Singh, T., and Scott, P.D. (2013). Polynomial-chaos-based Bayesian approach for state and parameter estimations. *Journal of Guidance, Control, and Dynamics*, 36(4), 1058–1074.

Mesbah, A., Streif, S., Findeisen, R., and Braatz, R.D. (2014). Stochastic nonlinear model predictive control with probabilistic constraints. In *2014 American Control Conference*, 2413–2419. IEEE.

Mühlpfordt, T., Paulson, J.A., Braatz, R.D., and Findeisen, R. (2016). Output feedback model predictive control with probabilistic uncertainties for linear systems. In *American Control Conference (ACC), 2016*, 2035–2040. IEEE.

Nagy, Z.K. and Braatz, R.D. (2003). Robust nonlinear model predictive control of batch processes. *AIChE Journal*, 49(7), 1776–1786.

Nagy, Z.K., Mahn, B., Franke, R., and Allgöwer, F. (2007). Evaluation study of an efficient output feedback nonlinear model predictive control for temperature tracking in an industrial batch reactor. *Control Engineering Practice*, 15(7), 839–850.

Nie, Y., Biegler, L.T., Villa, C.M., and Wassick, J.M. (2013a). Reactor modeling and recipe optimization of polyether polyol processes: Polypropylene glycol. *AIChE Journal*, 59(7), 2515–2529.

Nie, Y., Biegler, L.T., Villa, C.M., and Wassick, J.M. (2013b). Reactor modeling and recipe optimization of ring-opening polymerization: Block copolymers. *Industrial & Engineering Chemistry Research*, 53(18), 7434–7446.

Patrinos, P., Sopasakis, P., Sarimveis, H., and Bemporad, A. (2014). Stochastic model predictive control for constrained discrete-time Markovian switching systems. *Automatica*, 50(10), 2504–2514.

Paulson, J.A., Mesbah, A., Streif, S., Findeisen, R., and Braatz, R.D. (2014). Fast stochastic model predictive control of high-dimensional systems. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, 2802–2809. IEEE.

Rawlings, J.B. and Amrit, R. (2009). Optimizing process economic performance using model predictive control. In *Nonlinear model predictive control*, 119–138. Springer.

Rivero, P. (2005). Calculation method of molecular weight averages in polymerization with chain-length-dependent termination. *Journal of Polymer Research*, 11(4), 309–315.

Sehr, M.A. and Bitmead, R.R. (2017). Particle model predictive control: Tractable stochastic nonlinear output-feedback MPC. *IFAC-PapersOnLine*, 50(1), 15361–15366.

Sopasakis, P., Herceg, D., Patrinos, P., and Bemporad, A. (2017). Stochastic economic model predictive control for Markovian switching systems. *IFAC-PapersOnLine*, 50(1), 524–530.

Stein, M. (1987). Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, 29(2), 143–151.

Streif, S., Karl, M., and Mesbah, A. (2014). Stochastic nonlinear model predictive control with efficient sample approximation of chance constraints. *arXiv preprint arXiv:1410.4535*.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1), 25–57.

Xiu, D. and Karniadakis, G.E. (2003). Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of computational physics*, 187(1), 137–167.

Appendix E

Output feedback stochastic nonlinear model predictive control of a polymerization batch process

This chapter is based on **Paper M**: E. Bradford, M. Reble, and L. Imsland. Output feedback stochastic nonlinear model predictive control of a polymerization batch process. In *2019 18th European Control Conference (ECC)*, pages 3144–3151. IEEE, 2019.

Output feedback stochastic nonlinear model predictive control of a polymerization batch process

Eric Bradford[1], Marcus Reble[2], and Lars Imsland[3]

***Abstract*— Nonlinear model predictive control (NMPC) is one of the few methods that can handle multivariate nonlinear control problems while accounting for process constraints. Many dynamic models are however affected by significant stochastic uncertainties that can lead to closed-loop performance problems and infeasibility issues. In this paper we propose a novel stochastic NMPC (SNMPC) algorithm to optimize a probabilistic objective while adhering chance constraints for feasibility in which only noisy measurements are observed at each sampling time. The system predictions are assumed to be both affected by parametric and additive stochastic uncertainties. In particular, we use polynomial chaos expansions (PCE) to expand the random variables of the uncertainties. These are updated using a PCE nonlinear state estimator and exploited in the SNMPC formulation. The SNMPC scheme was verified on a complex polymerization semi-batch reactor case study.**

I. INTRODUCTION

Batch processes play a vital role for the manufacture of high value products in many sectors of the chemical industry, such as pharmaceuticals, polymers, biotechnology, and food. The main reason for the continued use of batch processes is their inherent flexibility to produce multiple products and deal with variations in feedstock, product specifications, and market demand. The control of batch reactors is often challenging due to their frequently highly nonlinear behaviour and operation at states that are not steady states. Therefore, there is an increased acceptance in industry for the application of nonlinear model predictive control (NMPC) to address these challenges [1].

Model predictive control (MPC) is an advanced control method that has been employed to a significant extent in industry due to its ability to deal with multivariate plants and process constraints. MPC solves an optimal control problem (OCP) based on an explicit dynamic model at each sampling time to determine a finite sequence of control actions [2]. Due to various uncertainties however the dynamic system behaviour may be significantly different from the predicted behaviour of the dynamic model. This may lead to constraint violations and a worse control performance. If we assume the uncertainties to lie in a bounded set, robust MPC (RMPC) methods are available to deal with this problem [3]. For robust NMPC, min-max NMPC [4] and tube-based NMPC [5] have been proposed among others. While these approaches can give stability and performance guarantees in the worst-case, the probability of occurrence of the worst-case realization may be very small and hence lead to a too conservative solution [6].

Alternatively stochastic MPC (SMPC) may be employed, for which the uncertainties are given by known probability density functions (pdf). Constraints and objective in this context are addressed in a probabilistic sense, allowing for a pre-defined level of constraint violations in probability and thereby alleviating the previously mentioned problem by trading-off risk with closed-loop performance [6]. SMPC has been largely focused on linear systems [7], such as tube-based SMPC [8], scenario-based SMPC [9] and SMPC using affine-parametrizations [10], while stochastic NMPC (SNMPC) has received relatively little attention.

The main difficulty of SNMPC is propagating continuous stochastic uncertainties through nonlinear equations without being prohibitively computationally expensive. Several methods have been proposed for approximating this case, some of which have been successfully applied to formulate SNMPC approaches: Unscented transformations [11], Polynomial chaos expansions (PCE) [12], Quasi Monte Carlo (MC) [13], Markov Chain MC [14], Gaussian processes [15], Gaussian mixtures (GM) [16], Fokker-Planck [17], linearization [18], and particle filters [19]. The control of systems with discrete stochastic uncertainties has been addressed in [20]. Most SNMPC work is based on full state feedback, but there are several algorithms that have been proposed for output feedback. The unscented transformation work in [11] assumes feedback from the Unscented Kalman filter, in [21] a probabilistic high-gain observer is proposed to be jointly used with a continuous-time SNMPC formulation and lastly [19] use the particle filter equations for both state estimation and uncertainty propagation.

PCE has received a lot of attention for SNMPC, which can be seen as a sampling-based MPC algorithm to approximate both probabilistic constraints and objectives. In [12] PCEs are used to approximate objectives and constraints in expectation. The work by [22] extends the approach to include chance constraints by using Chebyshev's inequality, while in [23] the chance constraints are instead approximated using MC sampling. This leads to computationally more expensive, but less conservative approximations of the chance constraints. PCE are only able to represent time-invariant uncertainties, which causes problems with commonly assumed time-varying uncertainties. This issue has been addressed in [24] for additive noise and in [25] for non-additive noise by using conditional probability rules to essentially deal with time-varying and time-invariant uncertainties separately.

1 E. Bradford and 3 L. Imsland are with the Faculty of Information Technology and Electrical Engineering, Department of Engineering Cybernetics, NTNU, 7491 Trondheim, Norway {eric.bradford,lars.imsland}@ntnu.no

2 M. Reble is with BASF SE, 67056 Ludwigshafen, Germany marcus.reble@basf.com

Apart from SNMPC, PCEs have also been extensively used for nonlinear filtering. Firstly, PCEs can be used as a cheap surrogate to obtain mean and covariance estimates of nonlinear transformations. This has been applied to yield various PCE Kalman filters (KF) for nonlinear estimation problems, including a PCE ensemble KF [26] and a PCE extended KF [27]. Apart from the PCE KFs, several authors have applied Bayes' rule directly to the PCE expansion to attain the posterior distribution of the states. In [28] PCEs are used to propagate uncertainties, from which the moments are fitted to a GM and used in Bayes' rule. Similarly in [29] the same procedure is used, however the update is carried out using linear update laws considering higher order moments. Using sampling the posterior moments of a PCE expansion can be obtained from Bayes' rule and used to fit a posterior PCE expansion by updating the coefficients as shown in [30]. The method in [31] similarly to [30] uses posterior moments from Bayes' rule, but accounts for time-varying additive disturbances and uses the PCE in addition for uncertainty propagation. [32] propose to use PCEs for uncertainty propagation in conjunction with a particle filter.

In this paper we extend the work in [33] that proposes a SNMPC algorithm for output feedback using the nonlinear filter proposed by [30] and a PCE SNMPC algorithm as in [22]. In the previous work additive process noise was ignored due to the issue time-varying uncertainties cause for PCE based methods. To address these issues we extended the approach in [30] to be able to handle additive disturbance noise and in addition formulate an efficient SNMPC algorithm using a sparse Gauss-Hermite (sGH) sampling rule. The framework is verified on an extensive case study of a semi-batch polymerization reaction, for which we directly minimise the required batch time subject to safety and product quality constraints. The paper is comprised of the following sections. In the subsequent section a general problem definition is given and the main algorithm is introduced. In section 3 we give some background on PCE. In section 4 the PCE state estimator is outlined, while in section 5 we introduce the Gauss-Hermite SNMPC formulation. Section 6 defines the case study, while in section 7 the results and discussions of the case study are presented. Lastly, in section 8 conclusions are given.

II. PROBLEM SETUP

The problem to be solved is outlined in this section. Consider a discrete-time system of nonlinear equations with stochastic parameters and additive disturbance noise:

$$\mathbf{x}'_{t+1} = \mathbf{f}'(\mathbf{x}'_t, \mathbf{u}_t) + \mathbf{w}_t, \quad \mathbf{x}'_0 = \mathbf{x}'_0(\boldsymbol{\xi}) \tag{1}$$

$$\mathbf{y}_t = \mathbf{h}'(\mathbf{x}'_t) + \boldsymbol{\nu}_t \tag{2}$$

where t is the discrete time, $\mathbf{x}' = [\mathbf{x}, \boldsymbol{\theta}]^T$ is an augmented state vector, $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$ are the system states, $\boldsymbol{\theta} \in \mathbb{R}^{n_\boldsymbol{\theta}}$ are parametric uncertainties, $\mathbf{u} \in \mathbb{R}^{n_\mathbf{u}}$ denote the control inputs, $\mathbf{f}'(\mathbf{x}'_t, \mathbf{u}_t) = [\mathbf{f}(\mathbf{x}'_t, \mathbf{u}_t), \boldsymbol{\theta}_t]^T$ are the dynamic equations for the augmented state vector, $\mathbf{f} : \mathbb{R}^{n_\mathbf{x}+n_\boldsymbol{\theta}} \times \mathbb{R}^{n_\mathbf{u}} \to \mathbb{R}^{n_\mathbf{x}}$ represents the nonlinear dynamic system for the states, $\mathbf{y} \in \mathbb{R}^{n_\mathbf{y}}$ denote the measurements, $\mathbf{h} : \mathbb{R}^{n_\mathbf{x}+n_\boldsymbol{\theta}} \to \mathbb{R}^{n_\mathbf{y}}$ are the output equations, $\boldsymbol{\nu} \in \mathbb{R}^{n_\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\boldsymbol{\nu})$ is additive measurement noise assumed to follow a zero mean multivariate normal distribution with known covariance matrix $\boldsymbol{\Sigma}_\boldsymbol{\nu}$, and $\mathbf{w} \in \mathbb{R}^{n_\mathbf{x}+n_\boldsymbol{\theta}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\mathbf{w})$ is additive disturbance noise assumed to follow a zero mean multivariate normal distribution with known covariance matrix $\boldsymbol{\Sigma}_\mathbf{w}$. The initial condition $\mathbf{x}'_0$ is assumed to follow a known probability distribution represented by a PCE with $\boldsymbol{\xi} \in \mathbb{R}^{n_\mathbf{x}+n_\boldsymbol{\theta}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For background information on PCEs refer to section III.

To approximately represent the probability distribution of $\mathbf{x}'_t$ at each discrete time we use PCEs. Let $\mathbf{x}'_t(\boldsymbol{\xi})$ correspond to the PCE of $\mathbf{x}'$ at time t. It is assumed that we are initially given a PCE of $\mathbf{x}'$ denoted by $\mathbf{x}'_0(\boldsymbol{\xi})$ as shown in Eq.1. Usually this initial probability distribution will be broad with a relatively large variance representing the uncertainty of the initial states and uncertain parameters. At each sampling time $t+1$ we measure $\mathbf{y}_{t+1}$ given by Eq.(2). This measurement is then used to determine $\mathbf{x}'_{t+1}(\boldsymbol{\xi})$ by updating the PCE representation of $\mathbf{x}'_t(\boldsymbol{\xi})$ using Bayes' rule. The nonlinear filter using PCEs is outlined in section IV. Given the PCE $\mathbf{x}'_t(\boldsymbol{\xi})$ at each discrete time t, we wish to control the dynamic system defined by Eq.(1) subject to chance constraints and a stochastic objective. To accomplish this we solve a probabilistic finite time-horizon optimal control problem repeatedly in MPC fashion at each time t:

$$\begin{aligned}
&\underset{\mathbf{U}_N}{\text{minimize}} \quad \mathbb{E}(J(N, \mathbf{x}'_t(\boldsymbol{\xi}), \mathbf{U}_N)) \\
&\text{subject to} \\
&\mathbf{x}'_{k+1} = \mathbf{f}'(\mathbf{x}'_k, \mathbf{u}_k) + \mathbf{w}_k && \forall k \in \mathbb{N}_k \\
&\mathbb{P}(g_j(\mathbf{x}'_k, \mathbf{u}_k) \leq 0) \geq 1 - \epsilon && \forall (k, j) \in \mathbb{N}_{k+1} \times \mathbb{N}_g \\
&\mathbb{P}(g_j^N(\mathbf{x}'_N, \mathbf{u}_N) \leq 0) \geq 1 - \epsilon && \forall j \in \mathbb{N}_g^N \\
&\mathbf{u}_k \in \mathbb{U}_k && \forall k \in \mathbb{N}_k \\
&\mathbf{x}'_0 = \mathbf{x}'_t(\boldsymbol{\xi})
\end{aligned} \tag{3}$$

where $\mathbb{N}_g = \{1, \ldots, n_g\}$, $\mathbb{N}_g^N = \{1, \ldots, n_g^N\}$, $\mathbb{N}_k = \{0, ..., N-1\}$, $\mathbb{N}_{k+1} = \{1, ..., N\}$, the expectation of $J(N, \mathbf{x}'_t(\boldsymbol{\xi}), \mathbf{U}_N)$ is the objective, N is the time horizon, the probability of the functions $g_j : \mathbb{R}^{n_\mathbf{x}+n_\boldsymbol{\theta}} \times \mathbb{R}^{n_\mathbf{u}} \to \mathbb{R}$ over all times and $g_j^N : \mathbb{R}^{n_\mathbf{x}+n_\boldsymbol{\theta}} \times \mathbb{R}^{n_\mathbf{u}} \to \mathbb{R}$ at the final time exceeding 0 should be less than ϵ, the constraints on the inputs are given by $\mathbb{U}_k \subset \mathbb{R}^{n_\mathbf{u}}$, and lastly $\mathbf{U}_N := [\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}]$ represents the control inputs.

