



Economic Nonlinear Model Predictive Control for Integrated and Optimized Non-Stationary Operation of Biotechnological Processes

Wahlgreen, Morten Ryberg

Publication date:
2023

Document Version
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):
Wahlgreen, M. R. (2023). *Economic Nonlinear Model Predictive Control for Integrated and Optimized Non-Stationary Operation of Biotechnological Processes*. Technical University of Denmark.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

 **DTU Compute**
Department of Applied Mathematics and Computer Science

Economic Nonlinear Model Predictive Control for Integrated and Optimized Non-Stationary Operation of Biotechnological Processes

Morten Wahlgreen Kaysfeld

Main supervisor:
John Bagterp Jørgensen

Co-supervisors:
Dimitri Boiroux, Allan Peter Engsig-Karup

Kongens Lyngby 2023



DTU Compute

**Department of Applied Mathematics and Computer Science
Technical University of Denmark**

Matematiktorvet

Building 303B

2800 Kongens Lyngby, Denmark

Phone +45 4525 3031

compute@compute.dtu.dk

www.compute.dtu.dk

Preface

This Ph.D. thesis was prepared at the department of Applied Mathematics and Computer Science (DTU Compute) at the Technical University of Denmark in partial fulfillment of the requirements for acquiring a Ph.D. degree in applied mathematics.

The work presented in this thesis was carried out under the supervision of Professor John Bagterp Jørgensen, (former) Associate Professor Dimitri Boiroux, and Associate Professor Allan Peter Engsig-Karup in the period from August 1st 2020 to July 31st 2023.

I was married on September 10th 2022 and changed name from Morten Ryberg Wahlgreen to Morten Wahlgreen Kaysfeld.

Kongens Lyngby, July 31, 2023

A handwritten signature in grey ink that reads "Morten Wahlgreen Kaysfeld". The signature is written in a cursive style with a large initial 'M'.

Morten Wahlgreen Kaysfeld

Acknowledgments

I would like to thank my supervisor Professor John Bagterp Jørgensen for his advice and supervision during this project. Also a thanks to my co-supervisors (former) Associate Professor Dimitri Boiroux and Associate Professor Allan Peter Engsig-Karup for interesting collaborations, and to Associate Professor Mario Zanon for hosting me at my external stay in Lucca, Italy. I would like to thank Asbjørn Thode Reenberg and Marcus Krogh Nielsen for great collaborations, support, and (almost) three years as office mates. Also a big thanks to my colleges at the Scientific Computing section at DTU Compute and in the research group for interesting discussions, coffee breaks, support, and making the everyday life special. A special thanks to my wife Kira for the constant support and positive inputs when times were hard during the project. Finally, I would like to thank my friends and family for their continued support and encouragement.

Summary

Biotechnological processes are used to produce many products in society, e.g., bioplastics, beer, vaccines, enzymes, and biopharmaceuticals. A promising biopharmaceutical is monoclonal antibodies (mAbs), which have been applied for treatment of various diseases including cancer, autoimmune disorders, and infectious diseases such as coronavirus disease 2019 (COVID-19). In 2017, the 6 top-selling biopharmaceuticals were mAbs. In 2022 the mAb market size was valued at 210.06 billion USD and the expected market size in 2030 is 494.52 billion USD.

The subject of this thesis is economic nonlinear model predictive control (ENMPC) for biotechnological processes and uncertainty quantification (UQ) of closed-loop systems. We develop an ENMPC algorithm for profit maximization of an mAb fermentation process and a high-performance Monte Carlo simulation toolbox for UQ of closed-loop systems. The purpose of this thesis is to demonstrate, by simulation, that ENMPC can increase the profit of the mAb fermentation process and that Monte Carlo simulation can quantify uncertainties and closed-loop performance for the process.

In this work, we develop 1) a high-performance Monte Carlo simulation toolbox for UQ of closed-loop systems, 2) a modeling methodology for processes consisting of reactive systems conducted in reactors, and 3) an ENMPC algorithm for profit maximization. In addition, we develop a target-tracking nonlinear model predictive control (NMPC) algorithm for testing purposes. This thesis introduces the Monte Carlo simulation toolbox, the modeling methodology, the NMPC algorithms, and a set of main numerical results. The Monte Carlo simulation toolbox applies Open Multi-Processing (OpenMP) for parallelization on shared memory architectures and shows almost linear parallel scaling for both proportional-integral-derivative (PID) controllers and the developed NMPC algorithms. The Monte Carlo simulation approach is applicable for quantification of uncertainties and closed-loop performance with respect to any selected key performance indicators (KPIs), which makes the approach versatile. The modeling methodology separates modeling of the reactor and the reactive system. In this way, the reactor equations are not required to be updated to change the considered process. We apply the modeling methodology to introduce three models in this thesis. The ENMPC algorithm consists of a continuous-discrete extended Kalman filter (CD-EKF) for state estimation and an economic regulator for profit maximization. We apply the ENMPC algorithm for profit maximization of an mAb fermentation process, where the economic regulator depends on mAb and glucose prices. The numerical results show that availability of Monte Carlo simulations enables closed-loop performance quantification with respect to selected KPIs and simplifies comparison between controllers. Closed-loop sim-

ulations with the ENMPC algorithm indicate that ENMPC can increase the profit of the mAb fermentation process. In addition, a set of operational insights summarizes the ENMPC operational strategy, which we apply to design a simple controller. Monte Carlo simulation quantifies the simple controller performance and show practically identical performance as the ENMPC algorithm. These results demonstrate the “*from-simple-via-complex-to-lucid*” approach, where the ENMPC technology enables design of a simple controller with practically identical performance.

This thesis consists of a summary report and a collection of seven research papers and a technical report. Six papers are published or accepted for publication in conference proceedings and one paper is submitted to Journal of Process Control. The technical report is not peer-reviewed.

Summary (danish)

Bioteknologiske processer bruges til at producere mange produkter i samfundet, f.eks. bioplast, øl, vacciner, enzymer og biofarmaceutiske midler. Et lovende biofarmaceutisk middel er monoklonale antistoffer (mAb'er), som er blevet anvendt til behandling af forskellige sygdomme, herunder kræft, autoimmune lidelser og infektionssygdomme som coronavirussygdom 2019 (COVID-19). I 2017 var de 6 bedst sælgende biofarmaceutiske midler mAb'er. I 2022 blev mAb-markedet værdisat til 210,06 milliarder amerikanske dollars, og den forventede markedsværdi i 2030 er 494,52 milliarder amerikanske dollars.

Denne afhandling omhandler økonomisk ikke-lineær model prædiktiv regulering (ENMPC) for bioteknologiske processer og usikkerhedskvantificering (UQ) af lukket-sløjfe systemer. Vi udvikler en ENMPC-algoritme til at maksimere profitten af en mAb-fermentationsproces og et høj-ydeevne Monte Carlo-simuleringsværktøj til UQ af lukket-sløjfe systemer. Formålet med denne afhandling er at demonstrere ved simulering, at ENMPC kan øge profitten af mAb-fermentationsprocessen og at Monte Carlo-simulation kan kvantificere usikkerheder og lukket-sløjfe-ydeevne for processen.

I dette arbejde udvikler vi 1) et høj-ydeevne Monte Carlo-simuleringsværktøj til UQ af lukket-sløjfe systemer, 2) en modelleringsmetodologi til processer bestående af reaktive systemer udført i reaktorer og 3) en ENMPC-algoritme til profitmaksimering. Derudover udvikler vi en referencesøgende NMPC-algoritme til testformål. Denne afhandling introducerer Monte Carlo simuleringsværktøjet, modelleringsmetodologien, NMPC-algoritmerne og en række numeriske hovedresultater. Monte Carlo simuleringsværktøjet anvender *Open Multi-Processing* (OpenMP) til parallelisering på delte hukommelsesarkitekturer og har næsten lineær parallel skalering både for *proportional-integral-derivative* (PID)-regulatorer og de udviklede NMPC-algoritmer. Monte Carlo simuleringsværktøjet kan anvendes til kvantificering af usikkerheder og lukket-sløjfe-ydeevne i forhold til valgte *key performance indicators* (KPI'er), hvilket gør tilgangen alsidig. Modelleringsmetodologien adskiller modelleringen af reaktoren og det reaktive system. På denne måde kræves der ikke opdatering af reaktorligningerne for at ændre den betragtede proces. Vi anvender modelleringsmetodologien til at introducere tre modeller i denne afhandling. ENMPC-algoritmen består af et kontinuert-diskret udvidet Kalman-filter (CD-EKF) til tilstandsestimering og en økonomisk regulator til profitmaksimering. Vi anvender ENMPC-algoritmen til profitmaksimering af en mAb-fermentationsproces, hvor den økonomiske regulator afhænger af mAb- og glukosepriser. De numeriske resultater viser, at tilgængeligheden af Monte Carlo-simuleringer muliggør kvantificering af lukket-sløjfe-ydeevne med hensyn til valgte KPI'er og forenkler sammenligningen mellem regulatorer. Lukket-sløjfe simuleringer med ENMPC-algoritmen antyder, at ENMPC

kan øge profitten af mAb-fermentationsprocessen. Derudover opsummerer en række operationelle indsigter ENMPC-operationsstrategien, som vi anvender til at designe en simpel regulator. Monte Carlo-simulering kvantificerer ydeevnen af den simple regulator og viser praktisk identisk ydeevne som ENMPC-algoritmen. Disse resultater demonstrerer tilgangen "*from-simple-via-complex-to-lucid*", hvor ENMPC-teknologien muliggør design af en simpel regulator med praktisk identisk ydeevne.

Denne afhandling består af en sammenfattende rapport og en samling af syv forskningsartikler samt en teknisk rapport. Seks artikler er blevet udgivet eller accepteret til udgivelse i konferenceprocedurer, og en artikel er indsendt til Journal of Process Control. Den tekniske rapport er ikke peer-reviewed.

List of abbreviations

BFGS	Broyden–Fletcher–Goldfarb–Shanno
BR	batch reactor
CAGR	compound annual growth rate
CD-EKF	continuous-discrete extended Kalman filter
COVID-19	coronavirus disease 2019
CPR	continuous perfusion reactor
CPU	central processing unit
CSTR	continuous stirred tank reactor
DOP	dynamic optimization problem
EKF	extended Kalman filter
ELISA	enzyme-linked immunosorbent assay
EMPC	economic model predictive control
EnKF	ensemble Kalman filter
ENMPC	economic nonlinear model predictive control
EUA	emergency use authorization
FBR	fed-batch reactor
IP	interior-point
IVP	initial value problem
KPI	key performance indicator
LS	least squares

mAb	monoclonal antibody
MHC	moving horizon control
MHE	moving horizon estimation
MPC	model predictive control
MPI	Message Passing Interface
NLP	nonlinear programming problem
NMPC	nonlinear model predictive control
NUMA	non-uniform memory access
OCP	optimal control problem
ODE	ordinary differential equation
OpenMP	Open Multi-Processing
P	proportional
PDE	partial differential equation
PFR	plug-flow reactor
PI	proportional-integral
PID	proportional-integral-derivative
QP	quadratic programming problem
RHC	receding horizon control
SARS-CoV-2	severe acute respiratory syndrome coronavirus 2
SDE	stochastic differential equation
SQP	sequential quadratic programming
UKF	unscented Kalman filter
UQ	uncertainty quantification
US FDA	United States Food and Drug Administration
ZOH	zero-order-hold

Contents

Preface	i
Acknowledgments	iii
Summary	v
Summary (danish)	vii
List of abbreviations	ix
Contents	xi
I Summary Report	1
1 Introduction	3
1.1 Motivation	3
1.2 Literature review	5
1.3 Objectives and contributions	9
1.4 Outline of the thesis	11
1.5 List of publications	12
2 Monte Carlo simulation based uncertainty quantification	15
2.1 Procedure for closed-loop simulation	16
2.2 Procedure for Monte Carlo simulation	18
2.3 Summary	19
3 Systematic process modeling	21
3.1 Modeling methodology	21
3.2 Biomass fermentation process conducted in a fed-batch reactor	25
3.3 Exothermic chemical reaction conducted in an adiabatic continuous stirred tank reactor	27
3.4 Monoclonal antibody fermentation process conducted in a continuous per- fusion reactor	30
3.5 Summary	34
4 Nonlinear model predictive control	37
4.1 Estimator	37

4.2	Regulator	39
4.3	Summary	43
5	Main results	45
5.1	Uncertainty quantification for biomass production	45
5.2	Uncertainty quantification of NMPC algorithm for a chemical reaction . .	49
5.3	Uncertainty quantification of ENMPC algorithm for mAb production . .	50
5.4	Summary	55
6	Conclusions	57
6.1	Suggestions for future work	58
	Bibliography	59
 II Appendix		 73
A	Paper I: CDC 2021	75
	A High-Performance Monte Carlo Simulation Toolbox for Uncertainty Quantification of Closed-loop Systems	75
B	Paper II: DYCOPS 2022	85
	On the Implementation of a Preconditioned Riccati Recursion based Primal-Dual Interior-Point Algorithm for Input Constrained Optimal Control Problems	85
C	Paper III: DYCOPS 2022	93
	Modeling and Simulation of Upstream and Downstream Processes for Monoclonal Antibody Production	93
D	Paper IV: FOCAPO/CPC 2023	101
	Model Predictive Control Tuning by Monte Carlo Simulation and Controller Matching	101
E	Paper V: ECC 2023	109
	Performance Quantification of a Nonlinear Model Predictive Controller by Parallel Monte Carlo Simulations of a Closed-loop System	109
F	Paper VI: IFAC WC 2023	117
	Dynamic Optimization for Monoclonal Antibody Production	117
G	Paper VII: JPC 2023	125
	Uncertainty Quantification of an Economic Nonlinear Model Predictive Controller for Monoclonal Antibody Production	125
H	Technical Report	143
	Numerical optimization packages for optimal control: QPIPM and NLPSQP .	143

Part I

Summary Report

CHAPTER 1

Introduction

The subject of this thesis is economic nonlinear model predictive control (ENMPC) for optimized operation of biotechnological processes and uncertainty quantification (UQ) of closed-loop systems. Model predictive control (MPC) applies the moving horizon optimization principle, where optimal open-loop control strategies are obtained as solution to optimal control problems (OCPs). We implement the first part of the open-loop control strategy in the process and solve a new OCP each time new measurements are available. This forms a closed-loop system. Closed-loop systems involving model-based controllers have many sources of uncertainty, e.g., process noise, measurement noise, disturbances, and uncertainty related to plant-model mismatch. The main motivation for this project is to develop advanced control technology for closed-loop operation of biotechnological processes and an UQ tool for quantification of uncertainties in closed-loop systems. In this work, we apply large-scale high-performance Monte Carlo simulation for UQ of stochastic closed-loop systems and develop an ENMPC algorithm for profit maximization. We design the ENMPC algorithm for a monoclonal antibody (mAb) fermentation process, which constitutes a relevant biotechnological process in society.

The remaining parts of this chapter are structured as follows. Section 1.1 provides a motivation for research within optimization of mAb production and development of both ENMPC technology and UQ tools for mAb production in closed-loop. In Section 1.2, we give a review of relevant literature related to mAb production, MPC technology, and numerical optimization. Then, we introduce the objectives and contributions of this thesis in Section 1.3. Section 1.4 presents the structure of the remaining parts of this thesis. Finally, Section 1.5 provides a list of publications included in this thesis and a list of relevant publications not included in this thesis.

1.1 Motivation

Biotechnological processes are essential for production of a variety of products in society, e.g., bio-plastics, beer, vaccines, enzymes, and biopharmaceuticals. Biopharmaceuticals are drugs produced using biological sources. An example of a promising biopharmaceutical is mAbs. MAbs are laboratory-produced proteins designed to recognize and target specific harmful substances such as viruses, cancer cells, or other abnormal cells. The first mAb was developed in 1975 and the first therapeutic mAb, muromonab-CD3 (Orthoclone OKT3), was fully licensed and approved by the United States Food and Drug Administration (US FDA) in 1986 (Köhler and Milstein, 1975; Sommerfeld and

Strube, 2005; Liu, 2014; Ecker et al., 2015; Lu et al., 2020). Research and development related to mAb production have increased drastically since discovery of the first mAb in 1975. Figure 1.1 shows the evolution of the US FDA approved mAbs in the period 1975–2019, which shows that the US FDA had approved 80 therapeutic mAbs by 2019. As a result, the mAb market size has experienced an exponential growth in the past years. Figure 1.2 illustrates the mAb market size growth in the period 1975–2019. In 2019, the mAb market size was valued at 150 billion USD (Lu et al., 2020). Already in 2022, the market size had increased to 210.06 billion USD and current projections indicate a compound annual growth rate (CAGR) of 11.04% from 2023 to 2030 resulting in an expected market size of 494.53 billion USD in 2030 (Grand View Research, 2023).

In conclusion, mAbs are biopharmaceuticals with a market size in exponential growth. MAbs have shown promising properties to target harmful substances and have been applied as treatment for a number of diseases including cancer. However, production of mAbs is both time-consuming and costly, which can ultimately lead to very expensive treatments (Hernandez et al., 2018). The increasing market size and costly production motivate research within optimization and control of mAb production processes. Development of ENMPC algorithms allows for optimization related to economic objectives in closed-loop. Therefore, ENMPC might offer opportunities to increase profit and availability of mAb treatments. However, quantification of uncertainties and closed-loop performance can be cumbersome when applying advanced control technology such as ENMPC. This motivates the development of a versatile UQ tool for closed-loop systems, which might offer valuable performance insights prior to experiments and thereby possibly reduce the production cost of mAbs.

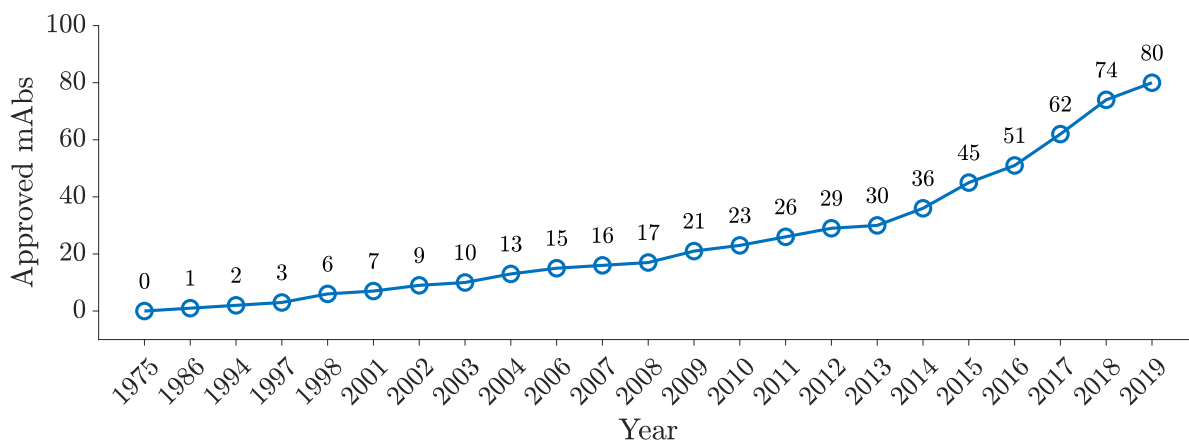


Figure 1.1: Number of approved mAbs by the US FDA from 1975–2019. This figure is based on data from Table 1 in (Lu et al., 2020).

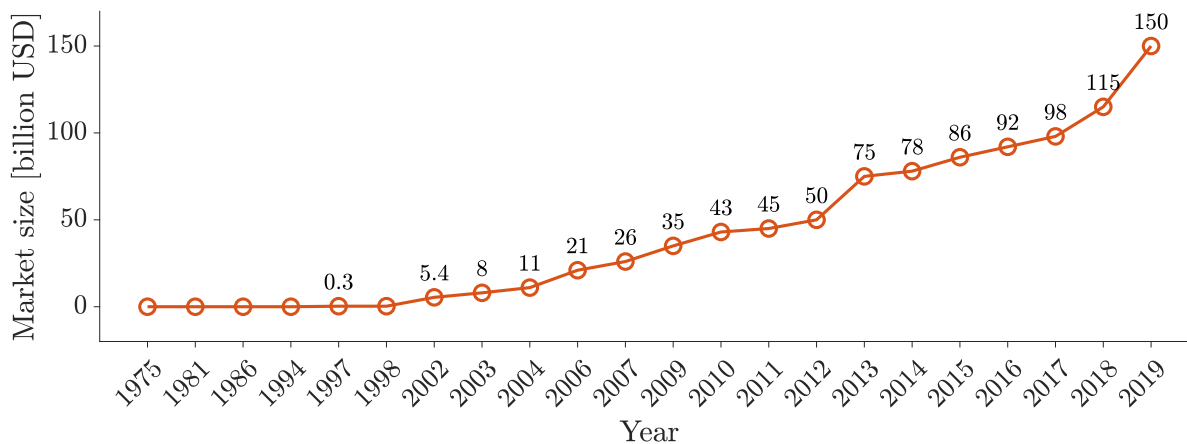


Figure 1.2: Evolution of mAb market size from 1975–2019. This figure is adapted from Fig. 1 in (Lu et al., 2020).

1.2 Literature review

In this section, we provide a review of literature relevant to this thesis. We cover literature related to 1) mAb production, 2) MPC technology, and 3) numerical optimization.

Monoclonal antibodies. As previously mentioned, the first mAb was discovered in 1975 (Köhler and Milstein, 1975). In 1986 the first therapeutic mAb, muromonab-CD3 (Orthoclone OKT3), was approved by the US FDA and the market size for mAbs has grown exponentially since (Sommerfeld and Strube, 2005; Liu, 2014; Ecker et al., 2015; Lu et al., 2020). As of 2022, the market size was 210.06 billion USD and it is expected to increase to 494.53 billion USD by 2030 (Grand View Research, 2023). MAbs are proteins designed to target harmful substances giving them significant relevance in medical applications. In 2017, the 6 top-selling biopharmaceuticals were mAbs and in 2021 mAbs represented more than half of the 20 top-selling biopharmaceuticals (Walsh, 2018; Walsh and Walsh, 2022). They have shown great effect against a number of diseases including cancer, autoimmune disorders, and infectious diseases (Carter, 2001; Melero et al., 2007; Hafeez et al., 2018; Otsubo and Yasui, 2022). MAbs were repeatedly proposed as treatments for coronavirus disease 2019 (COVID-19) in the recent global pandemic of COVID-19 caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) (Jahanshahlu and Rezaei, 2020; Marovich et al., 2020; Wang et al., 2020; Zhou et al., 2020). During the pandemic, the US FDA issued multiple emergency use authorizations (EUAs) for mAb treatments of COVID-19, e.g., for casirivimab and imdevimab the 21st of November 2020 and for bamlanivimab and etesevimab the 9th of February 2021 (U.S. Food and Drug Administration (FDA), 2020, 2021). Treatments involving mAbs are usually very expensive and a study from 2018 showed that among 107 unique mAb-indication combinations an average annual price of mAb treatments was 96,731 USD (Hernandez et al., 2018). The study also showed that the average price of a cancer drug has doubled in the last decade.

Production of mAbs is a complex process involving several steps, 1) *immunization*, where animal injection of the antigen of interest triggers the animals immune system to form antibody producing B cells, 2) *cell fusion*, where hybridoma cells are formed by fusing the antibody producing B cells with myeloma cells, 3) *screening*, where the screening technique enzyme-linked immunosorbent assay (ELISA) is applied to identify hybridoma cells that produce the desired antibody, 4) *cloning*, where the antibody producing hybridoma cells are grown in a bioreactor to increase the number of identical cells and produce mAbs, and 5) *mAb purification*, where chromatography techniques are applied to separate the produced mAbs from impurities in the production medium (Schofield, 2023). The process of growing cells in a bioreactor is an upstream biotechnological fermentation process, whereas the purification steps are downstream processes. This work focuses on the upstream fermentation process for production of mAbs in bioreactors. The bioreactor is a controlled environment, where the growth of hybridoma cells can be monitored and factors such as temperature, pH, dissolved oxygen concentration, and available nutrients can be controlled to improve the growth conditions during the fermentation (Erickson, 2009).

Mechanistic modeling of biotechnological processes is a useful tool that provides a deeper understanding of the fermentation process. The models are also essential for model-based optimization and new models for simulation and optimization are continuously developed (Bailey, 1998; Provost and Bastin, 2004; Liu et al., 2008; Craven et al., 2012; Glen et al., 2018; Kyriakopoulos et al., 2018; Sha et al., 2018; Badr et al., 2021; Kumar et al., 2022). Models vary in complexity and many different factors can be incorporated, e.g., the impact of glucose concentration, lactate concentration, and temperature (Fan et al., 2015; Sissolak et al., 2019). The cells often consume glucose as their main source of carbon and energy (Fan et al., 2015), where the glucose consuming metabolism produces lactate as a bi-product (Li et al., 2012; Kumar et al., 2022). The concentration of both glucose and lactate in the bioreactor impact the growth rate of cells and their productivity (Li et al., 2012; Dean and Reddy, 2013; Pereira et al., 2018; Vergara et al., 2018). There exist many methods to optimize mAb fermentation processes, e.g., exploring new cell culture systems, optimizing bioreactor designs and operation, and developing advanced modeling and control strategies (Fong et al., 1997; Butler, 2005; Li et al., 2010; Hong et al., 2018).

Model predictive control. The development of MPC (also referred to as moving horizon control (MHC) or receding horizon control (RHC)) traces back to the 1970s and 1980s (Garcia and Prett, 1986; Morari et al., 1988). MPC is a closed-loop feedback strategy, where optimal open-loop control strategies are repeatedly computed based on available feedback (Rawlings et al., 2017). The MPC algorithm implements the first part of the optimal open-loop strategy in the process and recomputes a new open-loop strategy when new feedback is available. Applications with MPC span many fields and MPC is widely applied in the industry (Qin and Badgwell, 2003; Forbes et al., 2015; Honc et al., 2016). In the late 1980s and early 1990s, the concept of nonlinear model predictive control (NMPC) appeared (Eaton et al., 1988; Rawlings et al., 1994; Biegler, 1998; Rawlings, 2000; Allgöwer et al., 2004). NMPC directly handles nonlinearities and

in turn provides a powerful technology for nonlinear systems. More recently, the new concept of economic model predictive control (EMPC) (or ENMPC specifically for nonlinear systems) appeared (Rawlings and Amrit, 2009; Amrit et al., 2011; Diehl et al., 2011; Angeli et al., 2012; Heidarinejad et al., 2012; Rawlings et al., 2012; Ellis et al., 2014). In EMPC algorithms, economic factors, such as profit maximization, are directly incorporated into the objective function rather than usual steady-state tracking objectives. Applications involving different MPC technologies span many fields of research. We provide a few examples from the process control literature (Diehl et al., 2002; Nagy and Braatz, 2003; Aehle et al., 2012; Craven et al., 2014; Huyck et al., 2014; Dewasme et al., 2015; Drejer et al., 2017; Ritschel et al., 2019).

MPC algorithms solve an OCP to achieve the optimal open-loop control strategy. The OCP involves dynamic state constraints for the considered system, which form an initial value problem (IVP). The initial condition for the IVP is the current vector of state variables. However, for most practical purposes, full state information is not available, since it is either not possible or not feasible to measure all states. In this case, state estimation techniques can estimate the states of the system based on the available measurements. For linear systems, the Kalman filter is an optimal state estimator (Kalman, 1960). A direct extension to nonlinear systems is the extended Kalman filter (EKF), which applies the original Kalman filter algorithm to a local linearization of the nonlinear system (Jazwinski, 1970). The EKF is a computationally efficient algorithm, which in many applications show great estimation properties (Schneider and Georgakis, 2013; Frogerais et al., 2012). However, the EKF may struggle in highly nonlinear systems due to linearization. There exist several other filtering methods, which can have better estimation properties in the case of high nonlinearity, e.g., the unscented Kalman filter (UKF) and the ensemble Kalman filter (EnKF) (Julier and Uhlmann, 2004; Gillijns et al., 2006; Nielsen et al., 2023). Alternatively, there exist optimization based methods like moving horizon estimation (MHE), where the estimation problem is an dynamic optimization problem (DOP) (Alessandri et al., 2010).

Numerical optimization and dynamic optimization problems. Numerical optimization is a broad and well-developed area of research (Fletcher, 1987; Gill et al., 1999; Nocedal and Wright, 1999). There exist various types of optimization problems and methods to solve them (Biegler and Grossmann, 2004). In this review, we restrict ourselves to quadratic programming problems (QPs) and nonlinear programming problems (NLPs), which are relevant for this thesis. QPs are a class of mathematical optimization problems that involve minimization of a quadratic objective function subject to linear equality constraints and inequality constraints. There exist different methods to solve QPs, where interior-point (IP) methods and active-set methods are widely applied (Goldfarb and Idnani, 1983; Powell, 1985; Mehrotra, 1992; Bartlett and Biegler, 2006). In IP methods, a sequence of barrier problems are solved to move towards the optimal solution while staying within the feasible region. Active-set methods update a so-called active-set that forms a smaller equality constrained QP to be solved in each iteration until the optimal solution is reached. NLPs are a broader class of mathematical optimization problems involving a nonlinear objective function subject to nonlinear equality

constraints and inequality constraints. Popular methods to solve NLPs are IP methods and sequential quadratic programming (SQP) methods (Gould and Toint, 2000; Forsgren et al., 2002; Wright, 2004; D’Apuzzo et al., 2010; Gondzio, 2012). SQP methods solve a QP-subproblem to obtain a search direction in each iteration. Stepping in the search direction advances towards the optimal solution. Many descriptions of the mentioned algorithms exist in the literature (Nocedal and Wright, 1999; Potra and Wright, 2000; Gill and Wong, 2012).

DOPs are special types of optimization problems involving dynamic constraints. OCPs arising in MPC applications are a special type of DOPs, where the dynamic constraints describe the evolution of state variables in time. The solution to an OCP is the control action (inputs) that minimizes some objective function subject to the dynamic constraints and possibly other process constraints, which makes OCPs infinite dimensional. Methods to solve OCPs are divided into two major categories namely direct and indirect methods. Application of indirect methods lead to a multiple-point boundary-value problem (Rao, 2009). In direct methods, which we consider in this thesis, the DOP is transcribed to an NLP by some discretization of the problem. The direct single-shooting method, direct multiple-shooting method, and direct collocation method (referred to without *direct* from now on) are popular choices to discretize OCPs (Bock and Plitt, 1984; Binder et al., 2001; Biegler, 2007; Schäfer et al., 2007; Diehl et al., 2009; Rao, 2009). We apply the multiple-shooting method in this thesis. The multiple-shooting method involves numerical solution of the state dynamics, for which Runge-Kutta schemes are popular choices (Butcher, 2000). The time interval is split into subintervals and the states at the beginning of each subinterval are decision variables. In each subinterval, an IVP is solved and continuity of the solution is enforced with a new set of equality constraints in the resulting NLP. The cost function is integrated with a quadrature rule consistent with the integration scheme in each subinterval (Rao, 2009). Single-shooting methods result in small and dense NLPs, while multiple-shooting and collocation methods result in large and sparse NLPs. Appropriate sparse NLP solvers make it computational tractable to apply both multiple-shooting and collocation methods even though they often result in very large optimization problems with many decision variables (Rao, 2009). An example of such a solver is IPOPT, which has also been interfaced with the symbolic automatic differentiation software CasADi (Wächter and Biegler, 2006; Andersson et al., 2019). In particular, multiple-shooting methods result in structured sparse NLPs. When solved with SQP algorithms, the QP-subproblem inherits a specific structure, where a Riccati recursion based QP solver can be utilized (Rao et al., 1998; Jørgensen, 2004; Jørgensen et al., 2012; Frison and Jørgensen, 2013; Frison et al., 2018; Frison and Diehl, 2020; Frison et al., 2020).

1.3 Objectives and contributions

In this work, the main objectives are to develop 1) a high-performance Monte Carlo simulation toolbox for UQ of closed-loop systems, 2) a systematic modeling methodology for processes consisting of reactive systems conducted in reactors, and 3) an ENMPC algorithm for profit maximization applied to an mAb fermentation process. High-performance Monte Carlo simulation constitutes a versatile UQ tool for closed-loop systems and systematic modeling of reactive systems contributes to simple implementation of various models. With development of ENMPC technology for mAb production, we apply advanced process control technology to optimize a relevant biotechnological process, and we demonstrate both the systematic modeling methodology and the Monte Carlo simulation toolbox for UQ. We apply an existing model for an mAb fermentation process and show with Monte Carlo simulation that the developed ENMPC algorithm improves the profit of the process compared to a base case operational strategy. The work in this thesis is a first step towards a Monte Carlo simulation tool for UQ of biotechnological processes and ENMPC technology for mAb production. Figure 1.3 illustrates the conceptual idea, where an engineer can remotely monitor the production plant, monitor the control system, and access cloud computing services to perform simulations and UQ. The main contributions of this thesis are listed in the following.

Monte Carlo simulation toolbox. We apply Monte Carlo simulations for UQ of closed-loop systems. For this purpose, we implement a high-performance Monte Carlo simulation toolbox in C, which applies Open Multi-Processing (OpenMP) for parallelization on shared memory architectures. The parallelization enables almost linear scaling on multiple central processing unit (CPU) cores due to complete independence of closed-loop simulations and a thread-safe implementation of the toolbox. In this work, we 1) demonstrate parallel scalability of the toolbox for closed-loop simulations involving both proportional-integral-derivative (PID) controllers and NMPC algorithms and 2) apply Monte Carlo simulation for UQ of different closed-loop systems.

Systematic modeling methodology. We develop a systematic modeling methodology for processes consisting of reactive systems conducted in reactors. The methodology separates modeling of the reactor and the reactive system. The reactor models consist of mass balance differential equations, and the reactive system models consist of a stoichiometric matrix and a reaction rate vector function. In this way, the reactor equations are not required to be updated to change the considered reactive system. In this work, the considered reactor types result in ordinary differential equation (ODE) models, however the methodology can be generalized for reactor types resulting in partial differential equation (PDE) models such as plug-flow reactors (PFRs). We apply the considered ODE models for simulation in the Monte Carlo simulation toolbox and in NMPC algorithms.

Economic nonlinear model predictive control algorithm. We develop an ENMPC algorithm for profit maximization, which we apply for an mAb fermentation process. We select the specific mAb fermentation process due to collaboration with the

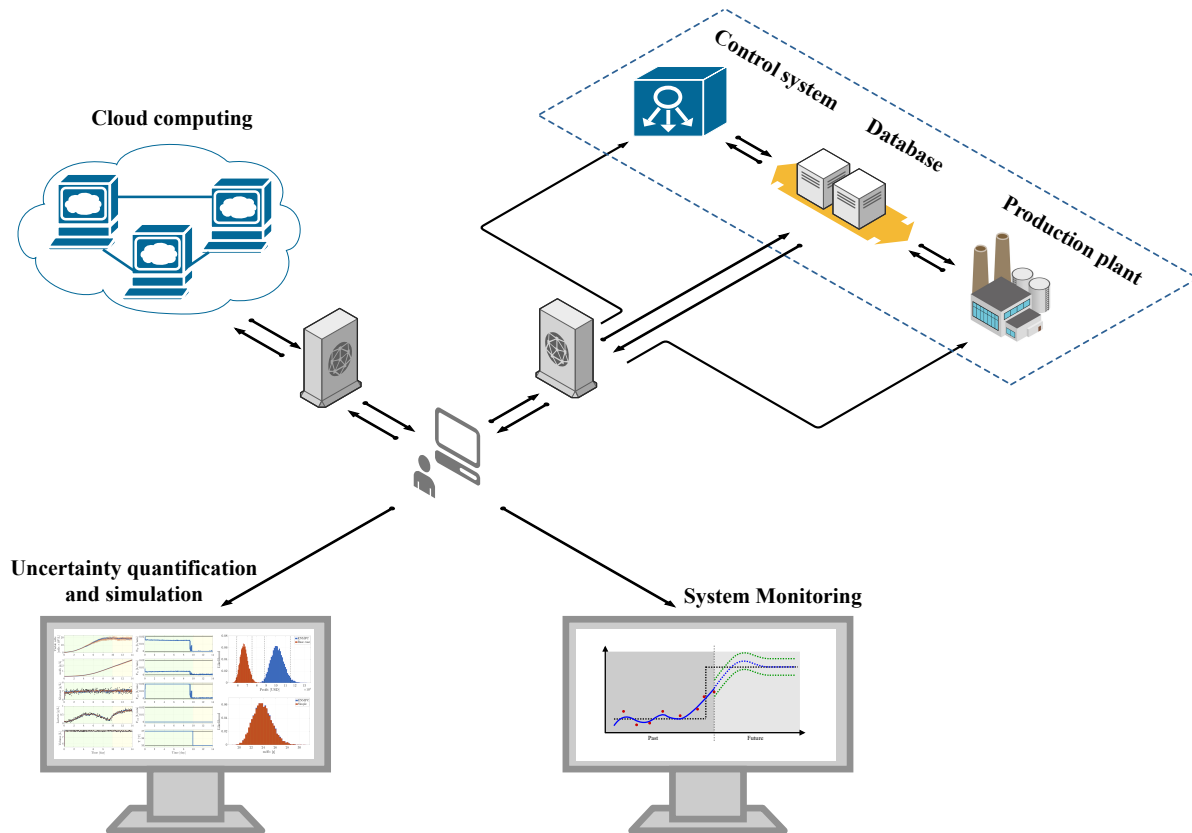


Figure 1.3: Conceptual idea of an advanced monitoring and simulation system. An engineer can remotely monitor both the plant and its control system while having access to cloud computing services. Cloud-based high-performance Monte Carlo simulation is applicable for UQ of the production process. Parts of this figure originally appeared in (Kaysfeld and Jørgensen, 2023).

main author of the original publication (Kumar et al., 2022). The collaboration yielded a dynamic optimization study of the mAb fermentation process (Appendix F), which later resulted in the ENMPC algorithm. The ENMPC algorithm applies a continuous-discrete extended Kalman filter (CD-EKF) as the state estimator and the regulator solves an economic OCP based on the CD-EKF state estimate. We apply a multiple-shooting method to transcribe the infinite dimensional OCP to a finite dimensional NLP and develop the SQP algorithm NLPSQP to solve the resulting NLP. NLPSQP applies a structure preserving Broyden–Fletcher–Goldfarb–Shanno (BFGS) update for Lagrangian Hessian approximation and a Riccati recursion based primal-dual IP algorithm to solve the QP-subproblem in each iteration. The ENMPC algorithm is thread-safe and implemented in C making it applicable in closed-loop simulations with the parallel Monte Carlo simulation toolbox.