The problem in Eq.(3) cannot be solved exactly since it requires the propagation of stochastic uncertainties through nonlinear transformations and involves chance constraints, which require the evaluation of multivariate integrals. Instead, a simplified problem is formulated in section V approximating Eq.(3) using a sparse Gauss-Hermite rule. Overall we propose to use PCEs introduced in section III to represent the probability distributions of the states $\mathbf{x}_t$ and the uncertain parameters $\boldsymbol{\theta}_t$ jointly denoted as $\mathbf{x}'_t$ at each sampling time t. The SNMPC algorithm formulation in section V exploits this uncertainty description to control the dynamic system in Eq.(1), while the measurements from Eq.(2) are utilised to update the PCE presentation of $\mathbf{x}'_t$

recursively as outlined in section IV. The proposed algorithm is summarised below as Algorithm 1.

Algorithm 1: Output feedback PCE SNMPC

Input : $\mathbf{f}'(\mathbf{x}', \mathbf{u})$, $\mathbf{h}(\mathbf{x}')$, $\mathbf{\Sigma_v}$, $\mathbf{\Sigma_w}$, $\mathbf{x}'_0(\boldsymbol{\xi})$

for *each sampling time* $t = 0, 1, 2, \ldots$ **do**

1) Solve PCE SNMPC problem (3) with $\mathbf{x}'_t(\boldsymbol{\xi})$ and obtain optimal control actions.
2) Apply the first control action to the plant.
3) Measure $\mathbf{y}_{t+1}$.
4) Apply PCE filter to update $\mathbf{x}'_t(\boldsymbol{\xi})$ to $\mathbf{x}'_{t+1}(\boldsymbol{\xi})$.

end

III. BACKGROUND: PCE

The polynomial chaos expansion (PCE) scheme will be briefly outlined in this section, for more information refer to [22], [34], [35]. In this work PCEs are used as an efficient means to represent random variables. It can be shown that a random variable γ with finite second order moments can be expressed as a convergent series expansion:

$$\gamma(\boldsymbol{\xi}) = \sum_{j=0}^{\infty} a_j \boldsymbol{\phi}_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi}) \tag{4}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{n_{\boldsymbol{\xi}}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a $n_{\boldsymbol{\xi}}$-dimensional random variable following a standard normal distribution with zero-mean and unit variance, a_j denote expansion coefficients and $\boldsymbol{\phi}_{\boldsymbol{\alpha}_j} = \prod_{i=1}^{n_{\boldsymbol{\xi}}} \phi_{\alpha_{ji}}(\xi_i)$ are multivariate polynomials with $\phi_{\alpha_{ji}}(\xi_i)$ being univariate polynomials of ξ_i of degree α_{ji}.

The univariate polynomials are chosen to satisfy an orthogonality property according to the pdf of $\boldsymbol{\xi}$, which in the case of standard Gaussian random variables are given by Hermite polynomials. Hermite polynomials He with degree j in terms of ξ_i can be expressed as:

$$He_j(\xi_i) = (-1)^j \exp\left(\frac{1}{2}\xi_i^2\right) \frac{d^j}{d\xi_i^j} \exp\left(-\frac{1}{2}\xi_i^2\right) \tag{5}$$

These orthogonal polynomials have the useful property:

$$\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle = \int \boldsymbol{\phi}_i(\boldsymbol{\xi}) \boldsymbol{\phi}_j(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi} = \delta_{ij} \langle \boldsymbol{\phi}_i^2 \rangle \tag{6}$$

where δ_{ij} is the Kronecker delta and $p(\boldsymbol{\xi})$ is the pdf of $\boldsymbol{\xi}$.

To approximate $\gamma(\boldsymbol{\xi})$ for practical reasons the PCE in Eq.(4) needs to be truncated:

$$\gamma(\boldsymbol{\xi}) = \sum_{0 \leq |\boldsymbol{\alpha}_j| \leq m} a_j \boldsymbol{\phi}_{\boldsymbol{\alpha}_j}(\boldsymbol{\xi}) = \mathbf{a}^T \boldsymbol{\phi}(\boldsymbol{\xi}) \tag{7}$$

where $\boldsymbol{\phi}(\cdot) = [\phi_1(\cdot), \ldots, \phi_L(\cdot)]^T$ contains the multivariate polynomials of the expansion, m denotes the order of truncation, and $|\boldsymbol{\alpha}_j| = \sum_{i=1}^{n_{\boldsymbol{\xi}}} \alpha_{ji}$. The truncated series consists of $L = \frac{(n_{\boldsymbol{\xi}}+m)!}{n_{\boldsymbol{\xi}}! m!}$ terms and $\mathbf{a} \in \mathbb{R}^L$ represents a vector of coefficients of these terms.

From Eq.(7) we have a PCE representation of γ parametrized by $\boldsymbol{\xi}$. PCE may also represent multivariate random variables as we require for $\mathbf{x}'$. Let a multivariate stochastic variable be given by $\boldsymbol{\gamma}(\boldsymbol{\xi}) = [\gamma_1(\boldsymbol{\xi}), \ldots, \gamma_{n_\gamma}(\boldsymbol{\xi})]^T \in \mathbb{R}^{n_\gamma = n_{\boldsymbol{\xi}}}$ with coefficients collected in $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_{n_\gamma}]$, which is parametrized in terms of standard normal variables $\boldsymbol{\xi}$ with the same dimension. The properties of $\boldsymbol{\gamma}(\boldsymbol{\xi})$ are dependent on the coefficients $\mathbf{A}$ of the expansion. Let each component of $\boldsymbol{\gamma}(\boldsymbol{\xi})$ be given by a truncated PCE with the same order of truncation and the same number of terms L, then the moments of $\boldsymbol{\gamma}(\boldsymbol{\xi})$ have a closed-form expression in terms of the PCE coefficients. The statistical moments of $\boldsymbol{\gamma}$ can be defined as:

$$M_{\mathbf{r}}(\mathbf{A}) = \int \prod_{i=1}^{n_{\boldsymbol{\xi}}} \gamma_i^{r_i}(\boldsymbol{\xi}) p(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{8}$$

where $\mathbf{r} \in \mathbb{R}^{n_{\boldsymbol{\xi}}}$ is a vector defining the moments with $k = \sum_{i=1}^{n_{\boldsymbol{\xi}}} r_i$ being the overall order.

The moments of the PCE expansion with the definition in Eq.(8) are [29]:

$$M_{\mathbf{r}}(\mathbf{A}) = \int \prod_{i=1}^{n_{\boldsymbol{\xi}}} (\mathbf{a}_i^T \boldsymbol{\phi}(\boldsymbol{\xi}))^{r_i} p(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{9}$$

IV. PCE STATE ESTIMATOR

The general outline for the PCE filter was taken from [30], [36], however these works do not consider additive disturbance noise. Let $D_t = \{\mathbf{y}_1 \ldots, \mathbf{y}_t\}$ be the measurements collected up to time t and $\mathbf{y}_t$ the most recent measurement. The state estimation step concerns the update of the states given the noisy measurements available. In our algorithm the uncertainties $\mathbf{x}'_t$ are given by PCEs. In particular, let $\mathbf{x}'_{t-1}(\boldsymbol{\xi})$ refer to the previously estimated PCE. Bayes's rule can be employed to update $\mathbf{x}'$ from $\mathbf{x}'_{t-1}|D_{t-1}$ to $\mathbf{x}'_t|D_t$ recursively as follows:

$$p(\mathbf{x}'_t|D_t) = \frac{p(\mathbf{x}'_t|D_{t-1}) p(\mathbf{y}_t|\mathbf{x}'_t, D_{t-1})}{p(\mathbf{y}_t|D_{t-1})} \tag{10}$$

Next we define each term on the RHS of Eq.(10), which are dependent on the dynamic and measurement equation.

1) $p(\mathbf{x}'_t|D_{t-1})$*:* Prior distribution of $\mathbf{x}'_t$ given the previous measurements D_{t-1}, which can be expressed as:

$$p(\mathbf{x}'_t|D_{t-1}) = \int p(\mathbf{x}'_t|\mathbf{x}'_{t-1}) p(\mathbf{x}'_{t-1}|D_{t-1}) d\mathbf{x}'_{t-1} \tag{11}$$

where $p(\mathbf{x}'_t|\mathbf{x}'_{t-1}) = \mathcal{N}(\mathbf{x}'_t|\mathbf{f}'(\mathbf{x}'_{t-1}, \mathbf{u}_{t-1}), \mathbf{\Sigma_w})$ is a multivariate normal pdf with mean given by the dynamics defined in Eq.(1) and the covariance by the disturbance noise evaluated at $\mathbf{x}_t$. It should be noted that without disturbance noise $p(\mathbf{x}'_t|D_{t-1}) = \int \delta(\mathbf{x}'_t - \mathbf{f}'(\mathbf{x}'_{t-1}, \mathbf{u}_{t-1})) p(\mathbf{x}'_{t-1}|D_{t-1}) d\mathbf{x}'_{t-1}$.

2) $p(\mathbf{y}_t|\mathbf{x}'_t, D_{t-1})$*:* The pdf of the current measurement $\mathbf{y}_t$ being observed given $\mathbf{x}'_t$, which can be given as follows:

$$p(\mathbf{y}_t|\mathbf{x}'_t, D_{t-1}) = \mathcal{N}(\mathbf{y}_t|\mathbf{h}(\mathbf{x}'_t), \mathbf{\Sigma_v}) \tag{12}$$

3) $p(\mathbf{y}_t|D_{t-1})$*:* Total probability of observation $\mathbf{y}_t$ given previous measurements can be expressed as:

$$p(\mathbf{y}_t|D_{t-1}) = \int p(\mathbf{y}_t|\mathbf{x}'_t, D_{t-1}) p(\mathbf{x}'_t|D_{t-1}) d\mathbf{x}'_t \tag{13}$$

If we take both sides of Eq.(10) times $\prod_{j=1}^{n_\xi}(x'_{tj})^{r_j}$ and integrate over both sides with respect to $\mathbf{x}'_t$ we obtain:

$$M_{\mathbf{r}}^{+} = \frac{\int \prod_{j=1}^{n_\xi}(x'_{tj})^{r_j} p(\mathbf{y}_t|\mathbf{x}'_t, D_{t-1}) p(\mathbf{x}'_t|D_{t-1}) d\mathbf{x}'_t}{p(\mathbf{y}_t|D_{t-1})} \quad (14)$$

where from Eq.(10) $M_{\mathbf{r}}^{+} = \int \prod_{j=1}^{n_\xi}(x'_{tj})^{r_j} p(\mathbf{x}'_t|D_t) d\mathbf{x}'_t$ and $k = \sum_{j=1}^{n_\xi} r_j$. Now $M_{\mathbf{r}}^{+}$ refers to the various k-th order moments of the updated distribution of $\mathbf{x}'_t$, $p(\mathbf{x}'_t|D_t)$.

Next we will deal with the approximation of the RHS of Eq.(14). Evaluating the various multivariate integrals required analytically is difficult and we therefore apply sampling. We are given a PCE expansion at time $t-1$, $\mathbf{x}'_{t-1}(\boldsymbol{\xi})$, corresponding to the pdf $p(\mathbf{x}'_{t-1}|D_{t-1})$ in Eq.(11). This distribution can be readily sampled, since $\boldsymbol{\xi}$ follows a standard normal distribution. Apart from sampling $\boldsymbol{\xi}$, we also require samples of the disturbance $\mathbf{w}$ to deal with the integrals over both $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$. It should be noted that Gauss-Hermite rules were not used for the sampling, since these showed poor convergence due to the high nonlinearity of the likelihood functions. Latin hypercube sampling was applied instead with the inverse normal cumulative transformation to improve the convergence rate over MC [37]:

$$\alpha = \frac{1}{N_s}\sum_{s=1}^{N_s} \mathcal{N}(\mathbf{y}_t|\mathbf{h}(\mathbf{x}_t'^{(s)}), \boldsymbol{\Sigma}_{\mathbf{v}}) \quad (15)$$

where α is the sample estimate of $p(\mathbf{y}(t)|D_{t-1})$, $\mathbf{x}_t'^{(s)} = \mathbf{f}'(\mathbf{x}_{t-1}(\boldsymbol{\xi}^{(s)}), \mathbf{u}_{t-1}) + \mathbf{w}^{(s)}$, N_s is the sample size, $\boldsymbol{\xi}^{(s)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{w}^{(s)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{w}})$ are the sample points.

Now using the sample estimate in Eq.(15) and applying a further sample estimate to Eq.(14) we obtain:

$$M_{\mathbf{r}}^{(s)+} = \frac{\sum_{s=1}^{N_s}\prod_{j=1}^{n_\xi}(x'_{tj}(\boldsymbol{\xi}^{(s)}))^{r_j}\mathcal{N}(\mathbf{y}_t|\mathbf{h}(\mathbf{x}_t'^{(s)}), \boldsymbol{\Sigma}_{\mathbf{v}})}{\alpha N_s} \quad (16)$$

where $M_{\mathbf{r}}^{(s)+}$ is an approximation of the RHS of Eq.(14).

To update $\mathbf{x}'_{t-1}(\boldsymbol{\xi})$ we match the moments found in Eq.(16) with those of the PCE $\mathbf{x}'_t(\boldsymbol{\xi})$, which are a function of its coefficients as shown in Eq.(9). The PCE is then fitted by solving a nonlinear least-squares optimization problem:

$$\hat{\mathbf{A}}_t = \arg\min_{\mathbf{A}_t} \sum_{k\leq m} ||M_{\mathbf{r}}^{+}(\mathbf{A}_t) - M_{\mathbf{r}}^{(s)+}||_2^2 \quad (17)$$

where $k = \sum_{j=1}^{n_\xi} r_j$ was defined above as the order of the moments and hence m defines the total order of moments we want to match. $M_{\mathbf{r}}^{+}(\mathbf{A}_t)$ is parametrized by $\mathbf{A}_t$ as shown in Eq.(9). The estimated coefficients $\hat{\mathbf{A}}_t$ then define the updated PCE $\mathbf{x}'_t(\boldsymbol{\xi})$ as required for Algorithm 1.

V. SPARSE GAUSS HERMITE SNMPC

In this section we outline a SNMPC formulation to approximately solve the OCP stated in Eq.(3). The problem of controlling a dynamic equation system given by Eq.(1) with initial conditions represented by PCEs and additive time-varying disturbance noise has been addressed in [24]. In this section we propose to use a similar approach with some changes. In [24] the initial conditions and parametric uncertainties are propagated by using PCEs, while the disturbance noise is propagated using linearization. We use a sGH rule instead of PCEs, which is significantly cheaper at estimating the mean and variance of nonlinear transformations. The GH sampling rule is particularly well-suited for uncertainties described by PCEs, since these are parametrized by standard normal distributed variables. In addition, it can be seen in [38] that for the same sample size the accuracy between GH sampling rules and PCE approximations is nearly identical.

In this work we use a sGH quadrature rule proposed in [39] to approximate mean and variance of the constraint and objective functions, which create a deterministic sample of $\boldsymbol{\xi}$ with corresponding weights. The sGH approximations of expectation and variances of a function $q(\boldsymbol{\xi})$ are:

$$\mathbb{E}\left[q(\boldsymbol{\xi})\right] \approx \mu_q = \sum_{q=1}^{N_q} w_q q(\boldsymbol{\xi}_q) \quad (18)$$

$$\mathbb{E}\left[\left(q(\boldsymbol{\xi}) - \mu_q\right)^2\right] \approx \sigma_q^2 = \sum_{q=1}^{N_q} w_q \left(q(\boldsymbol{\xi}_q) - \mu_q\right)^2 \quad (19)$$

where $\boldsymbol{\xi}_q$ and w_q are given by the sGH quadrature rule with overall N_q points. μ_q and σ_q^2 are the sGH mean and variance approximation respectively.

The above approximation is utilised to account for the contribution of the initial PCE $\mathbf{x}'_t(\boldsymbol{\xi})$, while the time-varying uncertainty $\mathbf{w}$ is accounted for by using linearisation. This is necessary, since $\mathbf{w}$ is time-varying and hence each $\mathbf{w}_t$ is a separate random variable leading otherwise to a too high dimension for sGH. Employing the law of total expectation we can deal with the uncertainties using sGH for $\mathbf{x}'_t(\boldsymbol{\xi})$ and $\mathbf{w}$ using linearization, which can be stated as [31]:

$$\mathbb{E}[q(\mathbf{x}'_t, \mathbf{w})] = \mathbb{E}_{\mathbf{x}'_t}[\mathbb{E}_{\mathbf{w}}[q(\mathbf{x}'_t, \mathbf{w})|\mathbf{x}'_t]] \quad (20)$$

$$\mathrm{Var}[q(\mathbf{x}'_t, \mathbf{w})] = \mathbb{E}_{\mathbf{x}'_t}[\mathrm{Var}_{\mathbf{w}}[q(\mathbf{x}'_t, \mathbf{w})|\mathbf{x}'_t]] + \mathrm{Var}_{\mathbf{x}'_t}[\mathbb{E}_{\mathbf{w}}[q(\mathbf{x}'_t, \mathbf{w})|\mathbf{x}'_t]] \quad (21)$$

By approximating the inner expectation and variance over $\mathbf{w}$ using linearisation, we arrive at:

$$\mathbb{E}[q(\mathbf{x}'_t, \mathbf{w})] \approx \mathbb{E}_{\mathbf{x}'_t}[q(\mathbf{x}'_t, \boldsymbol{\mu}_{\mathbf{w}})] \quad (22)$$

$$\mathrm{Var}[q(\mathbf{x}'_t, \mathbf{w})] \approx \mathbb{E}_{\mathbf{x}'_t}[\mathbf{Q}\boldsymbol{\Sigma}_{\mathbf{w}}\mathbf{Q}^T] + \mathrm{Var}_{\mathbf{x}'_t}[q(\mathbf{x}'_t, \boldsymbol{\mu}_{\mathbf{w}})] \quad (23)$$

where $\boldsymbol{\mu}_{\mathbf{w}}$ denotes the mean of $\mathbf{w}$, $\mathbf{Q} = \frac{\partial q}{\partial \mathbf{w}}|_{\mathbf{x}'_t, \boldsymbol{\mu}_{\mathbf{w}}}$ and $\boldsymbol{\Sigma}_{\mathbf{w}}$ is the covariance of $\mathbf{w}$.

The remaining expectation and variances can then be approximated by creating samples of $\mathbf{x}'_t$ using the sGH rule. Chance constraints are multivariate integrals that are very difficult to estimate online. We use Chebychev's inequality to robustly reformulate the chance constraints in terms of only the mean and variance of the constraint function. Let γ be a random variable with a finite variance, then [22]:

$$\kappa_\epsilon\sqrt{\sigma_\gamma^2} + \hat{\gamma} \leq 0,\ \kappa_\epsilon = \sqrt{(1-\epsilon)/\epsilon} \Rightarrow \mathbb{P}\left(\gamma \leq 0\right) \geq 1 - \epsilon \quad (24)$$

where $\epsilon \in (0, 1) \subset \mathbb{R}$ is the probability that γ exceeds 0, $\hat{\gamma}$ and σ_γ^2 are the mean and variance of γ respectively.

Next we state the finite-horizon stochastic OCP problem that approximates Eq.(3) using the above results. For the SNMPC algorithm we use linearization to account for and propagate the uncertainty of $\mathbf{w}$ similar to an extended Kalman filter based NMPC algorithm [1]. This is carried out for each sample generated by the sGH rule and the overall mean and variance of the objective and constraint function are then found using the law of total expectation. The chance constraints are then further approximated using Chebychev's inequality in Eq.(24). First we create a sGH quadrature sample design with N_q points, $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{N_q}\}$ each with corresponding weights w_q. Assuming we are at time t and are hence given a PCE representation $\mathbf{x}'_t(\boldsymbol{\xi})$, the SNMPC problem can be stated as follows:

$$
\begin{aligned}
\underset{\mathbf{U}_N}{\text{minimize}} \quad & \sum_{i=1}^{N_q} w_q J(N, \mathbf{x}'_t(\boldsymbol{\xi}_i), \mathbf{U}_N) \\
\text{subject to} \quad & \\
& \boldsymbol{\mu}^{(i)}_{\mathbf{x}',k+1} = \mathbf{f}(\boldsymbol{\mu}^{(i)}_{\mathbf{x}',k}, \mathbf{u}_k) && \forall (k,i) \in \mathbb{N}_k \times \mathbb{N}_q \\
& \boldsymbol{\Sigma}^{(i)}_{\mathbf{x}',k+1} = \mathbf{F}^{(i)}_k \boldsymbol{\Sigma}^{(i)}_{\mathbf{x}',k} \mathbf{F}^{(i)T}_k + \boldsymbol{\Sigma}_{\mathbf{w}} && \forall (k,i) \in \mathbb{N}_k \times \mathbb{N}_q \\
& \mu^{(k)}_{g^j} + \kappa_\epsilon \sigma^{(k)}_{g^j} \leq 0 && \forall (k,j) \in \mathbb{N}_{k+1} \times \mathbb{N}_g \\
& \mu_{g^N_j} + \kappa_\epsilon \sigma_{g^N_j} \leq 0 && \forall j \in \mathbb{N}_g \\
& \mathbf{u}_k \in \mathbb{U}_k && \forall k \in \mathbb{N}_k \\
& \boldsymbol{\mu}^{(i)}_{\mathbf{x}',0} = \mathbf{x}'_t(\boldsymbol{\xi}_i) && \forall i \in \mathbb{N}_q
\end{aligned}
\tag{25}
$$

where $\boldsymbol{\mu}^{(i)}_{\mathbf{x}',k}$ and $\boldsymbol{\Sigma}^{(i)}_{\mathbf{x}',k+1}$ represent the mean and covariance of $\mathbf{x}'_k$ for sample i, $\mathbf{F}^{(i)}_k = \frac{\partial \mathbf{f}'}{\partial \mathbf{x}'}|_{\boldsymbol{\mu}^{(i)}_{\mathbf{x}',k}, \mathbf{u}_k}$ denotes the linearised dynamic equation system at time k for sample i, $\mu_{g^{(k)}_j} = \sum_{i=1}^{N_q} w_i g_j(\boldsymbol{\mu}^{(i)}_{\mathbf{x}',k}, \mathbf{u}_k)$ is the mean of the constraint $g^{(k)}_j$, $\sigma^2_{g^{(k)}_j} = \sum_{i=1}^{N_q} w_i \mathbf{G}^{(i)}_{g^{(k)}_j} \boldsymbol{\Sigma}^{(i)}_{\mathbf{x}',k} \mathbf{G}^{(i)T}_{g^{(k)}_j} + \sum_{i=1}^{N_q} w_i \left(g_j(\boldsymbol{\mu}^{(i)}_{\mathbf{x}',k}, \mathbf{u}_k) - \mu^2_{g^{(k)}_j} \right)^2$ is the variance of constraint $g^{(k)}_j$, $\mathbf{G}^{(i)}_{g^{(k)}_j} = \frac{\partial g_j}{\partial \mathbf{x}'}|_{\boldsymbol{\mu}^{(i)}_{\mathbf{x}',k}, \mathbf{u}_k}$ is the Jacobian matrix for the constraint $g^{(k)}_j$ for sample i, $\mu_{g^N_j} = \sum_{i=1}^{N_q} w_i g^N_j(\boldsymbol{\mu}^{(i)}_{\mathbf{x}',N}, \mathbf{u}_N)$ is the mean of constraint g^N_j, $\sigma^2_{g^N_j} = \sum_{i=1}^{N_q} w_i \mathbf{G}^{(i)T}_{g^N_j} \boldsymbol{\Sigma}^{(i)}_{\mathbf{x}',k} \mathbf{G}^{(i)}_{g^N_j} + \sum_{i=1}^{N_q} w_i \left(g_j(\boldsymbol{\mu}^{(i)}_{\mathbf{x}',k}, \mathbf{u}_k) - \mu^2_{g^N_j} \right)^2$ is the variance of constraint g^N_j and lastly $\mathbf{G}^{(i)T}_{g^N_j} = \frac{\partial g^N_j}{\partial \mathbf{x}'}|_{\boldsymbol{\mu}^{(i)}_{\mathbf{x}',N}, \mathbf{u}_N}$ is the Jacobian matrix for constraint g^N_j for sample i.

Solving Eq.(25) at each time t gives the required control inputs for Algorithm 1.

VI. CASE STUDY

The algorithm outlined in section II is employed for the control of a semi-batch polymerization reaction involving the production of polyol from propylene oxide (PO) in a shrinking horizon fashion. A schematic of the process is shown in Fig. 1. A complex model for this process has been proposed in [40], which was employed in [41] for NMPC and in [42] for multi-stage NMPC. The computational times reported in these papers is relatively high at around 30 seconds to minutes. This is due to the model being highly nonlinear and requiring a separate balance equation for each polymer with a specific chain length. To reduce computational times we therefore simplified the model using the method of moments [43] to derive differential equations for the average molecular weight. Further, we disregard the balance equations for the unsaturated proportion of the polymer and assume that there is no water or methanol present in the reactor initially. In the aforementioned work perfect temperature control was assumed. We added a heat balance in this work due to the importance of temperature control for safety reasons. The simplified ordinary equation system can be stated as follows:

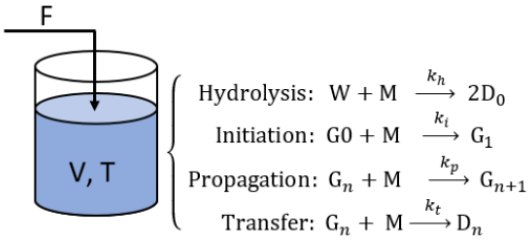


Fig. 1. F is the monomer feedrate, V and T are the volume and temperature of the liquid in the reactor respectively, W is water, M is the monomer, D_n and G_n are the dormant and active product chains with length n respectively.

$$\dot{m} = FMW_{\text{PO}}, \quad m(0) = m_0(\boldsymbol{\xi}) \tag{26a}$$

$$\dot{T} = \frac{(-\Delta H_p) k_p n_C PO}{V m C_{pb}} - \frac{UA(T - T_C)}{m C_{pb}} - \frac{FMW_{\text{PO}} C_{pf}(T - T_f)}{m C_{pb}}, \quad T(0) = T_0(\boldsymbol{\xi}) \tag{26b}$$

$$\dot{PO} = F - \frac{n_C(k_p + k_t) PO}{V}, \quad PO(0) = PO_0(\boldsymbol{\xi}) \tag{26c}$$

$$\dot{\gamma}_1 = \frac{k_p n_C PO}{V}, \quad \gamma_1(0) = \gamma_{1_0}(\boldsymbol{\xi}) \tag{26d}$$

where m[g] is the liquid mass in the reactor, F[mol/s] is the feed rate of the monomer, T[K] is the temperature of the reactor, PO[mol] is the amount of monomer and γ_1 is the first moment and hence the average molecular weight of the polymer chains, $MW_{\text{PO}} = 58.08$g/mol is the molecular weight of PO, ΔH_p is the enthalpy of the propagation reaction, $k_p = A_p \exp(-E_{Ap}/RT)$, n_C is the amount of catalyst, V is the volume of the liquid in the reactor, C_{pb} and C_{pf} are the heat capacities of the bulk liquid and the monomer feed respectively, $k_t = A_t \exp(-E_{At}/RT)$ and T_C[K] is the cooling water temperature. $m_0(\boldsymbol{\xi})$, $T_0(\boldsymbol{\xi})$, $PO_0(\boldsymbol{\xi})$, and $\gamma_{1_0}(\boldsymbol{\xi})$ are the initial PCE expansions of the states m, T, PO, and γ_1 respectively.