1.4 Outline of the thesis

Chapter 2. We introduce the procedure for closed-loop simulation and Monte Carlo simulation. The considered closed-loop system formulation consists of a stochastic continuous-discrete system and a controller. The continuous-discrete system involves a stochastic differential equation (SDE), a static measurement function corrupted by measurement noise, and a static output function. We apply an Euler-Maruyama scheme for integration of SDEs to form discrete-time systems. Together with a discrete-time controller formulation, we form a discrete-time closed-loop system for simulation. Finally, we introduce the implemented high-performance Monte Carlo simulation toolbox for UQ of closed-loop systems. The toolbox applies OpenMP for parallelization on shared memory architectures. This chapter is based on Appendices A, D, and E.

Chapter 3. This chapter presents the systematic modeling methodology for systems of reactions conducted in reactors. We provide mass balance ODE models for three reactor types, 1) fed-batch reactors (FBRs), 2) continuous stirred tank reactors (CSTRs), and 3) continuous perfusion reactors (CPRs). We provide stoichiometry and kinetics for three reactive systems. Selected combinations of reactors and reactive systems result in three models, 1) a biomass fermentation process conducted in an FBR, 2) an exothermic chemical reaction conducted in an adiabatic CSTR, and 3) an mAb fermentation process conducted in a CPR. This chapter is based on Appendices C, D, F, and G.

Chapter 4. This chapter introduces an NMPC formulation consisting of an estimator and a regulator. The estimator is the CD-EKF and the regulator solves an OCP based on the state estimate. We introduce the OCP with a general objective function and apply a multiple shooting method to transcribe the infinite dimensional OCP to a finite dimensional NLP. We introduce two objective functions, 1) a target-tracking objective function and 2) a profit maximization objective function. The first objective function leads to a target-tracking NMPC algorithm and the second objective function leads to an ENMPC algorithm, where economic objectives are directly incorporated in the objective function. This chapter is based on Appendices B, E, G, and H.

Chapter 5. This chapter presents the main numerical results and demonstrates the performance of the Monte Carlo simulation toolbox and NMPC algorithms. We 1) show almost linear parallel scaling for Monte Carlo simulation of closed-loop systems with a PID controller and apply the toolbox for UQ of the biomass fermentation process, 2) show similar parallel scaling for closed-loop systems with NMPC algorithms and apply Monte Carlo simulation for performance quantification of the target-tracking NMPC algorithm for temperature tracking in the chemical reaction, and 3) apply Monte Carlo simulation to show that the ENMPC algorithm improves the profit of the mAb fermentation process compared to a base case operational strategy. This chapter is based on Appendices A, E, and G.

Chapter 6. We give the final conclusions and summarize the main contributions. We provide suggestions for future work and improvements of the presented work.

1.5 List of publications

In this section, we present a list of papers and a technical report. The work is conducted in collaboration with co-authors. We include a list of work included in this thesis and a list of non-included work relevant to the thesis. This thesis is a summary of the listed publications.

1.5.1 Publications included in the thesis

Peer-reviewed conference papers

1. Appendix F (Kaysfeld et al., 2023a).
Morten Wahlgreen Kaysfeld, Deepak Kumar, Marcus Krogh Nielsen, John Bagterp Jørgensen. Dynamic Optimization for Monoclonal Antibody Production. Proceedings of the 22nd World Congress of the International Federation of Automatic Control (IFAC WC), Yokohama, Japan, July 9-14, 2023.
2. Appendix E (Kaysfeld et al., 2023b).
Morten Wahlgreen Kaysfeld, Mario Zanon, John Bagterp Jørgensen. Performance Quantification of a Nonlinear Model Predictive Controller by Parallel Monte Carlo Simulations of a Closed-loop System. Proceedings of the European Control Conference (ECC), Bucharest, Romania, June 13-16, 2023.
3. Appendix D (Wahlgreen et al., 2023).
Morten Ryberg Wahlgreen, John Bagterp Jørgensen, Mario Zanon. Model Predictive Control Tuning by Monte Carlo Simulation and Controller Matching. Proceedings of Foundations of Computer Aided Process Operations / Chemical Process Control (FOCAPO/CPC), San Antonio, Texas, USA, January 8–12, 2023.
4. Appendix C (Wahlgreen et al., 2022).
Morten Ryberg Wahlgreen, Kristian Meyer, Tobias K. S. Ritschel, Allan Peter Engsig-Karup, Krist V. Germaey, John Bagterp Jørgensen. Modeling and Simulation of Upstream and Downstream Processes for Monoclonal Antibody Production. Proceedings of the 13th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS), Busan, Republic of Korea, June 14–17, 2022.
5. Appendix B (Wahlgreen and Jørgensen, 2022).
Morten Ryberg Wahlgreen, John Bagterp Jørgensen. On the Implementation of a Preconditioned Riccati Recursion based Primal-Dual Interior-Point Algorithm for Input Constrained Optimal Control Problems. Proceedings of the 13th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS), Busan, Republic of Korea, June 14–17, 2022.

6. Appendix A (Wahlgreen et al., 2021).

Morten Ryberg Wahlgreen, Asbjørn Thode Reenberg, Marcus Krogh Nielsen, Anton Rydahl, Tobias K. S. Ritschel, Bernd Dammann, John Bagterp Jørgensen. A High-Performance Monte Carlo Simulation Toolbox for Uncertainty Quantification of Closed-loop Systems. Proceedings of the 60th IEEE Conference on Decision and Control (CDC), Austin, TX, USA, December 14-17, 2021.

Journal paper

1. Appendix G (Kaysfeld and Jørgensen, 2023).

Morten Wahlgreen Kaysfeld, John Bagterp Jørgensen. Uncertainty Quantification of an Economic Nonlinear Model Predictive Controller for Monoclonal Antibody Production. Journal of Process Control, 2023. In submission.

Technical report

1. Morten Wahlgreen Kaysfeld, John Bagterp Jørgensen. Numerical optimization packages for optimal control: QPIPM and NLPSQP. DTU Library 2023.

1.5.2 Publications not included in the thesis

1. Thomas Emil Ryde, Morten Ryberg Wahlgreen, Marcus Krogh Nielsen, Steen Hørsholt, Sten Bay Jørgensen, John Bagterp Jørgensen. Optimal Feed Trajectories for Fedbatch Fermentation with Substrate Inhibition Kinetics. Proceedings of the 16th IFAC Symposium on Advanced Control of Chemical Processes (ADCHEM), Venice, Italy, June 13-16, 2021.
2. John Bagterp Jørgensen, Tobias K. S. Ritschel, Dimitri Boiroux, Eskild Schroll-Fleischer, Morten Ryberg Wahlgreen, Marcus Krogh Nielsen, Hao Wu, Jakob Kjøbsted Huusom. Simulation of NMPC for a Laboratory Adiabatic CSTR with an Exothermic Reaction. Proceedings of the European Control Conference (ECC), St. Petersburg, Russia, May 12-15, 2020.
3. Morten Ryberg Wahlgreen, Eskild Schroll-Fleischer, Dimitri Boiroux, Tobias K. S. Ritschel, Hao Wu, Jakob Kjøbsted Huusom, John Bagterp Jørgensen. Nonlinear Model Predictive Control for an Exothermic Reaction in an Adiabatic CSTR. Proceedings of the 6th Conference on Advances in Control and Optimization of Dynamical Systems (ACODS), Chennai, India, February 16-19, 2020.

CHAPTER 2

Monte Carlo simulation based uncertainty quantification

In this chapter, we introduce the proposed Monte Carlo simulation approach for uncertainty quantification (UQ) of closed-loop systems. The considered closed-loop systems consist of a stochastic continuous-discrete system and a discrete-time controller. The stochastic continuous-discrete system involves a stochastic differential equation (SDE), a static measurement function corrupted by measurement noise, and a static output function. We discretize the continuous-time SDE with an Euler-Maruyama scheme to form a stochastic discrete-time system. The controller consists of an estimator, a regulator, and a predictor. We represent the controller in discrete-time similar to the discrete-time system. In combination, the stochastic discrete-time system and the discrete-time controller form a discrete-time closed-loop system. There are many sources of uncertainty in a closed-loop system, e.g., process noise, measurement noise, disturbances, and uncertainty related to plant-model mismatch. To quantify the uncertainties of closed-loop systems, we propose a Monte Carlo simulation approach. The Monte Carlo simulation approach utilizes repeated simulation of the closed-loop system with different realizations of selected uncertain quantities in the process. The output of the simulations is a data set with selected key performance indicators (KPIs) for each simulation. With this approach, the uncertainties related to specific quantities can be quantified with respect to the selected KPIs, which makes the approach versatile. The Monte Carlo simulation approach is computationally tractable due to the implementation of a high-performance Monte Carlo simulation toolbox for closed-loop systems in C. The toolbox is parallelized with Open Multi-Processing (OpenMP) for shared memory architectures (Dagum and Menon, 1998). The independence of closed-loop simulations and the thread-safe implementation of the toolbox lead to almost linear parallel scaling for Monte Carlo simulations.

This chapter is based on the papers in Appendices A, D, and E. In Appendix A, we introduce the Monte Carlo simulation toolbox and apply the toolbox for UQ of a biomass fermentation process conducted in a fed-batch reactor (FBR). Appendix A also demonstrates almost linear parallel scaling for Monte Carlo simulation of closed-loop systems with proportional-integral-derivative (PID) controllers. Appendix D tunes a

model predictive control (MPC) algorithm with controller matching to a Monte Carlo simulation tuned proportional-integral (PI) controller. In Appendix E, we extended the toolbox with a nonlinear model predictive control (NMPC) algorithm and show similar parallel scaling for Monte Carlo simulation of closed-loop systems with the NMPC algorithm.

2.1 Procedure for closed-loop simulation

In this section, we introduce the procedure for simulation of closed-loop systems. The closed-loop system consists of a process with an output, sensors for measurements, and a controller. Figure 2.1 provides an illustration of the closed-loop system.

2.1.1 Process, sensors, and output

We model the stochastic process as an SDE, the measurements as a static function corrupted by measurement noise, and the output as a static function. This forms a stochastic continuous-discrete system in the form

$$x(t_0) = x_0, \quad (2.1a)$$

$$dx(t) = \overbrace{f(t, x(t), u(t), d(t), p)dt}^{\text{drift term}} + \overbrace{\sigma(t, x(t), u(t), d(t), p)d\omega(t)}^{\text{diffusion term}}, \quad (2.1b)$$

$$y(t_i) = g(t_i, x(t_i), p) + v(t_i, p), \quad (2.1c)$$

$$z(t) = h(t, x(t), p). \quad (2.1d)$$

$x(t)$ are states, $u(t)$ are inputs, $d(t)$ are disturbances, and p are parameters. The function $f(\cdot)$ is the drift function, $\sigma(\cdot)$ is the diffusion function, $g(\cdot)$ is the measurement function, and $h(\cdot)$ is the output function. The diffusion term involves a standard Wiener process, $\omega(t)$, i.e., $d\omega(t) \sim N_{iid}(0, Idt)$, and the measurements are corrupted by normally distributed noise, $v(t_i, p) \sim N_{iid}(0, R_v(t_i, p))$. Measurements are available with sampling

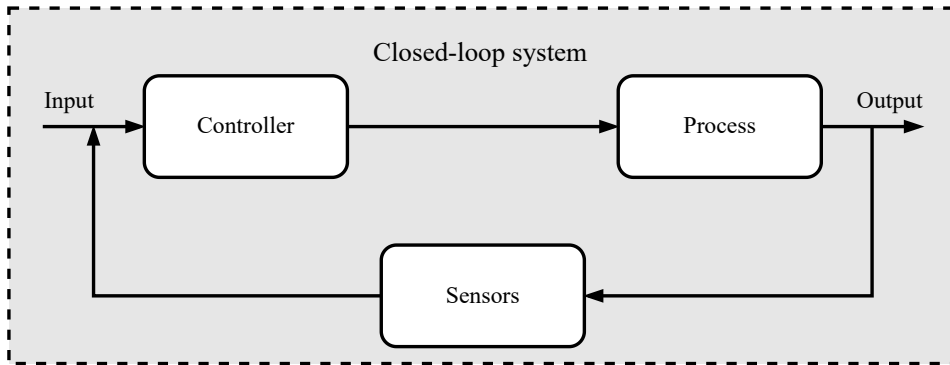


Figure 2.1: Illustration of the closed-loop system consisting of a process, a set of sensors, and a controller.

time T_s such that $t_{i+1} = t_i + T_s$. We assume zero-order-hold (ZOH) parameterization of the inputs, $u(t)$, and disturbances, $d(t)$, i.e.,

$$u(t) = u(t_i), \quad t_i \leq t < t_{i+1}, \quad (2.2a)$$

$$d(t) = d(t_i), \quad t_i \leq t < t_{i+1}. \quad (2.2b)$$

We apply a numerical explicit-explicit (Euler-Maruyama) scheme to solve the continuous SDE (2.1b) (Higham, 2001). Let N_s denote the number of internal steps, i.e., the number of Euler-Maruyama steps of fixed size Δt to integrate the SDE between t_i and t_{i+1} . Then, the Euler-Maruyama scheme applied to (2.1b) results in

$$t_{i,n+1} = t_{i,n} + \Delta t, \quad (2.3a)$$

$$x_{i,n+1} = x_{i,n} + f(t_{i,n}, x_{i,n}, u_i, d_i, p)\Delta t + \sigma(t_{i,n}, x_{i,n}, u_i, d_i, p)\Delta\omega_{i,n}, \quad (2.3b)$$

for $n = 0, 1, \dots, N_s - 1$ with $t_{i,n} = t_i + n\Delta t$, $x_{i,n} = x(t_{i,n})$, $u_i = u(t_i)$, $d_i = d(t_i)$, $\Delta\omega_{i,n} \sim N_{iid}(0, I\Delta t)$, $t_{i,0} = t_i$, $x_{i,0} = x_i$, $t_{i+1} = t_{i,N_s}$, and $x_{i+1} = x_{i,N_s}$. Let $\Phi(\cdot)$ denote an operator that applies the Euler-Maruyama scheme (2.3) and returns x_{i+1} . We express the process, sensors, and output as a stochastic discrete-time system in the form

$$x_{i+1} = \Phi(t_i, x_i, u_i, d_i, w_i, p), \quad (2.4a)$$

$$y_i = g(t_i, x_i, p) + v_i, \quad (2.4b)$$

$$z_i = h(t_i, x_i, p). \quad (2.4c)$$

$y_i = y(t_i)$, $z_i = z(t_i)$, $v_i = v(t_i, p)$, and $w_i = [\Delta\omega_{i,0}; \Delta\omega_{i,1}; \dots; \Delta\omega_{i,N_s-1}]$. In this work, we consider only non-stiff SDEs, which makes the Euler-Maruyama scheme a reasonable choice. We point out that we have implemented an implicit-explicit scheme as well, which should be applied for stiff SDEs (Tian and Burrage, 2001).

2.1.2 Controller

The controller consists of an estimator, a regulator, and a predictor. Figure 2.2 illustrates the controller and its connection to the process. We express the controller in discrete time similar to the discrete-time system (2.4). The discrete-time controller is

$$x_{i+1}^c = \kappa(t_i, x_i^c, u_i, d_i^c, y_{i+1}, p_c), \quad (2.5a)$$

$$u_i = \lambda(t_i, x_i^c, p_c), \quad (2.5b)$$

$$z_i^c = \mu(t_i, x_i^c, p_c). \quad (2.5c)$$

x_i^c are estimated states, u_i are the computed control signal, z_i^c are predictions, p_c are parameters in the controller, and d_i^c are disturbances known to the controller, which can differ from the real disturbances, d_i . The estimator $\kappa(\cdot)$, the regulator $\lambda(\cdot)$, and the predictor $\mu(\cdot)$ depend on the selected controller and can for some controllers be empty.

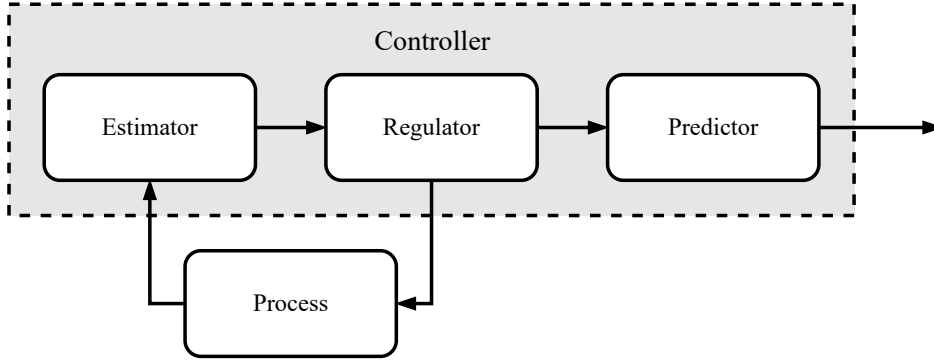


Figure 2.2: Illustration of the controller consisting of an estimator, a regulator, and a predictor. The estimator receives measurements from the process and the regulator returns a control signal back to the process. The predictor computes predictions of the system, which can be applied for, e.g., visualization.

2.1.3 Discrete-time closed-loop system

We combine the discrete-time system (2.4) and the discrete-time controller (2.5) to form the following discrete-time closed-loop system

$$y_i = g(t_i, x_i, p) + v_i, \quad (\text{Measurement}) \quad (2.6a)$$

$$z_i = h(t_i, x_i, p), \quad (\text{Output}) \quad (2.6b)$$

$$x_i^c = \kappa(t_{i-1}, x_{i-1}^c, u_{i-1}, d_{i-1}^c, y_i, p_c), \quad (\text{Estimation}) \quad (2.6c)$$

$$u_i = \lambda(t_i, x_i^c, p_c), \quad (\text{Regulation}) \quad (2.6d)$$

$$z_i^c = \mu(t_i, x_i^c, p_c), \quad (\text{Prediction}) \quad (2.6e)$$

$$x_{i+1} = \Phi(t_i, x_i, u_i, d_i, w_i, p), \quad (\text{Simulation}) \quad (2.6f)$$

for $i = 0, 1, \dots, N_{\text{sim}} - 1$. N_{sim} is the number of samples in the closed-loop simulation.

2.2 Procedure for Monte Carlo simulation

In this section, we introduce the Monte Carlo simulation procedure. We apply repeated simulation of the discrete-time closed-loop system (2.6), where each simulation includes a different realization of selected uncertain quantities. We save selected KPIs relevant to the specific system in each simulation. The KPI data can be applied for, e.g., UQ of the closed-loop performance and tuning of the controller. Figure 2.3 illustrates the Monte Carlo simulation approach for UQ of closed-loop systems. Algorithm 1 outlines the procedure for Monte Carlo simulation of closed-loop systems, where $x_i^{(j)}$, $u_i^{(j)}$, $d_i^{(j)}$, $z_i^{(j)}$, $p^{(j)}$, $y_i^{(j)}$, $v_i^{(j)}$, $w_i^{(j)}$, $x_i^{c,(j)}$, $z_i^{c,(j)}$, and $d_i^{c,(j)}$ denote closed-loop data for the j 'th Monte Carlo simulation and N_{mc} denotes the total number of Monte Carlo simulations. The procedure is versatile and enables the user to select uncertain quantities for realization and KPIs for quantification.

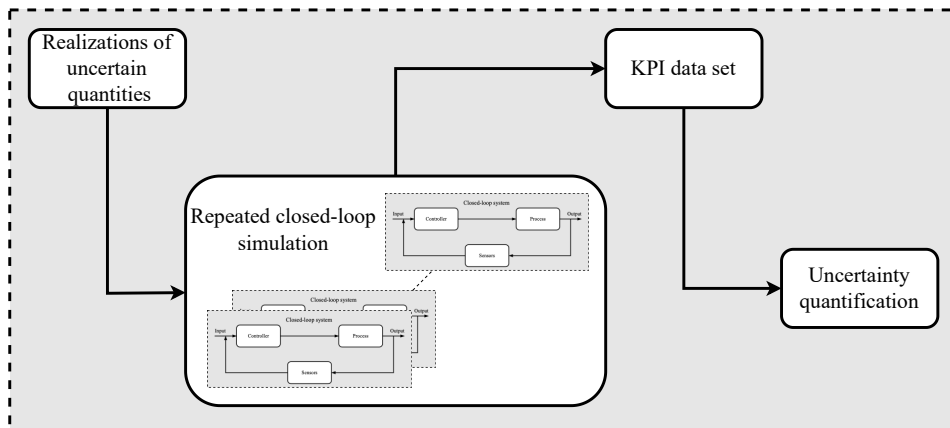


Figure 2.3: Illustration of Monte Carlo simulation approach for UQ of closed-loop systems. We select realizations of uncertain quantities and perform repeated closed-loop simulations. The resulting KPI data is applicable for UQ of the closed-loop performance.

2.2.1 High-performance Monte Carlo simulation toolbox for closed-loop systems

We implement a high-performance Monte Carlo simulation toolbox for closed-loop systems, which makes large-scale Monte Carlo simulation studies computational tractable. The toolbox implements the Monte Carlo simulation procedure presented in Algorithm 1. Closed-loop simulations are independent, which makes the outer loop of the Monte Carlo simulation approach well-suited for parallelization. The toolbox is implemented in C and applies OpenMP for parallelization on shared memory architectures. The implementation is thread-safe due to external memory allocation and internal distribution of the allocated memory. The toolbox supports user-implemented controllers, which are required to be thread-safe to achieve parallel scaling. In the case of optimization-based controllers, such as MPC algorithms, the optimization algorithm is also required to be thread-safe. For this purpose, we implement a thread-safe sequential quadratic programming (SQP) algorithm in C, which we discuss further in Chapter 4. Due to independence of closed-loop simulations and a thread-safe implementation of the toolbox and controllers, the Monte Carlo simulation toolbox has almost linear parallel scaling for both PID controllers and NMPC algorithms as demonstrated in Chapter 5. Currently, the toolbox supports variations in 1) model parameters (in both the simulation model and the controller model), 2) controller parameters, 3) process noise realizations, 4) initial conditions, and 5) disturbances (known and unknown to the controller).

2.3 Summary

In this chapter, we introduced the procedure for closed-loop simulation and Monte Carlo simulation. A closed-loop simulation consisted of a stochastic continuous-discrete system

Algorithm 1 Monte Carlo simulation procedure**Result:** Statistics and output data (KPIs)

```

1: for  $j = 1, 2, \dots, N_{\text{mc}}$  do ▷ Monte Carlo loop
2:   for  $i = 0, 1, \dots, N_{\text{sim}} - 1$  do ▷ Closed-loop simulation
3:     // Measurement
4:      $y_i^{(j)} = g(t_i, x_i^{(j)}, p^{(j)}) + v_i^{(j)}$ 
5:     // Output
6:      $z_i^{(j)} = h(t_i, x_i^{(j)}, p^{(j)})$ 
7:     // Estimation
8:      $x_i^{c,(j)} = \kappa(t_{i-1}, x_{i-1}^{c,(j)}, u_{i-1}^{(j)}, d_{i-1}^{c,(j)}, y_i^{(j)}, p_c^{(j)})$ 
9:     // Regulation
10:     $u_i^{(j)} = \lambda(t_i, x_i^{c,(j)}, p_c^{(j)})$ 
11:    // Prediction
12:     $z_i^{c,(j)} = \mu(t_i, x_i^{c,(j)}, p_c^{(j)})$ 
13:    // Simulation
14:     $x_{i+1}^{(j)} = \Phi(t_i, x_i^{(j)}, u_i^{(j)}, d_i^{(j)}, w_i^{(j)}, p^{(j)})$ 
15:  end for
16:  // Final measurement and output
17:   $y_{N_{\text{sim}}}^{(j)} = g(t_{N_{\text{sim}}}, x_{N_{\text{sim}}}^{(j)}, p^{(j)}) + v_{N_{\text{sim}}}^{(j)}$ 
18:   $z_{N_{\text{sim}}}^{(j)} = h(t_{N_{\text{sim}}}, x_{N_{\text{sim}}}^{(j)}, p^{(j)})$ 
19: end for

```

Note: This algorithm is adapted from its original appearance in (Wahlgreen et al., 2021).

that involves an SDE, a static measurement function corrupted by measurement noise, and a static output function. The SDE was discretized with an Euler-Maruyama scheme to form a stochastic discrete-time system. We considered controllers in discrete time with an estimator, a regulator, and a predictor. In combination, the discrete-time system and the discrete-time controller constituted a discrete-time closed-loop system. We introduced the Monte Carlo simulation procedure, which was based on repeated closed-loop simulation with different realizations of uncertain quantities. For computational tractability, we implemented a high-performance Monte Carlo simulation toolbox in C, which applies OpenMP for parallelization on shared memory architectures. The toolbox saves selected KPIs, which can be applied for UQ. The Monte Carlo simulation toolbox constitutes a versatile UQ tool, where the user can select any uncertain quantities for realization and any KPIs for quantification.

CHAPTER 3

Systematic process modeling

In this chapter, we introduce a systematic modeling methodology for processes consisting of reactive systems conducted in reactors. The modeling methodology provides a systematic and compact representation of models, where the reactor model and the reaction model is separated. Mass balances in the form of ordinary differential equations (ODEs) represent the reactor and the reaction model consists of stoichiometry and kinetics. In this way, processes can be changed without updating the reactor equations. The methodology presents models as ODEs in the form

$$\dot{x}(t) = f(t, x(t), u(t), d(t), p), \quad x(t_0) = x_0. \quad (3.1)$$

The ODE (3.1) constitutes the drift term of the stochastic differential equation (SDE) (2.1b) in the stochastic continuous-discrete system (2.1). We consider three reactor types and three processes, which form three models for 1) a biomass fermentation process conducted in a fed-batch reactor (FBR), 2) an exothermic chemical reaction conducted in an adiabatic continuous stirred tank reactor (CSTR), and 3) a monoclonal antibody (mAb) fermentation process conducted in a continuous perfusion reactor (CPR). We separately present the mass balance ODEs for each reactor type and the stoichiometric and kinetic models for the three processes. For each model, we introduce a static measurement function, a diffusion matrix function to extend the ODE models to SDE models, and operational information, e.g., available flow streams for the reactor. The three models are implemented in C for application in Monte Carlo simulation of closed-loop systems.

This chapter is based on the papers in Appendices C, D, F, and G. Appendix C introduces the modeling methodology and demonstrates the methodology on a fermentation process conducted in an FBR. In Appendix D, we apply the methodology to introduce a model for an exothermic chemical reaction conducted in an adiabatic CSTR. Appendices F and G applies the methodology for an mAb fermentation process conducted in a CPR.

3.1 Modeling methodology

In this section, we present the systematic modeling methodology for processes consisting of reactive systems conducted in reactors. The methodology separates the modeling of

the reactor and the reactions. The reactor model arises from mass balances and forms a system of ODEs for the reactor types considered in this thesis. The methodology is generalizable for reactor types giving rise to partial differential equations (PDEs), e.g., Appendix C applies the methodology for a chromatography process. The reaction model consists of stoichiometry and kinetics for the specific system of reactions.

3.1.1 Reactor modeling

We consider a reactor model as a virtual version of the physical reactor, in which any set of reactions can be conducted. This requires separation of the reactor model from the reaction model. The reactor mass balances are given by

$$\text{Acc} = \text{In} - \text{Out} + \text{Prod}. \quad (3.2)$$

The *Acc* term represents the change in the volume and the modeled components. The *In* and *Out* terms depend on the specific reactor type and the selection of inlet and outlet streams available in the reactor type. The *Prod* term models the reaction inside the reactor, which depends on the stoichiometry and kinetics for the specific system of reactions.

In the following, we introduce the mass balances for three different reactor types, FBRs, CSTRs, and CPRs. We assume that the density of a well-stirred reactor is constant. The reactor contains a set of modeled components denoted \mathcal{C} and a set of inlet streams denoted \mathcal{S} . We denote the set of reactions conducted in the reactor \mathcal{R} , which contains reaction numbers.

3.1.1.1 Mass balances for fed-batch reactors

We assume that FBRs have no outlet flow streams and a set of inlet streams with flow rates F_{in} . Figure 3.1(a) illustrates the FBR. The resulting system of ODEs is

$$\frac{dV}{dt} = e^\top F_{\text{in}}, \quad (3.3a)$$

$$\frac{dm}{dt} = C_{\text{in}} F_{\text{in}} + RV. \quad (3.3b)$$

$V \in \mathbb{R}$ is the volume, $e \in \mathbb{R}^{|\mathcal{S}|}$ is a vector of ones, $F_{\text{in}} \in \mathbb{R}^{|\mathcal{S}|}$ are the inlet flow rates, $m \in \mathbb{R}^{|\mathcal{C}|}$ are the component states measured with extensive properties, e.g., mass or amount, $C_{\text{in}} \in \mathbb{R}^{|\mathcal{C}| \times |\mathcal{S}|}$ contains the inlet stream concentrations, $c = m/V \in \mathbb{R}^{|\mathcal{C}|}$ are the component concentrations, and $R \in \mathbb{R}^{|\mathcal{C}|}$ are the component production rates computed as

$$R = S^\top r. \quad (3.4)$$

$S \in \mathbb{R}^{|\mathcal{R}| \times |\mathcal{C}|}$ is the stoichiometric matrix and $r = r(c, p) \in \mathbb{R}^{|\mathcal{R}|}$ are the reaction rates, where p includes a set of kinetic parameters. The FBR model (3.3) represents a batch reactor (BR) when $F_{\text{in}} = 0$.

3.1.1.2 Mass balances for continuous stirred tank reactors

We assume that CSTRs have a single outlet flow stream with flow rate F_{out} and a set of inlet streams with flow rates F_{in} . Figure 3.1(b) illustrates the CSTR. The resulting system of ODEs is

$$\frac{dV}{dt} = e^\top F_{\text{in}} - F_{\text{out}}, \quad (3.5a)$$

$$\frac{dm}{dt} = C_{\text{in}} F_{\text{in}} - c F_{\text{out}} + RV, \quad (3.5b)$$

where $F_{\text{out}} \in \mathbb{R}$ is the outlet flow rate. The CSTR model (3.5) represents an FBR when $F_{\text{out}} = 0$.

3.1.1.3 Mass balances for continuous perfusion reactors

We assume that CPRs have a single outlet flow stream with flow rate F_{out} , a single perfusion stream with flow rate F_{per} , and a set of inlet flow streams with flow rates F_{in} . Figure 3.1(c) illustrates the CPR. The system of ODEs is

$$\frac{dV}{dt} = e^\top F_{\text{in}} - F_{\text{out}} - F_{\text{per}}, \quad (3.6a)$$

$$\frac{dm}{dt} = C_{\text{in}} F_{\text{in}} - c F_{\text{out}} - C_{\text{per}} c F_{\text{per}} + RV. \quad (3.6b)$$

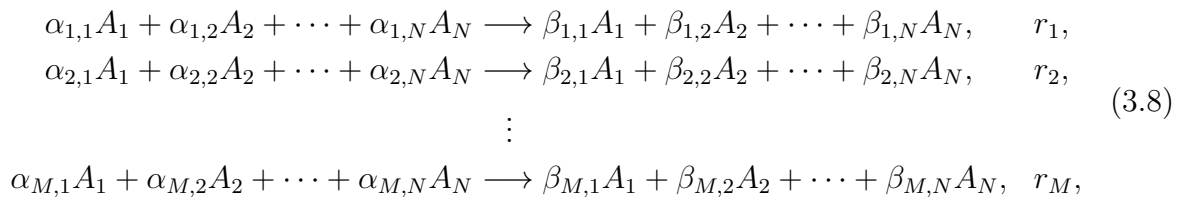
$F_{\text{per}} \in \mathbb{R}$ is the perfusion flow rate and $C_{\text{per}} \in \mathbb{R}^{|\mathcal{C}| \times |\mathcal{C}|}$ is a diagonal matrix with perfusion removal percentages. The CPR model (3.6) represents a CSTR when $F_{\text{per}} = 0$.

3.1.2 Reaction modeling

The production rate computation $R = S^\top r$ constitute the reaction model, which requires the stoichiometric matrix, S , and the reaction rate vector function (reaction kinetics), r . We consider a general system of M reactions and N components, i.e., the component set and reaction set are

$$\mathcal{C} = \{A_1, A_2, \dots, A_N\}, \quad \mathcal{R} = \{1, 2, \dots, M\}. \quad (3.7)$$

The reactive system is in the form



where r_i for $i \in \mathcal{R}$ are reaction rates for each reaction. The stoichiometric matrix provides a compact representation of reactive systems in the form (3.8). We define

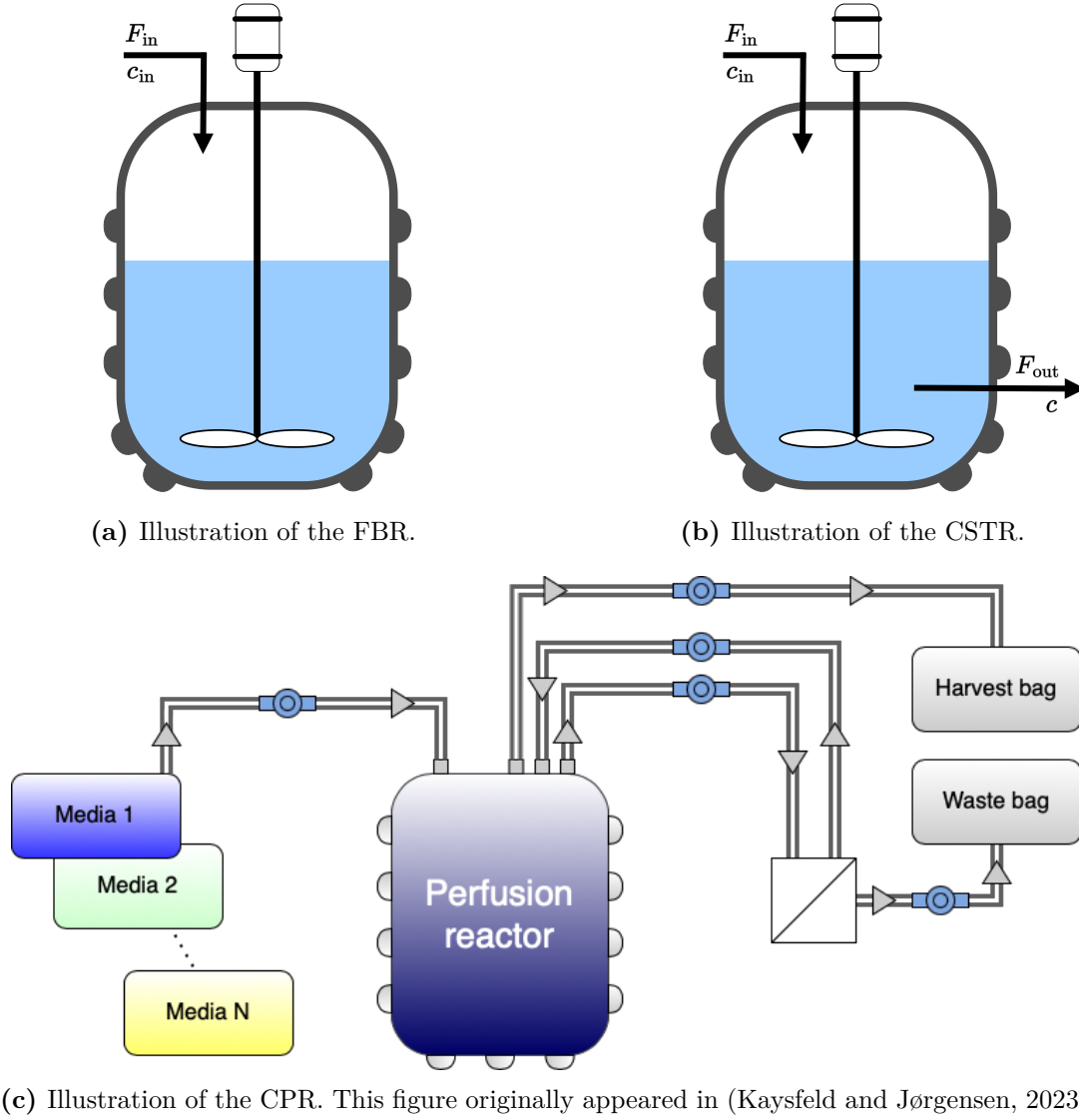


Figure 3.1: Illustrations of the modeled FBR, CSTR, and CPR.

the stoichiometric matrix such that each column represents a component and each row represents a reaction. The resulting stoichiometric matrix for (3.8) is in the form

$$S = \begin{bmatrix} A_1 & A_2 & \cdots & A_N \\ \beta_{1,1} - \alpha_{1,1} & \beta_{1,2} - \alpha_{1,2} & \cdots & \beta_{1,N} - \alpha_{1,N} \\ \beta_{2,1} - \alpha_{2,1} & \beta_{2,2} - \alpha_{2,2} & \cdots & \beta_{2,N} - \alpha_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{M,1} - \alpha_{M,1} & \beta_{M,2} - \alpha_{M,2} & \cdots & \beta_{M,N} - \alpha_{M,N} \end{bmatrix} \begin{matrix} 1 \\ 2 \\ \vdots \\ M \end{matrix}. \quad (3.9)$$

The reaction rate, $r_i = r_i(c, p)$ for $i \in \mathcal{R}$, is a function of the component concentrations c and kinetic parameters included in p . We define the reaction rate vector function, $r = r(c, p)$, as

$$r(c, p) = \begin{bmatrix} r_1(c, p) & r_2(c, p) & \cdots & r_M(c, p) \end{bmatrix}^\top. \quad (3.10)$$

For a specific process, we define the stoichiometric matrix S and the reaction rate vector function $r(c, p)$, which together represent the reaction model.