The missing parameter values including the vaporliquid equilibrium equations and temperature correlations can be found in [40]. The aim of the control algorithm is to minimize the required remaining batch time (t_f[s]) to achieve a specified number average molecular weight

(NAMW[g/mol]) of the final product of 350g/mol and ensuring that the amount of monomer (PO) at the end of the batch does not exceed 1000ppm. The definitions NAMW can be found in [40], which requires the zeroth moment γ_0[mol]. Due the assumptions made this zeroth moment is a constant in the defined problem and given in Tab. I. For safety reasons the reactor temperature T[K] is constrained to remain below 420K. The control variables are the monomer feed rate F[mol/s] to the reactor and the cooling water temperature T_C[K]. The chance of constraint violation was set to 0.05. Apart from the uncertainty of the states, the variables pre-exponential coefficient of the propagation kinetic constant (A_p[m^3/mol/s]) and the heat transfer coefficient (UA[W/K]) were assumed to be uncertain and added to the state vector as shown in Eq.(1). Measurements during the reaction are the pressure (P[bar]) of the reactor, the temperature (T[K]) of the reactor, and the amount of monomer (PO[mol]). The discretization of Eq.(26) was carried-out utilising orthogonal collocation. The resulting optimization problems for both the sGH SNMPC problem and the PCE state estimator were solved using Casadi [44] in conjunction with IPOPT [45]. The control problem to be solved is specified in Tab. I.

TABLE I

SPECIFICATIONS OF CONTROL PROBLEM

Augmented state ($\mathbf{x}'$)	m[g], PO[mol], T[K], γ_1[mol]
Disturbance noise	$\mathbf{\Sigma_w} = \text{diag}(0, 0, 1, 2, 50, 200)$
Outputs ($\mathbf{y}$)	P[bar], T[K], PO[mol]
Output noise	$\mathbf{\Sigma_\nu} = \text{diag}(0.25, 0.001, 1000)$
Inputs (u)	F[mol], T_C[K]
Uncertainties	A_p[m^3/mol/s], n_C[mol]
Objective	minimize t_f[s]
Path constraints	$T[K] - 420 \leq 0$
1st end constraint	$350 - \text{NAMW[g/mol]} \leq 0$
2nd end constraint	PO[ppm] $- 1000 \leq 0$
Probability	$\epsilon = 0.05$
Input constraints	$0 \leq F$[mol/s] ≤ 10, $298.15 \leq T_C$[K]
sGH SNMPC	sGH accuracy = 2, sGH manner = 1
$\mathbf{x}'$-PCE	PCE order = 2
PCE filter	Samples = 4000, Moments considered = 4
Discretization	N = 8, Degree = 5
Initial PCE $m_0(\boldsymbol{\xi})$	1537710
Initial PCE $PO_0(\boldsymbol{\xi})$	10000 + $1000\xi_2$
Initial PCE $T_0(\boldsymbol{\xi})$	378.15 + $4\xi_3$
Initial PCE $\gamma_{1\,0}(\boldsymbol{\xi})$	10000 + $500\xi_4$
Initial PCE $A_{P\,0}(\boldsymbol{\xi})$	8504 + $1000\xi_5$
Initial PCE $UA_0(\boldsymbol{\xi})$	40000 + $4000\xi_6$
Reactor specs.	V_R = 17m^3, n_C = 1000mol, γ_0 = 10000mol

VII. RESULTS AND DISCUSSIONS

In this section we verify Algorithm 1 by applying it to the previously specified control problem. First we run the algorithm on a specific realization of the initial uncertainties on $\mathbf{x}'$ as specified by the initial PCE in Tab. I, which is as follows: $\mathbf{x}' = [1537710, 12371, 375.2, 9855, 10100, 36301]^T$. The disturbances $\mathbf{w}$ and measurement noise $\boldsymbol{\nu}$ are randomly sampled. The results of this run are shown in Fig. 2.

The two graphs in the first row of Fig. 2 show the evolution of the pdf of the two uncertain parameters A_p and UA from its PCE. Firstly, it can be seen that both parameters are significantly better approximated at the final time than initially through the measurement updates. In particular, the distribution of A_p starts out at around 7500, but thereafter rapidly approaches its true value shown by the vertical black line at around 10000. Nonetheless some bias remains towards a lower value, however with some probability to take its true value or higher. UA on the other hand does not have such a bias and converges quickly to its true value, but the distribution remains relatively broad due to influence of the disturbance and measurement noise. The next two rows show the trajectories of the 4 states as a continuous blue line, while the black crosses and error bars represent the state estimates with a 95% confidence interval of the PCE expansion. The mass and monomer have near exact state estimates due to the assumed low disturbance noise. The monomer has a slight deviation initially but quickly converges to the true trajectory once the first measurement becomes available. The ppm of the monomer at the final time was found to be 916 and hence 84 less than required. This can be explained by the underestimation of A_p, which means the SNMPC will run longer to ensure a sufficiently low ppm is reached. Temperature has some significant uncertainty, which is accounted for in the SNMPC algorithm to not violate the constraint to remain below 420K. Lastly, the first moment over-shoots the constraint required to reach 350mol/g NAMW and instead reaches a NAMW of 385mol/g. This can again be explained by the underestimation of A_p and hence increasing the batch time to reach the required NAMW and ppm in at least 95% of possible cases. This can also be seen in the graph in the last row, where at first the sampling time decreases due to less uncertainty but then increases again to ensure the end-point constraints. In the 4th row the control inputs are shown, which are as expected. First the feed rate of the monomer is set to the maximum before it is set to zero to ensure a low ppm a the end of the batch process. The reaction rate is highest initially at which point the cooling temperature is at its lower bound and after which only moderate cooling temperatures are required to prevent constraint violations of the reactor temperature.

Next we ran 100 MC simulations of the control problem. This is compared to 100 MC simulations of a nominal NMPC algorithm using the mean value of the PCE expansion as state estimate from the PCE state estimator, but ignoring the shape of the probability distribution otherwise. This is done to show the importance of accounting for the inherent uncertainty in the problem. The results of these MC simulations are highlighted in Fig. 3. The first graph illustrates that while the nominal NMPC method manages in only 49% of cases to realize the required NAMW, the SNMPC approach due its increased conservativeness manages to fulfil the NAMW constraint in 99% of the simulations. Similarly the second graphs shows that the SNMPC approach manages to reduce the amount of monomer below the required threshold of 1000 ppm in 97% of realizations, while the nominal NMPC adheres this end-point constraint in only 52% of the simulations. The next graph shows that this increased robustness comes at the price of on-average longer batch times. The nominal NMPC took on average 4800 seconds, while the

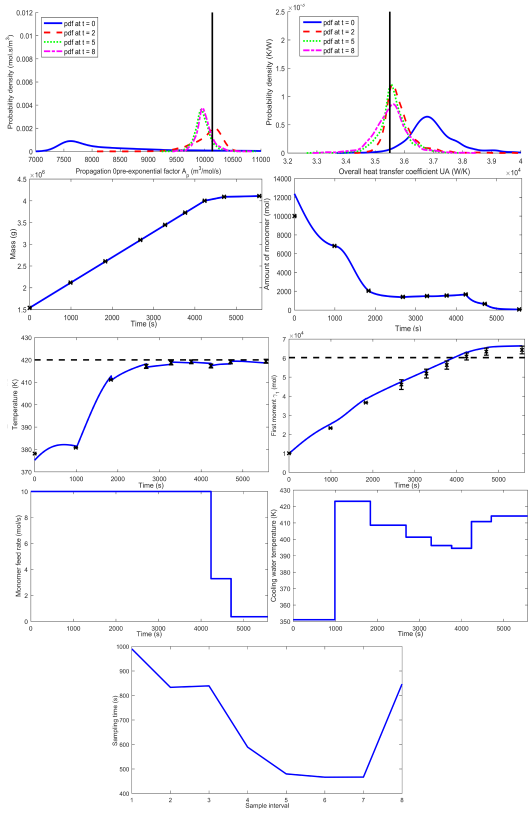


Fig. 2. First row shows the pdfs at $t = 0, 2, 5, 8$ for A_p and UA, 2nd and 3rd row show the trajectories of the states with the corresponding state estimates as black-crosses with 95% confidence intervals, 4th row illustrates the control inputs and the last row the changes in the sampling time.

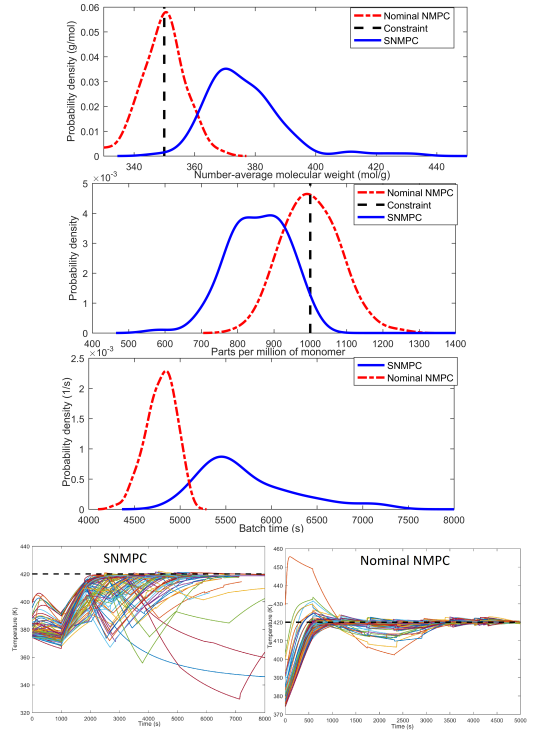


Fig. 3. Probability densities of NAMW, ppm of monomer, and batch time at final time and temperature trajectories of SNMPC and nominal NMPC based on 100 MC simulations

SNMPC had average batch times of 9300 seconds. The final graphs illustrate significantly better temperature control of the SNMPC approach compared to the nominal NMPC. In particular at the beginning ignoring the uncertainty of the overall heat transfer coefficient UA leads to constraint violations of up to 30K, while later on not accounting for the disturbance also leads to constraint violations for nominal NMPC method. The SNMPC method on the other hand shows close to no constraint violations.

VIII. CONCLUSIONS

In conclusion, we have presented a new algorithm for output feedback SNMPC that utilises a PCE nonlinear state estimator to approximate the probability distribution of states and uncertain parameters at each sampling time. This PCE representation is then exploited in a SNMPC formulation to account for both the initial value uncertainty using a sGH sampling rule and additive disturbance noise employing linearisation. Objectives and constraints were based on general nonlinear functions. A semi-batch reactor case study verified that the SNMPC framework is able to control the system with large initial uncertainties and additive disturbance noise. The PCE nonlinear state estimator is shown to able to accurately update the distribution of the states and uncertain parameters, while considering the uncertainty information from the distribution to avoid constraint violations. Furthermore, it was shown that ignoring the uncertainty informations leads to 50% constraint violations of the end-point constraints and large overshoots of the temperature path-constraint.

ACKNOWLEDGMENTS

Eric Bradford gratefully appreciates BASF for hosting his placement over the course of the Marie-Curie project and the helpful input and discussions for this work. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 675215.

REFERENCES

[1] Z. K. Nagy and R. D. Braatz, "Robust nonlinear model predictive control of batch processes," *AIChE Journal*, vol. 49, no. 7, pp. 1776–1786, 2003.

[2] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[3] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*. Springer, 1999, pp. 207–226.

[4] H. Chen, C. W. Scherer, and F. Allgower, "A game theoretic approach to nonlinear robust receding horizon control of constrained systems," in *Proceedings of the 1997 American Control Conference*, vol. 5. IEEE, 1997, pp. 3073–3077.

[5] D. Q. Mayne, E. C. Kerrigan, E. J. Van Wyk, and P. Falugi, "Tubebased robust nonlinear model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[6] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems*, vol. 36, no. 6, pp. 30–44, 2016.

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

[8] M. Cannon, B. Kouvaritakis, S. V. Rakovic, and Q. Cheng, "Stochastic tubes in model predictive control with probabilistic constraints," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 194–200, 2011.

[9] G. Schildbach, L. Fagiano, C. Frei, and M. Morari, "The scenario approach for stochastic model predictive control with bounds on closed-loop constraint violations," *Automatica*, vol. 50, no. 12, pp. 3009–3018, 2014.

[10] D. H. Van Hessem and O. H. Bosgra, "A conic reformulation of model predictive control including bounded and stochastic disturbances under state and input constraints," in *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 4. IEEE, 2002, pp. 4643–4648.

[11] E. Bradford and L. Imsland, "Economic Stochastic Model Predictive Control Using the Unscented Kalman Filter," *IFAC-PapersOnLine*, vol. 51, no. 18, pp. 417–422, 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405896318320196

[12] L. Fagiano and M. Khammash, "Nonlinear stochastic model predictive control via regularized polynomial chaos expansions," in *51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 142–147.

[13] E. Bradford and L. Imsland, "Expectation constrained stochastic nonlinear model predictive control of a batch bioreactor," *Computer Aided Chemical Engineering*, vol. 40, pp. 1621–1626, 2017.

[14] J. M. Maciejowski, A. L. Visintini, and J. Lygeros, "NMPC for complex stochastic systems using a Markov chain Monte Carlo approach," in *Assessment and Future Directions of Nonlinear Model Predictive Control*. Springer, 2007, pp. 269–281.

[15] E. Bradford and L. Imsland, "Stochastic Nonlinear Model Predictive Control Using Gaussian Processes," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 1027–1034.

[16] F. Weissel, M. F. Huber, and U. D. Hanebeck, "Stochastic nonlinear model predictive control based on Gaussian mixture approximations," in *Informatics in Control, Automation and Robotics*. Springer, 2009, pp. 239–252.

[17] E. A. Buehler, J. A. Paulson, and A. Mesbah, "Lyapunov-based stochastic nonlinear model predictive control: Shaping the state probability distribution functions," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 5389–5394.

[18] M. Cannon, D. Ng, and B. Kouvaritakis, "Successive linearization NMPC for a class of stochastic nonlinear systems," in *Nonlinear Model Predictive Control*. Springer, 2009, pp. 249–262.

[19] M. A. Sehr and R. R. Bitmead, "Particle model predictive control: Tractable stochastic nonlinear output-feedback MPC," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 15 361–15 366, 2017.

[20] P. Patrinos, P. Sopasakis, H. Sarimveis, and A. Bemporad, "Stochastic model predictive control for constrained discrete-time Markovian switching systems," *Automatica*, vol. 50, no. 10, pp. 2504–2514, 2014.

[21] T. Homer and P. Mhaskar, "Output-Feedback Lyapunov-Based Predictive Control of Stochastic Nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 571–577, 2018.

[22] A. Mesbah, S. Streif, R. Findeisen, and R. D. Braatz, "Stochastic nonlinear model predictive control with probabilistic constraints," in *2014 American Control Conference*. IEEE, 2014, pp. 2413–2419.

[23] S. Streif, M. Karl, and A. Mesbah, "Stochastic nonlinear model predictive control with efficient sample approximation of chance constraints," *arXiv preprint arXiv:1410.4535*, 2014.

[24] V. A. Bavdekar and A. Mesbah, "Stochastic nonlinear model predictive control with joint chance constraints," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 270–275, 2016.

[25] J. A. Paulson and A. Mesbah, "An efficient method for stochastic optimal control with joint chance constraints for nonlinear systems," *International Journal of Robust and Nonlinear Control*, 2017.

[26] J. Li and D. Xiu, "A generalized polynomial chaos based ensemble Kalman filter with high accuracy," *Journal of computational physics*, vol. 228, no. 15, pp. 5454–5469, 2009.

[27] E. D. Blanchard, A. Sandu, and C. Sandu, "A polynomial chaos-based Kalman filter approach for parameter estimation of mechanical systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 132, no. 6, p. 61404, 2010.

[28] P. Dutta and R. Bhattacharya, "Nonlinear estimation of hypersonic state trajectories in Bayesian framework with polynomial chaos," *Journal of guidance, control, and dynamics*, vol. 33, no. 6, pp. 1765–1778, 2010.

[29] ——, "Nonlinear estimation with polynomial chaos and higher order moment updates," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 3142–3147.

[30] R. Madankan, P. Singla, T. Singh, and P. D. Scott, "Polynomial-chaos-based Bayesian approach for state and parameter estimations," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 4, pp. 1058–1074, 2013.

[31] V. A. Bavdekar and A. Mesbah, "A polynomial chaos-based nonlinear Bayesian approach for estimating state and parameter probability distribution functions," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 2047–2052.

[32] R. Pandurangan, A. Chaudhuri, and S. Gupta, "The use of polynomial chaos for parameter identification from measurements in nonlinear dynamical systems," *ZAMMJournal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 95, no. 12, pp. 1372–1392, 2015.

[33] E. Bradford, M. Reble, A. Bouaswaig, and L. Imsland, "Economic stochastic nonlinear model predictive control of a semi-batch polymerization reaction," in *Dynamics and Control of Process Systems, including Biosystems - 12th DYCOPS*. IFAC, 2019, p. submitted.

[34] D. Xiu and G. E. Karniadakis, "Modeling uncertainty in flow simulations via generalized polynomial chaos," *Journal of computational physics*, vol. 187, no. 1, pp. 137–167, 2003.

[35] M. Eldred and J. Burkardt, "Comparison of Non-Intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Quantification," in *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, 2009, p. 976. [Online]. Available: http://arc.aiaa.org/doi/10.2514/6.2009-976

[36] T. Mühlpfordt, J. A. Paulson, R. D. Braatz, and R. Findeisen, "Output feedback model predictive control with probabilistic uncertainties for linear systems," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 2035–2040.

[37] M. Stein, "Large sample properties of simulations using Latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[38] J. H. Panchal, S. R. Kalidindi, and D. L. McDowell, "Key computational modeling issues in integrated computational materials engineering," *Computer-Aided Design*, vol. 45, no. 1, pp. 4–25, 2013.

[39] B. Jia, M. Xin, and Y. Cheng, "Sparse-grid quadrature nonlinear filtering," *Automatica*, vol. 48, no. 2, pp. 327–341, 2012.

[40] Y. Nie, L. T. Biegler, C. M. Villa, and J. M. Wassick, "Reactor modeling and recipe optimization of polyether polyol processes: Polypropylene glycol," *AIChE Journal*, vol. 59, no. 7, pp. 2515–2529, 2013.

[41] T. Y. Jung, Y. Nie, J. H. Lee, and L. T. Biegler, "Model-based online optimization framework for semi-batch polymerization reactors," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 164–169, 2015.

[42] H. Jang, J. H. Lee, and L. T. Biegler, "A robust NMPC scheme for semi-batch polymerization reactors," *IFAC-PapersOnLine*, vol. 49, no. 7, pp. 37–42, 2016.

[43] Y. Nie, L. T. Biegler, C. M. Villa, and J. M. Wassick, "Reactor modeling and recipe optimization of ring-opening polymerization: Block copolymers," *Industrial & Engineering Chemistry Research*, vol. 53, no. 18, pp. 7434–7446, 2013.

[44] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, pp. 1–36, 2018.

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

References

[1] V. O. Adesanya, M. P. Davey, S. A. Scott, and A. G. Smith. Kinetic modelling of growth and storage molecule production in microalgae under mixotrophic and autotrophic conditions. *Bioresour. Technol.*, 157:293–304, 2014.

[2] S. Aiba. Growth kinetics of photosynthetic microorganisms. In *Microbial reactions*, pages 85–156. Springer, 1982.

[3] S. Aksoy and R. M. Haralick. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern recognition letters*, 22(5):563–582, 2001.

[4] M. Alamir. On the use of supervised clustering in stochastic NMPC design. *arXiv preprint arXiv:1811.09069*, 2018.

[5] T. Alamo, R. Tempo, A. Luque, and D. R. Ramirez. Randomized methods for design of uncertain systems: Sample complexity and sequential algorithms. *Automatica*, 52:160–172, 2015.

[6] F. Allgöwer, R. Findeisen, and Z. K. Nagy. Nonlinear model predictive control: From theory to application. *J. Chin. Inst. Chem. Engrs*, 35(3):299–315, 2004.

[7] M. Alvarez, D. Luengo, and N. D. Lawrence. Latent force models. In *Artificial Intelligence and Statistics*, pages 9–16, 2009.

[8] J. Andersson. A general-purpose software framework for dynamic optimization. *Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10*, 2013.

[9] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, pages 1–36, 2018.

[10] L. E. Andersson, E. Bradford, and L. Imsland. Distributed learning for wind farm optimization with Gaussian processes. In *2020 American Control Conference (ACC)*, accepted, 2020.

[11] L. E. Andersson, E. Bradford, and L. Imsland. Gaussian processes modifier adaptation with uncertain inputs using distributed learning and optimization on a wind farm. In *IFAC 2020 - 21st IFAC World Congress*, submitted, 2020.

[12] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[13] A. Ashoori, B. Moshiri, A. Khaki-Sedigh, and M. R. Bakhtiari. Optimal control of a nonlinear fed-batch fermentation process using model predictive approach. *Journal of Process Control*, 19(7):1162–1173, 2009.

[14] A. Azarpour, T. N. G. Borhani, S. R. W. Alwi, Z. A. Manan, and M. I. A. Mutalib. A generic hybrid model development for process analysis of industrial fixed-bed catalytic reactors. *Chemical Engineering Research and Design*, 117:149–167, 2017.

[15] C. Baroi and A. K. Dalai. Review on Biodiesel Production from Various Feedstocks Using 12-Tungstophosphoric Acid (TPA) as a Solid Acid Catalyst Precursor. *Ind. Eng. Chem. Res.*, 53(49):18611–18624, dec 2014.

[16] V. A. Bavdekar and A. Mesbah. A polynomial chaos-based nonlinear Bayesian approach for estimating state and parameter probability distribution functions. In *American Control Conference (ACC), 2016*, pages 2047–2052. IEEE, 2016.

[17] V. A. Bavdekar and A. Mesbah. Stochastic model predictive control with integrated experiment design for nonlinear systems. *IFAC-PapersOnLine*, 49(7):49–54, 2016.

[18] V. A. Bavdekar and A. Mesbah. Stochastic nonlinear model predictive control with joint chance constraints. *IFAC-PapersOnLine*, 49(18):270–275, 2016.

[19] F. A. Bayer, M. Lorenzen, M. A. Müller, and F. Allgöwer. Robust economic Model Predictive Control using stochastic information. *Automatica*, 74:151–161, 2016.

[20] K. J. Beers and K. J. Beers. *Numerical methods for chemical engineering: applications in Matlab*. Cambridge University Press, 2007.

[21] A. Bemporad. Reducing conservativeness in predictive control of constrained systems with disturbances. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)*, volume 2, pages 1384–1389. IEEE, 1998.

[22] A. Bemporad and M. Morari. Robust model predictive control: A survey. In *Robustness in identification and control*, pages 207–226. Springer, 1999.

[23] F. Berkenkamp and A. P. Schoellig. Safe and robust learning control with Gaussian processes. In *2015 European Control Conference (ECC)*, pages 2496–2501. IEEE, 2015.

[24] D. Bernardini and A. Bemporad. Scenario-based model predictive control of stochastic constrained linear systems. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 6333–6338. IEEE, 2009.

[25] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 2. Athena Scientific, 3 edition, 2011.

[26] N. Bhutani, G. P. Rangaiah, and A. K. Ray. First-principles, data-based, and hybrid modeling and optimization of an industrial hydrocracking unit. *Industrial & engineering chemistry research*, 45(23):7807–7816, 2006.

[27] L. T. Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, volume 10. Siam, 2010.

[28] L. T. Biegler and J. B. Rawlings. Optimization approaches to nonlinear model predictive control. Technical report, Argonne National Lab., IL (USA), 1991.

[29] D. Bonvin, B. Srinivasan, and D. Hunkeler. Control and optimization of batch processes. *IEEE Control Systems Magazine*, 26(6):34–45, 2006.

[30] A. Bonzanini, T. Santos, and A. Mesbah. Tube-based stochastic nonlinear model predictive control: A comparative study on constraint tightening. *IFAC-PapersOnLine*, 52:598–603, 2019.

[31] E. Bradford and L. Imsland. Expectation constrained stochastic nonlinear model predictive control of a batch bioreactor. *Computer Aided Chemical Engineering*, 40:1621–1626, 2017.

[32] E. Bradford and L. Imsland. Stochastic Nonlinear Model Predictive Control with State Estimation by Incorporation of the Unscented Kalman Filter. *arXiv preprint arXiv:1709.01201*, 2017.

[33] E. Bradford and L. Imsland. Economic Stochastic Model Predictive Control Using the Unscented Kalman Filter. *IFAC-PapersOnLine*, 51(18):417–422, 2018.

[34] E. Bradford and L. Imsland. Economic Stochastic Model Predictive Control Using the Unscented Kalman Filter. *IFAC-PapersOnLine*, 51(18):417–422, 2018.

[35] E. Bradford and L. Imsland. Stochastic NMPC of Batch Processes Using Parameterized Control Policies. *Computer Aided Chemical Engineering*, 44:625–630, 2018.

[36] E. Bradford and L. Imsland. Stochastic Nonlinear Model Predictive Control Using Gaussian Processes. In *2018 European Control Conference (ECC)*, pages 1027–1034, 2018.

[37] E. Bradford and L. Imsland. Output feedback stochastic nonlinear model predictive control for batch processes. *Computers & Chemical Engineering*, 126:434–450, 2019.

[38] E. Bradford and L. Imsland. Stochastic nonlinear model predictive control of a batch fermentation process. *Computer Aided Chemical Engineering*, 46:1237–1242, 2019.

[39] E. Bradford and L. Imsland. Combining Gaussian processes and polynomial chaos expansions for stochastic nonlinear model predictive control. *Journal of Process Control*, submitted, 2020.

[40] E. Bradford, L. Imsland, and E. A. del Rio-Chanona. Nonlinear model predictive control with explicit back-offs for Gaussian process state space models. In *58th Conference on decision and control (CDC)*, pages 4747–4754. IEEE, 2019.

[41] E. Bradford, L. Imsland, M. Reble, and E. A. del Rio-Chanona. Hybrid Gaussian process modelling applied to economic stochastic model predictive control of batch processes. In *Progress on Economic and Distributed Model Predictive Control and Applications*. Springer, submitted, 2020.

[42] E. Bradford, L. Imsland, D. Zhang, and E. A. del Rio-Chanona. Stochastic data-driven model predictive control using Gaussian processes. *Computers & Chemical Engineering*, accepted, 2020.

[43] E. Bradford, M. Reble, A. Bouaswaig, and L. Imsland. Economic stochastic nonlinear model predictive control of a semi-batch polymerization reaction. *IFAC-PapersOnLine*, 52(1):667–672, 2019.

[44] E. Bradford, M. Reble, A. Bouaswaig, and L. Imsland. Economic stochastic nonlinear model predictive control of a semi-batch polymerization reaction. In *Dynamics and Control of Process Systems, including Biosystems - 12th DYCOPS*, page accepted. IFAC, 2019.

[45] E. Bradford, M. Reble, and L. Imsland. Output feedback stochastic nonlinear model predictive control of a polymerization batch process. In *2019 18th European Control Conference (ECC)*, pages 3144–3151. IEEE, 2019.

[46] E. Bradford, A. M. Schweidtmann, and A. Lapkin. Efficient multiobjective optimization employing Gaussian processes, spectral sampling and a genetic algorithm. *Journal of Global Optimization*, 71(2):407–438, 2018.

[47] E. Bradford, A. M. Schweidtmann, D. Zhang, K. Jing, and E. A. del Rio-Chanona. Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate Gaussian processes. *Computers & Chemical Engineering*, 118:143–158, 2018.

[48] S. Brahim-Belhouari and A. Bermak. Gaussian process for nonstationary time series prediction. *Comput. Stat. Data Anal.*, 47(4):705–712, 2004.

[49] L. Brennan and P. Owende. Biofuels from microalgae-A review of technologies for production, processing, and extractions of biofuels and co-products. *Renew. Sustain. Energy Rev.*, 14(2):557–577, feb 2010.

[50] E. A. Buehler, J. A. Paulson, and A. Mesbah. Lyapunov-based stochastic nonlinear model predictive control: Shaping the state probability distribution functions. In *2016 American Control Conference (ACC)*, pages 5389–5394. IEEE, 2016.