3.2 Biomass fermentation process conducted in a fed-batch reactor

In this section, we introduce a biomass fermentation process with stoichiometry and kinetics for *Methylococcus Capsulatus* (Olsen et al., 2010; Ryde et al., 2021). The process model has a single reaction and two components, 1) the substrate and 2) the biomass. The fermentation is conducted in an FBR in the form (3.3), and we assume that the system is oxygen saturated (Ryde et al., 2021). In the following, we define the stoichiometry and kinetics for the fermentation process and define the operational information for the FBR. We refer to previous work for the model parameters and more details (Ryde et al., 2021).

3.2.1 Stoichiometry and kinetics

The process consists of a single reaction. The component set and reaction set are

$$\mathcal{C} = \{S, X\}, \quad \mathcal{R} = \{1\}, \quad (3.11)$$

where S is the substrate and X is the biomass. The reaction describes the consumption of substrate to produce biomass as



γ is the stoichiometric coefficient. The stoichiometric matrix, S (not to be confused with the substrate component), is

$$S = \begin{array}{c} \begin{array}{cc} S & X \\ \begin{bmatrix} -\gamma & 1 \end{bmatrix} & 1 \end{array} \end{array}. \quad (3.13)$$

The reaction is governed by Haldane growth kinetics resulting in the reaction rate function

$$r_1(c) = \mu(c_S)c_X, \quad (3.14)$$

where the specific growth rate, $\mu(c_S)$, is

$$\mu(c_S) = \mu_{\max} \frac{c_S}{K_S + c_S + c_S^2/K_I}. \quad (3.15)$$

μ_{\max} , K_S , and K_I are parameters in the model.

3.2.2 Operation

We operate the FBR with two inlet streams, 1) a pure water inlet stream and 2) a stream with substrate at a constant concentration $c_{S,\text{in}}$. The inlet stream set and corresponding inlet flow rate vector are

$$\mathcal{S} = \{S_W, S_S\}, \quad F_{\text{in}} = \begin{bmatrix} F_W \\ F_S \end{bmatrix}, \quad (3.16)$$

where S_W is the water stream with flow rate F_W and S_S is the substrate stream with flow rate F_S . The inlet concentration matrix, C_{in} , is

$$C_{\text{in}} = \begin{bmatrix} 0 & c_{S,\text{in}} \\ 0 & 0 \end{bmatrix}. \quad (3.17)$$

We consider the two inlet flow rates, F_W and F_S , as manipulated inputs to the system.

3.2.3 Deterministic model

The stoichiometry and kinetics (3.11)-(3.15) constitute the reaction model for the fermentation process. Together with the FBR model (3.3) and the operational information (3.16)-(3.17), the model is complete. In the general form (3.1), the fed-batch fermentation model has

$$x = \begin{bmatrix} V \\ m_S \\ m_X \end{bmatrix}, \quad u = \begin{bmatrix} F_W \\ F_S \end{bmatrix}, \quad d = [\quad], \quad f = \begin{bmatrix} e^\top F_{\text{in}} \\ C_{\text{in}} F_{\text{in}} + RV \end{bmatrix}. \quad (3.18)$$

V [m³] is the volume, m_S [kg] is the substrate mass, m_X [kg] is the biomass mass, F_W [m³/h] is the water inlet flow rate, and F_S [m³/h] is the substrate inlet flow rate.

3.2.4 Diffusion matrix and measurement function

We extend the deterministic model (3.18) with a diffusion term to get an SDE in the form (2.1b). We model stochastic variations in the three states with a constant diagonal diffusion matrix

$$\sigma = \begin{bmatrix} \sigma_V & & \\ & \sigma_S & \\ & & \sigma_X \end{bmatrix}. \quad (3.19)$$

The diffusion parameters, σ_V , σ_S , and σ_X , are parameters in the model. We apply a static measurement function for substrate concentration measurements at discrete times

$$y(t_i) = g(t_i, x(t_i), p) = c_S = \frac{m_S}{V}. \quad (3.20)$$

Substrate concentration measurements are corrupted by measurement noise with covariance matrix

$$R_v = \sigma_{y,S}^2, \quad (3.21)$$

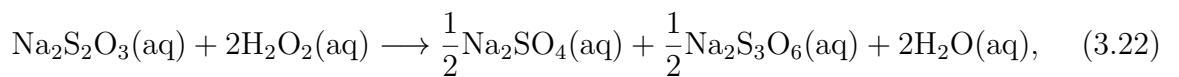
where $\sigma_{y,S}$ is a parameter for estimation or selected based on measurement equipment specifications.

3.3 Exothermic chemical reaction conducted in an adiabatic continuous stirred tank reactor

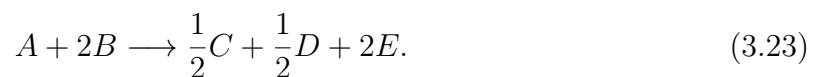
In this section, we introduce an exothermic chemical reaction conducted in a laboratory scale adiabatic CSTR (Wahlgreen et al., 2020; Jørgensen et al., 2020). The reaction has five components (two reactants and three products) and is exothermic, i.e., produces heat. We model the two reactants and the temperature, and we apply the CSTR model (3.5) to represent the reactor. In the following, we define the stoichiometry and kinetics for the chemical reaction and define the operational information for the CSTR. We refer to previous work for the model parameters and more details (Wahlgreen et al., 2020; Jørgensen et al., 2020).

3.3.1 Stoichiometry and kinetics

The process consists of a single reaction. The exothermic chemical reaction is



which we express as



We consider the thermal energy as a product and disregard the components C , D , and E . The component set and reaction set are

$$\mathcal{C} = \{A, B, T\}, \quad \mathcal{R} = \{1\}, \quad (3.24)$$

where A and B are reactants and T is the thermal energy. The reduced reaction is



where we consider $\beta = -\Delta H_r/(\rho c_P)$ as the stoichiometric coefficient for T . ΔH_r is the enthalpy of reaction, ρ is the density of the aqueous mixture, and c_P is the specific heat capacity. The concentration of the thermal energy component, c_T , is the temperature of the mixture. Based on the components (3.24) and the reaction (3.25), the stoichiometric matrix is

$$S = \begin{array}{ccc} & \text{A} & \text{B} & \text{T} \\ & -1 & -2 & \beta \\ & & & 1 \end{array} \cdot \quad (3.26)$$

The reaction rate is

$$r_1(c) = k(c_T)c_A c_B, \quad (3.27)$$

where the rate function, $k(c_T)$, is given by the Arrhenius expression

$$k(c_T) = k_0 \exp\left(-\frac{E_a}{R c_T}\right). \quad (3.28)$$

k_0 is the Arrhenius constant, E_a is the activation energy, and R is the gas constant.

3.3.2 Operation

We operate the CSTR with a single inlet media flow stream with flow rate F . The inlet stream has a temperature of $c_{T,\text{in}}$ and contains both reactants at concentrations $c_{A,\text{in}}$ and $c_{B,\text{in}}$. The volume of the CSTR is constant due to overflow into a waste tank. We model the overflow as an outlet flow stream with flow rate $F_{\text{out}} = F$ to maintain the constant volume. The inlet stream set and corresponding inlet flow rate vector are

$$\mathcal{S} = \{S_M\}, \quad F_{\text{in}} = [F], \quad (3.29)$$

where S_M is the media inlet flow stream with flow rate F . The inlet concentration matrix, C_{in} , is

$$C_{\text{in}} = \begin{bmatrix} c_{A,\text{in}} \\ c_{B,\text{in}} \\ c_{T,\text{in}} \end{bmatrix}. \quad (3.30)$$

We consider the inlet and outlet flow rate, F , as the single manipulated input to the system.

3.3.3 Deterministic model

The stoichiometry and kinetics (3.24)-(3.28) constitute the reaction model for the exothermic chemical reaction. Together with the CSTR model (3.5) and the operational information (3.29)-(3.30), the model is complete. In the general form (3.1), the model for the exothermic chemical reaction conducted in an adiabatic CSTR has

$$x = \begin{bmatrix} n_A \\ n_B \\ n_T \end{bmatrix}, \quad u = [F], \quad d = [\quad], \quad f = C_{\text{in}}F - cF + RV. \quad (3.31)$$

n_A [mol] is the mole number of reactant A , n_B [mol] is the mole number of reactant B , n_T [K·L] is the total thermal energy, and F [L/s] is the inlet and outlet flow rate. The volume, V [L], is a model parameter rather than a state, since the CSTR has constant volume. Notice, we apply the notation n instead of m for the component states in this model, since the unit of reactant components is moles.

3.3.4 Model reduction

The model (3.31) has three states and one input. The three-state model is exactly represented by a one-state model at steady-state, which enables a model reduction based on steady-state assumptions (Wahlgreen et al., 2020). At steady-state, the reactant concentrations, c_A and c_B , are functions of the temperature, c_T , as

$$c_A(c_T) = c_{A,\text{in}} + \frac{1}{\beta}(c_{T,\text{in}} - c_T), \quad c_B(c_T) = c_{B,\text{in}} + \frac{2}{\beta}(c_{T,\text{in}} - c_T). \quad (3.32)$$

By elimination of the two reactant states, the model has only the thermal energy state remaining. The stoichiometric matrix and inlet concentration matrix are

$$S = [\beta], \quad C_{\text{in}} = [c_{T,\text{in}}]. \quad (3.33)$$

The reaction rate is

$$r_1(c_T) = k(c_T)c_A(c_T)c_B(c_T), \quad (3.34)$$

and the complete one-state model in the general form (3.1) has

$$x = [n_T], \quad u = [F], \quad d = [\quad], \quad f = C_{\text{in}}F - cF + RV. \quad (3.35)$$

n_T [K·L] is the total thermal energy and F [L/s] is the inlet and outlet flow rate.

3.3.5 Diffusion matrix and measurement function

We extend the deterministic models (3.31) and (3.35) with diffusion terms to get SDEs in the form (2.1b). We model stochastic variations in the three-state model and in the

one-state model with diagonal diffusion matrices

$$\sigma_{3d} = \begin{bmatrix} \sigma_A & & \\ & \sigma_B & \\ & & \sigma_T \end{bmatrix}, \quad \sigma_{1d} = [\sigma_T]. \quad (3.36)$$

We select $\sigma_A = \sigma_B = 0$ and $\sigma_T = F\bar{\sigma}_T$ with $\bar{\sigma}_T$ being a parameter. This particular choice of diffusion matrix models stochastic variations in the inlet temperature, $c_{T,\text{in}}$. We apply a static measurement function for temperature measurements at discrete times in both the three-state and the one-state model

$$y(t_i) = g(t_i, x(t_i), p) = c_T = \frac{n_T}{V}. \quad (3.37)$$

The temperature measurements are corrupted by measurement noise with covariance matrix

$$R_v = \sigma_{y,T}^2, \quad (3.38)$$

where $\sigma_{y,T}$ is a parameter for estimation or selected based on measurement equipment specifications.

3.4 Monoclonal antibody fermentation process conducted in a continuous perfusion reactor

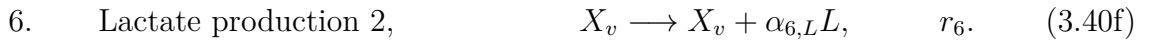
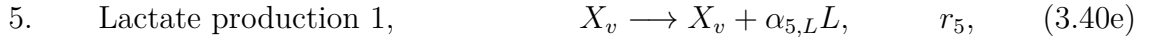
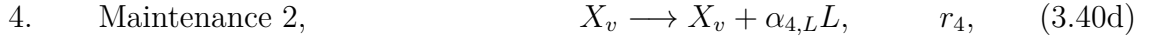
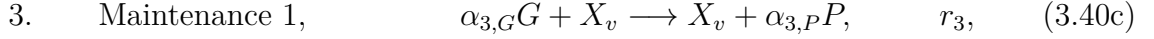
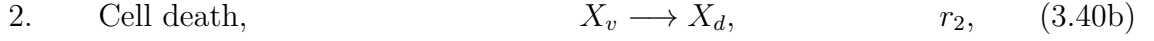
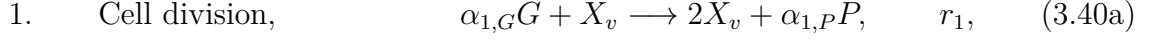
In this section, we introduce an mAb fermentation process conducted in a CPR (Kumar et al., 2022). In Appendix F, we extend the original model with a glucose inhibition term and a smooth maximum approximation for product inhibition. The process model consists of six reactions and includes five components, 1) viable cells, 2) dead cells, 3) glucose, 4) lactate, and 5) the product (mAb). Factors like oxygen and pH are not included in the model, but are part of the reactor operation (Kumar et al., 2022). The fermentation is conducted in a CPR, which we model in the general form (3.6). In the following, we define the stoichiometry and kinetics for the fermentation process and define the operational information for the CPR. Appendices F and G present the model parameters and more details.

3.4.1 Stoichiometry and kinetics

The mAb fermentation process consists of six reactions conducted in a CPR. The component set and reaction set are

$$\mathcal{C} = \{X_v, X_d, G, L, P\}, \quad \mathcal{R} = \{1, 2, 3, 4, 5, 6\}, \quad (3.39)$$

where X_v are viable cells, X_d are dead cells, G is glucose, L is lactate, and P is the product. The six stoichiometric reactions are



Here, $\alpha_{1,G}$, $\alpha_{1,P}$, $\alpha_{3,G}$, $\alpha_{3,P}$, $\alpha_{4,L}$, $\alpha_{5,L}$, and $\alpha_{6,L}$ are stoichiometric coefficients. The stoichiometric matrix for the six reactions (3.40) is

$$S = \begin{array}{ccccc} & X_v & X_d & G & L & P \\ \left[\begin{array}{cccccc} 1 & 0 & -\alpha_{1,G} & 0 & \alpha_{1,P} \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_{3,G} & 0 & \alpha_{3,P} \\ 0 & 0 & 0 & \alpha_{4,L} & 0 \\ 0 & 0 & 0 & \alpha_{5,L} & 0 \\ 0 & 0 & 0 & \alpha_{6,L} & 0 \end{array} \right] & \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \end{array} . \quad (3.41)$$

The reaction rates are all dependent on the viable cell density, c_{X_v} , and specific rate functions. The reaction rates are

$$r_1 = \mu_X(c, T)c_{X_v}, \quad r_2 = \mu_D(T)c_{X_v}, \quad r_3 = \mu_{m_1}c_{X_v}, \quad (3.42a)$$

$$r_4 = \mu_{m_2}c_{X_v}, \quad r_5 = \mu_{L,p_1}(c, T)c_{X_v}, \quad r_6 = \mu_{L,p_2}(c)c_{X_v}, \quad (3.42b)$$

where T is the temperature. We represent the reaction rates (3.42) as a vector function,

$$r = [r_1 \ r_2 \ r_3 \ r_4 \ r_5 \ r_6]^\top. \quad (3.43)$$

The vector function, $r = r(c, T, p)$, is a function of the concentrations, c , the temperature, T , and kinetic parameters included in p . The specific growth rate for viable cells is

$$\mu_X = \mu_{X,\max} f_{lim} f_{L,inh} f_{G,inh} f_{P,inh} f_{temp}, \quad (3.44)$$

where $\mu_{X,\max}$ is the maximum growth rate, f_{lim} is the limiting growth term, $f_{L,inh}$ is the lactate inhibition term, $f_{G,inh}$ is the glucose inhibition term, $f_{P,inh}$ is the product inhibition term, and f_{temp} specifies the temperature dependency of the specific growth rate. The five terms are

$$f_{lim} = \frac{c_G}{K_G c_{X_v} + c_G}, \quad f_{L,inh} = \frac{K_{I,L}}{K_{I,L} + c_L}, \quad (3.45a)$$

$$f_{G,inh} = 1 - s_\gamma(c_G, \bar{c}_G), \quad f_{P,inh} = \max_\alpha(0, 1 - K_{I,PCP}), \quad (3.45b)$$

$$f_{temp} = \exp\left(-\frac{K_1}{T}\right). \quad (3.45c)$$

In the glucose inhibition term, we apply the sigmoid function, s_γ , and in the product inhibition term, we apply the smooth maximum approximation, \max_α . The two functions are

$$s_\gamma(c_G, \bar{c}_G) = \frac{1}{1 + \exp(-\gamma(c_G - \bar{c}_G))}, \quad (3.46a)$$

$$\max_\alpha(x_1, x_2) = \frac{x_1 + x_2 + \sqrt{(x_1 - x_2)^2 + \alpha^2}}{2}. \quad (3.46b)$$

The original model did not include the glucose inhibition term and the product inhibition term was linear without the soft maximum function (Kumar et al., 2022). The specific cell death rate function is

$$\mu_D = \mu_{D,\max} f_{D,temp}. \quad (3.47)$$

$\mu_{D,\max}$ is the maximum death rate and $f_{D,temp}$ specifies the temperature dependency of the specific death rate given as

$$f_{D,temp} = \exp\left(-\frac{K_2}{T}\right). \quad (3.48)$$

The two maintenance specific rate functions, μ_{m_1} and μ_{m_2} , and the two lactate production specific rate functions, μ_{L,p_1} and μ_{L,p_2} , model the passive consumption of glucose to produce lactate and product. The rate functions are

$$\mu_{m_1} = \bar{\mu}_{m_1}, \quad \mu_{m_2} = \bar{\mu}_{m_2} \frac{L_{\max,2} - c_L}{L_{\max,2}}, \quad (3.49a)$$

$$\mu_{L,p_1} = \mu_X \frac{L_{\max,1} - c_L}{L_{\max,1}}, \quad \mu_{L,p_2} = \bar{\mu}_{L,p_2} \frac{L_{\max,1} - c_L}{L_{\max,1}}. \quad (3.49b)$$

The following kinetic and stoichiometric coefficients are parameters in the model: μ_{\max} , $\mu_{D,\max}$, $\bar{\mu}_{m_1}$, $\bar{\mu}_{m_2}$, $\bar{\mu}_{L,p_2}$, K_1 , K_2 , K_G , $K_{I,L}$, $K_{I,P}$, $L_{\max,1}$, $L_{\max,2}$, \bar{c}_G , γ , α , $\alpha_{1,G}$, $\alpha_{1,P}$, $\alpha_{3,G}$, $\alpha_{3,P}$, $\alpha_{4,L}$, $\alpha_{5,L}$, and $\alpha_{6,L}$.

3.4.2 Operation

We operate the CPR with two inlet streams, a perfusion stream, an outlet stream, and a temperature regulation system. We assume the temperature regulation system to be an ideal controller such that the temperature, T , is a manipulated input in the model. The two inlet streams are 1) a pure water inlet stream and 2) a glucose inlet stream with glucose at constant concentration $c_{G,\text{in}}$. The inlet stream set and corresponding inlet flow rate vector are

$$\mathcal{S} = \{S_W, S_G\}, \quad F_{\text{in}} = \begin{bmatrix} F_W \\ F_G \end{bmatrix}, \quad (3.50a)$$

where S_W is a water stream with flow rate F_W and S_G is a glucose stream with flow rate F_G . The inlet concentration matrix, C_{in} , is

$$C_{in} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & c_{G,in} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \quad (3.51)$$

The perfusion stream is an outlet stream with flow rate F_{per} , where cells and product are assumed to be completely filtrated and kept in the reactor. Only spend media, i.e., glucose and lactate, is removed from the reactor through the perfusion stream. We model the filtration behavior of the perfusion stream with a diagonal perfusion matrix, C_{per} . The value of each diagonal element is between 0 and 1, where 0 means no removal of the component and 1 means complete removal of the component. Accordingly, the perfusion matrix, C_{per} , is

$$C_{per} = \text{diag}([0 \ 0 \ 1 \ 1 \ 0]), \quad (3.52)$$

which models full glucose and lactate removal through the perfusion stream. The CPR has an outlet stream, which harvests the reactor content with flow rate F_{out} . We consider the four flow rates, F_W , F_G , F_{per} , and F_{out} , and the temperature, T , as manipulated inputs to the system.

3.4.3 Deterministic model

The stoichiometry and kinetics (3.39)-(3.49) constitute the reaction model for the mAb fermentation process. Together with the CPR model (3.6) and the operational information (3.50)-(3.52), the model is complete. In the general form (3.1), the model for the mAb fermentation process conducted in a CPR has

$$x = \begin{bmatrix} V \\ m_{X_v} \\ m_{X_d} \\ m_G \\ m_L \\ m_P \end{bmatrix}, \quad u = \begin{bmatrix} F_W \\ F_G \\ F_{per} \\ F_{out} \\ T \end{bmatrix}, \quad d = [\quad], \quad f = \begin{bmatrix} e^\top F_{in} - F_{out} - F_{per}, \\ C_{in} F_{in} - c F_{out} - C_{per} c F_{per} + RV \end{bmatrix}. \quad (3.53)$$

V [L] is the volume, m_{X_v} [cells $\times 10^9$] is the number of viable cells, m_{X_d} [cells $\times 10^9$] is the number of dead cells, m_G [g] is the glucose mass, m_L [g] is the lactate mass, m_P [g] is the product (mAb) mass, F_W [L/min] is the pure water stream flow rate, F_G [L/min] is the glucose stream flow rate, F_{per} [L/min] is the perfusion stream flow rate, F_{out} [L/min] is the outlet stream flow rate, and T [K] is the temperature.

3.4.4 Diffusion matrix and measurement function

We extend the deterministic model (3.53) with a diffusion term to get an SDE in the form (2.1b). We model stochastic variations in the number of viable cells, m_{X_v} , the glucose mass, m_G , and the lactate mass, m_L , with a diagonal diffusion matrix

$$\sigma = \begin{bmatrix} 0 & 0 & 0 \\ \sigma_{X_v}(c_{X_v}) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \sigma_G(c_G) & 0 \\ 0 & 0 & \sigma_L(c_L) \\ 0 & 0 & 0 \end{bmatrix}. \quad (3.54)$$

We apply concentration dependent diffusion given as

$$\sigma_{X_v}(c_{X_v}) = \bar{\sigma}_{X_v} c_{X_v}, \quad \sigma_G(c_G) = \bar{\sigma}_G c_G, \quad \sigma_L(c_L) = \bar{\sigma}_L c_L, \quad (3.55)$$

where $\bar{\sigma}_{X_v}$, $\bar{\sigma}_G$, and $\bar{\sigma}_L$ are parameters in the model. We apply a static measurement function for discrete-time measurements of the volume, V , the glucose concentration, c_G , and the lactate concentration, c_L ,

$$y(t_i) = g(t_i, x(t_i), p) = \begin{bmatrix} V \\ c_G \\ c_L \end{bmatrix} = \begin{bmatrix} V \\ m_G/V \\ m_L/V \end{bmatrix}. \quad (3.56)$$

The three measurements are corrupted by independent measurement noise with diagonal covariance matrix

$$R_v = \begin{bmatrix} \sigma_{y,V}^2 & & \\ & \sigma_{y,G}^2 & \\ & & \sigma_{y,L}^2 \end{bmatrix}, \quad (3.57)$$

where $\sigma_{y,V}$, $\sigma_{y,G}$, and $\sigma_{y,L}$ are parameters for estimation or selected based on measurement equipment specifications.

3.5 Summary

In this chapter, we introduced a general modeling methodology for processes consisting of reactive systems conducted in reactors. The methodology separately models the reactor and the reactions. In this way, processes can be changed without updating the reactor equations. We demonstrated the methodology representation of three reactor types, 1) FBRs, 2) CSTRs, and 3) CPRs. Mass balances in the form of ODEs constituted the reactor model. The reaction models were based on stoichiometric and kinetic considerations for a specific process. The models were completed by specification of the stoichiometric matrix and a reaction rate vector function together with operational information for the

chosen reactor. We introduced stoichiometry and kinetics for three reactive systems. In combination with the three reactor types, we presented three models, 1) a biomass fermentation process conducted in an FBR, 2) an exothermic chemical reaction conducted in an adiabatic CSTR, and 3) an mAb fermentation process conducted in a CPR. The general modeling methodology provided ODE models for the three systems, and we extended the models with diffusion terms to model stochastic variations. This resulted in SDE models. The stochastic models are applicable for Monte Carlo simulation of closed-loop systems and in nonlinear model predictive control (NMPC) algorithms.

CHAPTER 4

Nonlinear model predictive control

This chapter presents a nonlinear model predictive control (NMPC) formulation consisting of an estimator and a regulator. The estimator is the continuous-discrete extended Kalman filter (CD-EKF) and the regulator solves an optimal control problem (OCP) based on the CD-EKF state estimate. The solution to the OCP is the input trajectory in a finite horizon, where only the part corresponding to the first control interval is implemented in the process. The OCP has a general objective function that can be selected based on the specific process. We apply a multiple-shooting method to transcribe the infinite dimensional OCP to a finite dimensional nonlinear programming problem (NLP). The structure of the resulting NLP motivates implementation of the Riccati recursion based sequential quadratic programming (SQP) algorithm NLPSQP. NLPSQP solves the quadratic programming problem (QP)-subproblem with the Riccati recursion based primal-dual interior-point (IP) method QPIPM. We introduce two objective functions, 1) a target-tracking objective function based on a weighted least squares (LS) cost function and 2) a profit maximization objective function based on process incomes and expenses. These objective functions lead to a target-tracking NMPC algorithm and an economic nonlinear model predictive control (ENMPC) algorithm. The NMPC algorithms and the two optimization packages, QPIPM and NLPSQP, are implemented thread-safe in C. As a result, the NMPC algorithms are applicable in the parallel Monte Carlo simulation toolbox introduced in Chapter 2.

This chapter is based on the papers in Appendices B, E, G, and H. Appendix B provides details about the initial implementation of QPIPM in Matlab. Appendix H gives a detailed description of the two software packages QPIPM and NLPSQP implemented in both Matlab and C. In Appendix E, we apply the target-tracking NMPC algorithm for temperature tracking of a chemical reaction conducted in an adiabatic continuous stirred tank reactor (CSTR). Appendix G applies the ENMPC algorithm for profit maximization of a monoclonal antibody (mAb) fermentation process.

4.1 Estimator

We consider controller models as stochastic continuous-discrete systems in the form (2.1). We assume zero-order-hold (ZOH) parameterization of the inputs and disturbances such that they are constant between every measurement sample, which are available

with sampling time T_s . When a noisy measurement vector, y_i , is available at time t_i , the estimator computes an estimate of the states, $\hat{x}_{i|i}$. For this purpose, we apply a CD-EKF, which consists of two steps, 1) a prediction step and 2) a filtering step (Jazwinski, 1970; Schneider and Georgakis, 2013). The CD-EKF computes the filtered state estimate $\hat{x}_{i|i}$ and its covariance $P_{i|i}$ from the measurement vector, y_i , the previous actually implemented inputs, u_{i-1} , the previous disturbances, d_{i-1} , and the previous filtered mean-covariance pair, $\hat{x}_{i-1|i-1}$ and $P_{i-1|i-1}$. At time t_i , the CD-EKF procedure is,

- 1) provide current measurement vector, y_i , previous actual input vector, u_{i-1} , previous disturbances, d_{i-1} , and previous filtered mean-covariance pair, $\hat{x}_{i-1|i-1}$ and $P_{i-1|i-1}$,
- 2) compute the one-step prediction, $\hat{x}_{i|i-1}$ and $P_{i|i-1}$, from $\hat{x}_{i-1|i-1}$, $P_{i-1|i-1}$, u_{i-1} , and d_{i-1} ,
- 3) compute the filtered state estimate $\hat{x}_{i|i}$ and its covariance $P_{i|i}$ from the measurement vector, y_i , and the one-step prediction, $\hat{x}_{i|i-1}$ and $P_{i|i-1}$.

The actually implemented inputs can differ from the input signal computed by the regulator due to, e.g., failure in communication. In the proposed procedure, we base the CD-EKF prediction on the inputs that have actually been implemented in the process rather than the computed input signal from the regulator. In the first iteration, we provide reasonable values for $\hat{x}_{-1|-1}$, $P_{-1|-1}$, u_{-1} , and d_{-1} to initialize the CD-EKF.

The CD-EKF constitutes the estimator, $\kappa(\cdot)$, in the general controller form (2.5).

4.1.1 Prediction

The CD-EKF computes a prediction of the state and covariance evolution one sampling time ahead, i.e., a one-step prediction. The predictions are computed from the following system of ordinary differential equations (ODEs)

$$\frac{d}{dt}\hat{x}_{i-1}(t) = f(t, \hat{x}_{i-1}(t), u_{i-1}, d_{i-1}, p), \quad (4.1a)$$

$$\begin{aligned} \frac{d}{dt}P_{i-1}(t) &= A_{i-1}(t)P_{i-1}(t) + P_{i-1}(t)A_{i-1}(t)^\top \\ &\quad + \sigma_{i-1}(t)\sigma_{i-1}(t)^\top, \end{aligned} \quad (4.1b)$$

where

$$A_{i-1}(t) = \frac{\partial}{\partial x} f(t, \hat{x}_{i-1}(t), u_{i-1}, d_{i-1}, p), \quad (4.2a)$$

$$\sigma_{i-1}(t) = \sigma(t, \hat{x}_{i-1}(t), u_{i-1}, d_{i-1}, p). \quad (4.2b)$$

The CD-EKF solves (4.1) for $t \in [t_{i-1}, t_i]$ and applies the previous filtered mean-covariance pair, $\hat{x}_{i-1|i-1}$ and $P_{i-1|i-1}$, as initial condition, i.e.,

$$\hat{x}_{i-1}(t_{i-1}) = \hat{x}_{i-1|i-1}, \quad P_{i-1}(t_{i-1}) = P_{i-1|i-1}. \quad (4.3a)$$

The one-step prediction is the state and covariance at t_i given as

$$\hat{x}_{i|i-1} = \hat{x}_{i-1}(t_i), \quad P_{i|i-1} = P_{i-1}(t_i). \quad (4.4)$$

The CD-EKF applies the one-step prediction (4.4) in the filtering step.

4.1.2 Filtering

In the filtering step, the CD-EKF computes the filtered mean-covariance pair, $\hat{x}_{i|i}$ and $P_{i|i}$. From the measurement vector, y_i , and the one-step prediction, $\hat{x}_{i|i-1}$ and $P_{i|i-1}$, the CD-EKF computes the filtered state estimate, $\hat{x}_{i|i}$, as

$$\hat{y}_{i|i-1} = g(t_i, \hat{x}_{i|i-1}, p), \quad C_i = \frac{\partial}{\partial x} g(t_i, \hat{x}_{i|i-1}, p), \quad (4.5a)$$

$$e_i = y_i - \hat{y}_{i|i-1}, \quad R_{e,i} = R_i + C_i P_{i|i-1} C_i^\top, \quad (4.5b)$$

$$\hat{x}_{i|i} = \hat{x}_{i|i-1} + K_i e_i, \quad K_i = P_{i|i-1} C_i^\top R_{e,i}^{-1}, \quad (4.5c)$$

where $R_i = R_v(t_i, p)$ denotes the covariance of the measurement noise $v(t_i, p)$ in (2.1). The covariance of $\hat{x}_{i|i}$ is

$$P_{i|i} = P_{i|i-1} - K_i R_{e,i} K_i^\top \quad (4.6a)$$

$$= (I - K_i C_i) P_{i|i-1} \quad (4.6b)$$

$$= (I - K_i C_i) P_{i|i-1} (I - K_i C_i)^\top + K_i R_i K_i^\top. \quad (4.6c)$$

The expression (4.6c) is the Joseph stabilizing form (Schneider and Georgakis, 2013), which ensures symmetry and positive semi-definiteness of the update. Therefore, (4.6c) is the preferred form even though (4.6a) is computationally more efficient.

The regulator receives the filtered state estimate, $\hat{x}_{i|i}$, once it is available from the CD-EKF. The CD-EKF applies both the state estimate, $\hat{x}_{i|i}$, and its covariance, $P_{i|i}$, in the next iteration to compute a new one-step prediction.

4.2 Regulator

At time t_i , the regulator computes an open-loop control strategy in a finite control horizon when the CD-EKF provides a state estimate. To compute open-loop strategies, the regulator solves an OCP based on the deterministic part of the stochastic differential equation (SDE) (2.1b) and the available state estimate, $\hat{x}_{i|i}$. We denote the finite control horizon T_h and split it into N intervals of size T_s , i.e., $T_h = NT_s$. The inputs and disturbances are constant in each interval due to ZOH parameterization. We define the start of each interval as $t_{i,k} = t_i + kT_s$ for $k = 0, \dots, N-1$, where $t_{i,0} = t_i$. We also denote the final time $t_{f,i} = t_{i,N} = t_i + T_h$. The solution to the OCP is the continuous state trajectory and the parameterized input trajectory, $[x(t), u(t)]_{t_i}^{t_{f,i}}$. We implement only

the inputs corresponding to the first interval, $[t_i, t_i + T_s[$, in the process. The regulator OCP is in the form

$$\min_{[x(t);u(t)]_{t_i}^{t_{f,i}}} \varphi_i = \varphi_i \left([x(t), u(t)]_{t_i}^{t_{f,i}} \right), \quad (4.7a)$$

$$\text{s.t.} \quad x(t_i) = \hat{x}_{i|i}, \quad (4.7b)$$

$$\dot{x}(t) = f(t, x(t), u(t), d(t), p), \quad t_i \leq t \leq t_{f,i}, \quad (4.7c)$$

$$u(t) = u_{i,k}, \quad t_{i,k} \leq t < t_{i,k+1}, \quad k = 0, \dots, N-1, \quad (4.7d)$$

$$d(t) = d_{i,k}, \quad t_{i,k} \leq t < t_{i,k+1}, \quad k = 0, \dots, N-1, \quad (4.7e)$$

$$x_{\min} \leq x(t) \leq x_{\max}, \quad t_i \leq t \leq t_{f,i}, \quad (4.7f)$$

$$u_{\min} \leq u(t) \leq u_{\max}, \quad t_i \leq t \leq t_{f,i}. \quad (4.7g)$$

The objective function (4.7a) defines the objective of the regulator, e.g., target-tracking or profit maximization as introduced later in this chapter. The dynamic state constraints (4.7b) and (4.7c) constitute an initial value problem (IVP) for the state dynamics with the state estimate, $\hat{x}_{i|i}$, as initial condition. (4.7d) and (4.7e) are ZOH parameterization of the inputs and disturbances. The finite input set $\{u_{i,k}\}_{k=0}^{N-1}$ uniquely defines the input trajectory solution $[u(t)]_{t_i}^{t_{f,i}}$ due to ZOH parameterization. Finally, (4.7f) are state bound constraints and (4.7g) are input bound constraints. The state bound constraints might require softening to avoid feasibility issues in the OCP.