[51] B. A. Calfa, I. E. Grossmann, A. Agarwal, S. J. Bury, and J. M. Wassick. Data-driven individual and joint chance-constrained optimization via kernel smoothing. *Computers & Chemical Engineering*, 78:51–69, 2015.

[52] E. F. Camacho and C. B. Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[53] P. J. Campo and M. Morari. Robust model predictive control. In *American Control Conference, 1987*, pages 1021–1026. IEEE, 1987.

[54] M. Cannon, Q. Cheng, B. Kouvaritakis, and S. V. Raković. Stochastic tube MPC with state estimation. *Automatica*, 48(3):536–541, 2012.

[55] M. Cannon, B. Kouvaritakis, and D. Ng. Probabilistic tubes in linear stochastic model predictive control. *Systems & Control Letters*, 58(10):747–753, 2009.

[56] M. Cannon, B. Kouvaritakis, S. V. Rakovic, and Q. Cheng. Stochastic tubes in model predictive control with probabilistic constraints. *IEEE Transactions on Automatic Control*, 56(1):194–200, 2010.

[57] M. Cannon, B. Kouvaritakis, S. V. Rakovic, and Q. Cheng. Stochastic tubes in model predictive control with probabilistic constraints. *IEEE Transactions on Automatic Control*, 56(1):194–200, 2011.

[58] M. Cannon, B. Kouvaritakis, and X. Wu. Probabilistic constrained MPC for multiplicative and additive stochastic uncertainty. *IEEE Transactions on Automatic Control*, 54(7):1626–1632, 2009.

[59] M. Cannon, D. Ng, and B. Kouvaritakis. Successive linearization NMPC for a class of stochastic nonlinear systems. In *Nonlinear Model Predictive Control*, pages 249–262. Springer, 2009.

[60] G. Cao, E. M.-K. Lai, and F. Alam. Gaussian process model predictive control of an unmanned quadrotor. *Journal of Intelligent & Robotic Systems*, 88(1):147–162, 2017.

[61] Y. Cao. A formulation of nonlinear model predictive control using automatic differentiation. *Journal of Process Control*, 15(8):851–858, 2005.

[62] B. Chachuat. Nonlinear and dynamic optimization: From theory to practice. Technical report, EPFL – École polytechnique fédérale de Lausanne, 2007.

[63] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998.

[64] H. Chen, C. W. Scherer, and F. Allgower. A game theoretic approach to nonlinear robust receding horizon control of constrained systems. In *Proceedings of the 1997 American Control Conference*, volume 5, pages 3073–3077. IEEE, 1997.

[65] G. Chowdhary, H. A. Kingravi, J. P. How, and P. A. Vela. Bayesian nonparametric adaptive control using gaussian processes. *IEEE transactions on neural networks and learning systems*, 26(3):537–550, 2015.

[66] W.-l. Chu. Biotechnological applications of microalgae. *Int. e-Journal Sci. Med. Educ.*, 6(126):24–37, 2012.

[67] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

[68] S. Conti, J. P. Gosling, J. E. Oakley, and A. O'Hagan. Gaussian process emulation of dynamic computer codes. *Biometrika*, 96(3):663–676, 2009.

[69] F. E. Curtis, A. Wachter, and V. M. Zavala. A sequential algorithm for solving nonlinear optimization problems with chance constraints. *SIAM Journal on Optimization*, 28(1):930–958, 2018.

[70] D. M. de la Penad, A. Bemporad, and T. Alamo. Stochastic programming applied to model predictive control. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 1361–1366. IEEE, 2005.

[71] M. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[72] M. P. Deisenroth. Efficient Reinforcement Learning using Gaussian Processes. *KIT Scientific Publishing*, 9, 2010.

[73] M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck. Analytic moment-based Gaussian process filtering. In *Proceedings of the 26th annual international conference on machine learning*, pages 225–232. ACM, 2009.

[74] E. A. del Rio-Chanona, X. Cong, E. Bradford, D. Zhang, and K. Jing. Review of advanced physical and data-driven models for dynamic bioprocess simulation: Case study of algae-bacteria consortium wastewater treatment. *Biotechnology and Bioengineering*, 116(2):342–353, 2018.

[75] E. A. del Rio-Chanona, P. Dechatiwongse, D. Zhang, G. C. Maitland, K. Hellgardt, H. Arellano-Garcia, and V. S. Vassiliadis. Optimal Operation Strategy for Biohydrogen Production. *Ind. Eng. Chem. Res.*, 54(24):6334–6343, jun 2015.

[76] E. A. del Rio-Chanona, P. Dechatiwongse, D. Zhang, G. C. Maitland, K. Hellgardt, H. Arellano-Garcia, and V. S. Vassiliadis. Optimal operation strategy for biohydrogen production. *Industrial & Engineering Chemistry Research*, 54(24):6334–6343, 2015.

[77] E. A. del Rio-Chanona, F. Fiorelli, D. Zhang, N. R. Ahmed, K. Jing, and N. Shah. An efficient model construction strategy to simulate microalgal lutein photo-production dynamic process. *Biotechnol. Bioeng.*, 114(11):2518–2527, nov 2017.

[78] E. A. del Rio-Chanona, J. E. A. Graciano, E. Bradford, and B. Chachuat. Modifier-Adaptation Schemes Employing Gaussian Processes and Trust Regions for Real-Time Optimization. *IFAC-PapersOnLine*, 52(1):52–57, 2019.

[79] E. A. del Rio-Chanona, E. Manirafasha, D. Zhang, Q. Yue, and K. Jing. Dynamic modeling and optimization of cyanobacterial C-phycocyanin production process by artificial neural network. *Algal Res.*, 13:7–15, jan 2016.

[80] E. A. del Rio-Chanona, N. rashid Ahmed, D. Zhang, Y. Lu, and K. Jing. Kinetic modeling and process analysis for Desmodesmus sp. lutein photo-production. *AIChE J.*, 63(7):2546–2554, jul 2017.

[81] E. A. del Rio-Chanona, D. Zhang, and V. S. Vassiliadis. Model-based real-time optimisation of a fed-batch cyanobacterial hydrogen production process using economic model predictive control strategy. *Chem. Eng. Sci.*, 142:289–298, mar 2016.

[82] E. A. del Rio-Chanona, J. Liu, J. L. Wagner, D. Zhang, Y. Meng, S. Xue, and N. Shah. Dynamic modeling of green algae cultivation in a photobioreactor for sustainable biodiesel production. *Biotechnology and bioengineering*, 115(2):359–370, 2018.

[83] P. Dutta and R. Bhattacharya. Nonlinear estimation with polynomial chaos and higher order moment updates. In *American Control Conference (ACC), 2010*, pages 3142–3147. IEEE, 2010.

[84] M. Ebden. Gaussian processes: A quick introduction. *arXiv Prepr. arXiv1505.02965*, 2015.

[85] M. Eldred and J. Burkardt. Comparison of Non-Intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Quantification. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, page 976. Aerospace Sciences Meetings, 2009.

[86] J. Fábregas, A. Otero, A. Maseda, and A. Domínguez. Two-stage cultures for the production of Astaxanthin from Haematococcus pluvialis. *J. Biotechnol.*, 89(1):65–71, jul 2001.

[87] L. Fagiano and M. Khammash. Nonlinear stochastic model predictive control via regularized polynomial chaos expansions. In *51st IEEE Conference on Decision and Control (CDC)*, pages 142–147. IEEE, 2012.

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

[89] M. Farrokhsiar and H. Najjaran. An unscented model predictive control approach to the formation control of nonholonomic mobile robots. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1576–1582. IEEE, 2012.

[90] Y. Feng, W. Barr, and W. F. Harper. Neural network processing of microbial fuel cell signals for the identification of chemicals present in water. *J. Environ. Manage.*, 120:84–92, 2013.

[91] E. Fernandez-Camacho and C. Bordons-Alba. *Model predictive control in the process industry*. Springer, 1995.

[92] R. Findeisen and F. Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control*, volume 11, pages 119–141. Technische Universiteit Eindhoven Veldhoven, 2002.

[93] T. F. Finkler, S. Lucia, M. B. Dogru, and S. Engell. Simple control scheme for batch time minimization of exothermic semibatch polymerizations. *Industrial & Engineering Chemistry Research*, 52(17):5906–5920, 2013.

[94] H. S. Fogler. *Elements of chemical reaction engineering*. Prentice-Hall International London;, 1999.

[95] A. I. J. Forrester and A. J. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1):50–79, 2009.

[96] G. François, B. Srinivasan, and D. Bonvin. Use of measurements for enforcing the necessary conditions of optimality in the presence of constraints and uncertainty. *Journal of Process Control*, 15(6):701–712, 2005.

[97] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen. Bayesian inference and learning in Gaussian process state-space models with particle MCMC. In *Advances in Neural Information Processing Systems*, pages 3156–3164, 2013.

[98] F. García-Camacho, L. López-Rosales, A. Sánchez-Mirón, E. Belarbi, Y. Chisti, and E. Molina-Grima. Artificial neural network modeling for predicting the growth of the microalga Karlodinium veneficum. *Algal Res.*, 14:58–64, mar 2016.

[99] A. Gelb. *Applied optimal estimation*. MIT press, 1974.

[100] A. Girard, C. E. Rasmussen, J. Q. Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs-application to multiple-step ahead time series forecasting. *Advances in neural information processing systems*, pages 545–552, 2003.

[101] G. C. Goodwin, J. Østergaard, D. E. Quevedo, and A. Feuer. A vector quantization approach to scenario generation for stochastic NMPC. In *Nonlinear Model Predictive Control*, pages 235–248. Springer, 2009.

[102] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski. Optimization over state feedback policies for robust control with constraints. *Automatica*, 42(4):523–533, 2006.

[103] A. Grancharova, J. Kocijan, and T. A. Johansen. Explicit stochastic nonlinear predictive control based on Gaussian process models. In *2007 European Control Conference (ECC)*, pages 2340–2347. IEEE, 2007.

[104] S. Gros. Dual-Mode Batch-to-Batch Optimization as a Markov Decision Process. *Industrial & Engineering Chemistry Research*, 58(30):13780–13791, 2019.

[105] S. Gros and M. Zanon. Data-driven economic NMPC using reinforcement learning. *IEEE Transactions on Automatic Control*, 2019.

[106] L. Grüne. Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems. *SIAM Journal on Control and Optimization*, 48(2):1206–1228, 2009.

[107] M. E. Gunay, F. Akpinar, Z. I. Onsan, and R. Yildirim. Investigation of water gas-shift activity of Pt-MOx-CeO2 Al2O3 using modular artificial neural networks. *Int. J. Hydrogen Energy*, 37(3):2094–2102, feb 2012.

[108] L. J. Halliwell. The Lognormal Random Multivariate. In *Casualty Actuarial Society E-Forum, Spring 2015*, 2015.

[109] T. Heine, M. Kawohl, and R. King. Robust model predictive control using the unscented transformation. In *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, pages 224–230. IEEE, 2006.

[110] T. A. N. Heirung and A. Mesbah. Stochastic Nonlinear Model Predictive Control with Active Model Discrimination: A Closed-Loop Fault Diagnosis Application. *IFAC-PapersOnLine*, 50(1):15934–15939, 2017.

[111] T. A. N. Heirung, J. A. Paulson, J. O'Leary, and A. Mesbah. Stochastic model predictive control—how does it work? *Computers & Chemical Engineering*, 114:158–170, 2018.

[112] M. W. Hermanto, R. D. Braatz, and M. Chiu. Integrated batch-to-batch and nonlinear model predictive control for polymorphic transformation in pharmaceutical crystallization. *AIChE journal*, 57(4):1008–1019, 2011.

[113] L. Hewing, A. Liniger, and M. N. Zeilinger. Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars. In *2018 European Control Conference (ECC)*, pages 1341–1348, 2018.

[114] L. Hewing and M. N. Zeilinger. Cautious Model Predictive Control using Gaussian Process Regression. *arXiv preprint arXiv:1705.10702*, 2017.

[115] D. M. Himmelblau. Accounts of Experiences in the Application of Artificial Neural Networks in Chemical Engineering. *Ind. Eng. Chem. Res.*, 47(16):5782–5796, aug 2008.

[116] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

[117] S.-H. Ho, Y. Xie, M.-C. Chan, C.-C. Liu, C.-Y. Chen, D.-J. Lee, C.-C. Huang, and J.-S. Chang. Effects of nitrogen source availability and bioreactor operating strategies on lutein production with Scenedesmus obliquus FSP-3. *Bioresour. Technol.*, 184:131–138, may 2015.

[118] P. Hokayem, E. Cinquemani, D. Chatterjee, F. Ramponi, and J. Lygeros. Stochastic receding horizon control with output feedback and bounded controls. *Automatica*, 48(1):77–88, 2012.

[119] T. Homem-de Mello and G. Bayraksan. Monte Carlo sampling-based methods for stochastic optimization. *Surveys in Operations Research and Management Science*, 19(1):56–85, 2014.

[120] T. Homer and P. Mhaskar. Output-Feedback Lyapunov-Based Predictive Control of Stochastic Nonlinear Systems. *IEEE Transactions on Automatic Control*, 63(2):571–577, 2018.

[121] M. A. Hosen, M. A. Hussain, and F. S. Mjalli. Control of polystyrene batch reactors using neural network based model predictive control (NNMPC): An experimental investigation. *Control Eng. Pract.*, 19(5):454–467, 2011.

[122] ICCA. The Global Chemical Industry: Catalyzing Growth and Addressing Our World's Sustainability Challenges. Technical report, International Council of Chemical Associations (ICCA), 2019.

[123] S. Iplikci. Support vector machines based generalized predictive control. *International Journal of Robust and Nonlinear Control: IFAC Affiliated Journal*, 16(17):843–862, 2006.

[124] T. S. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In *ISMB*, volume 99, pages 149–158, 1999.

[125] D. Jang, J. Yoo, C. Y. Son, H. J. Kim, and K. H. Johansson. Networked operation of a UAV using Gaussian process-based delay compensation and model predictive control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9216–9222. IEEE, 2019.

[126] H. Jang, J. H. Lee, and L. T. Biegler. A robust NMPC scheme for semi-batch polymerization reactors. *IFAC-PapersOnLine*, 49(7):37–42, 2016.

[127] B. Jia, M. Xin, and Y. Cheng. Sparse-grid quadrature nonlinear filtering. *Automatica*, 48(2):327–341, 2012.