4.2.1 The multiple-shooting method

The OCP (4.7) is finite in the inputs due to ZOH parameterization and infinite in the states. We apply a multiple-shooting method to transcribe the infinite dimensional OCP to a finite dimensional NLP. The multiple-shooting method splits the finite control horizon, T_h , into control intervals. In this work, we assume that control intervals are equidistant, non-overlapping, and coincide with the sampling time, i.e., there are N control intervals of size T_s and the k 'th control interval is defined as $[t_{i,k}, t_{i,k+1}]$. The states at the left boundary of each control interval are considered as decision variables in the resulting NLP, i.e., $\{x_{i,k+1}\}_{k=0}^{N-1}$ with $x_{i,0} = \hat{x}_{i|i}$ being a parameter. We solve an IVP for the state dynamics in each control interval with the corresponding decision variable as initial condition. A set of new NLP constraints ensures continuity in the solution. Figure 4.1 illustrates the multiple-shooting method. The multiple-shooting method transcribes the OCP (4.7) to an NLP in the form

$$\min_{\{u_{i,k}, x_{i,k+1}\}_{k=0}^{N-1}} \varphi_i = \varphi_i \left(\{u_{i,k}, x_{i,k+1}\}_{k=0}^{N-1} \right), \quad (4.8a)$$

$$\text{s.t.} \quad R_{i,k} = x_{i,k+1} - F(t_{i,k}, x_{i,k}, u_{i,k}, d_{i,k}, p) = 0, \quad k = 0, \dots, N-1, \quad (4.8b)$$

$$x_{\min} \leq x_{i,k+1} \leq x_{\max}, \quad k = 0, \dots, N-1, \quad (4.8c)$$

$$u_{\min} \leq u_{i,k} \leq u_{\max}, \quad k = 0, \dots, N-1, \quad (4.8d)$$

where $F(\cdot)$ constitutes a numerical integration scheme to solve the state dynamic IVP with initial condition $x_{i,k}$. Often, the objective function (4.8a) depends continuously

on the states, e.g., by integration of a state dependent cost function. In this case, we formulate the integral as an IVP and solve it simultaneously with the state equations in each control interval. The constraint (4.8b) is a set of continuity constraints that forces the dynamic solution in each control interval to coincide with the corresponding decision variables. The constraints (4.8c) and (4.8d) constitute the state and input bound constraints.

To efficiently solve the NLP (4.8), we require gradient information about the objective function (4.8a) and the nonlinear equality constraints (4.8b). The gradients require sensitivities of the numerical integration scheme $F(\cdot)$ and the objective function evaluation. We apply a forward algorithmic differentiation approach to obtain the required sensitivities (Andersson, 2013; Gebremedhin and Walther, 2019). If the objective function (4.8a) is partially separable, which we assume in this work, the NLP has a specific structure, where the Lagrangian Hessian is block diagonal. When solved with an SQP algorithm, the QP-subproblem inherits a specific sparse structure, which can be utilized by a Riccati recursion based QP-solver (Rao et al., 1998; Jørgensen, 2004; Frison and Jørgensen, 2013; Frison and Diehl, 2020; Wahlgreen and Jørgensen, 2022; Kaysfeld et al., 2023b). This motivates the implementation of the Riccati recursion based SQP algorithm NLPSQP to solve OCPs in NMPC regulators. We implement NLPSQP thread-safe in C such that the software is applicable in parallel Monte Carlo simulations. The SQP algorithm solves the QP-subproblem with the Riccati recursion based primal-dual IP algorithm QPIPM, which we also implement thread-safe in C. NLPSQP applies a structure preserving Broyden–Fletcher–Goldfarb–Shanno (BFGS) update to avoid the need of second order derivative evaluations (Bock and Plitt, 1984). Appendix H pro-

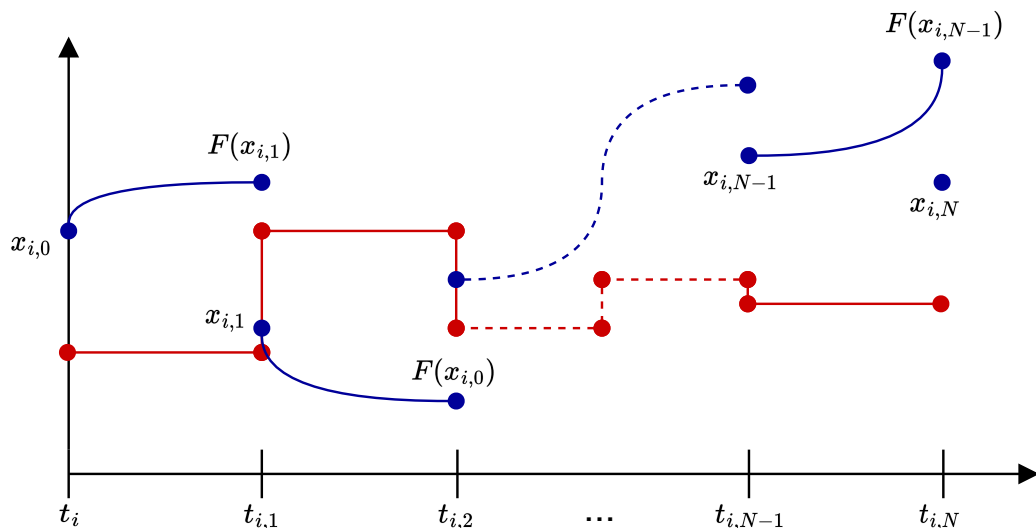


Figure 4.1: Illustration of the multiple-shooting method. The blue curve is the discontinuous state trajectory and the red curve is the ZOH parameterized input trajectory. We enforce continuity in the state trajectory by imposing continuity constraints in the resulting NLP.

vides details on QPIPM and NLPSQP. The implementations of QPIPM and NLPSQP are still in development and the main strengths of the implementations are their parallel scalability making them applicable with the Monte Carlo simulation toolbox introduced in Chapter 2. Multiple improvements can be considered to increase the computational performance of QPIPM and NLPSQP.

The solution to the NLP (4.8) constitutes both the regulator, $\lambda(\cdot)$, and the predictor, $\mu(\cdot)$, in the general controller form (2.5). The regulator extracts the inputs corresponding to the first control interval, i.e., $u_i = u_{i,0}$, and implements these inputs in the process. The predictor applies the inputs and states, $\{u_{i,k}, x_{i,k+1}\}_{k=0}^{N-1}$, as predictions in the finite horizon.

4.2.2 Target-tracking NMPC objective function

The first objective function considered in this work is for output target-tracking. We apply two different formulations for the OCP objective function, 1) integration of a cost function over the control horizon and 2) point-wise evaluation of the cost function at discrete points in the control horizon. The cost function is a weighted LS function for output target-tracking in the form

$$l(t, x(t), u(t), p) = l(z(t)) = \|z(t) - \bar{z}(t)\|_{Q_z}^2. \quad (4.9)$$

The output, $z(t)$, is defined in (2.1d), $\bar{z}(t)$ is the output target, and Q_z is a weight matrix. The integral objective function is in the form

$$\begin{aligned} \varphi_i &= \int_{t_i}^{t_{f,i}} l(t, x(t), u(t), p) dt \\ &= \sum_{k=0}^{N-1} \int_{t_{i,k}}^{t_{i,k+1}} l(t, x(t), u_{i,k}, p) dt, \\ &= \left\{ \sum_{k=0}^{N-1} \int_{t_{i,k}}^{t_{i,k+1}} l(t, x(t), u_{i,k}, p) dt : \right. \\ &\quad x(t_{i,0}) = \hat{x}_{i|i}, \\ &\quad x(t_{i,k}) = x_{i,k}, \quad k = 1, \dots, N-1, \\ &\quad \left. \dot{x}(t) = f(t, x(t), u_{i,k}, d_{i,k}, p), \quad t_{i,k} \leq t \leq t_{i,k+1}, \quad k = 0, \dots, N-1 \right\}. \end{aligned} \quad (4.10)$$

$l(\cdot)$ is the selected cost function (4.9) and the second equality is due to ZOH parameterization of the inputs. The third equality arises from multiple-shooting discretization. In practice, we formulate the objective integral as an IVP and solve it simultaneously with the state equations in each control interval. The point-wise objective function is a simplification of the integral objective function, where we evaluate the cost function only in the available decision variables. The point-wise objective function is

$$\varphi_i = l(t_{i,0}, u_{i,0}, p)T_s + \sum_{k=1}^{N-1} l(t_{i,k}, x_{i,k}, u_{i,k}, p)T_s + l(t_{i,N}, x_{i,N}, p)T_s. \quad (4.11)$$

The computational requirement for the point-wise objective function (4.11) is lower compared to the integral objective function (4.10), however it considers only point-wise information about the cost function. In Chapter 5, we apply the point-wise formulation (4.11) for temperature target-tracking in the CSTR model introduced in Chapter 3.

4.2.3 Profit maximization ENMPC objective function

The second objective function maximizes profit and leads to an ENMPC algorithm. We define profit as the difference between the incomes of the process and the expenses of the process. Therefore, the profit maximization objective function is

$$\varphi_i = -\varphi_{\text{profit},i} = -(\varphi_{\text{income},i} - \varphi_{\text{expense},i}), \quad (4.12)$$

where $\varphi_{\text{income},i}$ is the economic income from the process and $\varphi_{\text{expense},i}$ is the economic expense from the process. Specification of $\varphi_{\text{income},i}$ and $\varphi_{\text{expense},i}$ depends on the specific process. In Chapter 5, we define these for profit maximization of the mAb fermentation process introduced in Chapter 3.

4.3 Summary

In this chapter, we presented an NMPC formulation that consisted of an estimator and a regulator. The estimator was the CD-EKF, which applies a prediction step and filtering step to compute state estimates. The regulator solved an OCP when the CD-EKF provides a state estimate based on new available feedback. The OCP involved a general objective function, and we applied a multiple-shooting approach to transcribe the infinite dimensional OCP to a finite dimensional NLP. We introduced two objective functions, 1) a target-tracking objective function based on a weighted LS cost function and 2) a profit maximization objective function based on process incomes and expenses. These objective functions gave rise to a target-tracking NMPC algorithm and an ENMPC algorithm for profit maximization. We implemented the NMPC algorithms in C and applied the thread-safe SQP algorithm NLPSQP to solve OCPs in the regulators. The NMPC algorithms are applicable in the Monte Carlo simulation toolbox for parallel Monte Carlo simulation of closed-loop systems.

CHAPTER 5

Main results

This chapter presents the main numerical results of the thesis. We apply the Monte Carlo simulation toolbox introduced in Chapter 2, the three models introduced in Chapter 3, and the nonlinear model predictive control (NMPC) algorithms introduced in Chapter 4, i.e., the target-tracking NMPC algorithm and the economic nonlinear model predictive control (ENMPC) algorithm for profit maximization. We demonstrate that the Monte Carlo simulation toolbox has almost linear parallel scaling for proportional-integral-derivative (PID) controllers and the developed NMPC algorithms. We apply the toolbox for 1) uncertainty quantification (UQ) of the biomass fermentation process conducted in a fed-batch reactor (FBR) and tuning of the PID controller in the closed-loop system, 2) UQ of a closed-loop system consisting of the exothermic chemical reaction conducted in an adiabatic continuous stirred tank reactor (CSTR) and the target-tracking NMPC algorithm, and 3) UQ of profit in the monoclonal antibody (mAb) fermentation process with the developed ENMPC algorithm. The results demonstrate applications of the Monte Carlo simulation toolbox for UQ of different closed-loop systems with controller of varying complexity and the performance of the ENMPC algorithm for profit maximization of the considered mAb fermentation process.

This chapter is based on the papers in Appendices A, E, and G. In Appendix A, we introduce the implemented Monte Carlo simulation toolbox and demonstrate almost linear parallel scaling for a closed-loop system with a PID controller. Appendix E extends the toolbox with the target-tracking NMPC algorithm based on the sequential quadratic programming (SQP) algorithm NLPSQP and shows that the toolbox achieves similar parallel scaling with the NMPC algorithm. Finally, Appendix G applies the developed ENMPC algorithm for profit maximization of the mAb fermentation process and applies the Monte Carlo simulation toolbox for UQ of the closed-loop system.

5.1 Uncertainty quantification for biomass production

In this section, we apply the Monte Carlo simulation toolbox for UQ of the biomass fermentation process introduced in Section 3.2. The process is conducted in an FBR with an upper volume limit of 12.39 m^3 . Measurements of the substrate concentration, c_S , are available with sampling time T_s and the stochastic differential equation (SDE) model includes variations in the three state variables. The closed-loop system has initial

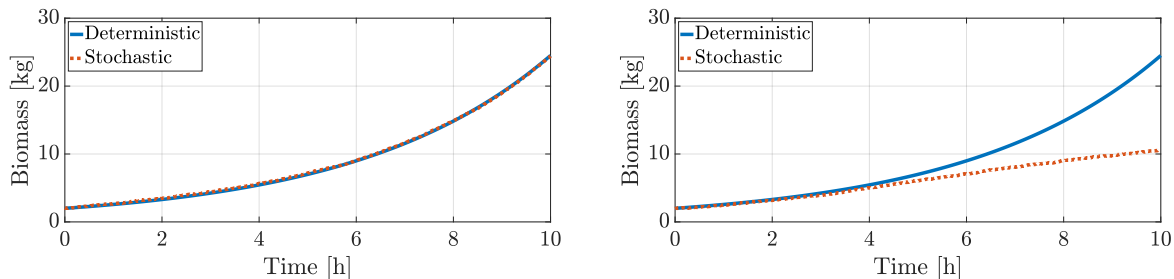
time $t_0 = 0.0$ h, final time $t_f = 10.0$ h, and sampling time $T_s = 36.0$ s. We provide a short motivation for the necessity of Monte Carlo simulations, demonstrate the parallel scaling properties of the toolbox, and apply Monte Carlo simulation for UQ of the process and tuning of the PID controller in the closed-loop system. The results are conducted on a dual-socket Intel® Xeon® Gold 6226R CPU @ 2.90GHz system with 16 cores in each non-uniform memory access (NUMA) node.

5.1.1 Motivation for Monte Carlo simulation

In the deterministic case, there exist infinitely many optimal open-loop operational strategies for maximization of the biomass production, which all maintain an optimal substrate concentration (Ryde et al., 2021). However, stochastic variations might lead to non-optimal substrate concentrations resulting in non-optimal biomass production in the open-loop system. To motivate this statement, we select one optimal operational strategy, the Bang-Bang strategy from the original paper (Ryde et al., 2021), and simulate the stochastic system twice. Figure 5.1 presents the two stochastic simulations compared to the deterministic simulation. The first stochastic simulation is almost identical to the deterministic simulation, but the second stochastic simulation results in low biomass production. It is hard to make reasonable conclusions about the process performance based on these two stochastic simulations. In conclusion, single or few stochastic simulations provide limited information about the uncertainty in the system. This motivates the application of Monte Carlo simulation for UQ of the process.

5.1.2 Parallel scaling with PID controller

We design a PID controller for tracking of the optimal substrate concentration by manipulating the substrate inlet stream flow rate, F_S . We perform 10,000 Monte Carlo simulations on different numbers of central processing unit (CPU) cores to get scale-up data for the Monte Carlo simulation toolbox. Figure 5.2 presents the scale-up results.



(a) Stochastic simulation 1. The stochastic simulation almost coincide with the deterministic simulation.

(b) Stochastic simulation 2. The stochastic simulation results in low biomass production compared to the deterministic simulation.

Figure 5.1: Stochastic simulations of the biomass fermentation model.

The Monte Carlo simulation toolbox shows almost linear parallel scaling on a single NUMA node and approximately 27.3 times speedup on 32 cores. The decrease in performance on the second NUMA node is expected since the toolbox is not designed for multiple NUMA nodes.

5.1.3 Uncertainty quantification and controller tuning

We apply the Monte Carlo simulation toolbox for UQ of the stochastic biomass fermentation process. We select the produced biomass as key performance indicator (KPI) for UQ. Figure 5.3 presents biomass production histograms with three different controllers. Each histogram is based on biomass production data from 30,000 closed-loop simulations with different process noise realizations, which takes less than a second to simulate. Figure 5.3(a) shows the open-loop performance of the Bang-Bang strategy resulting in a long-tailed biomass distribution with high probability of low biomass production. Figure 5.3(b) presents the closed-loop performance of a proportional (P) controller, which improves the biomass production in both mean and variance compared to the open-loop strategy. Finally, we apply the Monte Carlo simulation toolbox to tune a PID controller based on the performance of different PID gain combinations in 303,000 closed-loop simulations. The tuning process takes approximately 7.5 s. Figure 5.3(c) shows the biomass distribution for the tuned PID controller. The tuned PID controller improves the mean and variance of the biomass production compared to both the open-loop strategy and the P controller. The results show that Monte Carlo simulation enable simple KPI comparison between different operational strategies.

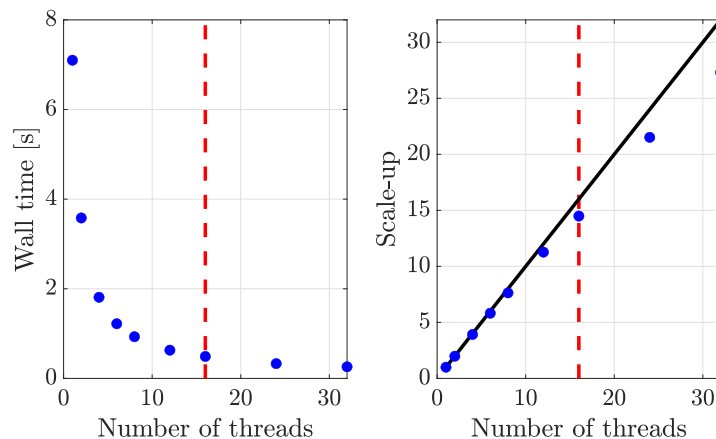
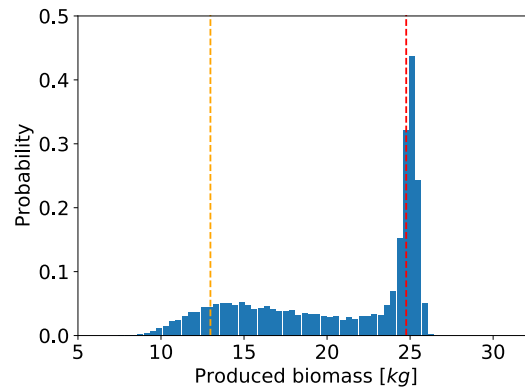
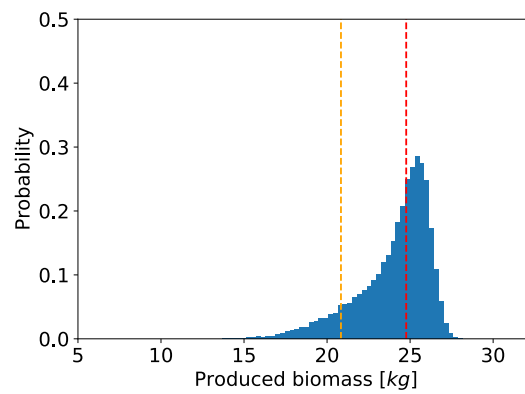


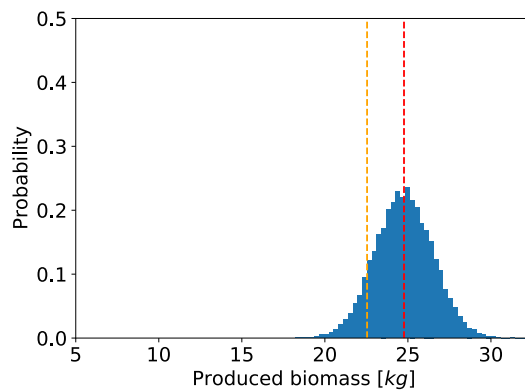
Figure 5.2: Parallel scaling of the Monte Carlo simulation toolbox with a PID controller based on 10,000 Monte Carlo simulations. The red dashed line shows the first 16 cores, i.e., the first NUMA node. We observe almost linear scaling on a single NUMA node and a scale-up of 27.3 on 32 cores. This figure originally appeared in (Wahlgreen et al., 2021).



(a) Open-loop controller.



(b) Sub-optimal P controller.



(c) Optimal PID controller.

Figure 5.3: Histograms for biomass production computed from 30,000 closed-loop simulations with different process noise realizations. The control strategy is specified in each subplot. The red dashed line is the produced biomass in the deterministic simulation and the orange dashed line is the 10% quantile. This figure originally appeared in (Wahlgren et al., 2021).

5.2 Uncertainty quantification of NMPC algorithm for a chemical reaction

In this section, we consider the model in Section 3.3 for an exothermic chemical reaction conducted in an adiabatic CSTR. The CSTR is laboratory-scale with a constant volume of 0.105 L. We consider the temperature as the output, i.e.,

$$z(t) = h(t, x(t), p) = \frac{n_T}{V} = c_T, \quad (5.1)$$

and apply the target-tracking NMPC algorithm in Chapter 4 for temperature tracking in the CSTR. The NMPC algorithm consists of the continuous-discrete extended Kalman filter (CD-EKF) and a tracking regulator based on the point-wise tracking objective formulation (4.11). The closed-loop system consists of the three-state model for simulation and the NMPC algorithm based on the reduced one-state model. Measurements of the temperature, c_T , are available with sampling time T_s and the SDE model includes variations in the inlet temperature, $c_{T,\text{in}}$. The initial time is $t_0 = 0.0$ s, the final time is $t_f = 600.0$ s, and the sampling time is $T_s = 1.0$ s.

The results are conducted on the same CPU as in the previous section. We apply a reference proportional-integral (PI) controller for comparison, but point out that this is solely to demonstrate the ease of comparing two controller types with Monte Carlo simulation. Figure 5.4 shows a single closed-loop simulation demonstrating that both controllers are able to track a variable temperature set-point in the process.

5.2.1 Parallel scaling with NMPC algorithm

We perform a parallel scaling study for the Monte Carlo simulation toolbox with the NMPC algorithm. Figure 5.5 shows that the Monte Carlo simulation toolbox achieves similar scaling with the NMPC algorithm as it did with the PID controller in Figure 5.2. The toolbox shows approximately 27 times speedup on 32 cores.

5.2.2 Comparing controllers with Monte Carlo simulation

We apply the Monte Carlo simulation toolbox to quantify the performance of the NMPC algorithm and the PI controller. As KPI, we select a scaled point-wise squared-2-norm metric given as

$$\Phi = \frac{1}{\bar{N} + 1} \sum_{i=0}^{\bar{N}} \|z(t_i) - \bar{z}(t_i)\|_2^2, \quad (5.2)$$

where $\bar{z}(t)$ is the output target and $\bar{N} = \frac{t_f - t_0}{T_s}$ is the number of samples in the closed-loop simulation. Figure 5.6 shows histograms for the two controllers based on KPI data

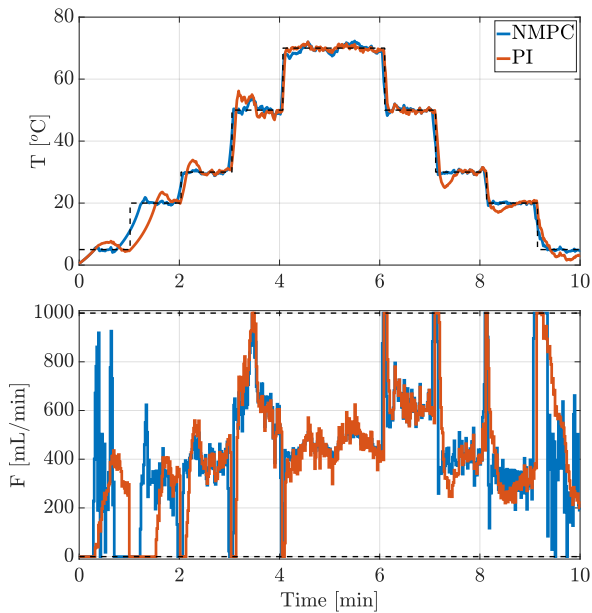


Figure 5.4: A single stochastic closed-loop simulation of the CSTR model with both the PI controller and the NMPC algorithm for temperature, $T = c_T$, set-point tracking. Both controllers are able to track the variable temperature set-point. This figure originally appeared in (Kaysfeld et al., 2023b).

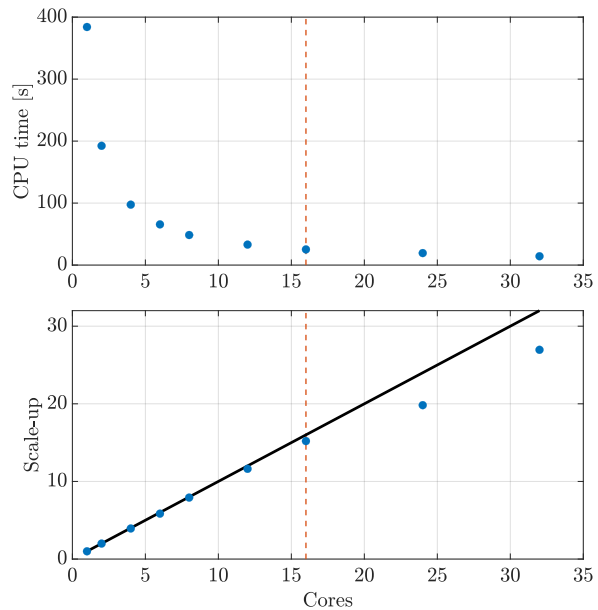


Figure 5.5: Parallel scaling of the Monte Carlo simulation toolbox with the NMPC algorithm. The red dashed line shows the first 16 cores, i.e., the first NUMA node. We observe similar parallel scaling as observed in Figure 5.2 and approximately 27 times speedup on 32 cores. This figure originally appeared in (Kaysfeld et al., 2023b).

from 30,000 closed-loop simulations. The histograms enable performance quantification based on KPI distributions, which makes comparison simple. As expected in this case, the NMPC algorithm outperforms the PI controller in both mean and variance. This result demonstrates that availability of large-scale Monte Carlo simulation data simplifies the process of comparing two controllers.

5.3 Uncertainty quantification of ENMPC algorithm for mAb production

In this section, we apply the developed ENMPC algorithm for profit maximization of the mAb fermentation process presented in Section 3.4. The process is conducted in a small-scale continuous perfusion reactor (CPR) with an upper volume limit of 8.0 L. Measurements of the volume, V , the glucose concentration, c_G , and the lactate concentration, c_L , are available with sampling time T_s and the SDE model includes variations in the number of viable cells, m_{X_v} , the glucose mass, m_G , and the lactate mass, m_L .

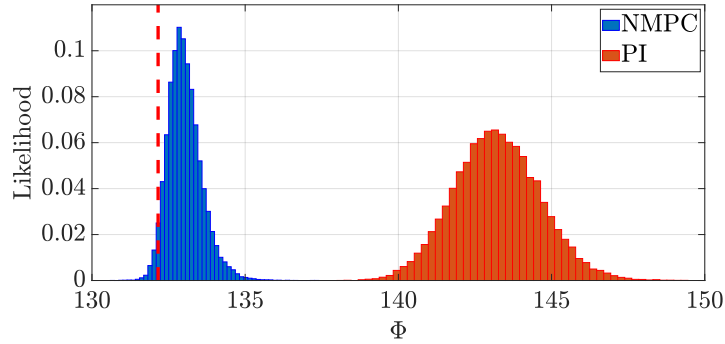


Figure 5.6: Histograms of the tracking KPI (5.2) for the PI controller and the NMPC algorithm based on 30,000 closed-loop simulations. The red dashed line is the performance of the NMPC algorithm in the deterministic case. The 30,000 simulations with the NMPC algorithm took approximately 55 min. This figure originally appeared in (Kaysfeld et al., 2023b).

The original paper on this process applied an operational strategy, which we denote the base case (Kumar et al., 2022). We design the closed-loop with initial time $t_0 = 0.0$ min, final time $t_f = 14 \cdot 24 \cdot 60 = 20,160.0$ min, and sampling time $T_s = 60.0$ min. We introduce the income and expense terms for the ENMPC regulator, show a single stochastic closed-loop simulation, apply Monte Carlo simulation for UQ of the profit and mAb production, and apply operational insights from the ENMPC algorithm to design a simple controller. We conduct the simulations on a Linux workstation with a 6 core Intel® Xeon® W-2235 CPU with frequency 3.80 GHz.

5.3.1 Income and expense term for mAb production

The profit objective term in the ENMPC regulator, $\varphi_{\text{profit},i}$ (4.12), includes an income term, $\varphi_{\text{income},i}$, and an expense term, $\varphi_{\text{expense},i}$. We apply simple expressions, where the income term is the price of produced mAb and the expense term is the price of spend glucose. We express the income as

$$\begin{aligned} \bar{\varphi}_{\text{income},i} &= \varphi_{P,i} = P_P \int_{t_i}^{t_{f,i}} R_P(t) V(t) dt \\ &= P_P (m_P(t_{f,i}) - m_P(t_i)) + P_P \int_{t_i}^{t_{f,i}} c_P(t) F_{\text{out}}(t) dt. \end{aligned} \quad (5.3)$$

P_P [USD/g] is the price of mAb (the product) and R_P is the product production rate as computed in (3.4). We assume that the outlet stream is only applied for harvesting at the end of the fermentation process, i.e., the reactor is in fed-batch perfusion mode with $F_{\text{out}} = 0$ throughout the fermentation. Instead of imposing a constraint in the optimizer, we implement the income term as

$$\varphi_{\text{income},i} = P_P (m_P(t_{f,i}) - m_P(t_i)), \quad (5.4)$$

where harvesting during the fermentation has a negative impact on the profit. This implies that the controller should set $F_{\text{out}} = 0$. The expense term is

$$\begin{aligned}\varphi_{\text{expense},i} &= \varphi_{G,i} = P_G \int_{t_i}^{t_{f,i}} c_{G,\text{in}} F_G(t) dt \\ &= P_G \sum_{k=0}^{N-1} c_{G,\text{in}} F_{G,i,k} T_s.\end{aligned}\tag{5.5}$$

P_G [USD/g] is the price of glucose and the second equality is due to zero-order-hold (ZOH) parameterization of the inputs such that $F_G(t) = F_{G,i,k}$ for $t_{i,k} \leq t < t_{i,k+1}$. We apply (5.4) and (5.5) to compute the profit term (4.12) for profit maximization of the mAb fermentation process.

5.3.2 Closed-loop simulation with ENMPC algorithm

We consider the ENMPC algorithm in closed-loop for profit maximization of the mAb fermentation process. The ENMPC consists of the CD-EKF and an economic regulator based on the income term (5.4) and expense term (5.5). Figure 5.7 presents a single stochastic closed-loop simulation of the system. The ENMPC algorithm operates the CPR in two phases, which we denote the *growth phase* and *production phase*. The process starts in the growth phase and the production phase begins when the controller reduces the temperature from the upper limit to the lower limit. The optimal operation provides a set of key insights for selection of the inputs. Table 5.1 summarizes the insights, and we notice that the ENMPC algorithm keeps the glucose concentration close to the point of glucose inhibition. The simulation resulted in a 24.82 g mAb production in the 14 days fermentation.

Table 5.1: Input key insights for optimal operation of the CPR. This table originally appeared in (Kaysfeld and Jørgensen, 2023).

Input	Growth phase	Production phase
F_W	Selected based on F_G and F_{per} such that the volume is constant at the upper limit.	Set to zero.
F_G	Selected such that the glucose concentration, c_G , is almost constant at 7.0 g/L.	Same operation as in the growth phase.
F_{per}	Fixed at the upper limit.	Selected equal to F_G such that the volume is constant at the upper limit.
F_{out}	Set to zero.	Set to zero.
T	Fixed at the upper limit.	Fixed at the lower limit.

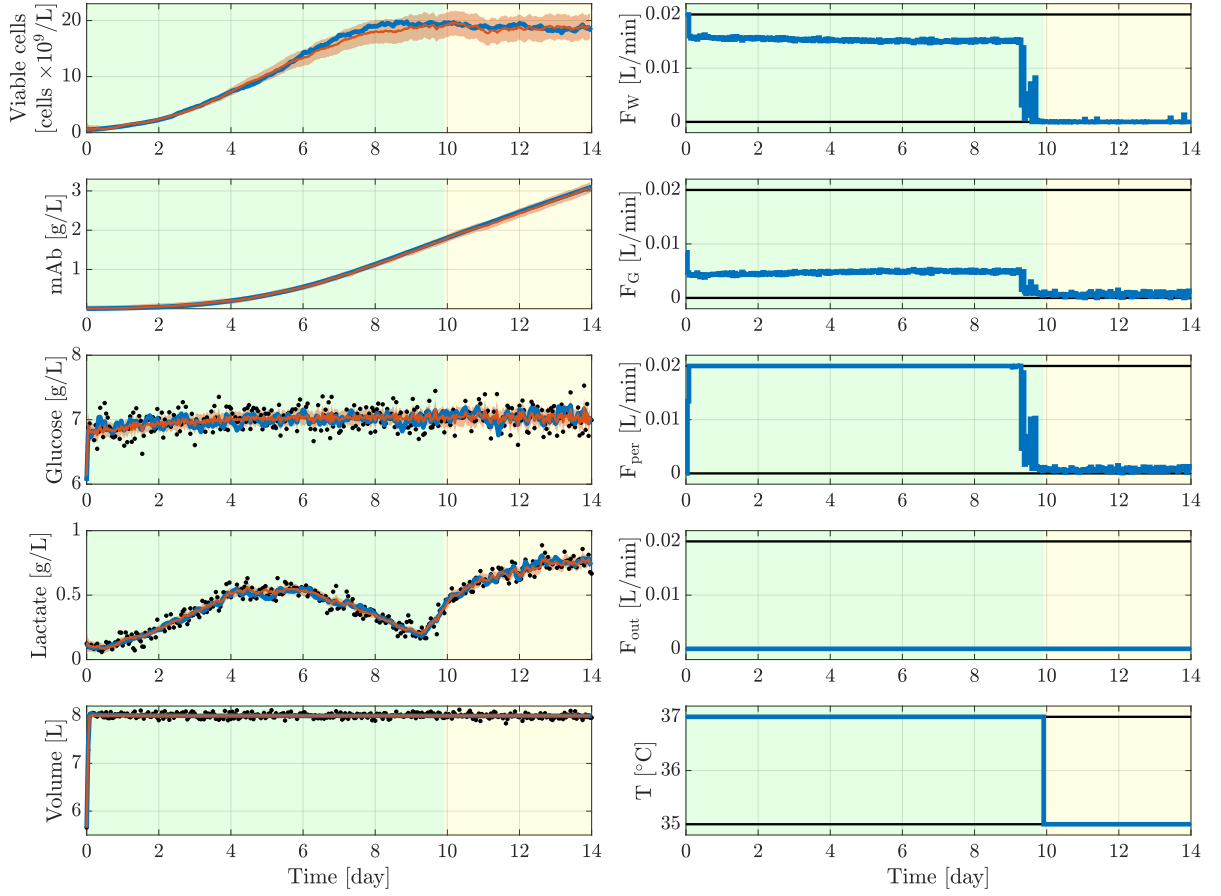


Figure 5.7: A single closed-loop simulation with the ENMPC algorithm. The blue lines are the simulation of the CPR, the red lines are estimates from the CD-EKF together with the red shaded 95% confidence interval for the estimates, and the black dots are measurements. We observe that the ENMPC algorithm operates the reactor in two phases: Growth phase (shaded green) and production phase (shaded yellow). The production phase is initialized after approximately 10 days, where the controller decreases the temperature. This figure originally appeared in (Kaysfeld and Jørgensen, 2023).

5.3.3 Uncertainty quantification

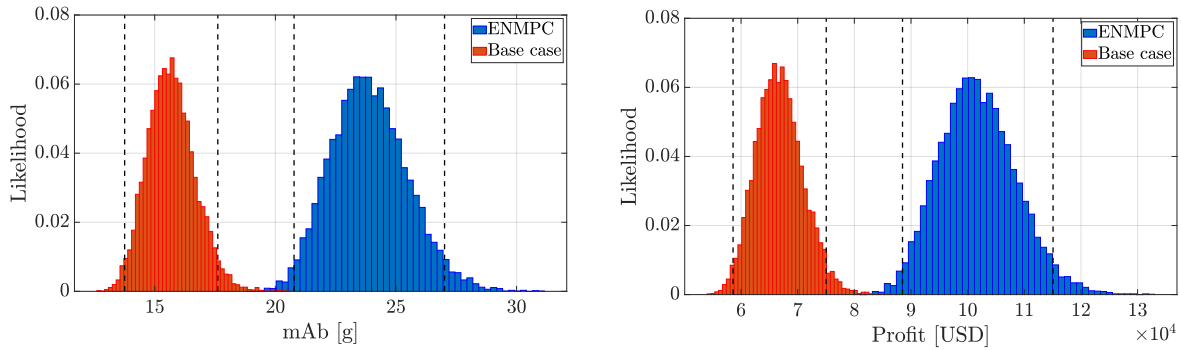
We apply the Monte Carlo simulation toolbox to perform UQ for the mAb fermentation process. We select two KPIs to evaluate the performance of the fermentation process given as

$$\Phi_1 = m_P(t_f) - m_P(t_0), \quad (\text{mAb production}) \quad (5.6a)$$

$$\Phi_2 = P_P(m_P(t_f) - m_P(t_0)) - P_G \sum_{i=0}^{N_{\text{sim}}-1} c_{G,\text{in}} F_{G,i} T_s, \quad (\text{Profit}) \quad (5.6b)$$

where $F_{G,i} = F_G(t_i)$ is the implemented glucose flow rate at time t_i and N_{sim} denotes the number of samples in the closed-loop simulation. The KPIs (5.6) are valid since $F_{\text{out}} = 0$ throughout closed-loop simulations. We gather KPI data in 10,000 stochastic closed-loop simulations with different process noise realizations. The simulations took approximately 6 h on 6 CPU cores. Figure 5.8 presents KPI histograms for the ENMPC algorithm and the base case strategy. The two KPIs have very similar distributions likely due to the low price of glucose compared to mAb. Table 5.2 presents selected statistical data for the simulations. The ENMPC algorithm increases the mean profit, but also increases the uncertainty measured in the range (difference between smallest and largest value) compared to the base case. The ENMPC algorithm operates the process close to a point of glucose inhibition, which might explain the increased uncertainty. There are no confidence interval overlap between the two operational strategies indicating a high statistical probability for the ENMPC algorithm to have higher profit.

We consider the CPU time for each call to the ENMPC algorithm in the 10,000 Monte Carlo simulations. Figure 5.9 presents a histogram for the CPU time of the $14 \cdot 24 \cdot 10,000 = 3.36 \cdot 10^6$ calls to the ENMPC algorithm. The Monte Carlo simulations show that in 99.9% of the calls, the computation time is less than 0.35 s and that the worst case CPU time is 3.2 s. The CPU times indicate that the algorithm is applicable in real time applications, where the time to compute the ENMPC response is negligible compared to the sampling time of $T_s = 60$ min. The low computation time is important for computational tractability of large-scale Monte Carlo simulation studies, where even lower CPU times would be beneficial and should be achievable by improving the computational efficiency of NLPSQP.



(a) Distribution of mAb production for the ENMPC algorithm and the base case strategy. The mean production is 23.89 g for the ENMPC algorithm and 15.68 g in the base case.

(b) Distribution of profit for the ENMPC algorithm and the base case strategy. The mean profit is 10.2×10^4 USD for the ENMPC algorithm and 6.68×10^4 USD in the base case.

Figure 5.8: Histograms of the mAb production and the profit based on 10,000 Monte Carlo simulations with the ENMPC algorithm in closed-loop and the base case strategy in open-loop. The black dashed lines show 95% confidence intervals. This figure originally appeared in (Kaysfeld and Jørgensen, 2023).

Table 5.2: Statistical data for the mAb production and the profit based on 10,000 closed-loop Monte Carlo simulations. The profit numbers (except percentages) are $\times 10^4$. The increase in percentage is relative to the base case, i.e., $(x_{\text{ENMPC}} - x_{\text{base case}})/x_{\text{base case}}$. This table originally appeared in (Kaysfeld and Jørgensen, 2023).

mAb production												
	Mean		min		max		Range		Std		95% CI	
ENMPC	23.89	[g]	18.83	[g]	31.13	[g]	12.29	[g]	1.59	[g]	[20.78, 27.04]	[g]
Base case	15.68	[g]	12.69	[g]	20.56	[g]	7.87	[g]	0.99	[g]	[13.75, 17.61]	[g]
Increase	52	[%]	48	[%]	51	[%]	56	[%]	62	[%]	[51, 53]	[%]

Profit $\times 10^4$												
	Mean		min		max		Range		Std		95% CI	
ENMPC	10.18	[USD]	8.02	[USD]	13.26	[USD]	5.24	[USD]	0.68	[USD]	[8.85, 11.51]	[USD]
Base case	6.68	[USD]	5.41	[USD]	8.76	[USD]	3.35	[USD]	0.42	[USD]	[5.86, 7.50]	[USD]
Increase	52	[%]	48	[%]	51	[%]	56	[%]	62	[%]	[51, 53]	[%]

5.3.4 Simple controller design and comparison

We apply the “*from-simple-via-complex-to-lucid*” approach and use the input key insights to design a simple controller (Stoustrup, 2013). We let a PI controller select the glucose flow rate, F_G , to track a glucose concentration of $c_G = 7.0$ g/L and set the remaining inputs according to Table 5.1. We apply the Monte Carlo simulation toolbox to quantify the performance of the simple controller. Figure 5.10 presents histograms of the mAb production for the closed-loop with the ENMPC algorithm and the simple controller. The simple controller and the ENMPC algorithm have practically identical performance.

5.4 Summary

In this chapter, we presented the main numerical results of the thesis. First, we demonstrated that the implemented Monte Carlo simulation toolbox has almost linear parallel scaling for closed-loop systems with a PID controller. We showed that Monte Carlo simulations are applicable for tuning of a PID controller and for UQ of a biomass fermentation process conducted in an FBR. Then, we showed similar linear parallel scaling for Monte Carlo simulations with the implemented target-tracking NMPC algorithm. We applied Monte Carlo simulation to quantify the performance of the NMPC algorithm for temperature set-point tracking of an exothermic chemical reaction conducted in an adiabatic CSTR. Finally, we applied the developed ENMPC algorithm for profit maximization of an mAb fermentation process conducted in a CPR. The profit was based on mAb and glucose prices. We quantified the ENMPC algorithm performance with Monte Carlo simulation, which showed a high statistical probability for the ENMPC algorithm to increase the profit of the process compared to a base case strategy. From the operational insights provided by the ENMPC algorithm, we developed a simple

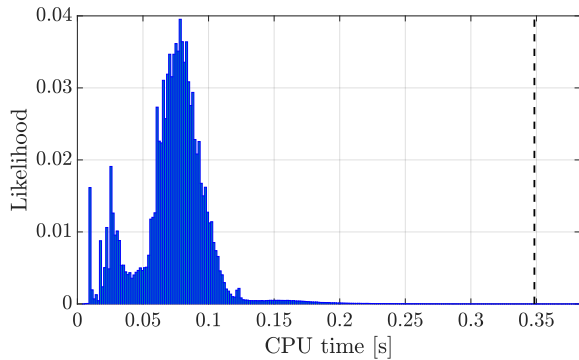


Figure 5.9: Histogram of CPU time for calls to the ENMPC algorithm in 10,000 Monte Carlo simulations. The black dashed line shows the 99.9% quantile, i.e., 99.9% of the CPU times are below 0.35 s. The worst case CPU time is 3.2 s. This figure originally appeared in (Kaysfeld and Jørgensen, 2023).

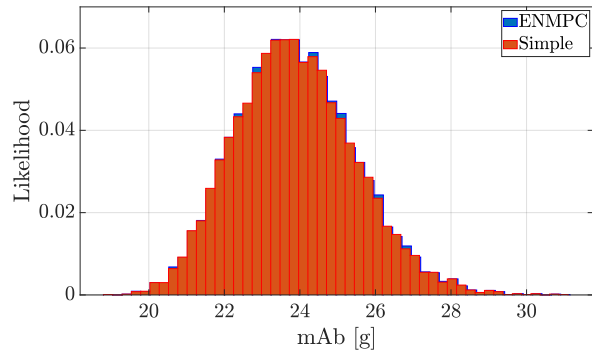


Figure 5.10: Histograms of mAb production for the ENMPC algorithm and the simple controller. The two controllers have practically identical performance. This figure originally appeared in (Kaysfeld and Jørgensen, 2023).

controller. Monte Carlo simulation showed that the simple controller and the ENMPC algorithm had practically identical performance. This result indicates that we can 1) utilize ENMPC for potentially increasing the profit in mAb fermentation processes either by implementation of ENMPC algorithms in the process or by collecting operational insights from ENMPC simulations to design simpler controllers and 2) apply Monte Carlo simulation to quantify the performance of the ENMPC algorithm and in this case show that the simple controller has practically the same performance as the ENMPC algorithm.

CHAPTER 6

Conclusions

In this chapter, we present the conclusions of the thesis. The main contributions of this work is 1) a Monte Carlo simulation toolbox for uncertainty quantification (UQ) of closed-loop systems, 2) a systematic modeling methodology for processes consisting of reactive systems conducted in reactors, and 3) an economic nonlinear model predictive control (ENMPC) algorithm for profit maximization applied to a monoclonal antibody (mAb) fermentation process.

This thesis presented a high-performance Monte Carlo simulation toolbox for UQ of closed-loop systems parallelized with Open Multi-Processing (OpenMP) for shared memory architectures. We introduced the procedure for simulation of closed-loop systems consisting of a stochastic continuous-discrete system and a controller. The stochastic continuous-discrete system involved a stochastic differential equation (SDE), a static measurement function corrupted by measurement noise, and a static output function. The controller consisted of an estimator, a regulator, and a predictor. We applied an Euler-Maruyama scheme to discretize the SDE and express the closed-loop system in discrete time. The Monte Carlo simulation approach was based on repeated simulation of the closed-loop system with different realizations of selected uncertain quantities in the process. During Monte Carlo simulation, we collected key performance indicator (KPI) data. The distribution of KPI data over large-scale Monte Carlo simulations was applicable for UQ of the closed-loop system. We implemented the Monte Carlo simulation toolbox in C for computational tractability of large-scale Monte Carlo simulations. The toolbox is thread-safe due to external memory allocation and internal distribution of the allocated memory. The thread-safe implementation enabled almost linear parallel scaling for closed-loop simulations with both proportional-integral-derivative (PID) controllers and nonlinear model predictive control (NMPC) algorithms.

The thesis introduced a systematic modeling methodology for processes consisting of reactive systems conducted in reactors. The methodology enabled complete separation of the reactor model and the reaction model. We demonstrated the methodology for three different reactors and three different reactive systems. A selected combination of reactors and reactions provided three ordinary differential equation (ODE) models, which we extended with diffusion terms to form SDE models. The models were for 1) a biomass fermentation process conducted in a fed-batch reactor (FBR), 2) an exothermic chemical reaction conducted in an adiabatic continuous stirred tank reactor (CSTR), and 3) an mAb fermentation process conducted in a continuous perfusion reactor (CPR). We applied the models for simulation and in NMPC algorithms.

The thesis presented an NMPC formulation consisting of an estimator and a regulator. The estimator was the continuous-discrete extended Kalman filter (CD-EKF)

and the regulator solved an optimal control problem (OCP) based on state estimates. The OCP had a general objective function, and we applied a multiple-shooting method to transcribe the infinite dimensional OCP to a finite dimensional nonlinear programming problem (NLP). We applied the sequential quadratic programming (SQP) algorithm NLPSQP to solve the resulting NLP. NLPSQP solves the quadratic programming problem (QP)-subproblem with the Riccati recursion based primal-dual interior-point (IP) algorithm QPIPM. We introduced two different objective functions resulting in a target-tracking NMPC algorithm and an ENMPC algorithm for profit maximization. The NMPC algorithms were thread-safe and integrated in the Monte Carlo simulation toolbox.

Finally, the thesis introduced the main numerical results, which showed that the Monte Carlo simulation toolbox had almost linear parallel scaling with both PID controllers and the developed NMPC algorithms. We applied the toolbox for UQ of 1) biomass production in the FBR, 2) temperature set-point tracking for the chemical reaction in the CSTR, and 3) profit and mAb production in the mAb fermentation process. These results demonstrated applications of the Monte Carlo simulation toolbox for UQ of three closed-loop systems with respect to different KPIs. In particular for the mAb fermentation process, we showed that the ENMPC algorithm improved the profit of the process compared to a base case operational strategy. The ENMPC strategy was summarized in a set of simple operational insights, which we applied to develop a simple controller. Monte Carlo simulation showed that the simple controller and the ENMPC algorithm had practically identical performance. This result was an example of the “*from-simple-via-complex-to-lucid*” approach, where the complex ENMPC algorithm contributed to the design of a simple controller with practically identical performance.

In conclusion, we developed a Monte Carlo simulation approach for UQ of closed-loop systems and demonstrated its parallel scaling properties for both PID controllers and NMPC algorithms. We developed an ENMPC algorithm for an mAb fermentation process and showed, with Monte Carlo simulation, that the ENMPC algorithm improved the profit of the process. Finally, we applied the “*from-simple-via-complex-to-lucid*” approach to implement a simple controller with similar performance as the ENMPC algorithm. These results are intended as the first step in the development of a fully integrated cloud-based high-performance Monte Carlo simulation framework for UQ of closed-loop systems and ENMPC technology for biotechnological processes.

6.1 Suggestions for future work

In this section, we discuss suggestions for future work based on the work in this thesis. We consider 1) the Monte Carlo simulation toolbox, 2) the optimization software, and 3) perspectives on mAb production and design of the ENMPC algorithm.

Monte Carlo simulation toolbox. The current version of the toolbox is written purely in C and application of the toolbox requires knowledge of the low-level programming language. We suggest to implement high-level interfaces in, e.g., Python and

Matlab, which would make the toolbox accessible for non-expert users. The toolbox is parallelized for shared memory architectures with OpenMP. We suggest to implement a distributed memory version with, e.g., Message Passing Interface (MPI), to enable applications with more central processing unit (CPU) cores. The distributed memory version would bring the toolbox closer to the vision of a cloud-based tool, where high-performance Monte Carlo simulations can be computed remotely for UQ of closed-loop systems.

Optimization software. The software packages QPIPM and NLPSQP are initial versions and are currently still in development. We suggest to run a detailed profiling of the software to detect points in the code, where computational efficiency could be improved. The packages include a Matlab version for development purposes. In Matlab, we implemented an integrated option to apply soft state constraints, which was not implemented in C due to time limitations of this project. We suggest to implement the soft state constraint option in C based on the Matlab version. This would provide more options to design NMPC algorithms with soft state constraints in the regulator. Finally, the main computational burden of NLPSQP lies in solving the QP-subproblem, i.e., the call to QPIPM. We suggest to investigate open-source options to solve the QP-subproblem that might improve computational performance. One option is HPIPM, which is also Riccati recursion based, have shown good computational properties for QPs in the required form, and should be thread-safe (Frison and Diehl, 2020).

MAb production. The results in this thesis were based on a rather simple model for mAb fermentation. While the model is previously validated with experimental data (Kumar et al., 2022), there are many factors not included in the model, e.g., pH dynamics and dissolved oxygen. We suggest to investigate important factors for inclusion in the model. This work considered the upstream part of the mAb production process. However, mAb production includes a number of purification steps after the upstream process, which usually include different types of chromatography processes. We suggest to apply the developed Monte Carlo simulation toolbox, modeling methodology, and ENMPC technology for simulation and optimal control of downstream processes for mAb production. Finally, we suggest to consider plant-wide optimization of the combined upstream and downstream process for mAb production and develop, e.g., an ENMPC algorithm for plant-wide control.

In this work, the ENMPC design included only the price of mAbs and glucose in the economic regulator. We suggest to include more important economic factors of the process to get a better measure of the profit, e.g., expenses of substrate feeds besides glucose. Finally, the ENMPC algorithm disregarded the value of harvested mAb, which as expected resulted in no outlet flow throughout the fermentation, i.e., $F_{\text{out}} = 0$. For generality, we suggest to include the value of harvested mAb and let the controller choose whether or not harvesting is beneficial. In recent work, we performed dynamic optimization studies, where we included the value of harvested mAb in the economic objective function. The initial results indicated that harvesting is not beneficial in a 14 days fermentation, but might be beneficial in a longer fermentation.

Bibliography

- Aehle, M., Bork, K., Schaepe, S., Kuprijanov, A., Horstkorte, R., Simutis, R., Lübbert, A., 2012. Increasing batch-to-batch reproducibility of CHO-cell cultures using a model predictive control approach. *Cytotechnology* 64, 623–634. doi:10.1007/s10616-012-9438-1.
- Alessandri, A., Baglietto, M., Battistelli, G., Zavala, V., 2010. Advances in moving horizon estimation for nonlinear systems, in: *Proceedings of the 49th IEEE Conference on Decision and Control (CDC)*, pp. 5681–5688. doi:10.1109/CDC.2010.5718126.
- Allgöwer, F., Findeisen, R., Nagy, Z.K., 2004. Nonlinear model predictive control: From theory to application. *Journal of The Chinese Institute of Chemical Engineers* 35, 299–315. doi:10.6967/JCICE.200405.0299.
- Amrit, R., Rawlings, J.B., Angeli, D., 2011. Economic optimization using model predictive control with a terminal cost. *Annual Reviews in Control* 35, 178–186. doi:10.1016/j.arcontrol.2011.10.011.
- Andersson, J., 2013. A general-purpose software framework for dynamic optimization. Ph.D. thesis. KU Leuven.
- Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M., 2019. CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation* 11, 1–36. doi:10.1007/s12532-018-0139-4.
- Angeli, D., Amrit, R., Rawlings, J.B., 2012. On average performance and stability of economic model predictive control. *IEEE Transactions on Automatic Control* 57, 1615–1626. doi:10.1109/TAC.2011.2179349.
- Badr, S., Okamura, K., Takahashi, N., Ubbenjans, V., Shirahata, H., Sugiyama, H., 2021. Integrated design of biopharmaceutical manufacturing processes: Operation modes and process configurations for monoclonal antibody production. *Computers & Chemical Engineering* 153, 107422. doi:10.1016/j.compchemeng.2021.107422.
- Bailey, J.E., 1998. Mathematical modeling and analysis in biochemical engineering: Past accomplishments and future opportunities. *Biotechnology Progress* 14, 8–20. doi:10.1021/bp9701269.

- Bartlett, R.A., Biegler, L.T., 2006. QPSchur: A dual, active-set, Schur-complement method for large-scale and structured convex quadratic programming. *Optimization Engineering* 7, 5–32. doi:10.1007/s11081-006-6588-z.
- Biegler, L.T., 1998. Advances in nonlinear programming concepts for process control. *Journal of Process Control* 8, 301–311. doi:10.1016/s0959-1524(98)00009-2.
- Biegler, L.T., 2007. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification* 46, 1043–1053. doi:10.1016/j.cep.2006.06.021.
- Biegler, L.T., Grossmann, I.E., 2004. Retrospective on optimization. *Computers & Chemical Engineering* 28, 1169–1192. doi:10.1016/j.compchemeng.2003.11.003.
- Binder, T., Blank, L., Bock, H.G., Bulirsch, R., Dahmen, W., Diehl, M., Kronseder, T., Marquardt, W., Schlöder, J.P., von Stryk, O., 2001. Introduction to model based optimization of chemical processes on moving horizons, in: *Online Optimization of Large Scale Systems*. Springer-Verlag Berlin Heidelberg, pp. 295–339. doi:10.1007/978-3-662-04331-8_18.
- Bock, H.G., Plitt, K.J., 1984. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes* 17, 1603–1608. doi:10.1016/S1474-6670(17)61205-9.
- Butcher, J.C., 2000. Numerical methods for ordinary differential equations in the 20th century. *Journal of Computational and Applied Mathematics* 125, 1–29. doi:10.1016/S0377-0427(00)00455-6.
- Butler, M., 2005. Animal cell cultures: recent achievements and perspectives in the production of biopharmaceuticals. *Applied Microbiology and Biotechnology* 68, 283–291. doi:10.1007/s00253-005-1980-8.
- Carter, P., 2001. Improving the efficacy of antibody-based cancer therapies. *Nature Reviews Cancer* 1, 118–129. doi:10.1038/35101072.
- Craven, S., Shirsat, N., Whelan, J., Glennon, B., 2012. Process model comparison and transferability across bioreactor scales and modes of operation for a mammalian cell bioprocess. *Biotechnology Progress* 29, 186–196. doi:10.1002/btpr.1664.
- Craven, S., Whelan, J., Glennon, B., 2014. Glucose concentration control of a fed-batch mammalian cell bioprocess using a nonlinear model predictive controller. *Journal of Process Control* 24, 344–357. doi:10.1016/j.jprocont.2014.02.007.
- Dagum, L., Menon, R., 1998. OpenMP: an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering* 5, 46–55. doi:10.1109/99.660313.

- D'Apuzzo, M., De Simone, V., di Serafino, D., 2010. On mutual impact of numerical linear algebra and large-scale optimization with focus on interior point methods. *Computational Optimization and Applications* 45, 283–310. doi:10.1007/s10589-008-9226-1.
- Dean, J., Reddy, P., 2013. Metabolic analysis of antibody producing CHO cells in fed-batch production. *Biotechnology and Bioengineering* 110, 1735–1747. doi:10.1002/bit.24826.
- Dewasme, L., Fernandes, S., Amribt, Z., Santos, L.O., Bogaerts, P., Vande Wouwer, A., 2015. State estimation and predictive control of fed-batch cultures of hybridoma cells. *Journal of Process Control* 30, 50–57. doi:10.1016/j.jprocont.2014.12.006.
- Diehl, M., Amrit, R., Rawlings, J.B., 2011. A Lyapunov function for economic optimizing model predictive control. *IEEE Transactions on Automatic Control* 56, 703–707. doi:10.1109/TAC.2010.2101291.
- Diehl, M., Bock, H.G., Schlöder, J.P., Findeisen, R., Nagy, Z., Allgöwer, F., 2002. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control* 12, 577–585. doi:10.1016/S0959-1524(01)00023-3.
- Diehl, M., Ferreau, H.J., Haverbeke, N., 2009. Efficient numerical methods for nonlinear MPC and moving horizon estimation, in: Magni, L., Raimondo, D.M., Allgöwer, F. (Eds.), *Nonlinear Model Predictive Control*. Springer. volume 384 of *Lecture Notes in Control and Information Sciences*, pp. 391–417. doi:10.1007/978-3-642-01094-1_32.
- Drejer, A., Ritschel, T., Jørgensen, S.B., Jørgensen, J.B., 2017. Economic optimizing control for single-cell protein production in a U-loop reactor, in: Espuña, A., Graells, M., Puigjaner, L. (Eds.), *27th European Symposium on Computer Aided Process Engineering (ESCAPE)*. volume 40 of *Computer Aided Chemical Engineering*, pp. 1759–1764. doi:10.1016/B978-0-444-63965-3.50295-6.
- Eaton, J.W., Rawlings, J.B., Edgar, T.F., 1988. Model-predictive control and sensitivity analysis for constrained nonlinear processes. *IFAC Proceedings Volumes* 21, 129–135. doi:10.1016/B978-0-08-035735-5.50022-X.
- Ecker, D.M., Jones, S.D., Levine, H.L., 2015. The therapeutic monoclonal antibody market. *mAbs* 7, 9–14. doi:10.4161/19420862.2015.989042.
- Ellis, M., Durand, H., Christofides, P.D., 2014. A tutorial review of economic model predictive control methods. *Journal of Process Control* 24, 1156–1178. doi:10.1016/j.jprocont.2014.03.010.
- Erickson, L.E., 2009. Bioreactors, in: Schaechter, M. (Ed.), *Encyclopedia of Microbiology* (3rd ed.). Elsevier, pp. 206–211. doi:10.1016/B978-012373944-5.00136-X.

- Fan, Y., Jimenez Del Val, I., Müller, C., Wagtberg Sen, J., Rasmussen, S.K., Kontoravdi, C., Weilguny, D., Andersen, M.R., 2015. Amino acid and glucose metabolism in fed-batch CHO cell culture affects antibody production and glycosylation. *Biotechnology and Bioengineering* 112, 521–535. doi:10.1002/bit.25450.
- Fletcher, R., 1987. *Practical Methods of Optimization*. 2nd ed., Wiley. doi:10.1002/9781118723203.
- Fong, W., Zhang, Y., Yung, P., 1997. Optimization of monoclonal antibody production: combined effects of potassium acetate and perfusion in a stirred tank bioreactor. *Cytotechnology* 24, 47–54. doi:10.1023/A:1007914004727.
- Forbes, M.G., Patwardhan, R.S., Hamadah, H., Gopaluni, R.B., 2015. Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine* 48, 531–538. doi:10.1016/j.ifacol.2015.09.022.
- Forsgren, A., Gill, P.E., Wright, M.H., 2002. Interior methods for nonlinear optimization. *SIAM Review* 44, 525–597.
- Frison, G., Diehl, M., 2020. HPIPM: a high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine* 53, 6563–6569. doi:10.1016/j.ifacol.2020.12.073.
- Frison, G., Jørgensen, J.B., 2013. Efficient implementation of the riccati recursion for solving linear-quadratic control problems, in: *Proceedings of the IEEE International Conference on Control Applications (CCA)*, pp. 1117–1122. doi:10.1109/CCA.2013.6662901.
- Frison, G., Kouzoupis, D., Sartor, T., Zanelli, A., Diehl, M., 2018. BLASFEO: Basic linear algebra subroutines for embedded optimization. *ACM Transactions on Mathematical Software* 44. doi:10.1145/3210754.
- Frison, G., Sartor, T., Zanelli, A., Diehl, M., 2020. The BLAS API of BLASFEO: Optimizing performance for small matrices. *ACM Transactions on Mathematical Software* 46. doi:10.1145/3378671.
- Frogerais, P., Bellanger, J.J., Senhadji, L., 2012. Various ways to compute the continuous-discrete extended Kalman filter. *IEEE Transactions on Automatic Control* 57, 1000–1004. doi:10.1109/TAC.2011.2168129.
- Garcia, C.E., Prett, D.M., 1986. Advances in industrial model-predictive control, in: *Chemical Process Control (CPCIII), Proceedings of the Third International Conference on Chemical Process Control*, pp. 245–293.
- Gebremedhin, A.H., Walther, A., 2019. An introduction to algorithmic differentiation. *WIREs Data Mining and Knowledge Discovery* 10. doi:10.1002/widm.1334.
- Gill, P.E., Murray, W., Wright, M.H., 1999. *Practical Optimization*. Academic Press.

- Gill, P.E., Wong, E., 2012. Sequential quadratic programming methods, in: Lee, J., Leyffer, S. (Eds.), *Mixed Integer Nonlinear Programming*. Springer. volume 154 of *The IMA Volumes in Mathematics and its Applications*, pp. 147–224. doi:10.1007/978-1-4614-1927-3_6.
- Gillijns, S., Mendoza, O.B., Chandrasekar, J., De Moor, B.L.R., Bernstein, D.S., Riedley, A., 2006. What is the ensemble kalman filter and how well does it work?, in: *Proceedings of the 2006 American Control Conference (ACC)*, pp. 4448–4453. doi:10.1109/ACC.2006.1657419.
- Glen, K.E., Cheeseman, E.A., Stacey, A.J., Thomas, R.J., 2018. A mechanistic model of erythroblast growth inhibition providing a framework for optimisation of cell therapy manufacturing. *Biochemical Engineering Journal* 133, 28–38. doi:10.1016/j.bej.2018.01.033.
- Goldfarb, D., Idnani, A., 1983. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming* 27, 1–33. doi:10.1007/BF02591962.
- Gondzio, J., 2012. Interior point methods 25 years later. *European Journal of Operational Research* 218, 587–601. doi:10.1016/j.ejor.2011.09.017.
- Gould, N.I.M., Toint, P.L., 2000. SQP methods for large-scale nonlinear programming, in: Powell, M., Scholtes, S. (Eds.), *System Modelling and Optimization*. volume 46, pp. 149–178. doi:10.1007/978-0-387-35514-6_7.
- Grand View Research, 2023. Monoclonal antibodies market size, share & trends analysis report by source type (chimeric, murine, humanized, human), by production type (in vivo, in vitro), by application, by end-use, by region, and segment forecasts, 2023 - 2030. URL: <https://www.grandviewresearch.com/industry-analysis/monoclonal-antibodies-market>. Accessed on 2023-05-11.
- Hafeez, U., Gan, H.K., Scott, A.M., 2018. Monoclonal antibodies as immunomodulatory therapy against cancer and autoimmune diseases. *Current Opinion in Pharmacology* 41, 114–121. doi:10.1016/j.coph.2018.05.010.
- Heidarinejad, M., Liu, J., Christofides, P.D., 2012. Economic model predictive control of nonlinear process systems using Lyapunov techniques. *AIChE Journal* 58, 855–870. doi:10.1002/aic.12672.
- Hernandez, I., Bott, S.W., Patel, A.S., Wolf, C.G., Hospodar, A.R., Sampathkumar, S., Shrank, W.H., 2018. Pricing of monoclonal antibody therapies: higher if used for cancer? *The American Journal of Managed Care* 24, 109–112.
- Higham, D.J., 2001. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM Review* 43, 525–546. doi:10.1137/S0036144500378302.

- Honc, D., Sharma K., R., Abraham, A., Dušek, F., Pappa, N., 2016. Teaching and practicing model predictive control. *IFAC-PapersOnLine* 49, 34–39. doi:10.1016/j.ifacol.2016.07.149.
- Hong, M.S., Severson, K.A., Jiang, M., Lu, A.E., Love, J.C., Braatz, R.D., 2018. Challenges and opportunities in biopharmaceutical manufacturing control. *Computers & Chemical Engineering* 110, 106–114. doi:10.1016/j.compchemeng.2017.12.007.
- Huyck, B., De Brabanter, J., De Moor, B., Van Impe, J.F., Logist, F., 2014. Online model predictive control of industrial processes using low level control hardware: A pilot-scale distillation column case study. *Control Engineering Practice* 28, 34–48. doi:10.1016/j.conengprac.2014.02.016.
- Jahanshahlu, L., Rezaei, N., 2020. Monoclonal antibody as a potential anti-COVID-19. *Biomedicine & Pharmacotherapy* 129, 110337. doi:10.1016/j.biopha.2020.110337.
- Jazwinski, A.H., 1970. *Stochastic processes and filtering theory*. volume 64. Academic Press.
- Julier, S.J., Uhlmann, J.K., 2004. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE* 92, 401–422. doi:10.1109/JPROC.2003.823141.
- Jørgensen, J.B., 2004. *Moving horizon estimation and control*. Ph.D. thesis. Technical University of Denmark.
- Jørgensen, J.B., Frison, G., Gade-Nielsen, N.F., Damman, B., 2012. Numerical methods for solution of the extended linear quadratic control problem. *IFAC Proceedings Volumes* 45, 187–193. doi:10.3182/20120823-5-NL-3013.00092.
- Jørgensen, J.B., Ritschel, T.K.S., Boiroux, D., Schroll-Fleischer, E., Wahlgreen, M.R., Krogh Nielsen, M., Wu, H., Huusom, J.K., 2020. Simulation of NMPC for a laboratory adiabatic CSTR with an exothermic reaction, in: *Proceedings of the 2020 European Control Conference (ECC)*, pp. 202–207. doi:10.23919/ECC51009.2020.9143733.
- Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* 82, 35–45. doi:10.1115/1.3662552.
- Kaysfeld, M.W., Jørgensen, J.B., 2023. Uncertainty quantification of an economic nonlinear model predictive controller for monoclonal antibody production. In submission for *Journal of Process Control* .
- Kaysfeld, M.W., Kumar, D., Nielsen, M.K., Jørgensen, J.B., 2023a. Dynamic optimization for monoclonal antibody production, in: *Proceedings of the 22nd World Congress of the International Federation of Automatic Control (IFAC)*. doi:10.48550/arXiv.2302.09932.

- Kaysfeld, M.W., Zanon, M., Jørgensen, J.B., 2023b. Performance quantification of a nonlinear model predictive controller by parallel Monte Carlo simulations of a closed-loop system, in: Proceedings of the 21st European Control Conference (ECC). doi:10.48550/arXiv.2212.02197.
- Köhler, G., Milstein, C., 1975. Continuous cultures of fused cells secreting antibody of predefined specificity. *Nature* 256, 495–497. doi:10.1038/256495a0.
- Kumar, D., Gangwar, N., Rathore, A.S., Ramteke, M., 2022. Multi-objective optimization of monoclonal antibody production in bioreactor. *Chemical Engineering and Processing - Process Intensification* 180, 108720. doi:10.1016/j.cep.2021.108720.
- Kyriakopoulos, S., Ang, K.S., Lakshmanan, M., Huang, Z., Yoon, S., Gunawan, R., Lee, D.Y., 2018. Kinetic modeling of mammalian cell culture bioprocessing: The quest to advance biomanufacturing. *Biotechnology Journal* 13, 1700229. doi:10.1002/biot.201700229.
- Li, F., Vijayasankaran, N., Shen, A.Y., Kiss, R., Amanullah, A., 2010. Cell culture processes for monoclonal antibody production. *MAbs* 2, 466–479. doi:10.4161/mabs.2.5.12720.
- Li, J., Wong, C.L., Vijayasankaran, N., Hudson, T., Amanullah, A., 2012. Feeding lactate for CHO cell culture processes: Impact on culture metabolism and performance. *Biotechnology and Bioengineering* 109, 1173–1186. doi:10.1002/bit.24389.
- Liu, J.K.H., 2014. The history of monoclonal antibody development - progress, remaining challenges and future innovations. *Annals of Medicine & Surgery* 3, 113–116. doi:10.1016/j.amsu.2014.09.001.
- Liu, Y.H., Bi, J.X., Zeng, A.P., Yuan, J.Q., 2008. A simple kinetic model for myeloma cell culture with consideration of lysine limitation. *Bioprocess and Biosystems Engineering* 31, 569–577. doi:10.1007/s00449-008-0204-x.
- Lu, R.M., Hwang, Y.C., Liu, I.J., Lee, C.C., Tsai, H.Z., Li, H.J., Wu, H.C., 2020. Development of therapeutic antibodies for the treatment of diseases. *Journal of Biomedical Science* 27. doi:10.1186/s12929-019-0592-z.
- Marovich, M., Mascola, J.R., Cohen, M.S., 2020. Monoclonal antibodies for prevention and treatment of COVID-19. *JAMA* 324, 131–132. doi:10.1001/jama.2020.10245.
- Mehrotra, S., 1992. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization* 2, 575–601. doi:10.1137/0802028.
- Melero, I., Hervas-Stubbs, S., Glennie, M., Pardoll, D.M., Chen, L., 2007. Immunostimulatory monoclonal antibodies for cancer therapy. *Nature Reviews Cancer* 7, 95–106. doi:10.1038/nrc2051.

- Morari, M., Garcia, C.E., Prett, D.M., 1988. Model predictive control: Theory and practice. IFAC Proceedings Volumes 21, 1–12. doi:10.1016/B978-0-08-035735-5.50006-1.
- Nagy, Z.K., Braatz, R.D., 2003. Robust nonlinear model predictive control of batch processes. AIChE Journal 49, 1776–1786. doi:10.1002/aic.690490715.
- Nielsen, M.K., Ritschel, T.K.S., Christensen, I., Dragheim, J., Huusom, J.K., Gernaey, K.V., Jørgensen, J.B., 2023. State estimation for continuous-discrete-time nonlinear stochastic systems, in: Proceedings of Foundations of Computer Aided Process Operations / Chemical Process Control (FOCAPO / CPC). doi:10.48550/arXiv.2212.02139.
- Nocedal, J., Wright, S.J., 1999. Numerical optimization. Springer.
- Olsen, D.F., Jørgensen, J.B., Villadsen, J., Jørgensen, S.B., 2010. Modeling and simulation of single cell protein production. IFAC Proceedings Volumes 43, 502–507. doi:10.3182/20100707-3-BE-2012.0099.
- Otsubo, R., Yasui, T., 2022. Monoclonal antibody therapeutics for infectious diseases: Beyond normal human immunoglobulin. Pharmacology & Therapeutics 240. doi:10.1016/j.pharmthera.2022.108233.
- Pereira, S., Kildegaard, H.F., Andersen, M.R., 2018. Impact of CHO metabolism on cell growth and protein production: An overview of toxic and inhibiting metabolites and nutrients. Biotechnology Journal 13, 1700499. doi:10.1002/biot.201700499.
- Potra, F.A., Wright, S.J., 2000. Interior-point methods. Journal of Computational and Applied Mathematics 124, 281–302. doi:10.1016/S0377-0427(00)00433-7.
- Powell, M.J.D., 1985. On the quadratic programming algorithm of Goldfarb and Idnani, in: Cottle, R. (Ed.), Mathematical Programming: Essays in Honor of George B. Dantzig Part II. Springer, Berlin, Heidelberg. volume 25 of *Mathematical Programming Studies*, pp. 46–61. doi:10.1007/BFb0121074.
- Provost, A., Bastin, G., 2004. Dynamic metabolic modelling under the balanced growth condition. Journal of Process Control 14, 717–728. doi:10.1016/j.jprocont.2003.12.004.
- Qin, S.J., Badgwell, T.A., 2003. A survey of industrial model predictive control technology. Control Engineering Practice 11, 733–764. doi:10.1016/S0967-0661(02)00186-7.
- Rao, A.V., 2009. A survey of numerical methods for optimal control, in: Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, pp. 497–528. Preprint (AAS 09-334).