[128] T. A. Johansen. Introduction to nonlinear model predictive control and moving horizon estimation. *Selected topics on constrained and nonlinear control*, 1:1–53, 2011.

[129] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001.

[130] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of global optimization*, 13(4):455–492, 1998.

[131] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[132] T. Y. Jung, Y. Nie, J. H. Lee, and L. T. Biegler. Model-based on-line optimization framework for semi-batch polymerization reactors. *IFAC-PapersOnLine*, 48(8):164–169, 2015.

[133] K. Kavsek-Biasizzo, I. Skrjanc, and D. Matko. Fuzzy predictive control of highly nonlinear pH process. *Computers & chemical engineering*, 21:S613–S618, 1997.

[134] S. S. a. Keerthi and E. G. Gilbert. Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations. *Journal of optimization theory and applications*, 57(2):265–293, 1988.

[135] M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

[136] P. Kersaudy, B. Sudret, N. Varsier, O. Picon, and J. Wiart. A new surrogate modeling technique combining Kriging and polynomial chaos expansions–Application to uncertainty analysis in computational dosimetry. *Journal of Computational Physics*, 286:103–117, 2015.

[137] T. Kim, W. Kim, S. Choi, and H. J. Kim. Path Tracking for a Skid-steer Vehicle using Model Predictive Control with On-line Sparse Gaussian Process. *IFAC-PapersOnLine*, 50(1):5755–5760, 2017.

[138] G. Kimaev and L. A. Ricardez-Sandoval. A comparison of efficient uncertainty quantification techniques for stochastic multiscale systems. *AIChE Journal*, 63(8):3361–3373, 2017.

[139] E. D. Klenske, M. N. Zeilinger, B. Schölkopf, and P. Hennig. Gaussian process-based predictive control for periodic error correction. *IEEE Transactions on Control Systems Technology*, 24(1):110–121, 2016.

[140] J. Ko and D. Fox. Learning GP-BayesFilters via Gaussian process latent variable models. *Autonomous Robots*, 30(1):3–23, 2011.

[141] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith. Dynamic systems identification with Gaussian processes. *Mathematical and Computer Modelling of Dynamical Systems*, 11(4):411–424, 2005.

[142] J. Kocijan and R. Murray-Smith. Nonlinear predictive control with a Gaussian process model. In *Switching and learning in feedback systems*, pages 185–200. Springer, 2005.

[143] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *American Control Conference, 2004. Proceedings of the 2004*, volume 3, pages 2214–2219. IEEE, 2004.

[144] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and B. Likar. Predictive control with Gaussian process models. In *EUROCON 2003. Computer as a Tool. The IEEE Region 8*, volume 1, pages 352–356. IEEE, 2003.

[145] J. Köhler, R. Soloperto, M. A. Müller, and F. Allgöwer. A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, 2019.

[146] R. W. Koller, L. A. Ricardez-Sandoval, and L. T. Biegler. Stochastic back-off algorithm for simultaneous design, control, and scheduling of multiproduct systems under uncertainty. *AIChE Journal*, 64(7):2379–2389, 2018.

[147] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause. Learning-based model predictive control for safe exploration and reinforcement learning. *arXiv preprint arXiv:1803.08287*, 2018.

[148] U. Konda, P. Singla, T. Singh, and P. D. Scott. State uncertainty propagation in the presence of parametric uncertainty and additive white noise. *Journal of Dynamic Systems, Measurement, and Control*, 133(5):51009, 2011.

[149] M. Korda, R. Gondhalekar, F. Oldewurtel, and C. N. Jones. Stochastic MPC framework for controlling the average constraint violation. *IEEE Transactions on Automatic Control*, 59(7):1706–1721, 2014.

[150] D. G. Krige. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139, 1951.

[151] T. T. Kristoffersen and C. Holden. Model predictive control and extended Kalman filter for a gas-liquid cylindrical cyclone. In *Control Technology and Applications (CCTA), 2017 IEEE Conference on*, pages 1248–1255. IEEE, 2017.

[152] M. Kuddus, P. Singh, G. Thomas, and A. Al-Hazimi. Recent developments in production and biotechnological applications of C-phycocyanin. *Biomed Res. Int.*, 2013:742859, jan 2013.

[153] N. D. Lawrence, G. Sanguinetti, and M. Rattray. Modelling transcriptional regulation using Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 785–792, 2007.

[154] J. H. Lee and N. L. Ricker. Extended Kalman filter based nonlinear model predictive control. *Industrial & Engineering Chemistry Research*, 33(6):1530–1541, 1994.

[155] J. H. Lee and Z. Yu. Worst-case formulations of model predictive control for systems with bounded parameters. *Automatica*, 33(5):763–781, 1997.

[156] K. S. Lee, I. Chin, H. J. Lee, and J. H. Lee. Model predictive control technique combined with iterative learning for batch processes. *AIChE Journal*, 45(10):2175–2187, 1999.

[157] S. H. Lee and W. Chen. A comparative study of uncertainty propagation methods for black-box-type problems. *Structural and Multidisciplinary Optimization*, 37(3):239, 2009.

[158] B. Likar and J. Kocijan. Predictive control of a gas–liquid separation plant based on a Gaussian process model. *Computers & chemical engineering*, 31(3):142–152, 2007.

[159] D. Limon, T. Alamo, D. M. Raimondo, D. M. De La Peña, J. M. Bravo, A. Ferramosca, and E. F. Camacho. Input-to-state stability: a unifying framework for robust model predictive control. In *Nonlinear model predictive control*, pages 1–26. Springer, 2009.

[160] C. Liu, A. Gray, C. Lee, J. K. Hedrick, and J. Pan. Nonlinear stochastic predictive control with unscented transformation for semi-autonomous vehicles. In *American Control Conference (ACC), 2014*, pages 5574–5579. IEEE, 2014.

[161] M. Lorenzen, F. Allgöwer, F. Dabbene, and R. Tempo. An improved constraint-tightening approach for stochastic MPC. In *2015 American Control Conference (ACC)*, pages 944–949. IEEE, 2015.

[162] S. Lucia. *Robust Multi-stage Nonlinear Model Predictive Control*. Citeseer, 2014.

[163] S. Lucia, J. A. E. Andersson, H. Brandt, M. Diehl, and S. Engell. Handling uncertainty in economic nonlinear model predictive control: A comparative case study. *Journal of Process Control*, 24(8):1247–1259, 2014.

[164] S. Lucia, T. Finkler, and S. Engell. Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9):1306–1319, 2013.

[165] J. M. Maciejowski. *Predictive control: with constraints*. Pearson education, 2002.

[166] J. M. Maciejowski, A. L. Visintini, and J. Lygeros. NMPC for complex stochastic systems using a Markov chain Monte Carlo approach. In *Assessment and Future Directions of Nonlinear Model Predictive Control*, pages 269–281. Springer, 2007.

[167] J. M. Maciejowski and X. Yang. Fault tolerant control using Gaussian processes and model predictive control. In *Control and Fault-Tolerant Systems (SysTol), 2013 Conference on*, pages 1–12. IEEE, 2013.

[168] R. Madankan, P. Singla, T. Singh, and P. D. Scott. Polynomial-chaos-based Bayesian approach for state and parameter estimations. *Journal of Guidance, Control, and Dynamics*, 36(4):1058–1074, 2013.

[169] M. Maiworm, D. Limon, J. M. Manzano, and R. Findeisen. Stability of gaussian process learning based output feedback model predictive control. *IFAC-PapersOnLine*, 51(20):455–461, 2018.

[170] A. Malek, L. C. Zullo, and P. Daoutidis. Modeling and Dynamic Optimization of Microalgae Cultivation in Outdoor Open Ponds. *Ind. Eng. Chem. Res.*, 55(12):3327–3337, mar 2016.

[171] D. L. Marruedo, T. Alamo, and E. F. Camacho. Input-to-state stable MPC for constrained discrete-time nonlinear systems with bounded additive uncertainties. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 4, pages 4619–4624. IEEE, 2002.

[172] T. M. Mata, A. A. Martins, and N. S. Caetano. Microalgae for biodiesel production and other applications: A review. *Renew. Sustain. Energy Rev.*, 14(1):217–232, jan 2010.

[173] D. Q. Mayne, E. C. Kerrigan, E. J. Van Wyk, and P. Falugi. Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 21(11):1341–1353, 2011.

[174] D. Q. Mayne, M. M. Seron, and S. V. Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[175] A. Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems*, 36(6):30–44, 2016.

[176] A. Mesbah, A. E. M. Huesman, H. J. M. Kramer, Z. K. Nagy, and P. M. J. Van den Hof. Real-time control of a semi-industrial fed-batch evaporative crystallizer using different direct optimization strategies. *AIChE journal*, 57(6):1557–1569, 2011.

[177] A. Mesbah, S. Streif, R. Findeisen, and R. D. Braatz. Stochastic nonlinear model predictive control with probabilistic constraints. In *2014 American Control Conference*, pages 2413–2419. IEEE, 2014.

[178] H. Michalska and D. Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on automatic control*, 38(11):1623–1633, 1993.

[179] M. S. Mohamed, J. S. Tan, R. Mohamad, M. N. Mokhtar, and A. B. Ariff. Comparative Analyses of Response Surface Methodology and Artificial Neural Network on Medium Optimization for Tetraselmis sp. FTC209 Grown under Mixotrophic Condition. *Sci. World J.*, 2013:1–14, 2013.

[180] J. Mohd Ali, M. A. Hussain, M. O. Tade, and J. Zhang. Artificial Intelligence techniques applied as estimator in chemical process systems - A literature survey. *Expert Syst. Appl.*, 42(14):5915–5931, 2015.

[181] P. E. Moraal and J. W. Grizzle. Observer design for nonlinear systems with discrete-time measurements. *IEEE Transactions on automatic control*, 40(3):395–404, 1995.

[182] T. Mühlpfordt, J. A. Paulson, R. D. Braatz, and R. Findeisen. Output feedback model predictive control with probabilistic uncertainties for linear systems. In *American Control Conference (ACC), 2016*, pages 2035–2040. IEEE, 2016.

[183] R. Murray-Smith, D. Sbarbaro, C. E. Rasmussen, and A. Girard. Adaptive, cautious, predictive control with Gaussian process priors. *IFAC Proceedings Volumes*, 36(16):1155–1160, 2003.

[184] Z. K. Nagy and R. D. Braatz. Robust nonlinear model predictive control of batch processes. *AIChE Journal*, 49(7):1776–1786, 2003.

[185] Z. K. Nagy and R. D. Braatz. Open-loop and closed-loop robust optimal control of batch processes using distributional and worst-case analysis. *Journal of process control*, 14(4):411–422, 2004.

[186] Z. K. Nagy and R. D. Braatz. Distributional uncertainty analysis using power series and polynomial chaos expansions. *Journal of Process Control*, 17(3):229–240, 2007.

[187] Z. K. Nagy, B. Mahn, R. Franke, and F. Allgöwer. Evaluation study of an efficient output feedback nonlinear model predictive control for temperature tracking in an industrial batch reactor. *Control Engineering Practice*, 15(7):839–850, 2007.

[188] Z. K. Nagy, B. Mahn, R. Franke, and F. Allgöwer. Real-time implementation of nonlinear model predictive control of batch processes in an industrial framework. In *Assessment and Future Directions of Nonlinear Model Predictive Control*, pages 465–472. Springer, 2007.

[189] N. Nasr, H. Hafez, M. H. El Naggar, and G. Nakhla. Application of artificial neural networks for modeling of biohydrogen production. *Int. J. Hydrogen Energy*, 38(8):3189–3195, 2013.

[190] M. Nayhouse, A. Tran, J. S.-I. Kwon, M. Crose, G. Orkoulas, and P. D. Christofides. Modeling and control of ibuprofen crystal growth and size distribution. *Chemical Engineering Science*, 134:414–422, 2015.

[191] R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[192] R. Nelofer, R. N. Ramanan, R. N. Z. R. A. Rahman, M. Basri, and A. B. Ariff. Comparison of the estimation capabilities of response surface methodology and artificial neural network for the optimization of recombinant lipase production by E. coli BL21. *J. Ind. Microbiol. Biotechnol.*, 39:243–254, 2012.

[193] Y. Nie, L. T. Biegler, C. M. Villa, and J. M. Wassick. Reactor modeling and recipe optimization of polyether polyol processes: Polypropylene glycol. *AIChE Journal*, 59(7):2515–2529, 2013.

[194] Y. Nie, L. T. Biegler, C. M. Villa, and J. M. Wassick. Reactor modeling and recipe optimization of ring-opening polymerization: Block copolymers. *Industrial & Engineering Chemistry Research*, 53(18):7434–7446, 2013.

[195] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[196] L. M. Ochoa-Estopier, M. Jobson, and R. Smith. Operational optimization of crude oil distillation systems using artificial neural networks. *Comput. Aided Chem. Eng.*, 30:982–986, 2012.

[197] C. Odabasi, M. E. Gunay, and R. Yildirim. Knowledge extraction for water gas shift reaction over noble metal catalysts from publications in the literature between 2002 and 2012. *Int. J. Hydrogen Energy*, 39(11):5733–5746, apr 2014.

[198] A. O'Hagan. Bayesian analysis of computer code outputs: a tutorial. *Reliability Engineering & System Safety*, 91(10):1290–1300, 2006.

[199] A. O'Hagan. Polynomial Chaos: A tutorial and critique from a statistician's perspective. *SIAM/ASA Journal on Uncertainty Quantification*, 20:1–20, 2013.

[200] A. O'Hagan and J. F. C. Kingman. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–42, 1978.

[201] F. Oldewurtel, C. N. Jones, and M. Morari. A tractable approximation of chance constrained stochastic MPC based on affine disturbance feedback. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4731–4736. IEEE, 2008.

[202] M. Ono. Joint chance-constrained model predictive control with probabilistic resolvability. In *American Control Conference (ACC), 2012*, pages 435–441. IEEE, 2012.

[203] N. E. Owen, P. Challenor, P. P. Menon, and S. Bennani. Comparison of surrogate-based uncertainty quantification methods for computationally expensive simulators. *SIAM/ASA Journal on Uncertainty Quantification*, 5(1):403–435, 2017.

[204] V. K. Pareek, M. P. Brungs, a.a Adesina, and R. Sharma. Artificial neural network modeling of a multiphase photodegradation system. *J. Photochem. Photobiol. A Chem.*, 149(1-3):139–146, 2002.

[205] P. Patrinos, P. Sopasakis, H. Sarimveis, and A. Bemporad. Stochastic model predictive control for constrained discrete-time Markovian switching systems. *Automatica*, 50(10):2504–2514, 2014.