- Rao, C.V., Wright, S.J., Rawlings, J.B., 1998. Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications* 99, 723–757. doi:10.1023/A:1021711402723.
- Rawlings, J.B., 2000. Tutorial overview of model predictive control. *IEEE Control Systems Magazine* 20, 38–52. doi:10.1109/37.845037.
- Rawlings, J.B., Amrit, R., 2009. Optimizing process economic performance using model predictive control, in: Magni, L., Raimondo, D.M., Allgöwer, F. (Eds.), *Nonlinear Model Predictive Control*. Springer, Berlin, Heidelberg. volume 384 of *Lecture Notes in Control and Information Sciences*, pp. 119–138. doi:10.1007/978-3-642-01094-1_10.
- Rawlings, J.B., Angeli, D., Bates, C.N., 2012. Fundamentals of economic model predictive control, in: *Proceedings of the 51st IEEE Conference on Decision and Control (CDC)*, pp. 3851–3861. doi:10.1109/CDC.2012.6425822.
- Rawlings, J.B., Mayne, D.Q., Diehl, M., 2017. *Model predictive control: Theory, computation, and design*. 2nd ed., Nob Hill Publishing.
- Rawlings, J.B., Meadows, E.S., Muske, K.R., 1994. Nonlinear model predictive control: A tutorial and survey. *IFAC Proceedings Volumes* 27, 185–197. doi:10.1016/S1474-6670(17)48151-1.
- Ritschel, T.K.S., Boiroux, D., Nielsen, M.K., Huusom, J.K., Jørgensen, S.B., Jørgensen, J.B., 2019. Economic optimal control of a U-loop bioreactor using simultaneous collocation-based approaches, in: *Proceedings of the 3rd IEEE Conference on Control Technology and Applications (CCTA)*, pp. 933–938. doi:10.1109/CCTA.2019.8920479.
- Ryde, T.E., Wahlgreen, M.R., Nielsen, M.K., Hørsholt, S., Jørgensen, S.B., Jørgensen, J.B., 2021. Optimal feed trajectories for fedbatch fermentation with substrate inhibition kinetics. *IFAC-PapersOnLine* 54, 318–323. doi:10.1016/j.ifacol.2021.08.261.
- Schneider, R., Georgakis, C., 2013. How to not make the extended Kalman filter fail. *Industrial & Engineering Chemistry Research* 52, 3354–3362. doi:10.1021/ie300415d.
- Schofield, D., 2023. Monoclonal antibody production explained. URL: https://www.evitria.com/journal/monoclonal-antibodies/monoclonal-antibody-production/?utm_term=monoclonal%20antibody%20production&utm_campaign=Recombinant+Antibodies+Europa&utm_source=adwords&utm_medium=ppc&hsa_acc=5241823564&hsa_cam=12363551991&hsa_grp=117442612986&hsa_ad=512764200911&hsa_src=g&hsa_tgt=kwd-193918576&hsa_kw=monoclonal%20antibody%20production&hsa_mt=p&hsa_net=adwords&hsa_ver=3&gclid=Cj0KCQjwyLGjBhDKARIsAFRNgW-hXv-5SFx46h8CxFr3ifaGyfwe4feiiXIwwA1CsdW5GS2WSTbMPx8aAubkEALw_wcB. Accessed on 2023-05-25.

- Schäfer, A., Kühn, P., Diehl, M., Schlöder, J., Bock, H.G., 2007. Fast reduced multiple shooting methods for nonlinear model predictive control. *Chemical Engineering and Processing: Process Intensification* 46, 1200–1214. doi:10.1016/j.cep.2006.06.024.
- Sha, S., Huang, Z., Wang, Z., Yoon, S., 2018. Mechanistic modeling and applications for CHO cell culture development and production. *Current Opinion in Chemical Engineering* 22, 54–61. doi:10.1016/j.coche.2018.08.010.
- Sissolak, B., Lingg, N., Sommeregger, W., Striedner, G., Vorauer-Uhl, K., 2019. Impact of mammalian cell culture conditions on monoclonal antibody charge heterogeneity: an accessory monitoring tool for process development. *Journal of Industrial Microbiology and Biotechnology* 46, 1167–1178. doi:10.1007/s10295-019-02202-5.
- Sommerfeld, S., Strube, J., 2005. Challenges in biotechnology production - generic processes and process optimization for monoclonal antibodies. *Chemical Engineering and Processing: Process Intensification* 44, 1123–1137. doi:10.1016/j.cep.2005.03.006.
- Stoustrup, J., 2013. Successful industry/academia cooperation: From simple via complex to lucid solutions. *European Journal of Control* 19, 358–368. doi:10.1016/j.ejcon.2013.06.001.
- Tian, T., Burrage, K., 2001. Implicit Taylor methods for stiff stochastic differential equations. *Applied Numerical Mathematics* 38, 167–185. doi:10.1016/S0168-9274(01)00034-4.
- U.S. Food and Drug Administration (FDA), 2020. Coronavirus (COVID-19) update: FDA authorizes monoclonal antibodies for treatment of COVID-19. Press release. URL: <https://www.fda.gov/news-events/press-announcements/coronavirus-covid-19-update-fda-authorizes-monoclonal-antibodies-treatment-covid-19>. Accessed on 2023-05-25.
- U.S. Food and Drug Administration (FDA), 2021. Coronavirus (COVID-19) update: FDA authorizes monoclonal antibodies for treatment of COVID-19. Press release. URL: <https://www.fda.gov/news-events/press-announcements/coronavirus-covid-19-update-fda-authorizes-monoclonal-antibodies-treatment-covid-19-0>. Accessed on 2023-05-25.
- Vergara, M., Torres, M., Müller, A., Avello, V., Acevedo, C., Berrios, J., Reyes, J.G., Valdez-Cruz, N.A., Altamirano, C., 2018. High glucose and low specific cell growth but not mild hypothermia improve specific r-protein productivity in chemostat culture of CHO cells. *PLOS ONE* 13, 1–22. doi:10.1371/journal.pone.0202098.
- Wahlgreen, M.R., Jørgensen, J.B., 2022. On the implementation of a preconditioned riccati recursion based primal-dual interior-point algorithm for input constrained optimal control problems. *IFAC-PapersOnLine* 55, 346–351. doi:10.1016/j.ifacol.2022.07.468.

- Wahlgreen, M.R., Jørgensen, J.B., Zanon, M., 2023. Model predictive control tuning by Monte Carlo simulation and controller matching, in: Proceedings of Foundations of Computer Aided Process Operations / Chemical Process Control (FOCAPO / CPC). doi:10.48550/arXiv.2212.02142.
- Wahlgreen, M.R., Meyer, K., Ritschel, T.K.S., Engsig-Karup, A.P., Gernaey, K.V., Jørgensen, J.B., 2022. Modeling and simulation of upstream and downstream processes for monoclonal antibody production. IFAC-PapersOnLine 55, 685–690. doi:10.1016/j.ifacol.2022.07.523.
- Wahlgreen, M.R., Schroll-Fleischer, E., Boiroux, D., Ritschel, T.K.S., Wu, H., Huusom, J.K., Jørgensen, J.B., 2020. Nonlinear model predictive control for an exothermic reaction in an adiabatic CSTR. IFAC-PapersOnLine 53, 500–505. doi:10.1016/j.ifacol.2020.06.084.
- Wahlgreen, M.R., Thode Reenberg, A., Nielsen, M.K., Rydahl, A., Ritschel, T.K.S., Dammann, B., Jørgensen, J.B., 2021. A high-performance Monte Carlo simulation toolbox for uncertainty quantification of closed-loop systems, in: Proceedings of the 60th IEEE Conference on Decision and Control (CDC), pp. 6755–6761. doi:10.1109/CDC45484.2021.9682781.
- Walsh, G., 2018. Biopharmaceutical benchmarks 2018. Nature Biotechnology 36, 1136–1145. doi:10.1038/nbt.4305.
- Walsh, G., Walsh, E., 2022. Biopharmaceutical benchmarks 2022. Nature Biotechnology 40, 1722–1760. doi:10.1038/s41587-022-01582-x.
- Wang, C., Li, W., Drabek, D., Okba, N.M.A., van Haperen, R., Osterhaus, A.D.M.E., van Kuppeveld, F.J.M., Haagmans, B.L., Grosveld, F., Bosch, B.J., 2020. A human monoclonal antibody blocking SARS-CoV-2 infection. Nature Communications 11, 2251. doi:10.1038/s41467-020-16256-y.
- Wright, M.H., 2004. The interior-point revolution in optimization: History, recent developments, and lasting consequences. Bulletin of the American Mathematical Society 42, 39–56. doi:10.1090/S0273-0979-04-01040-7.
- Wächter, A., Biegler, L., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106, 25–57. doi:10.1007/s10107-004-0559-y.
- Zhou, P., Yang, X.L., Wang, X.G., Hu, B., Zhang, L., Zhang, W., Si, H.R., Zhu, Y., Li, B., Huang, C.L., Chen, H.D., Chen, J., Luo, Y., Guo, H., Jiang, R.D., Liu, M.Q., Chen, Y., Shen, X.R., Wang, X., Zheng, X.S., Zhao, K., Chen, Q.J., Deng, F., Liu, L.L., Yan, B., Zhan, F.X., Wang, Y.Y., Xiao, G.F., Shi, Z.L., 2020. A pneumonia outbreak associated with a new coronavirus of probable bat origin. Nature 579, 270–273. doi:10.1038/s41586-020-2012-7.

Part II

Appendix

APPENDIX **A**

Paper I: CDC 2021

A High-Performance Monte Carlo Simulation Toolbox for Uncertainty Quantification of Closed-loop Systems

Authors:

Morten Ryberg Wahlgreen, Asbjørn Thode Reenberg, Marcus Krogh Nielsen, Anton Rydahl, Tobias K. S. Ritschel, Bernd Dammann, John Bagterp Jørgensen

Published in:

Proceedings of the 60th IEEE Conference on Decision and Control (CDC), Austin, TX,
USA, December 14-17, 2021.

<https://doi.org/10.1109/CDC45484.2021.9682781>

A High-Performance Monte Carlo Simulation Toolbox for Uncertainty Quantification of Closed-loop Systems

Morten Ryberg Wahlgreen, Asbjørn Thode Reenberg, Marcus Krogh Nielsen, Anton Rydahl, Tobias K. S. Ritschel, Bernd Dammann, John Bagterp Jørgensen

Abstract—We apply Monte Carlo simulation for performance quantification and tuning of controllers in nonlinear closed-loop systems. Computational feasibility of large-scale Monte Carlo simulation is achieved by implementation of a parallelized high-performance Monte Carlo simulation toolbox for closed-loop systems in C for shared memory architectures. The toolbox shows almost linear scale-up on 16 CPU cores on a single NUMA node, and a scale-up of 27.3 on two NUMA nodes with a total of 32 CPU cores. We demonstrate performance quantification and tuning of a PID controller for a bioreactor in fed-batch operation. We perform 30,000 closed-loop simulations of the fed-batch reactor within 1 second. This is approximately a 2300 times computational performance increase compared to a serial reference implementation in Matlab. Additionally, we apply Monte Carlo simulation to perform automatic tuning of the PID controller based on maximizing average produced biomass within 8 seconds.

I. INTRODUCTION

In closed-loop systems, we encounter unknown quantities that need to be estimated, e.g., model parameters. Additionally, it can be beneficial to quantify controller performance. Currently, there exist well-defined methods for parameters estimation [1], [2] and tuning of controllers in linear systems [3]–[5]. However, for nonlinear systems, quantification of controller performance and tuning is not as well developed. We propose a Monte Carlo simulation brute-force technique for automatic performance quantification and tuning of controllers in linear and nonlinear systems. With the Monte Carlo approach, we can tune controllers with any performance measure, e.g., maximizing economic yield or minimizing the risk of low production, such as in modern control applications [6], [7]. The Monte Carlo simulation technique is made computationally feasible by implementation of a high-performance Monte Carlo simulation toolbox parallelized for shared memory architectures in C.

Monte Carlo simulation is a widely used technique for quantification of uncertainties. It is applied in various areas, e.g., portfolio management and epidemiology [8]–[10]. Monte Carlo simulation uses random sampling to obtain numerical results about deterministic quantities. However, the method requires many samples to be effective, and thus computational efficiency becomes a bottleneck. The development in central processing unit (CPU) technology increases the number of possible applications for Monte

Carlo simulation. However, trends in CPU development show that the clock frequency of new CPUs is no longer increasing due to power consumption and heat issues [11]. Instead, new CPUs have increased performance by increasing the number of cores. Consequently, the full potential of modern CPUs is only achieved with parallelized software executed on multi-core processors. Not all problems are parallelizable, but Monte Carlo simulation is a prime example of a parallelizable problem, as each simulation is independent of all other simulations. To achieve the full potential of Monte Carlo simulation on modern CPUs, we require state-of-the-art parallelized software in high-performance languages.

In this paper, we present closed-loop systems based on a stochastic continuous-discrete model, a stochastic differential equation (SDE) solver, and a controller. We introduce our implementation of a high-performance Monte Carlo simulation toolbox for closed-loop systems. Additionally, we introduce the SDE solvers and controllers contained in the toolbox. Furthermore, we demonstrate applications of the toolbox on a bioreactor in fed-batch operation [12]. In particular, we demonstrate that Monte Carlo simulation can be used for performance quantification and tuning of a proportional–integral–derivative (PID) controller.

The remaining part of the paper is organized as follows. Section II introduces the stochastic continuous-discrete system, two SDE solvers, and four controllers of increasing complexity. Section III presents the Monte Carlo simulation scheme for closed-loop systems and introduces our toolbox for Monte Carlo simulation. Section IV presents an example application of the toolbox. Section V presents our conclusion.

II. CLOSED-LOOP SIMULATIONS

This section presents our representation of closed-loop systems for simulation. Our simulations consist of 1) an SDE model with discrete measurements, represented as a stochastic continuous-discrete model, 2) an SDE solver, and 3) a controller.

A. Stochastic continuous-discrete system

We consider stochastic continuous-discrete systems in the form

$$x(t_0) = x_0, \quad (1a)$$

$$dx(t) = f(t, x(t), u(t), d(t), p_f)dt + \sigma(t, x(t), u(t), d(t), p_\sigma)d\omega(t), \quad (1b)$$

$$y(t_k) = g(t_k, x(t_k), p_g) + v(t_k, p_v), \quad (1c)$$

$$z(t) = h(t, x(t), p_h), \quad (1d)$$

M. R. Wahlgreen, A. T. Reenberg, M. K. Nielsen, A. Rydahl, T. K. S. Ritschel, B. Dammann, and J. B. Jørgensen are with the Department of Applied Mathematics and Computer Science, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark.

Corresponding author: J. B. Jørgensen (E-mail: jbj@dtu.dk).

where $x(t)$ are the states, $u(t)$ are inputs, $d(t)$ are disturbances, and $p_f, p_\sigma, p_g, p_v,$ and p_h are parameters. Additionally, x_0 is a normally distributed initial condition, $\omega(t)$ is a standard Wiener process, and $v(t_k, p_v)$ is normally distributed measurement noise at discrete time, i.e.,

$$x_0 \sim N(\bar{x}_0, P_0), \quad (2a)$$

$$d\omega(t) \sim N_{iid}(0, Idt), \quad (2b)$$

$$v(t_k, p_v) \sim N_{iid}(0, R(t_k, p_v)). \quad (2c)$$

Measurements are sampled with sampling time Δt , such that, $t_{k+1} = t_k + \Delta t$. We use zero-order-hold parameterization of the inputs and disturbances:

$$u(t) = u_k, \quad t_k \leq t < t_{k+1}, \quad (3a)$$

$$d(t) = d_k, \quad t_k \leq t < t_{k+1}. \quad (3b)$$

B. Stochastic differential equation solvers

SDE solvers are required to simulate stochastic continuous-discrete systems in the form (1). We consider an explicit-explicit (Euler-Maruyama) solver and an implicit-explicit solver [13], [14].

1) Explicit-explicit (Euler-Maruyama):

$$t_{k,n+1} = t_{k,n} + \Delta t, \quad (4a)$$

$$x_{k,n+1} = x_{k,n} + f(t_{k,n}, x_{k,n}, u_k, d_k, p_f)\Delta t + \sigma(t_{k,n}, x_{k,n}, u_k, d_k, p_\sigma)\Delta\omega_{k,n}, \quad (4b)$$

2) Implicit-explicit:

$$t_{k,n+1} = t_{k,n} + \Delta t, \quad (5a)$$

$$x_{k,n+1} = x_{k,n} + f(t_{k,n+1}, x_{k,n+1}, u_k, d_k, p_f)\Delta t + \sigma(t_{k,n}, x_{k,n}, u_k, d_k, p_\sigma)\Delta\omega_{k,n}, \quad (5b)$$

where $t_{k,0} = t_k, x_{k,0} = x_k,$ and $\Delta\omega_{k,n} \sim N_{iid}(0, I\Delta t)$.

Let N_k denote the number of steps of size Δt in the interval $[t_k, t_{k+1}]$. Then

$$t_{k+1} = t_{k,N_k}, \quad (6a)$$

$$x_{k+1} = x_{k,N_k}. \quad (6b)$$

The explicit-explicit solver is suitable for non-stiff systems, whereas the implicit-explicit solver is suitable for stiff systems.

We let Φ represent the discretization of the state equation, (1b), with either the explicit-explicit solver or the implicit-explicit solver. To compactly describe simulation of closed-loop systems, we introduce the notation for a discretized version of (1),

$$x_{k+1} = \Phi(t_k, x_k, u_k, d_k, w_k, p_f, p_\sigma), \quad (7a)$$

$$y_k = g(t_k, x_k, p_g) + v_k, \quad (7b)$$

$$z_k = h(t_k, x_k, p_h), \quad (7c)$$

where $v_k = v(t_k, p_v)$.

C. Controller

The digital discrete-time controller in typical model-based control applications is represented as the dynamic system

$$x_k^c = \kappa(t_{k-1}, x_{k-1}^c, y_k, u_{k-1}, p_\kappa), \quad (8a)$$

$$u_k = \lambda(t_k, x_k^c, p_\lambda), \quad (8b)$$

$$z_k^c = \mu(t_k, x_k^c, p_\mu), \quad (8c)$$

where x_k^c are estimated states, z_k^c are predictions of the outputs and manipulated inputs, $\kappa(\cdot)$ is a state estimator, $\lambda(\cdot)$ is a regulator, and $\mu(\cdot)$ is a predictor.

We consider four controllers of increasing complexity.

1) *Open-loop controller (no feedback)*: The open-loop controller does not include feedback and outputs a target value for the inputs

$$u_k = \lambda(\cdot) = \bar{u}_k. \quad (9)$$

The functions $\kappa(\cdot)$ and $\mu(\cdot)$ are not necessary for the open-loop controller.

2) *Proportional-integral-derivative controller*: The continuous PID controller is given by

$$u(t) = \bar{u}(t) + K_p e(t) + K_i \int_{t_0}^t e(\tau) d\tau + K_d \frac{de(t)}{dt}, \quad (10)$$

where $K_p, K_i,$ and K_d are gain constants. The error, $e(t)$, is the difference between the set point, $\bar{y}(t)$, and the measurement, $y(t)$, i.e.,

$$e(t) = \bar{y}(t) - y(t). \quad (11a)$$

Notice, for the PID controller the output is assumed to be measured, i.e., $\bar{y}(t) = \bar{z}(t)$. It can be advantageous to let the derivative term act on the measurements, $y(t)$, rather than the error, $e(t)$ [15], [16]. We get,

$$u(t) = \bar{u}(t) + K_p e(t) + K_i \int_{t_0}^t e(\tau) d\tau - K_d \frac{dy(t)}{dt}. \quad (12)$$

We discretize the PID controller (12) as

$$e_k = \bar{y}_k - y_k^F, \quad (13a)$$

$$P_k = K_p e_k, \quad (13b)$$

$$I_k = I_{k-1} + T_s K_i e_k, \quad (13c)$$

$$D_k = -\frac{K_d}{T_s} (y_k^F - y_{k-1}^F), \quad (13d)$$

$$u_k = \bar{u}_k + P_k + I_k + D_k, \quad (13e)$$

where T_s is the sampling time and filtered measurements, y_k^F , are computed from the discrete-time low-pass filter

$$y_k^F = (1 - \alpha)y_{k-1}^F + \alpha y_k, \quad (14)$$

with $\alpha \in [0, 1]$.

For the PID controller, $\lambda(\cdot)$ is given by (13e), $\kappa(\cdot)$ is given by (14), and $\mu(\cdot)$ is not necessary.

3) *PID controller with clipping*: We incorporate input bounds with clipping. The PID controller with clipping is,

$$\tilde{u}_k = \bar{u}_k + P_k + I_k + D_k, \quad (15a)$$

$$u_k = \max(u_{\min}, \min(u_{\max}, \tilde{u}_k)). \quad (15b)$$

4) *Nonlinear model predictive control*: The nonlinear model predictive controller (NMPC) includes a continuous-discrete extended Kalman filter (CD-EKF) based on (1) for state estimation and prediction [17], [18]. Given the filtered state-covariance pair, $\hat{x}_{k-1|k-1}$ and $P_{k-1|k-1}$, the CD-EKF obtains a one-step prediction

$$\hat{x}_{k|k-1} = \hat{x}_{k-1}(t_k), \quad (16a)$$

$$P_{k|k-1} = P_{k-1}(t_k), \quad (16b)$$

as the solution to

$$\frac{d}{dt}\hat{x}_{k-1}(t) = f(t, \hat{x}_{k-1}(t), u_{k-1}, d_{k-1}, p_f), \quad (17a)$$

$$\begin{aligned} \frac{d}{dt}P_{k-1}(t) &= A_{k-1}(t)P_{k-1}(t) + P_{k-1}(t)A_{k-1}(t)' \\ &+ \sigma_{k-1}(t)\sigma_{k-1}(t)', \end{aligned} \quad (17b)$$

for $t_{k-1} \leq t \leq t_k$ with initial condition

$$\hat{x}_{k-1}(t_{k-1}) = \hat{x}_{k-1|k-1}, \quad (18a)$$

$$P_{k-1}(t_{k-1}) = P_{k-1|k-1}, \quad (18b)$$

and

$$A_{k-1}(t) = \frac{\partial}{\partial x}f(t, \hat{x}_{k-1}(t), u_{k-1}, d_{k-1}, p_f), \quad (19a)$$

$$\sigma_{k-1}(t) = \sigma(t, \hat{x}_{k-1}(t), u_{k-1}, d_{k-1}, p_\sigma). \quad (19b)$$

The CD-EKF obtains a filtered state estimate, $\hat{x}_{k|k}$, and its covariance, $P_{k|k}$, from the one-step prediction, $\hat{x}_{k|k-1}$ and $P_{k|k-1}$, and the measurement, y_k . The CD-EKF computes the predicted measurement and derivative,

$$\hat{y}_{k|k-1} = g(t_k, \hat{x}_{k|k-1}, p_g), \quad (20a)$$

$$C_k = \frac{\partial}{\partial x}g(t_k, \hat{x}_{k|k-1}, p_g), \quad (20b)$$

the innovation and its covariance,

$$e_k = y_k - \hat{y}_{k|k-1}, \quad (21a)$$

$$R_{e,k} = C_k P_{k|k-1} C_k' + R_k, \quad (21b)$$

and the Kalman gain,

$$K_{f_x,k} = P_{k|k-1} C_k' R_{e,k}^{-1}. \quad (22)$$

We obtain the estimated state-covariance pair from (20)-(22) as

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_{f_x,k} e_k, \quad (23a)$$

$$P_{k|k} = P_{k|k-1} - K_{f_x,k} R_{e,k} K_{f_x,k}'. \quad (23b)$$

The CD-EKF is the state estimator, $\kappa(\cdot)$, that computes filtered state estimates, $x_k^c = \hat{x}_{k|k}$, from measurements y_k , and one-step state prediction, $\hat{x}_{k|k-1}$.

The NMPC uses a regulator based on a weighted least-squares objective and regularization of the input rate-of-movement. This regulator can be expressed in terms of the optimal control problem (OCP)

$$\min_{x,u} \varphi_k = \varphi_{z,k} + \varphi_{\Delta u,k}, \quad (24a)$$

$$s.t. \quad x(t_k) = \hat{x}_{k|k}, \quad (24b)$$

$$\dot{x}(t) = f(t, x, u, d, p_f), \quad t_k \leq t \leq t_k + T, \quad (24c)$$

$$z(t) = h(t, x, p_h), \quad (24d)$$

$$u(t) = u_{k+j}, \quad j \in \mathcal{N}, \quad t_{k+j} \leq t \leq t_{k+j+1}, \quad (24e)$$

$$d(t) = d_{k+j}, \quad j \in \mathcal{N}, \quad t_{k+j} \leq t \leq t_{k+j+1}, \quad (24f)$$

$$u_l \leq u_{k+j} \leq u_u, \quad j \in \mathcal{N}, \quad (24g)$$

$$\Delta u_l \leq \Delta u_{k+j} \leq \Delta u_u, \quad j \in \mathcal{N}, \quad (24h)$$

with $f(t, x, u, d, p_f) = f(t, x(t), u(t), d(t), p_f)$ and the objective terms

$$\varphi_{z,k} = \frac{1}{2} \int_{t_k}^{t_k+T} \|W_z(z(t) - \bar{z}(t))\|_2^2 dt, \quad (25a)$$

$$\varphi_{\Delta u,k} = \frac{1}{2} \sum_{j=0}^{N-1} \|\bar{W}_{\Delta u} \Delta u_{k+j}\|_2^2, \quad (25b)$$

where $\bar{W}_{\Delta u} = W_{\Delta u}/T_s$. The term $\varphi_{z,k}$ is output target tracking and $\varphi_{\Delta u,k}$ is input rate of movement penalty. We use the prediction and control horizon, T , defined as $T = NT_s$, where T_s is the sampling time and N is the discrete prediction and control horizon. Additionally, we define $\mathcal{N} = \{0, 1, \dots, N-1\}$ such that $t_{k+j} = t_k + jT_s$ for $j \in \mathcal{N}$. We solve the OCP with a simultaneous approach, where we discretize each control interval with M time steps using Euler's implicit method.

We denote the optimal solution $\{\hat{x}_{k+j+1|k}, \hat{u}_{k+j|k}\}_{j \in \mathcal{N}}$. The input corresponding to the first control interval, $u_k = \hat{u}_{k|k} = \lambda(\cdot)$, is part of the solution of this optimal control problem. Only u_k is implemented in the system. Furthermore, $\{\hat{z}_{k+j+1|k}, \hat{u}_{k+j|k}\}_{j=0}^{N-1} = z_k^c = \mu(\cdot)$ is the predicted output and the predicted manipulated inputs from the controller that can be used for visualization.

D. Simulation of closed-loop systems

We compactly write a closed-loop simulation as

$$y_k = g(t_k, x_k, p_g) + v_k, \quad (26a)$$

$$z_k = h(t_k, x_k, p_h), \quad (26b)$$

$$x_k^c = \kappa(t_{k-1}, x_{k-1}^c, y_k, u_{k-1}, p_\kappa), \quad (26c)$$

$$u_k = \lambda(t_k, x_k^c, p_\lambda), \quad (26d)$$

$$z_k^c = \mu(t_k, x_k^c, p_\mu), \quad (26e)$$

$$x_{k+1} = \Phi(t_k, x_k, u_k, d_k, w_k, p_f, p_\sigma), \quad (26f)$$

for $k = 0, 1, \dots, N_s - 1$.

III. MONTE CARLO SIMULATION

Our Monte Carlo simulations are based on the closed-loop simulation (26). We perform N_{mc} distinct closed-loop simulations for different, e.g., process noise realizations. Algorithm 1 presents an overview of the Monte Carlo simulation scheme. For sufficiently large N_{mc} , the computed Monte

Algorithm 1: Monte Carlo simulation

Result: Statistics and output data
// Monte Carlo loop
for $i = 1, 2, \dots, N_{mc}$ **do**
 // Closed loop simulation
 for $k = 0, 1, \dots, N_s - 1$ **do**
 // Measurement
 $y_k^{(i)} = g(t_k, x_k^{(i)}, p_g^{(i)}) + v_k^{(i)}$
 // Output
 $z_k^{(i)} = h(t_k, x_k^{(i)}, p_h^{(i)})$
 // State estimation
 $x_k^{c,(i)} = \kappa(t_{k-1}, x_{k-1}^{c,(i)}, y_k^{(i)}, u_{k-1}^{(i)}, p_\kappa^{(i)})$
 // Regulator
 $u_k^{(i)} = \lambda(t_k, x_k^{c,(i)}, p_\lambda^{(i)})$
 // Output prediction
 $z_k^{c,(i)} = \mu(t_k, x_k^{c,(i)}, p_\mu^{(i)})$
 // Simulator
 $x_{k+1}^{(i)} = \Phi(t_k, x_k^{(i)}, u_k^{(i)}, d_k^{(i)}, w_k^{(i)}, p_f^{(i)}, p_\sigma^{(i)})$
 end
 // Final measurement and output
 $y_{N_s}^{(i)} = g(t_{N_s}, x_{N_s}^{(i)}, p_g^{(i)}) + v_{N_s}^{(i)}$
 $z_{N_s}^{(i)} = h(t_{N_s}, x_{N_s}^{(i)}, p_h^{(i)})$
end

Carlo data can quantify uncertainties in the closed-loop system. Possible applications are; estimation of unknown model parameters, tuning controllers, and testing performance of controllers on different noise realizations.

A. Toolbox

We implement a Monte Carlo simulation toolbox for closed-loop systems in C. The toolbox provides an interface for closed-loop Monte Carlo simulations that currently includes implementations of

- an explicit-explicit Euler-Maruyama SDE solver,
- an implicit-explicit SDE solver,
- an open-loop controller,
- a single-input single-output (SISO) PID controller with clipping, and
- an NMPC based on the CD-EKF and a simultaneous approach combined with IPOPT [19].

The toolbox includes three test examples, and the user can provide a set of model functions for a system and perform Monte Carlo simulations with the toolbox. Additionally, the toolbox allows for user-provided controllers and SDE solvers with specific interfaces. This allows the user to test and benchmark controllers and SDE solvers using Monte Carlo simulations. The toolbox supports perturbations of model parameters, controller parameters, noise realizations, initial conditions, and disturbances.

We include a parallelized version with OpenMP for shared memory architectures. Each worker is assigned distinct closed-loop simulations. Such parallelization requires that

TABLE I
PARAMETERS FOR FED-BATCH REACTOR.

Variable	Value	Unit
μ_{\max}	0.37	1/h
K_S	0.021	kg/m ³
K_I	0.38	kg/m ³
γ_s	1.777	kg substrate/kg biomass
$c_{S,in}$	10.0	kg/m ³

TABLE II
INITIAL CONDITION AND OPERATIONAL BOUNDS.

Variable	Value	Unit
V_0	1.00	m ³
$c_{X,0}$	2.00	kg/m ³
$c_{S,0}$	0.0893	kg/m ³
V_{\max}	12.39	m ³
$c_{X,\max}$	2.00	kg/m ³
$c_{S,\max}$	3.00	kg/m ³
$F_{S,\max}$	10.00	m ³
$F_{W,\max}$	10.00	m ³

each worker has access to a local workspace for the SDE solver and the controller to avoid data races. Additionally, some controllers utilize information from previous steps, e.g., the integral term of a PID controller or the CD-EKF for an NMPC. Each worker also requires a local version of such information. The Monte Carlo simulation toolbox distributes memory blocks to each local worker, such that workers do not have overlapping cache lines. This consideration is essential for achieving optimal parallel performance.

We demonstrate some applications of the toolbox in section IV.

IV. BIOREACTOR IN FED-BATCH OPERATION

A. Model

We consider the SDE model for a bioreactor in fed-batch operation [12],

$$dV = (F_S + F_W)dt + \sigma_1 d\omega_1(t), \quad (27a)$$

$$dm_X = (R_X V)dt + \sigma_2 d\omega_2(t), \quad (27b)$$

$$dm_S = (F_S c_{S,in} + R_S V)dt + \sigma_3 d\omega_3(t), \quad (27c)$$

where $m_X = c_X V$, $m_S = c_S V$, and

$$R_X = r, \quad r = \mu(c_S) c_X, \quad (28a)$$

$$R_S = -\gamma r, \quad \mu(c_S) = \mu_{\max} \frac{c_S}{K_S + c_S + c_S^2/K_I}. \quad (28b)$$

We represent the system as a stochastic continuous-discrete model in the form (1), where $x(t) = [V(t); m_X(t); m_S(t)]$, $y(t_k) = c_S(t_k) + v(t_k)$, and $z(t) = c_S(t)$.

Table I presents the parameters of the system and Table II presents the initial conditions and operational bounds of the system.

B. Control strategy

We operate the bioreactor with an open-loop input trajectory, $\bar{u} = [\bar{F}_W; \bar{F}_S]$. Additionally, we use a SISO PID

TABLE III
SYSTEM INFORMATION.

Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit
CPU(s):	32
Thread(s) per core:	1
Core(s) per socket:	16
Socket(s):	2
NUMA node(s):	2
Model name:	Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz
CPU MHz:	2900.000
L1d cache:	32 kB
L1i cache:	32 kB
L2 cache:	1024 kB
L3 cache:	22528 kB
RAM:	384 GB

controller with clipping, (15), that manipulate the substrate inlet, F_S , to achieve optimal substrate concentration, c_S^* , achieved at the maximum of $\mu(c_S)$,

$$c_S^* = \sqrt{K_I K_S}. \quad (29)$$

Thus, $\bar{z}(t) = c_S^*$. We use the bang-bang open-loop trajectory [12]. In the deterministic case, the bang-bang trajectory was one among infinitely many optimal solutions. However, it was the least sensitive to uncertainties. The inputs are computed as,

$$\bar{F}_W = \begin{cases} F_{W,\max}, & 0 \leq t \leq t_{switch}, \\ 0, & t_{switch} \leq t \leq t_f, \end{cases} \quad (30a)$$

$$\bar{F}_S = \frac{F_W c_S^* + \gamma_s \beta^*(t)}{c_{S,in} - c_S^*}, \quad (30b)$$

where

$$\beta^*(t) = \mu(c_S^*) c_{X,\max} V_0 \exp(\mu(c_S^*) t). \quad (31)$$

C. Simulation of the true system

We simulate the fed-batch reactor in closed-loop with an Euler-Maruyama solver. The reactor runs for 10 hours from time $t_0 = 0$ to $t_f = 10$ h. The sampling time is $T_s = 36$ seconds resulting in $N_s = 1000$ steps. At each step, we solve the SDE with $N_k = 10$ Euler-Maruyama steps.

D. Monte Carlo simulations of fed-batch reactor

Here, we demonstrate an application of the Monte Carlo simulation toolbox. The simulations are conducted on a dual-socket Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz system (see Table III for CPU details).

1) *Scaling*: Fig. 1 shows the wall time and the scale-up for 10,000 Monte Carlo simulations. We observe close to linear scaling within one non-uniform memory access (NUMA) node, and slightly decreasing scale-up when exceeding one socket. We point out that the toolbox is not optimized to utilize multiple NUMA nodes, so a decrease in performance on more than one socket is expected.

2) *Open-loop controller*: We perform Monte Carlo simulations for the fed-batch reactor in open-loop. Fig. 3(a) shows a probability density function (PDF) plot for 30,000 realizations of process noise. The mean produced biomass is $\bar{m}_X(t_f) = 20.69$ kg.

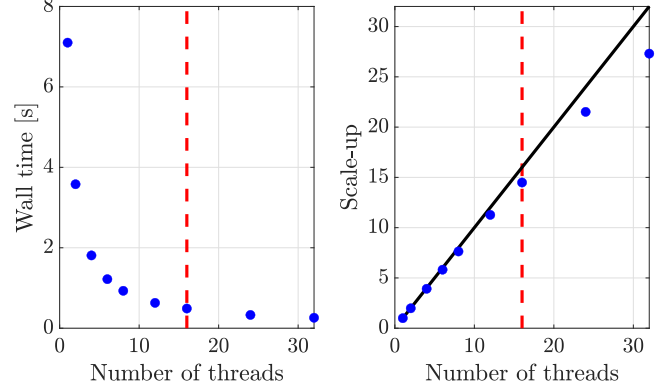


Fig. 1. Wall time and scale-up plots for 10,000 Monte Carlo simulations. The red dashed line is the number of cores on a single NUMA node. We get a scale-up of 27.3 on 32 cores.

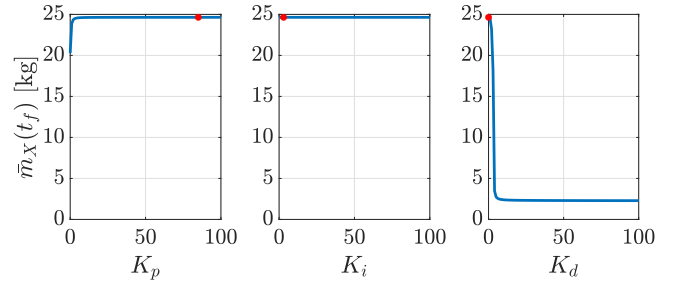
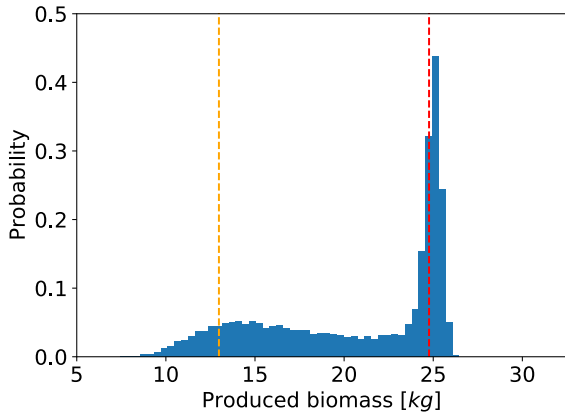


Fig. 2. Tuning of PID gains. Left: locate $K_p = 85$ as the optimum. Middle: locate $K_i = 3$ as the optimum. Right: locate $K_d = 0$ as the optimum. Total Monte Carlo simulations: $3 \cdot 101,000 = 303,000$. Computation time: ~ 7.50 seconds.

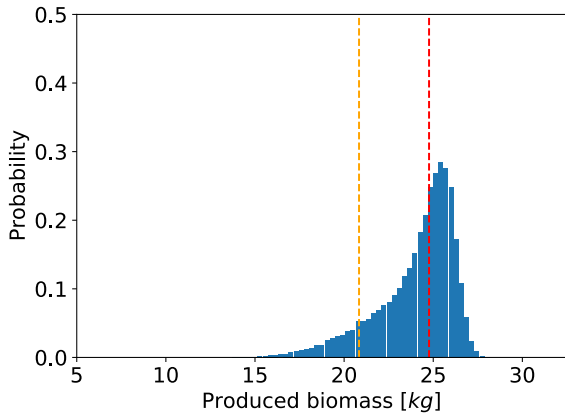
3) *Quantification of PID controller performance*: Consider a PID controller with $K_p = 1.0$, $K_i = 0.0$, and $K_d = 0.0$. We perform a Monte Carlo simulation with 30,000 process noise realizations. Fig. 3(b) presents a PDF plot of the produced biomass. The PDF follows a long-tailed distribution towards the lower values of produced biomass with mean produced biomass $\bar{m}_X(t_f) = 24.04$ kg, i.e., a 16.19% increase in biomass production compared to the open-loop controller. However, low produced biomass, for some realizations of process noise, indicates poor controller performance. The computation time of the Monte Carlo simulation is 0.77s. That is approximately a 2300 times speed-up compared to a reference serial implementation in Matlab.

4) *Tuning*: We apply Monte Carlo simulation to tune the value of K_p , K_i , and K_d in the PID controller. The tuning is based on maximizing the average produced biomass, $\bar{m}_X(t_f)$, for 1000 realizations of process noise. We point out that the tuning could have been based on other factors, e.g., maximizing the 10% quantile. We investigate 101 equidistant values in $[0, 100]$ of K_p , K_i , and K_d . Thus, the tuning of each gain requires 101,000 closed loop simulations. Fig. 2 shows the tuning results of the PID controller. The optimal parameters for the PID gains are $K_p = 85$, $K_i = 3$, and $K_d = 0$.

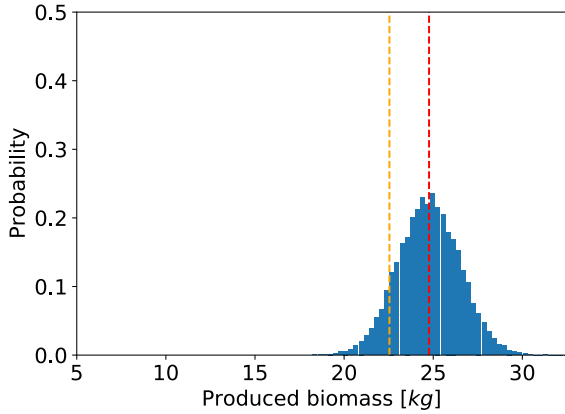
Fig. 3(c) shows a PDF plot for 30,000 noise realizations



(a) Open-loop controller. Computation time: 0.78s.



(b) Sub-optimal PID controller. Computation time: 0.77s.



(c) Optimal PID controller. Computation time: 0.76s.

Fig. 3. Probability density function of biomass production computed from 30,000 closed-loop simulations with different process noise realizations. The closed-loop consists of the fed-batch model, the Euler-Maruyama SDE solver, and a controller specified in the subplot. The red dashed line is the produced biomass in a simulation without process noise and the orange dashed line is the 10% quantile.

with the tuned PID controller. The PDF is almost normally distributed with mean produced biomass, $\bar{m}_X(t_f) = 24.76$. Compared to the non-optimal PID controller, the tuned controller results in an 2.98% increased average biomass

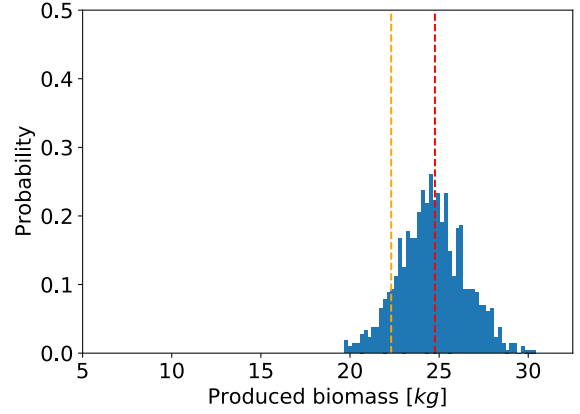


Fig. 4. Probability density function of biomass production computed from 1000 NMPC closed-loop simulations with different process noise realizations. The red dashed line is the produced biomass in a simulation without process noise and the orange dashed line is the 10% quantile. The computation time is ~ 30 min.

production and reduced risk of low biomass production. It is evident that the tuning improved the performance of the PID controller.

5) *NMPC*: Initial investigation of an NMPC based on the open source optimization software, IPOPT, does not show the same scaling as the PID controller. We observe a significant stall time in the memory allocation with `malloc()`, when increasing the number of threads. These calls to `malloc()` are located in IPOPT and give reason to believe that IPOPT has internal memory allocation. Each memory allocation has a lock that interrupts all activity, i.e., stalling all threads. In future work, we will expand the toolbox to include an NMPC based on optimization software that does not have internal memory allocation. We believe that scaling similar to the PID case can be achieved with such an NMPC. Fig. 4 presents a PDF plot for 1000 process noise realizations. The NMPC and the tuned PID controller show similar performance. The experiment is conducted on 6 cores as performance decreases above 6 cores due to the problem mentioned above. The computation time is ~ 30 min.

V. CONCLUSION

The paper presents a Monte Carlo simulation approach for performance quantification and tuning of controllers in linear and nonlinear systems. The approach is computationally feasible due to the implementation of a parallel high-performance Monte Carlo simulation toolbox in C for closed-loop systems. In particular, we demonstrate performance quantification and tuning of a PID controller for a bioreactor in fed-batch operation. Our results show that large-scale Monte Carlo simulations can be performed within seconds. The computational performance of the toolbox show approximately a 2300 times speed-up compared to a serial reference implementation in Matlab.

High-performance closed-loop Monte-Carlo simulations, as illustrated in this paper, has countless applications in systems and control. Drug dosing, as in treatment of diabetes,

is a very prominent example of this where several dosing strategies must be compared by their probability density functions [20], [21].

REFERENCES

- [1] D. Boiroux, T. K. S. Ritschel, N. K. Poulsen, H. Madsen, and J. B. Jørgensen, "Efficient Computation of the Continuous-Discrete Extended Kalman Filter Sensitivities Applied to Maximum Likelihood Stimulation," *58th Conference on Decision and Control*, pp. 6983–6988, 2019.
- [2] H. G. Bock, E. Kostina, and J. P. Schlöder, "Numerical Methods for Parameter Estimation in Nonlinear Differential Algebraic Equations," *GAMM-Mitteilungen*, vol. 30, no. 2, pp. 376–408, 2007.
- [3] S. Skogestad, "Simple analytic rules for model reduction and PID controller tuning," *Journal of Process Control*, p. 291–309, 2003.
- [4] D. Olesen, J. K. Huusom, and J. B. Jørgensen, "A Tuning Procedure for ARX-based MPC," *IEEE Multi-conference on Systems and Control*, pp. 188–193, 2013.
- [5] D. H. Olesen, J. K. Huusom, and J. B. Jørgensen, "A tuning procedure for ARX-based MPC of multivariate processes," *American Control Conference*, pp. 1721–1726, 2013.
- [6] T. K. S. Ritschel, D. Boiroux, M. K. Nielsen, J. K. Huusom, S. B. Jørgensen, and J. B. Jørgensen, "Economic Optimal Control of a U-loop Bioreactor using Simultaneous Collocation-based Approaches," in *2019 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2019, pp. 933–938.
- [7] —, "Economic Nonlinear Model Predictive Control of a U-loop Bioreactor," in *2020 European Control Conference (ECC)*. IEEE, 2020, pp. 208–213.
- [8] L. J. Hong, S. Juneja, and J. Luo, "Estimating sensitivities of portfolio credit risk using Monte Carlo," *Informa Journal on Computing*, vol. 26, no. 4, pp. 848–865, 2014.
- [9] A. Shlyakhter, L. Mirny, A. Vlasov, and R. Wilson, "Monte Carlo Modeling of Epidemiologic Studies," *Human and Ecological Risk Assessment*, vol. 2, no. 4, pp. 920–936, 1996.
- [10] J. B. Jørgensen, D. Boiroux, and Z. Mahmoudi, "An artificial pancreas based on simple control algorithms and physiological insight," *12th Symposium IFAC on Dynamics and Control of Process Systems, including Biosystems (DYCOPS 2019), Florianópolis - SC, Brazil*, April 23–26 2019.
- [11] S. J. Ratick and G. Schwarz, "Monte Carlo Simulation," in *Encyclopedia of Human Geography (Second Edition)*. Elsevier, 2009, ch. 14, pp. 175–185.
- [12] P. Ross, "Why CPU Frequency Stalled," *IEEE Spectrum*, vol. 45, no. 4, p. 72, 2008.
- [13] T. E. Ryde, M. R. Wahlgreen, M. K. Nielsen, S. Hørsholt, S. B. Jørgensen, and J. B. Jørgensen, "Optimal Feed Trajectories for Fed-batch Fermentation with Substrate Inhibition Kinetics," *International Symposium on Advanced Control of Chemical Processes, Accepted*, 2021.
- [14] D. J. Higham, "An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations," *SIAM Review*, vol. 43, no. 3, pp. 525–546, 2001.
- [15] T. Tian and K. Burrage, "Implicit Taylor methods for stiff stochastic differential equations," *Applied Numerical Mathematics*, vol. 38, pp. 167–185, 2001.
- [16] B. Wittenmark, K. Johan Åström, and K.-E. Årzén, in *Computer Control: An Overview*. IFAC professional brief, 2009.
- [17] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.
- [18] M. R. Wahlgreen, E. Schroll-Fleischer, D. Boiroux, T. K. S. Ritschel, H. Wu, J. K. Huusom, and J. B. Jørgensen, "Nonlinear Model Predictive Control for an Exothermic Reaction in an adiabatic CSTR," *6th Conference on Advances in Control and Optimization of Dynamical Systems ACODS, Chennai, India*, February 16–19 2020.
- [19] J. B. Jørgensen, T. K. S. Ritschel, D. Boiroux, E. Schroll-Fleischer, M. R. Wahlgreen, M. K. Nielsen, H. Wu, and J. K. Huusom, "Simulation of NMPC for a Laboratory Adiabatic CSTR with an Exothermic Reaction," *Proceedings of 2020 European Control Conference*, pp. 202–207, 2020.
- [20] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.
- [21] D. Boiroux and J. Jørgensen, "Nonlinear Model Predictive Control and Artificial Pancreas Technologies," *IEEE Conference on Decision and Control (CDC), Miami Beach, Florida, USA*, December 17–19 2018.

APPENDIX B

Paper II: DYCOPS 2022

On the Implementation of a Preconditioned Riccati Recursion based Primal-Dual Interior-Point Algorithm for Input Constrained Optimal Control Problems

Authors:

Morten Ryberg Wahlgreen, John Bagterp Jørgensen

Published in:

Proceedings of the 13th IFAC Symposium on Dynamics and Control of Process
Systems, including Biosystems (DYCOPS), Busan, Republic of Korea, June 14–17,
2022.

<https://doi.org/10.1016/j.ifacol.2022.07.468>

On the Implementation of a Preconditioned Riccati Recursion based Primal-Dual Interior-Point Algorithm for Input Constrained Optimal Control Problems^{*}

Morten Ryberg Wahlgreen^{*} John Bagterp Jørgensen^{*}

^{*} Department of Applied Mathematics and Computer Science,
Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark.

Abstract:

We present a preconditioned interior-point algorithm tailored for input constrained quadratic programmings (QPs) arising in optimal control problems (OCPs). The implicit approach to OCPs results in large sparse QPs, which we utilized by a tailored Riccati recursion algorithm. The Riccati recursion algorithm requires the solution of a set of small dense linear sub-systems of equations. The proposed preconditioner is an easily invertible diagonal matrix, which we apply in every linear sub-system of equations. We solve a target tracking OCP for a linearized modified quadruple tank system in Matlab. The computational results indicate that ill-conditioning in the sub-systems are reduced and that the additional CPU time for preconditioning is negligible. Additionally, the paper presents a detailed description of the proposed algorithm and serves as an implementation guide for the algorithm.

Copyright © 2022 The Authors. This is an open access article under the CC BY-NC-ND license (<https://creativecommons.org/licenses/by-nc-nd/4.0/>)

Keywords: Interior-point method, Quadratic programming, Optimal Control Problem, Riccati recursion, Preconditioning

1. INTRODUCTION

Interior-point methods have been an integrated part of optimization since the 1960s (Forsgren et al., 2002; Wright, 2004). In the 1960s, interior-point methods were mainly applied in problems with nonlinear constraints. Today, the applications span many different types of problems including linear programming (LP), quadratic programming (QP), and nonlinear programming (NLP) (Astfalk et al., 1992; Vanderbei, 1999; Byrd et al., 1999; Nocedal and Wright, 1999; Gertz and Wright, 2003). Furthermore, LPs, QPs, and NLPs are an essential component in development of advanced control algorithms such as model predictive control (MPC) (Rawlings et al., 1994). Interior-point methods suffer from unavoidable ill-conditioning as the iterations approach the solution (Murray, 1971). Preconditioning approaches are often proposed in relation to inexact interior-point methods due to the inherent need of well-conditioned systems in iterative solvers for linear systems (Bergamaschi et al., 2004; Shahzad et al., 2010; Cui et al., 2019). However, proposed preconditioning in relation to exact methods is sparse.

In linear model predictive control (LMPC), QPs arise in the form of optimal control problems (OCPs) (Borrelli et al., 2009). Usually either explicit or implicit approaches are applied to express OCPs as QPs. Explicit approaches result in small dense QPs, while implicit methods result in large sparse QPs (Shahzad et al., 2010). In the implicit case, Riccati recursion has been proposed as an efficient sparse solver with linear complexity in the control

horizon (Rao et al., 1998; Frison and Jørgensen, 2013). The Riccati recursion based interior-point method suffers from unavoidable ill-conditioning as all other interior-point methods. We propose a diagonal preconditioner for Riccati recursion based interior-point methods at the cost of negligible additional CPU time. We suggest that preconditioning at the cost of negligible CPU time improve the overall quality of the algorithm.

In this paper, we propose a preconditioned interior-point method for input constrained QPs arising in OCPs. We introduce the well-known Riccati recursion for solution of structured linear systems and propose a preconditioner for the Riccati based interior-point method. We present a detailed description of the proposed interior-point method, where we emphasize how to exploit the structure of box-constraints to reduce CPU time. As such, the paper serves as an implementation guide for a well-conditioned Riccati recursion based interior-point method for QPs in OCPs. The results in the paper are based on a Matlab implementation of the proposed interior-point method. We demonstrate applications of the implementation on a target tracking OCP for a linearized modified quadruple tank system.

The Matlab implementation is an essential building block in the development of a Riccati recursion based sequential quadratic programming (SQP) algorithm for NLPs arising in nonlinear model predictive control (NMPC). Additionally, it serves as a prototype for an C implementation of a LMPC, which can be applied for parallel Monte Carlo simulation of closed loop systems (Wahlgreen et al., 2021).

^{*} Corresponding author: J. B. Jørgensen (E-mail: jbjo@dtu.dk).

The remaining part of the paper is organized as follows. Section 2 introduces the main parts of the primal-dual interior-point algorithm. Section 3 presents the proposed preconditioner. Section 4 presents the Riccati recursion algorithm. Section 5 presents an example application of our interior-point algorithm. Section 6 presents our conclusion.

2. PRIMAL-DUAL INTERIOR-POINT ALGORITHM

This section presents a detailed description of the primal-dual interior-point algorithm for general box-constrained QPs. The algorithm is a predictor-corrector with scaled KKT-violation convergence criterion. We specialize the algorithm to OCPs in Section 4.

2.1 Quadratic programming

We consider box-constrained QPs on the form,

$$\min_x \quad f(x) = \frac{1}{2}x^T Hx + g^T x, \quad (1a)$$

$$s.t. \quad A^T x = b, \quad (1b)$$

$$l \leq x \leq u, \quad (1c)$$

where $H \in \mathbb{R}^{n \times n}$, $g \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $l \in \mathbb{R}^n$, $u \in \mathbb{R}^n$, and $x \in \mathbb{R}^n$ are the decision variables.

The Lagrangian function for (1) is,

$$\mathcal{L}(x, y, z_l, z_u) = \frac{1}{2}x^T Hx + g^T x - y^T (A^T x - b) - z_l^T (x - l) - z_u^T (u - x), \quad (2)$$

where $y \in \mathbb{R}^m$, $z_l \in \mathbb{R}^n$, and $z_u \in \mathbb{R}^n$ are the Lagrange multipliers for the equality constraints, lower bound constraints, and upper bound constraints, respectively.

The first order KKT-conditions for (1) are,

$$Hx + g - Ay - z_l + z_u = 0, \quad (3a)$$

$$b - A^T x = 0, \quad (3b)$$

$$s_l + l - x = 0, \quad s_u + x - u = 0, \quad (3c)$$

$$s_{l,i} z_{l,i} = 0, \quad s_{u,i} z_{u,i} = 0, \quad (3d)$$

$$(z_l, z_u, s_l, s_u) \geq 0, \quad (3e)$$

where $i = 1, \dots, n$ and the slack variables, s_l and s_u , are

$$s_l = x - l \geq 0, \quad s_u = u - x \geq 0. \quad (4)$$

We write the KKT-conditions, (3), as the nonlinear system of equations,

$$\begin{bmatrix} r_L \\ r_A \\ r_{B_l} \\ r_{B_u} \\ r_{S_l Z_l} \\ r_{S_u Z_u} \end{bmatrix} = \begin{bmatrix} Hx + g - Ay - z_l + z_u \\ b - A^T x \\ s_l + l - x \\ s_u + x - u \\ S_l Z_l e \\ S_u Z_u e \end{bmatrix} = 0, \quad (5a)$$

$$(z_l, z_u, s_l, s_u) \geq 0, \quad (5b)$$

where $Z_l = \text{diag}(z_l)$, $Z_u = \text{diag}(z_u)$, $S_l = \text{diag}(s_l)$, $S_u = \text{diag}(s_u)$, and e is a vector of ones of proper dimension. We apply Newton's method to solve (5), which yields the following linear system of equations for the Newton search direction,

$$\begin{bmatrix} H & -A & -I & I & 0 & 0 \\ -A^T & 0 & 0 & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & I & 0 \\ I & 0 & 0 & 0 & 0 & I \\ 0 & 0 & S_l & 0 & Z_l & 0 \\ 0 & 0 & 0 & S_u & 0 & Z_u \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z_l \\ \Delta z_u \\ \Delta s_l \\ \Delta s_u \end{bmatrix} = - \begin{bmatrix} r_L \\ r_A \\ r_{B_l} \\ r_{B_u} \\ r_{S_l Z_l} \\ r_{S_u Z_u} \end{bmatrix}. \quad (6)$$

We introduce the equivalent system,

$$\begin{bmatrix} H & -A & -C & 0 \\ -A^T & 0 & 0 & 0 \\ -C^T & 0 & 0 & I \\ 0 & 0 & S & Z \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_L \\ r_A \\ r_C \\ r_{SZ} \end{bmatrix}, \quad (7)$$

with $C = [I, -I]$, $Z = \text{diag}([Z_l; Z_u])$, $S = \text{diag}([S_l; S_u])$, $\Delta z = [\Delta z_l; \Delta z_u]$, $\Delta s = [\Delta s_l; \Delta s_u]$, $r_C = [r_{B_l}; r_{B_u}]$, and $r_{SZ} = [r_{S_l Z_l}; r_{S_u Z_u}]$. We point out that (7) only serves as notation, while (6) is the preferred form for implementation as it exploits the identity structure of C .

The solution of (7), $(\Delta x, \Delta y, \Delta z, \Delta s)$, serves as a step direction for the algorithm. In each iteration, $[l]$, the algorithm performs the step,

$$(x, y, z, s) \leftarrow (x, y, z, s) + \eta \alpha (\Delta x, \Delta y, \Delta z, \Delta s), \quad (8)$$

where $\eta = 0.995$ and the step-size, α , ensures $(z, s) \geq 0$.

2.2 Predictor-corrector

Our algorithm applies the Mehrotra predictor-corrector method (Mehrotra, 1992). The method considers the following form of (7),

$$\begin{bmatrix} H & -A & -C & 0 \\ -A^T & 0 & 0 & 0 \\ -C^T & 0 & 0 & I \\ 0 & 0 & S & Z \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_L \\ r_A \\ r_C \\ \bar{r}_{SZ} \end{bmatrix}, \quad (9)$$

where \bar{r}_{SZ} varies in the predictor and corrector phase. In the predictor phase, we set $\bar{r}_{SZ} = r_{SZ}$ and denote the solution to (9), $(\Delta x^{aff}, \Delta y^{aff}, \Delta z^{aff}, \Delta s^{aff})$. This direction is called the affine direction. In the corrector phase, we set $\bar{r}_{SZ} = r_{SZ} + \Delta S^{aff} \Delta Z^{aff} - \sigma \mu e$ and denote the solution $(\Delta x, \Delta y, \Delta z, \Delta s)$. We compute the duality gap, μ , and centering parameter, σ , as,

$$\mu^{aff} = \frac{(z + \alpha^{aff} \Delta z)^T (s + \alpha^{aff} \Delta s^{aff})}{m}, \quad (10)$$

$$\mu = \frac{s^T z}{m}, \quad \sigma = \left(\frac{\mu^{aff}}{\mu} \right)^3,$$

where $m = 2n$ for box-constrained QPs (1).

2.3 Augmented form

We write (9), i.e., (6), in the augmented form,

$$\begin{bmatrix} \bar{H} & -A \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = - \begin{bmatrix} -\bar{r}_L \\ r_A \end{bmatrix}, \quad (11)$$

where

$$\bar{H} = H + D_l + D_u, \quad (12)$$

$$\bar{r}_L = -r_L + (S_l^{-1} Z_l) (r_{B_l} - Z_l^{-1} \bar{r}_{S_l Z_l}) - (S_u^{-1} Z_u) (r_{B_u} - Z_u^{-1} \bar{r}_{S_u Z_u}), \quad (13)$$

with $D_l = \text{diag}(z_l/s_l)$ and $D_u = \text{diag}(z_u/s_u)$. The variables $\bar{r}_{S_l Z_l}$ and $\bar{r}_{S_u Z_u}$ are,

$$A = \begin{bmatrix} -B_0^T & I & & & & \\ & -A_1^T & -B_1^T & I & & \\ & & \ddots & \ddots & \ddots & \\ & & & -A_{N-1}^T & -B_{N-1}^T & I \end{bmatrix}^T, \quad (27d)$$

$$b = [\tilde{b}_0 \ b_1 \ \cdots \ b_{N-1}]^T, \quad (27e)$$

$$l = [u_{\min,0} \ -\infty \ u_{\min,1} \ \cdots \ u_{\min,N-1} \ -\infty]^T, \quad (27f)$$

$$u = [u_{\max,0} \ \infty \ u_{\max,1} \ \cdots \ u_{\max,N-1} \ \infty]^T, \quad (27g)$$

where $\tilde{b}_0 = b_0 + A_0^T x_0$.

4.2 Riccati recursion based interior-point method for OCPs

We consider the augmented system, (11), in the interior-point algorithm for the OCP (23) (for $N = 3$),

$$\begin{bmatrix} \bar{R}_0 & & & B_0 & & \\ & Q_1 & M_1 & -I & A_1 & \\ & M_1^T & \bar{R}_1 & & B_1 & \\ & & & & -I & A_2 \\ & & & Q_2 & M_2 & B_2 \\ & & & M_2^T & \bar{R}_2 & \\ & & & & & P_3 \\ \hline B_0^T & -I & & & & \\ & A_1^T & B_1^T & -I & & \\ & & A_2^T & B_2^T & -I & \end{bmatrix} \begin{bmatrix} \Delta u_0 \\ \Delta x_1 \\ \Delta u_1 \\ \Delta x_2 \\ \Delta u_2 \\ \Delta x_3 \\ \Delta y_0 \\ \Delta y_1 \\ \Delta y_2 \end{bmatrix} = - \begin{bmatrix} \bar{r}_0 \\ q_1 \\ \bar{r}_1 \\ q_2 \\ \bar{r}_2 \\ p_3 \\ \tilde{b}_0 \\ b_1 \\ b_2 \end{bmatrix}, \quad (28)$$

where $\bar{R}_k = R_k + D_{l,k} + D_{u,k}$, due to (12), (27f), and (27g). We compute the right hand side of the KKT system, (28), exactly as the right hand side of (11).

We exploit the sparse structure of (28) with a Riccati recursion based linear equation solver with linear complexity in the horizon, N (Rao et al., 1998). Algorithm 1 and 2 presents the factorization and solution phase of the Riccati recursion algorithm (Jørgensen, 2004). Notice that Riccati recursion requires solution of a set of small dense sub-systems of linear equations.

4.3 Preconditioned Riccati recursion

Unavoidable ill-conditioning of the interior-point method arises in \bar{R}_k . Consequently, $R_{e,k}$, becomes increasing ill-conditioned. We propose to apply the diagonal preconditioner, (21), to improve conditioning of the sub-systems containing $R_{e,k}$. Consequently, the preconditioned systems are,

$$K_k = -\tilde{R}_{e,k}^{-1} \left[\tilde{P}_k^{-1} (M_k^T + B_k P_{k+1} A_k^T) \right], \quad (29a)$$

$$a_k = -\tilde{R}_{e,k}^{-1} \left[\tilde{P}_k^{-1} (r_k + B_k (P_{k+1} b_k + p_{k+1})) \right], \quad (29b)$$

$$a_0 = -\tilde{R}_{e,0}^{-1} \left[\tilde{P}_0^{-1} (r_0 + B_0 (P_1 b_0 + p_1)) \right], \quad (29c)$$

where $\tilde{R}_{e,k} = \tilde{P}_k^{-1} R_{e,k}$ and $\tilde{P}_k = P(R_{e,k})$.

4.4 Algorithm

Algorithm 3 presents an implementation guide of the preconditioned Riccati recursion based interior-point method.

5. RESULTS

This section presents our results based on an example application. We implement the proposed Riccati recursion

Algorithm 1: Riccati factorization

Input: $\{\bar{R}_k, Q_k, M_k, A_k, B_k\}_{k=0}^{N-1}, P_N$.

1. Compute,

$$R_{e,k} = \bar{R}_k + B_k P_{k+1} B_k^T, \quad (30a)$$

$$K_k = -R_{e,k}^{-1} (M_k^T + B_k P_{k+1} A_k^T), \quad (30b)$$

$$P_k = Q_k + A_k P_{k+1} A_k^T - K_k^T R_{e,k} K_k, \quad (30c)$$

for $k = N - 1, N - 2, \dots, 1$ and

$$R_{e,0} = \bar{R}_0 + B_0 P_1 B_0^T. \quad (31)$$

Return: $\{R_{e,k}, P_{k+1}\}_{k=0}^{N-1}, \{K_k\}_{k=1}^{N-1}$.

Algorithm 2: Riccati solution

Input: $\{Q_k, M_k, A_k, B_k, R_{e,k}, P_{k+1}\}_{k=0}^{N-1}, \{K_k\}_{k=1}^{N-1}$.

1. Compute,

$$a_k = -R_{e,k}^{-1} (\bar{r}_k + B_k (P_{k+1} b_k + p_{k+1})), \quad (32a)$$

$$p_k = q_k + A_k (P_{k+1} b_k + p_{k+1}) + K_k^T (\bar{r}_k + B_k (P_{k+1} b_k + p_{k+1})), \quad (32b)$$

for $k = N - 1, N - 2, \dots, 1$ and

$$a_0 = -R_{e,0}^{-1} (\bar{r}_0 + B_0 (P_1 b_0 + p_1)). \quad (33)$$

2. Compute the solution, $\{\Delta u_k, \Delta x_{k+1}\}_{k=0}^{N-1}$,

$$\Delta u_0 = a_0, \quad (34a)$$

$$\Delta x_1 = B_0^T \Delta u_0 + \tilde{b}_0, \quad (34b)$$

and

$$\Delta u_k = K_k \Delta x_k + a_k, \quad (35a)$$

$$\Delta x_{k+1} = A_k^T \Delta x_k + B_k^T \Delta u_k + b_k, \quad (35b)$$

for $k = 1, 2, \dots, N - 1$.

3. Compute the Lagrange multipliers, $\{\Delta y_k\}_{k=0}^{N-1}$,

$$\Delta y_{N-1} = P_N \Delta x_N + p_N, \quad (36a)$$

$$\Delta y_{k-1} = A_k \Delta y_k + Q_k \Delta x_k + M_k \Delta u_k + q_k, \quad (36b)$$

for $k = N - 1, N - 2, \dots, 1$.

Return: $\{\Delta u_k, \Delta x_{k+1}, \Delta y_k\}_{k=0}^{N-1}$.

based primal-dual interior-point algorithm in Matlab and consider a linearized modified quadruple tank system.

5.1 Modified quadruple tank system

We apply a deterministic nonlinear model for the mass balances of the quadruple tank system of the form (Azam and Jørgensen, 2015),

$$\dot{x}(t) = f(t, x(t), u(t), d(t), p), \quad (37a)$$

$$z(t) = h(x(t)), \quad (37b)$$

where $x \in \mathbb{R}^4$ is the four masses, $u \in \mathbb{R}^2$ is the two flow rates, $d \in \mathbb{R}^2$ is the two disturbance flows in the top tanks, p are the parameters, and $z \in \mathbb{R}^2$ is the heights in the bottom tanks. We linearize the model at the steady state, $x_s = [2110.2; 1761.2; 680.6; 394.0]$ [g], achieved for $u_s = [250; 325]$ [cm³/s] and $d_s = [100; 100]$ [cm³/s]. The output steady state is $z_s = [55.51; 46.33]$ [cm]. Additionally, we compute the exact discretization of the linear model with sampling time, $T_s = 15$ [s]. The result is a linear state space model,

$$x_{k+1} = A_k x_k + B_k u_k + E_k d_k, \quad (38a)$$

$$z_k = C_{z,k} x_k, \quad (38b)$$

Algorithm 3: Preconditioned Riccati recursion based primal-dual interior-point algorithm

Input: H, g, A, b, l, u (as in (27)), x_0, ϵ .

- Initialize: $y = 0, z_l = 1, z_u = 1, s_l = 1, s_u = 1$.
- Calculate the scaled KKT-violation, ξ , in (18).

while $\xi > \epsilon$ **do**
1. Predictor phase:

- Setup augmented system (11) with (14).
- Compute factorization, $\{R_{e,k}, P_{k+1}\}_{k=0}^{N-1}$ and $\{K_k\}_{k=1}^{N-1}$, with Algorithm 1.
- Solve the augmented system, (11), with Algorithm 2 for Δx^{aff} and Δy^{aff} .
- Compute $\Delta z_l^{aff}, \Delta z_u^{aff}, \Delta s_l^{aff}$, and Δs_u^{aff} in (16).
- Compute the affine step size, α^{aff} , with (17).
- Compute the duality gap and centering parameter in (10).

2. Corrector phase:

- Setup right-hand side of (11) with (15).
- Solve the augmented system (11) with Algorithm 2 for Δx and Δy .
- Compute $\Delta z_l, \Delta z_u, \Delta s_l$, and Δs_u in (16).
- Compute the step size, α , with (17).

3. Update $(x, y, z_l, z_u, s_l, s_u)$ according to (8).

4. Calculate the scaled KKT-violation, ξ , in (18).

Return: x, y, z_l, z_u, s_l, s_u .

where

$$A_k = \begin{bmatrix} 0.8659 & 0 & 0.1246 & 0 \\ 0 & 0.8659 & 0 & 0.1246 \\ 0 & 0 & 0.8659 & 0 \\ 0 & 0 & 0 & 0.8659 \end{bmatrix}, \quad (39a)$$

$$B_k = \begin{bmatrix} 9.7793 & 0.3926 \\ 0.2944 & 8.3822 \\ 0 & 5.5882 \\ 4.1911 & 0 \end{bmatrix}, \quad E_k = \begin{bmatrix} 0.9814 & 0 \\ 0 & 0.9814 \\ 13.970 & 0 \\ 0 & 13.970 \end{bmatrix}, \quad (39b)$$

$$C_{z,k} = \begin{bmatrix} 0.0026 & 0 & 0 & 0 \\ 0 & 0.0026 & 0 & 0 \end{bmatrix}, \quad (39c)$$

 for all k .

We consider the target tracking OCP,

$$\min_{x,u} \phi = \frac{1}{2} \sum_{k=1}^N \|z_k - \bar{z}_k\|_{Q_z}^2 + \frac{1}{2} \sum_{k=0}^{N-1} \|u_k - \bar{u}_k\|_{Q_u}^2, \quad (40a)$$

$$s.t. \quad x_0 = \hat{x}_0, \quad (40b)$$

$$x_{k+1} = A_k x_k + B_k u_k + E_k d_k, \quad (40c)$$

$$z_k = C_z x_k, \quad (40d)$$

$$u_{\min,k} \leq u_k \leq u_{\max,k}, \quad (40e)$$

where $k = 0, \dots, N-1$. In the general form, (23), we have $Q_k = C_{z,k}^T Q_z C_{z,k}$, $M_k = 0$, $R_k = Q_u$, $P_N = C_{z,N}^T Q_z C_{z,N}$, $q_k = -(Q_z C_z)^T \bar{z}_k$, $p_N = -(Q_z C_z)^T \bar{z}_N$, $r_k = -Q_u \bar{u}_k$, and $b_k = E_k d_k$ for all k . Additionally, we use $Q_z = I$, $Q_u = 0$, $u_{\min,k} = [0; 0]$, $u_{\max,k} = [500; 500]$, and $d_k = [100; 100]$ for all k , with initial condition, $x_0 = x_s$, discrete horizon, $N = 200$, and a variable target, \bar{z}_k , over the horizon. We state the OCP in deviation variables due to the linearization.

Figure 1 shows the solution to the OCP, (40). The OCP is solved with the proposed interior-point algorithm.

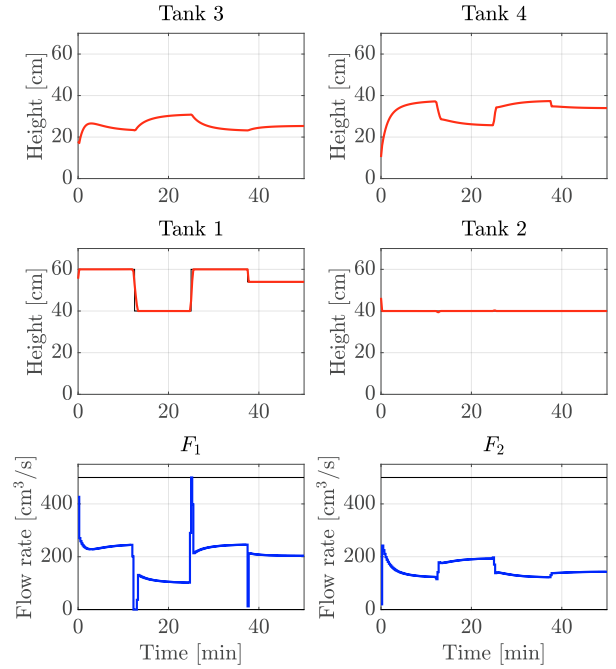


Fig. 1. Solution to target tracking OCP for linearized modified quadruple tank system with $N = 200$ and a variable target, \bar{z} . The controller is able to track the target in tank 1 and 2 (black line).

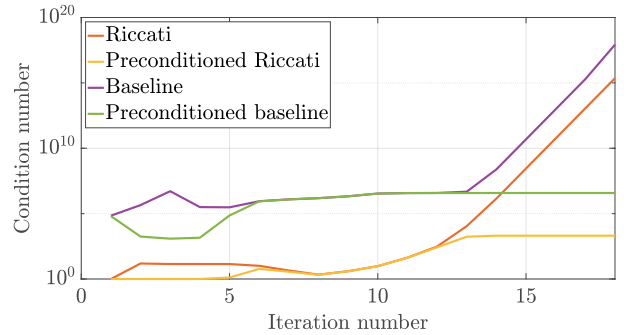


Fig. 2. Condition number of the system matrix in each iteration of the interior-point algorithm with four modes.

5.2 Condition number

We consider the interior-point algorithm in four modes,

- Baseline,
- Preconditioned Baseline,
- Riccati,
- Preconditioned Riccati,

where baseline means dense solution of the augmented system, (11). We compare the condition number of the system matrix with and without preconditioning at each iteration. For the Riccati solver, we consider the worst case condition number of the sub-system matrices. Fig. 2 presents the results. It is evident that the preconditioner improves the conditioning of the system matrices in both the baseline and Riccati recursion based interior-point algorithm.