[206] J. A. Paulson and A. Mesbah. Nonlinear model predictive control with explicit backoffs for stochastic systems under arbitrary uncertainty. *IFAC-PapersOnLine*, 51(20):523–534, 2018.

[207] J. A. Paulson and A. Mesbah. An efficient method for stochastic optimal control with joint chance constraints for nonlinear systems. *International Journal of Robust and Nonlinear Control*, 29(15):5017–5037, 2019.

[208] P. Petsagkourakis, I. Sandoval, E. Bradford, D. Zhang, and E. A. del Rio-Chanona. Reinforcement Learning for Batch Bioprocess Optimization. *Computers & Chemical Engineering*, 133:106649, 2019.

[209] P. Petsagkourakis, I. Sandoval, E. Bradford, D. Zhang, and E. A. del Rio-Chanona. Constrained Reinforcement Learning for Dynamic Optimization under Uncertainty. In *IFAC 2020 - 21st IFAC World Congress*, submitted, 2020.

[210] P. Petsagkourakis, I. O. Sandoval, E. Bradford, D. Zhang, and E. A. del Rio-Chanona. Reinforcement Learning for Batch-to-Batch Bioprocess Optimisation. *Computer Aided Chemical Engineering*, 46:919–924, 2019.

[211] S. Piche, B. Sayyar-Rodsari, D. Johnson, and M. Gerules. Nonlinear model predictive control using neural networks. *IEEE Control Systems Magazine*, 20(3):53–62, 2000.

[212] F. Pourboghrat, H. Pongpairoj, Z. Liu, F. Farid, F. Pourboghrat, and B. Aazhang. Dynamic neural Networks for modeling and control of nonlinear systems. *Intelligent Automation & Soft Computing*, 9(2):61–70, 2003.

[213] M. Prandini, S. Garatti, and J. Lygeros. A randomized approach to stochastic model predictive control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 7315–7320. IEEE, 2012.

[214] J. A. Primbs and C. H. Sung. Stochastic receding horizon control of constrained linear systems with state and control multiplicative noise. *IEEE transactions on Automatic Control*, 54(2):221–230, 2009.

[215] J. Prüher and O. Straka. Gaussian process quadrature moment transform. *IEEE Transactions on Automatic Control*, 63(9):2844–2854, 2017.

[216] D. C. Psichogios and L. H. Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992.

[217] J. Quinonero-Candela, A. Girard, and C. E. Rasmussen. Prediction at an uncertain input for Gaussian processes and relevance vector machines-application to multiple-step ahead time-series forecasting. Technical report, Technical University of Denmark (DTU), 2002.

[218] J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11(Jun):1865–1881, 2010.

[219] C. E. Rasmussen. Evaluation of Gaussian processes and other methods for non-linear regression. *University of Toronto*, PhD thesis, 1996.

[220] C. E. Rasmussen and C. K. Williams. Gaussian processes for machine learning. *The MIT Press*, 2006.

[221] S. Rasoulian and L. A. Ricardez-Sandoval. A robust nonlinear model predictive controller for a multiscale thin film deposition process. *Chemical Engineering Science*, 136:38–49, 2015.

[222] A. Raue, M. Schilling, J. Bachmann, A. Matteson, M. Schelke, D. Kaschek, S. Hug, C. Kreutz, B. D. Harms, F. J. Theis, et al. Lessons learned from quantitative dynamical modeling in systems biology. *PloS one*, 8(9):e74335, 2013.

[223] J. B. Rawlings and R. Amrit. Optimizing process economic performance using model predictive control. In *Nonlinear model predictive control*, pages 119–138. Springer, 2009.

[224] J. B. Rawlings and D. Q. Mayne. *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[225] V. Rostampour, P. M. Esfahani, and T. Keviczky. Stochastic nonlinear model predictive control of an uncertain batch polymerization reactor. *IFAC-PapersOnLine*, 48(23):540–545, 2015.

[226] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989.

[227] P. D. Sampson and P. Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417):108–119, 1992.

[228] S. Sarkka, M. A. Alvarez, and N. D. Lawrence. Gaussian process latent force models for learning and stochastic control of physical systems. *IEEE Transactions on Automatic Control*, 2018.

[229] S. Särkkä, J. Hartikainen, L. Svensson, and F. Sandblom. On the relation between Gaussian process quadratures and sigma-point methods. *arXiv preprint arXiv:1504.05994*, 2015.

[230] G. Schildbach, L. Fagiano, C. Frei, and M. Morari. The scenario approach for stochastic model predictive control with bounds on closed-loop constraint violations. *Automatica*, 50(12):3009–3018, 2014.

[231] R. Schobi, B. Sudret, and J. Wiart. Polynomial-chaos-based Kriging. *International Journal for Uncertainty Quantification*, 5(2), 2015.

[232] P. O. M. Scokaert and D. Q. Mayne. Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43(8):1136–1142, 1998.

[233] M. A. Sehr and R. R. Bitmead. Particle model predictive control: Tractable stochastic nonlinear output-feedback MPC. *IFAC-PapersOnLine*, 50(1):15361–15366, 2017.

[234] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[235] B. W. Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[236] D. Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[237] D. Simon. Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *IET Control Theory & Applications*, 4(8):1303–1318, 2010.

[238] I. M. Sobol. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and computers in simulation*, 55(1-3):271–280, 2001.

[239] R. Soloperto, M. A. Müller, S. Trimpe, and F. Allgöwer. Learning-Based Robust Model Predictive Control with State-Dependent Uncertainty. *IFAC-PapersOnLine*, 51(20):442–447, 2018.

[240] M. Stein. Large sample properties of simulations using Latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[241] V. Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[242] S. Streif, M. Karl, and A. Mesbah. Stochastic nonlinear model predictive control with efficient sample approximation of chance constraints. *arXiv preprint arXiv:1410.4535*, 2014.

[243] Z. Sun, T. Li, Z.-g. Zhou, and Y. Jiang. Microalgae as a Source of Lutein: Chemistry, Biosynthesis, and Carotenogenesis. *Microalgae Biotechnology*, pages 37–58, 2015.

[244] Z. Sun, S. J. Qin, A. Singhal, and L. Megan. Performance monitoring of model-predictive controllers via model residual assessment. *Journal of Process Control*, 23(4):473–482, 2013.

[245] S. Sundararajan and S. S. Keerthi. Predictive approaches for choosing hyperparameters in Gaussian processes. *Neural computation*, 13(5):1103–1118, 2001.

[246] B. Tamburic, F. W. Zemichael, G. C. Maitland, and K. Hellgardt. Parameters affecting the growth and hydrogen production of the green alga Chlamydomonas reinhardtii. *Int. J. Hydrogen Energy*, 36(13):7872–7876, jul 2011.

[247] A. P. Teixeira, N. Carinhas, J. M. L. Dias, P. Cruz, P. M. Alves, M. J. T. Carrondo, and R. Oliveira. Hybrid semi-parametric mathematical systems: Bridging the gap between systems biology and process engineering. *Journal of biotechnology*, 132(4):418–425, 2007.

[248] P. Terwiesch, M. Agarwal, and D. W. T. Rippin. Batch unit optimization with imperfect modelling: a survey. *Journal of Process Control*, 4(4):238–258, 1994.

[249] J. Umlauft, T. Beckers, and S. Hirche. Scenario-based Optimal Control for Gaussian Process State Space Models. In *2018 European Control Conference (ECC)*, pages 1386–1392. IEEE, 2018.

[250] J. Umlauft, T. Beckers, M. Kimmel, and S. Hirche. Feedback linearization using Gaussian processes. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5249–5255. IEEE, 2017.

[251] R. Urtasun, D. J. Fleet, and P. Fua. 3D people tracking with Gaussian process dynamical models. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 238–245. IEEE, 2006.

[252] J. Valappil and C. Georgakis. Nonlinear model predictive control of end-use properties in batch reactors. *AIChE Journal*, 48(9):2006–2021, 2002.

[253] R. Van Der Merwe and E. A. Wan. The square-root unscented Kalman filter for state and parameter-estimation. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 6, pages 3461–3464. IEEE, 2001.

[254] S. Vats and S. Negi. Use of artificial neural network (ANN) for the development of bioprocess using Pinus roxburghii fallen foliages for the release of polyphenols and reducing sugars. *Bioresour. Technol.*, 140:392–398, 2013.

[255] M. E. Villanueva, R. Quirynen, M. Diehl, B. Chachuat, and B. Houska. Robust MPC via min-max differential inequalities. *Automatica*, 77:311–321, 2017.

[256] A. L. Visintini, W. Glover, J. Lygeros, and J. Maciejowski. Monte Carlo optimization for conflict resolution in air traffic control. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):470–482, 2006.

[257] A. Völz and K. Graichen. Gradientenbasierte stochastische modellprädiktive Regelung unter Verwendung der Unscented-Transformation. *at-Automatisierungstechnik*, 64(8):658–670, 2016.

[258] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[259] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models. In *NIPS*, volume 18, page 3, 2005.

[260] Y. Wang, C. Ocampo-Martinez, and V. Puig. Stochastic model predictive control based on Gaussian processes applied to drinking water networks. *IET Control Theory & Applications*, 10(8):947–955, 2016.

[261] F. Weissel, M. F. Huber, and U. D. Hanebeck. Stochastic nonlinear model predictive control based on Gaussian mixture approximations. In *Informatics in Control, Automation and Robotics*, pages 239–252. Springer, 2009.

[262] G.-C. Wick. The evaluation of the collision matrix. *Physical review*, 80(2):268, 1950.

[263] C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In *Advances in neural information processing systems*, pages 514–520, 1996.

[264] Z. Wu, D. Rincon, and P. D. Christofides. Real-Time Adaptive Machine-Learning-Based Predictive Control of Nonlinear Processes. *Industrial & Engineering Chemistry Research*, jul 2019.

[265] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine learning-based predictive control of nonlinear processes. Part I: Theory. *AIChE Journal*, 65(11), 2019.

[266] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine-learning-based predictive control of nonlinear processes. Part II: Computational implementation. *AIChE Journal*, 65(11), 2019.

[267] X.-C. Xi, A.-N. Poo, and S.-K. Chou. Support vector regression model predictive control on a HVAC plant. *Control Engineering Practice*, 15(8):897–908, 2007.

[268] Y. Xie, S.-H. Ho, C.-N. N. Chen, C.-Y. Chen, I.-S. Ng, K.-J. Jing, J.-S. Chang, and Y. Lu. Phototrophic cultivation of a thermo-tolerant Desmodesmus sp. for lutein production: effects of nitrate concentration, light intensity and fed-batch operation. *Bioresour. Technol.*, 144:435–44, sep 2013.

[269] Z. Xiong and J. Zhang. Modelling and optimal control of fed-batch processes using a novel control affine feedforward neural network. *Neurocomputing*, 61(1-4):317–337, 2004.

[270] D. Xiu and G. E. Karniadakis. Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of computational physics*, 187(1):137–167, 2003.

[271] J. Yan and R. R. Bitmead. Incorporating state estimation into model predictive control and its application to network traffic control. *Automatica*, 41(4):595–604, 2005.

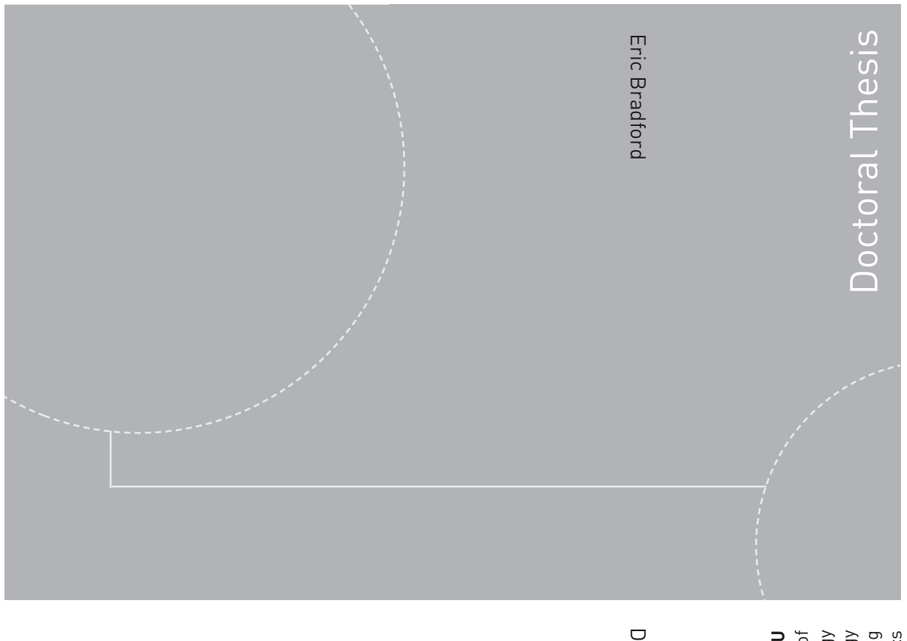
[272] X. Yang and J. Maciejowski. Risk-sensitive model predictive control with Gaussian process models. *IFAC-PapersOnLine*, 48(28):374–379, 2015.

[273] H.-W. Yen, C.-H. Sun, and T.-W. Ma. The Comparison of Lutein Production by Scenesdesmus sp. in the Autotrophic and the Mixotrophic Cultivation. *Appl. Biochem. Biotechnol.*, 164(3):353–361, jun 2011.

[274] D. Zhang, P. Dechatiwongse, E. A. Del-Rio-Chanona, K. Hellgardt, G. C. Maitland, and V. S. Vassiliadis. Analysis of the cyanobacterial hydrogen photoproduction process via model identification and process simulation. *Chem. Eng. Sci.*, 128:130–146, 2015.

[275] D. Zhang, N. Xiao, K. Mahbubani, E. del Rio-Chanona, N. Slater, and V. Vassiliadis. Bioprocess modelling of biohydrogen production by Rhodopseudomonas palustris: Model development and effects of operating conditions on hydrogen yield and glycerol conversion efficiency. *Chem. Eng. Sci.*, 130:68–78, jul 2015.

ISBN 978-82-326-4612-8 (printed version)
ISBN 978-82-326-4613-5 (electronic version)
ISSN 1503-8181



Doctoral theses at NTNU, 2020:132

Eric Bradford

Stochastic nonlinear model predictive control for chemical batch processes

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Doctoral theses at NTNU, 2020:132

NTNU

NTNU
Norwegian University of
Science and Technology