5.3 CPU time

We apply the interior-point algorithm to solve the OCP for increasing control horizon, N . Fig. 3 presents the CPU

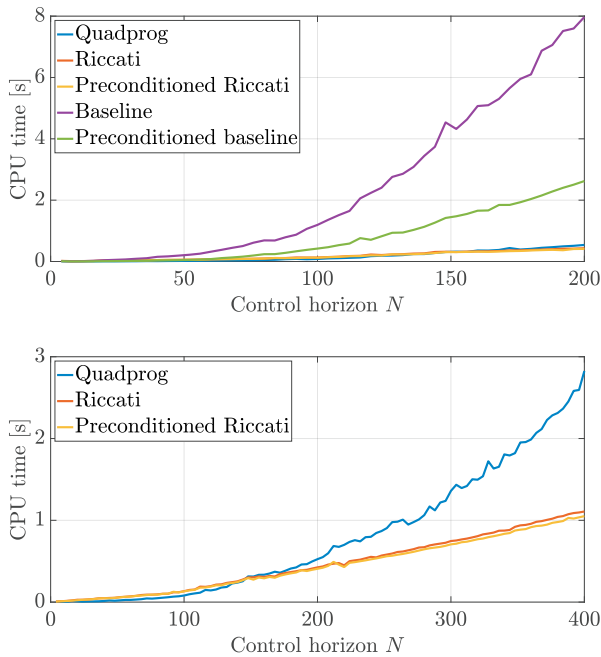


Fig. 3. CPU time for `quadprog` and interior-point algorithm. Top: Baseline and Riccati. Bottom: Riccati.

time for the interior-point method and `quadprog`. We observe that the Riccati based algorithm scales linearly in N and it is evident that the Riccati based algorithm outperforms the other algorithms for large N as expected. Additionally, the preconditioning does not increase CPU time rather it decreases the CPU time in these experiments. We point out that the implicit approach to the OCP results in a large sparse QP. Dense solvers like `quadprog` benefit from explicit approaches that produce small dense QPs. Thus, a fair CPU time comparison would require an explicit approach for `quadprog`. Such comparison is out of scope of this paper.

6. CONCLUSION

The paper presents a preconditioned Riccati recursion based interior-point algorithm tailored for QPs arising in input constrained OCPs. We implement the interior-point algorithm in Matlab and solve an OCP for target tracking of a linearized modified quadruple tank system. The results show that the diagonal preconditioner improves conditioning of the linear sub-systems of equations in the Riccati recursion.

This paper contains a detailed description of the proposed algorithm and serves as an implementation guide.

REFERENCES

Astfalk, G., Lustig, I., Marsten, R., and Shanno, D. (1992). The Interior-Point Method for Linear-Programming. *IEEE Software*, 9(4), 61–68.

Azam, S.N.M. and Jørgensen, J.B. (2015). Modeling and simulation of a modified quadruple tank system. *IEEE International Conference on Control Systems, Computing and Engineering (ICCSCE)*, 365–370.

Bergamaschi, L., Gondzio, J., and Zilli, G. (2004). Preconditioning Indefinite Systems in Interior Point Methods for Optimization. *Computational Optimization and Applications*, 28(2), 149–171.

Borrelli, F., Pekar, J., Baotić, M., and Stewart, G. (2009). On the Computation of Linear Model Predictive Control laws. *IEEE Conference on Decision and Control (CDC)*.

Byrd, R.H., Hribar, M.E., and Nocedal, J. (1999). An Interior Point Algorithm for Large-scale Nonlinear Programming. *SIAM Journal on Optimization*, 9(4), 877–900.

Cui, Y., Morikuni, K., Tsuchiya, T., and Hayami, K. (2019). Implementation of Interior-point Methods for LP based on Krylov Subspace Iterative Solvers with Inner-iteration Preconditioning. *Computational Optimization and Applications*, 74(1), 143–176.

Forsgren, A., Gill, P.E., and Wright, M.H. (2002). Interior Methods for Nonlinear Optimization. *SIAM Review*, 44(4), 525–597.

Frison, G. and Jørgensen, J.B. (2013). Efficient Implementation of the Riccati Recursion for Solving Linear-Quadratic Control Problems. *IEEE International Conference on Control Applications (CCA), Hyderabad, India*, 1117–1122.

Gertz, E.M. and Wright, S.J. (2003). Object-Oriented Software for Quadratic Programming. *ACM Transactions on Mathematical Software*, 29(1), 58–81.

Jørgensen, J.B. (2004). *Moving Horizon Estimation and Control*. Ph.D. thesis, Technical University of Denmark.

Mehrotra, S. (1992). On The Implementation of a Primal-dual Interior Point Method. *SIAM Journal on Optimization*, 2(4), 575–601.

Murray, W. (1971). Analytical expressions for the eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions. *Journal of Optimization Theory and Applications*, 7(3), 189–196.

Nocedal, J. and Wright, S.J. (1999). *Numerical Optimization*. Springer.

Rao, C.V., Wright, S.J., and Rawlings, J.B. (1998). Application of Interior-Point Methods to Model Predictive Control. *Journal of Optimization Theory and Applications*, 99(3), 723–757.

Rawlings, J., Meadows, E., and Muske, K. (1994). Nonlinear Model Predictive Control: A Tutorial and Survey. *IFAC Advanced Control and Chemical Processes, Kyoto, Japan*.

Shahzad, A., Kerrigan, E.C., and Constantinides, G.A. (2010). A Fast Well-conditioned Interior Point Method for Predictive Control. *IEEE Conference on Decision and Control (CDC), Atlanta, USA*.

Vanderbei, R.J. (1999). LOQO:an interior point code for quadratic programming. *Optimization Methods and Software*, 11(1-4), 451–484.

Wahlgreen, M.R., Reenberg, A.T., Nielsen, M.K., Rydahl, A., Ritschel, T.K.S., Dammann, B., and Jørgensen, J.B. (2021). A High-Performance Monte Carlo Simulation Toolbox for Uncertainty Quantification of Closed-loop Systems. *IEEE Conference on Decision and Control (CDC), Accepted*.

Wright, M.H. (2004). The interior-point revolution in optimization: history, recent developments, and lasting consequences. *American Mathematical Society*, 42, 39–56.

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.

APPENDIX C

Paper III: DYCOPS 2022

Modeling and Simulation of Upstream and Downstream Processes for Monoclonal Antibody Production

Authors:

Morten Ryberg Wahlgreen, Kristian Meyer, Tobias K. S. Ritschel, Allan Peter
Engsig-Karup, Krist V. Gernaey, John Bagterp Jørgensen

Published in:

Proceedings of the 13th IFAC Symposium on Dynamics and Control of Process
Systems, including Biosystems (DYCOPS), Busan, Republic of Korea, June 14–17,
2022.

<https://doi.org/10.1016/j.ifacol.2022.07.523>

Modeling and Simulation of Upstream and Downstream Processes for Monoclonal Antibody Production

Morten Ryberg Wahlgreen* Kristian Meyer^{**,***}
Tobias K. S. Ritschel* Allan Peter Engsig-Karup*
Krist V. Gernaey** John Bagterp Jørgensen^{*,****}

* Department of Applied Mathematics and Computer Science,
Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark.

** Department of Chemical and Biochemical Engineering, PROSYS,
Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark.

*** MCT Bioseparation ApS, DK-2800 Kgs. Lyngby, Denmark.

**** 2-control ApS, DK-7400 Herning, Denmark.

Abstract:

We present a numerical case study for modeling and simulation of upstream and downstream processes for monoclonal antibody (mAb) production. We apply a systematic and intuitive modeling methodology for an existing upstream process and downstream process. The resulting models are based on differential mass balances and kinetic expressions for the reactions and adsorption. Mass balances for the fedbatch reactor yield a model consisting of five ordinary differential equations (ODEs). The downstream process is conducted batchwise in a chromatographic column for capture of mAbs. Mass balances of the chromatographic column yield a system of partial differential equations (PDEs). The chromatographic model applies the nonlinear shrinking core adsorption isotherm model for transition between the mobile phase and the stationary phase. We apply a high-order spectral nodal continuous Galerkin scheme for spatial discretization of the chromatographic column, which result in a semi-discrete ODE formulation. The resulting simulation model, coupling the upstream and downstream processes in batchwise mAb production, can be used as a benchmark for numerical estimation, control and optimization studies.

Copyright © 2022 The Authors. This is an open access article under the CC BY-NC-ND license (<https://creativecommons.org/licenses/by-nc-nd/4.0/>)

Keywords: Monoclonal antibody production, fedbatch reactor, capture chromatography, shrinking core adsorption isotherm, process modeling.

1. INTRODUCTION

Monoclonal antibodies (mAbs) is a class of biopharmaceuticals representing the 6 top-selling biopharmaceutical products in 2017 (Walsh, 2018). There has been a huge increase in the sale of mAbs the previous years, and the yearly sale is expected to grow to 130-200 billion US dollars in 2022 (Grilo and Mantalaris, 2019). As such, new technology is needed to keep up with the increasing demand. The emerging need for huge mAb production has led to recent research in optimization of biotechnological processes for mAb production (Badr et al., 2021; Gomis-Fons et al., 2021). Most optimization is related to the development of mathematical models to support operation of the processes.

Modeling of upstream and downstream processes is well developed. However, often the model presentations are unnecessarily complicated and the chain rule is applied to the mass balance equations for fedbatch reactor models. Due to systematic and intuitive modeling and numerical considerations, we present the differential mass balance

models in compact forms without using the chain rule. When used systematically, this modeling methodology has significant advantages in the modeling phase and in the numerical implementation phase. Additionally, the models presented in this paper are well-suited for model based optimization such as model predictive control (MPC). We intend to apply the developed models in model based optimization in future work and the key objective of the paper is to present a well-defined simulation model for batchwise mAb production, that can be used as a benchmark in numerical simulation, control and optimization studies.

This paper presents a numerical case study that combines modeling of an upstream and downstream process for mAb production. Fig. 1 presents an overview of the reactor (upstream process) and chromatographic column (downstream process). We apply the proposed modeling methodology to existing models for the upstream process and downstream process (Badr et al., 2021). We show that the methodology allows for easy modeling of fedbatch reactors for upstream processes. For the downstream process, the methodology results in a general compact model, which is even applicable for a wide range of reactions conducted in columns. We apply a high-order spectral nodal con-

* Corresponding author: J. B. Jørgensen (E-mail: jbjo@dtu.dk).

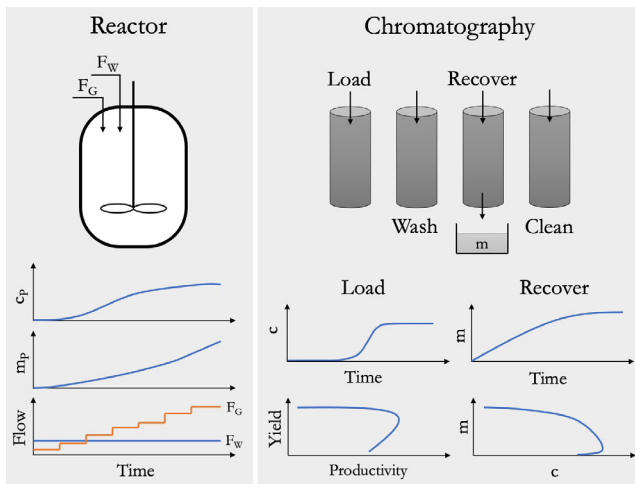


Fig. 1. Overview of the fedbatch reactor for fermentation and the chromatographic column operated batchwise.

tinuous Galerkin method (CGM) for space discretization of the chromatographic column. The method is based on Legendre polynomials on a Gauss-Lobatto grid. Recent results show that high order methods are very suitable for chromatographic processes as they outperform widely applied finite volume (FV) methods (Hørsholt et al., 2019a,b; Meyer et al., 2020). The chromatography model applies a nonlinear shrinking core adsorption isotherm model for transition between the mobile phase and the stationary phase. We demonstrate that the upstream and downstream model can be applied for simulation of mAb production and capture of mAbs respectively. The simulations of the chromatographic column show a trade-off between the productivity and yield of the column in the loading phase and a similar trade-off for captured mAbs and concentration of mAbs in the recovery phase. The trade-off indicates that optimization of the process can be advantageous.

The remaining part of the paper is organized as follows. Section 2 introduces the upstream process. Section 3 presents the downstream chromatographic process. Section 4 describes the applied methods for simulation. Section 5 presents the results. Section 6 presents the conclusion.

2. UPSTREAM PROCESS

We consider an upstream process for monoclonal antibody production conducted in a fedbatch reactor. The model was originally presented by Badr et al. (2021). We apply a systematic modeling methodology that is well-suited for modeling of the reactor in simulation, control, and optimization studies.

2.1 General model for fedbatch fermentation

The mass (mole) balance for well-mixed fedbatch fermentation, assuming constant and identical density, can be compactly states as (Ryde et al., 2021),

$$\frac{dV}{dt} = e^T F, \quad (1a)$$

$$\frac{dn}{dt} = C_{in} F + R V. \quad (1b)$$

Let \mathcal{C} denote the set of components (molecules) and let \mathcal{S} denote the set of inlet streams. $V \in \mathbb{R}$ is the medium volume, $e \in \mathbb{R}^{|\mathcal{S}|}$ is a vector of ones, $F \in \mathbb{R}^{|\mathcal{S}|}$ is the vector of flow rates in the inlet streams, $n \in \mathbb{R}^{|\mathcal{C}|}$ is the vector of mole numbers for each component in the reactor, $C_{in} \in \mathbb{R}^{|\mathcal{C}| \times |\mathcal{S}|}$ is a matrix of concentrations in the inlet streams, and $R \in \mathbb{R}^{|\mathcal{C}|}$ is the production rate vector. The concentration vector, $c \in \mathbb{R}^{|\mathcal{C}|}$, is

$$c = \frac{n}{V}. \quad (2)$$

Let \mathcal{R} denote the set of reactions and let $S \in \mathbb{R}^{|\mathcal{R}| \times |\mathcal{C}|}$ denote the stoichiometric matrix for the considered reactions and components. $r \in \mathbb{R}^{|\mathcal{R}|}$ denotes the reaction rate vector. Specification of the stoichiometry and kinetics, i.e. S and $r = r(c)$, enables computation of the production rate,

$$r = r(c), \quad (3a)$$

$$R = S^T r, \quad (3b)$$

and completes the model.

In addition, the masses of the components, $m \in \mathbb{R}^{|\mathcal{C}|}$, may be of interest. It is computed by

$$m = M_w \odot n, \quad (4)$$

in which \odot denotes elementwise multiplication and $M_w \in \mathbb{R}^{|\mathcal{C}|}$ is the molecular mass vector.

Remark. The model (1)-(4) is a general compact form, which is easy to implement. Additionally, the model equations reduce the actual modeling to specification of: \mathcal{C} , \mathcal{R} , S , $r(c)$, C_{in} , F , and M_w , together with selection of initial conditions for (1).

Usual practice is to apply the chain rule to the mass balance equations (1) to represent the states as concentrations, c , rather than mole numbers, n . The result is a set of equations on the form,

$$\frac{dV}{dt} = e^T F, \quad (5a)$$

$$\frac{dc}{dt} = (C_{in} - ce^T) \frac{F}{V} + S^T r(c). \quad (5b)$$

However, we strongly recommend to apply (1) rather than (5), as (1) is more intuitive and application of the chain rule is only valid for an infinitely small, dt . As such, the formulations (1) and (5) are not numerical equivalent when solved with numerical solvers. The novelty of this work lies in the intuitive and compact formulation of the model (1).

2.2 Reaction stoichiometry and kinetics

We demonstrate the application of the general modeling methodology on a fedbatch fermentation model for mAb production (Badr et al., 2021). The model consists of four components,

$$\mathcal{C} = \{X, G, L, P\}, \quad (6)$$

where X is viable cells, G is glucose, L is lactate, and P is the product (mAbs), and five reactions,

$$\mathcal{R} = \{1, 2, 3, 4, 5\}. \quad (7)$$

The stoichiometry of the process is,

1. Cell production, $\alpha_{1,G}G + X \longrightarrow 2X$, r_1 ,
2. Cell death, $X \longrightarrow \alpha_{2,G}G$, r_2 ,
3. Cell maintenance, $\alpha_{3,G}G + X \longrightarrow X$, r_3 ,
4. Lactate production, $X \longrightarrow X + \alpha_{4,L}L$, r_4 ,
5. Product formation, $X \longrightarrow X + \alpha_{5,P}P$, r_5 ,

which can be represented by the stoichiometric matrix,

$$S = \begin{bmatrix} X & G & L & P \\ 1 & -\alpha_{1,G} & 0 & 0 \\ -1 & \alpha_{2,G} & 0 & 0 \\ 0 & -\alpha_{3,G} & 0 & 0 \\ 0 & 0 & \alpha_{4,L} & 0 \\ 0 & 0 & 0 & \alpha_{5,P} \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \quad (8)$$

The reaction rates, $r = [r_1 \ r_2 \ r_3 \ r_4 \ r_5]^T$, are,

$$r_1 = \mu_X(c_G, c_L)c_X, \quad r_2 = \mu_D(c_G, c_L)c_X, \quad (9a)$$

$$r_3 = \mu_M c_X, \quad r_4 = \mu_L c_X, \quad (9b)$$

$$r_5 = \mu_P c_X, \quad (9c)$$

where $\mu_X(c_G, c_L)$ and $\mu_D(c_G, c_L)$ are governed by Monod growth kinetics,

$$\mu_X = \mu_{max} \left(\frac{c_G}{K_G + c_G} \right) \left(\frac{K_L}{K_L + c_L} \right), \quad (10a)$$

$$\mu_D = k_d \left(\frac{c_L}{K_{DL} + c_L} \right) \left(\frac{K_{DG}}{K_{DG} + c_G} \right), \quad (10b)$$

and μ_M , μ_L , and μ_P are parameters for estimation.

2.3 Inlet streams

The reactor is operated in fedbatch mode with two inlets; 1) an inlet containing glucose and 2) a pure water inlet. We denote the flow rate of the water inlet and glucose inlet F_W and F_G , respectively. The concentration of glucose in the glucose inlet is denoted $c_{G,in}$. Hence,

$$C_{in} = \begin{bmatrix} 0 & 0 \\ c_{G,in} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad F = \begin{bmatrix} F_G \\ F_W \end{bmatrix}, \quad S = \{S_G, S_W\}. \quad (11)$$

The split into a water inlet and a substrate inlet makes the model affine in the inlet flow rates (Ryde et al., 2021). This is beneficial in optimization studies with the model.

3. DOWNSTREAM PROCESS

We consider a downstream chromatographic process for capture of the product P , i.e., mAbs. Similarly to the upstream process, the chromatographic model was originally presented by Badr et al. (2021). We point out that P does not distinguish between a product with deficiencies (from the reactor) and a product of higher quality (after the chromatographic process). We demonstrate that our systematic modeling methodology is applicable for the downstream process and results in a general model formulation for the capture chromatography process.

We assume that the process is conducted in a column with diameter, d_c , packed with a porous media with porosity, ε_c . We denote the total volume of the column, V , which allows for definition of the liquid volume, $V_l = \varepsilon_c V$, and the stationary volume, $V_s = (1 - \varepsilon_c)V$. The stationary volume, V_s , contains a number of spherical particles,

N_p , with porosity, ε_p . The components of the model are the mobile phase concentration, $c(t, z)$, the pore phase concentration of free molecules, $c_p(t, z)$, and the pore phase concentration of bound molecules at different binding sites, $q(t, z)$. Additionally, we derive the general mass balance model under the following assumptions (Meyer, 2020),

- The column is homogeneously packed with particles.
- The particles are porous and spherical with constant diameter.
- The mobile phase density is constant.
- The viscosity is constant.
- Operational conditions are isothermal and adiabatic.
- No convection inside particles.
- No radial dispersion in the mobile phase.
- No diffusion in the pore phase.

3.1 General model for chromatography

The mass balances in the chromatography model are

$$\partial_t c = -\partial_z N + R, \quad (12a)$$

$$\partial_t c_p = R_p, \quad (12b)$$

$$\partial_t q = R_q, \quad (12c)$$

where (12a) is based on the mobile liquid volume and (12b)-(12c) are based on the pore volume in the particles. N denotes the flux in the mobile liquid phase and R denotes the transport (production) rate from the mobile liquid volume to the particle pore volume. Similarly, R_p and R_q denote the transport of molecules to the free and bound particle pore volume, respectively. We impose a convective flow inlet and outlet boundary condition,

$$N(t, 0) = v c_{in}(t), \quad (13a)$$

$$N(t, L) = v c(t, L), \quad (13b)$$

together with the following initial condition,

$$c(0, z) = c_0(z), \quad (14a)$$

$$c_p(0, z) = c_{p,0}(z), \quad (14b)$$

$$q(0, z) = q_0(z). \quad (14c)$$

The flux in the mobile liquid phase, N , is governed by advection (convection) and Fickian diffusion,

$$N = v c + J, \quad (15a)$$

$$J = -D \partial_z c, \quad (15b)$$

where v is the linear mobile liquid phase velocity in the column, J denotes Fick diffusion, and D is the diffusion coefficient.

The transport rate from the mobile liquid phase to the pore volume, R , is based on the open surface area of the particles. The open particle surface area per liquid volume in the chromatography column is

$$\begin{aligned} \phi_l &= \frac{N_p A_p \varepsilon_p}{V_l} = \frac{V_s}{V_p} \frac{A_p \varepsilon_p}{V_l} \\ &= \frac{(1 - \varepsilon_c) V}{(4/3)\pi (d_p/2)^3} \frac{4\pi (d_p/2)^2}{\varepsilon_c V} \varepsilon_p = \frac{1 - \varepsilon_c}{\varepsilon_c} \varepsilon_p \frac{6}{d_p}, \end{aligned} \quad (16)$$

where V_p is the volume of a particle and d_p is the particle diameter. The open pore surface area per pore volume is

$$\phi_p = \frac{N_p A_p \varepsilon_p}{N_p V_p \varepsilon_p} = \frac{4\pi (d_p/2)^2}{(4/3)\pi (d_p/2)^3} = \frac{6}{d_p}. \quad (17)$$

Consequently, the transport rates are

$$R = -\phi_l r_p, \quad (18a)$$

$$R_p = \phi_p r_p + S_p^T r_q, \quad (18b)$$

$$R_q = S_q^T r_q, \quad (18c)$$

where $S = [S_p \ S_q]$ is the stoichiometric matrix for the transport rates (reaction rates) in the pores of the particle. r_p is the kinetic expression for the transport of molecules from the mobile liquid volume to the pore space of the particles. The transport, r_p , is given per open surface area. r_q is the kinetic expression for the transport of molecules in the particle pore spaces. These transport (reaction) rates can be expressed as

$$r_p = r_p(c, c_p), \quad (19a)$$

$$r_q = r_q(c_p, q). \quad (19b)$$

Remark. The model (12)-(19) is a general and compact formulation of the chromatographic column under the given assumptions. The mobile phase transport equation, (12a), can even be applied to model general reactions conducted in columns, e.g., reactions conducted in a plug-flow reactor.

Usual practice is to insert expressions (15)-(18) into (12) and get

$$\partial_t c = -v \partial_z c + D \partial_{zz} c - \frac{1 - \epsilon_c}{\epsilon_c} \epsilon_p \frac{6}{d_c} r_p, \quad (20a)$$

$$\partial_t c_p = \frac{6}{d_c} r_p + S_q^T r_q, \quad (20b)$$

$$\partial_t q = S_q^T r_q, \quad (20c)$$

and even further insert selections of S_p , S_q , r_p , and r_q . We strongly recommend to apply the formulation (12) as it reduces the actual modeling of the chromatographic column to selection of production rates, R , R_p , and R_q , i.e., selection of S_p , S_q , r_p , and r_q , together with selection of inlet concentration, $c_{in}(t)$, and initial conditions (14). Also, the model (12)-(19) is a more intuitive formulation compared to (20). The novelty of this work is the introduction of the very compact model (12).

3.2 Shrinking core adsorption isotherm

We demonstrate the general modeling methodology on a capture chromatography model (Badr et al., 2021). The model applies a shrinking core adsorptions isotherm, which has previously been proposed as a valid model for modeling of chromatographic processes (Baur et al., 2016a,b). As such, we select the quantities, S_p , S_q , r_p , and r_q based on the shrinking core adsorption model.

The kinetic expression for the transport rate from the mobile phase to the pore space of the particles are

$$r_p = k(c - c_p), \quad (21)$$

where k is the transport coefficient.

The shrinking core adsorption isotherm assumes two sites for adsorption, $q = [q_1, q_2]^T$. The stoichiometry of the transport in the pores of the particles is

$$S = [S_p \ S_q] = \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad (22)$$

and the corresponding kinetic expression is

$$r_q = \begin{bmatrix} k_{A,1} \left(c_p (q_{\text{sat}} - q_1) - \frac{q_1}{k_{eq}} \right) \\ k_{A,2} \left(c_p (q_1 - q_2) - \frac{q_2}{k_{eq}} \right) \end{bmatrix}, \quad (23)$$

where $k_{A,1}$ and $k_{A,2}$ are the adsorption rate constants for each site, q_{sat} is the saturation capacity, and k_{eq} is the equilibrium constant of the adsorption process.

The transport coefficient, k , is given as a combination of the two site contributions as,

$$\frac{1}{k} = \frac{1}{k_F} + \frac{1}{k_S}, \quad (24)$$

where k_F is the film transfer coefficient and k_S is the pore transfer coefficient. The two transfer coefficients can be computed as,

$$k_F = \frac{1.09 u_{sf}}{\epsilon_c}, \quad (25)$$

$$k_S = D_s \frac{(1 - \alpha)^{1/3}}{1 - (1 - \alpha)^{1/3}}, \quad (26)$$

where D_s is a fitting parameter, $u_{sf} = v \epsilon_c$ is the superficial velocity, and

$$\alpha = \frac{q_1}{q_{\text{sat}}} \frac{1/k_{eq} + c_{in}}{c_{in}}. \quad (27)$$

The combination of the shrinking core model and the general PDE for chromatographic processes, (12), results in nonlinear chromatography described by a nonlinear PDE.

3.3 Loading of column

We select a rectangular pulse inlet boundary condition,

$$c_{in}(t) = \begin{cases} 0, & t < t_1 \\ c_f, & t_1 \leq t \leq t_2 \\ 0, & t > t_2 \end{cases}, \quad (28)$$

where c_f is the feed concentration, t_1 is the start of the loading phase, and t_2 is the end of the loading phase.

3.4 Yield and productivity in loading phase

The capture chromatography process has four phases as illustrated in Fig. 1; 1) loading of column, 2) washing of column, 3) recovery of product, and 4) cleaning of column. We assume constant operation time for phase 2-4 and denote the time in those phases t_c . The time in phase 2-4 is given as $t_c = 21\text{CV}$ (column volumes) (Badr et al., 2021), i.e., the time it takes the liquid to run through the column 21 times. As such, we consider the yield, Y , and the productivity, Q , of the column,

$$Y(t) = \frac{m}{m_{in}}, \quad (29)$$

$$Q(t) = \frac{m}{t + t_c}, \quad (30)$$

where $m = m(t)$ [g] is the total accumulated mass of captured mAbs in the column at time t , and $m_{in} = m_{in}(t)$ is the total accumulated mass of mAbs injected to the column at time t .

Table 1. Parameters for reactor model.

Parameter	Value	Unit
K_G	0.0	[mmol/L]
K_L	7.10	[mmol/L]
K_{DG}	1.54	[mmol/L]
K_{DL}	0.0	[mmol/L]
μ_{max}	5.17×10^{-2}	[h ⁻¹]
k_d	2.32×10^{-2}	[h ⁻¹]
μ_M, μ_L, μ_P	1.0	[h ⁻¹]
$\alpha_{1,G}$	7.52×10^{-10}	[mmol/cell]
$\alpha_{2,G}$	$= \alpha_{1,G}$	[mmol/cell]
$\alpha_{3,G}$	82.3×10^{-12}	[mmol/cell]
$\alpha_{4,L}$	2.76×10^{-11}	[mmol/cell]
$\alpha_{5,P}$	5.45×10^{-15}	[mmol/cell]
M_P	$\approx 0.15 \times 10^3$	[g/mmol]
$c_{G,in}$	130	[g/L]
M_G	0.180156	[g/mmol]

Table 2. Parameters for chromatographic process.

Parameter	Value (Load/Recover)	Unit
L	2.0	[cm]
d_c	15.0	[cm]
ε_c	0.36	[-]
d_p	85.0×10^{-4}	[cm]
ε_p	0.52	[-]
u_{sf}	1.33	[cm/min]
D	5.0×10^3	[cm ² /min]
c_f	2.4847 / 0.0	[g/L]
t_1	0 / 0	[min]
t_2	180 / 30	[min]
D_s	2.23×10^{-3}	[cm/min]
$k_{A,1}$	6.77×10^4	[min ⁻¹]
$k_{A,2}$	3.18×10^4	[min ⁻¹]
k_{eq}	61.47 / 0.001	[L/g]
q_{sat}	69.10	[g/L]
t_c	45.36	[min]

4. SIMULATION

The reactor model (1) consists of a system of non-stiff ODEs, which we solve in Matlab with `ode45`.

We apply a high-order CGM for spatial discretization of the chromatography model (12) (Hesthaven and Warburton, 2008). We solve the resulting system of semi-discrete ODEs in Matlab with an implicit variable step-size solver, `ode15s`, as the spatially discretized system is stiff.

5. RESULTS

This section presents simulation results for both models.

5.1 Upstream simulation

We simulate the reactor model (1). Table 1 presents the model parameters (Badr et al., 2021). Fig. 2 presents the simulation. The final concentration of the product, mAbs, is 2.48 [g/L], which we apply as inlet concentration, c_f , for the capture chromatography process.

5.2 Downstream simulation

We simulate the loading phase and recovery phase of the capture chromatographic model (12). Table 2 provides the parameters for the simulation adapted from Badr et al. (2021). Fig. 3 presents the result. Fig. 3a and

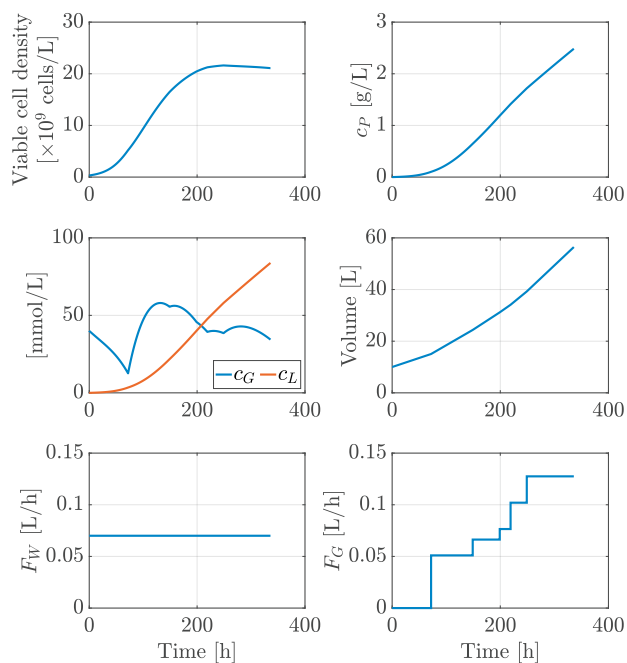


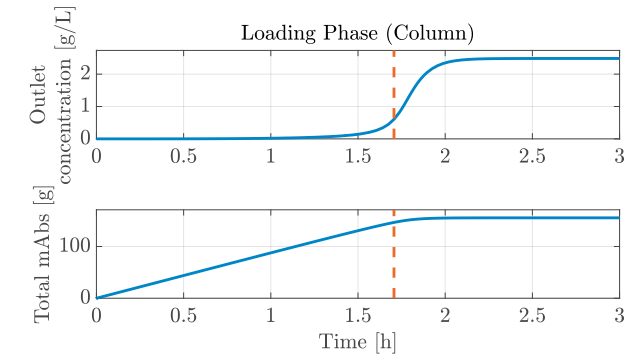
Fig. 2. Simulation of upstream process. States at the final time: $c_P = 2.48$ [g/L], $V = 56.44$ [L], and $m_P = 140.25$ [g].

3b presents the loading phase results, where we observe a breakthrough in the column and a trade-off between yield and productivity of the column. Fig. 3c and 3d presents the recovery phase results, where we observe a trade-off between recovered mAb mass and recovered mAb concentration.

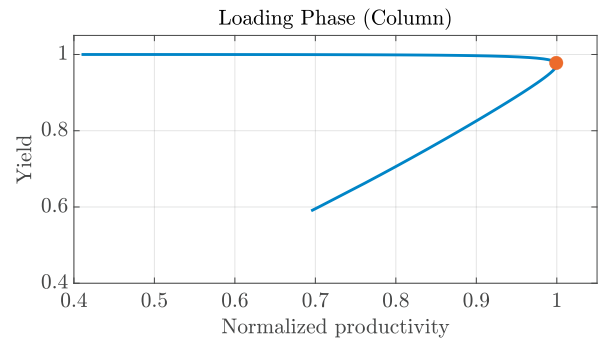
6. CONCLUSION

The paper presents a systematic methodology for upstream and downstream process modeling. The methodology simplifies the model presentation and reduces the actual modeling to selection of stoichiometric matrices and reaction kinetics for both upstream and downstream processes. We demonstrate the modeling methodology on an existing model for mAb production in a fedbatch reactor and a chromatography model for capture of mAbs. The upstream process model is a five component ODE, while the downstream process is a nonlinear PDE. The chromatographic model applies a nonlinear shrinking core adsorption isotherm to model the transition between the mobile phase and the stationary phase in the chromatographic column. We discretize the nonlinear PDE in space with a high-order spectral continuous Galerkin scheme that can be expanded to a multi-element spectral scheme for better resolution and scalability. Simulation of the upstream and downstream processes shows that the modeling methodology works as intended. In particular, the chromatographic process results in a Pareto front for the yield and productivity in the loading phase and a Pareto front for the concentration of mAbs and captured mAbs in the recovery phase.

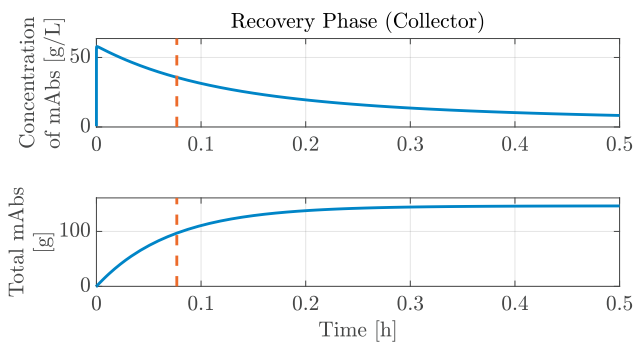
The proposed modeling methodology is well-suited for model-based optimization of the upstream and downstream processes.



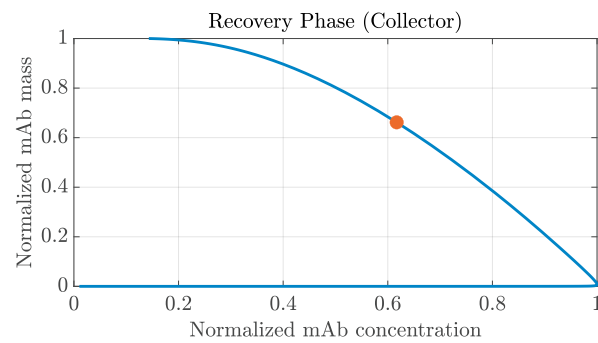
(a) Loading phase of the column. 1) The mobile phase concentration in the outlet of the column, where a breakthrough is observed, and 2) The total mAbs captured in the column. The red dotted line indicates a possible operating point for ending the loading phase.



(b) The normalized productivity vs. the yield of the loading phase. The plot forms a Pareto front. The red dot indicates the operating point indicated in Fig. 3a.



(c) Recovery phase of the column after the loading operation indicated on Fig. 3a. 1) The concentration of mAbs in the collector, and 2) The total mAbs in the collector. The red dotted line indicates a possible operating point for ending the recovery phase.



(d) Normalized concentration vs. normalized total product of the recovery phase. The plot forms a Pareto front. The red dot indicates the operating point indicated in Fig. 3c.

Fig. 3. Simulation of loading phase and recovery phase of the chromatographic column.

REFERENCES

- Badr, S., Okamura, K., Takahashi, N., Ubbenjans, V., Shirahata, H., and Sugiyama, H. (2021). Integrated design of biopharmaceutical manufacturing processes: Operation modes and process configurations for monoclonal antibody production. *30th European Symposium on Computer Aided Process Engineering (ESCAPE)*, 153, 107422.
- Baur, D., Angarita, M., Müller-Späth, T., and Morbidelli, M. (2016a). Optimal model-based design of the twin-column CaptureSMB process improves capacity utilization and productivity in protein A affinity capture. *Biotechnology Journal*, 11(1), 135–145.
- Baur, D., Angarita, M., Müller-Späth, T., Steinebach, F., and Morbidelli, M. (2016b). Comparison of batch and continuous multi-column protein A capture processes by optimal design. *Biotechnology Journal*, 11(7), 920–931.
- Gomis-Fons, J., Yamane-Nolin, M., Andersson, N., and Nilsson, B. (2021). Optimal loading flow rate trajectory in monoclonal antibody capture chromatography. *Journal of Chromatography A*, 1635, 461760.
- Grilo, A.L. and Mantalaris, A. (2019). The Increasingly Human and Profitable Monoclonal Antibody Market. *Trends in Biotechnology*, 37(1), 9–16.
- Hesthaven, J.S. and Warburton, T. (2008). *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, volume 54. Springer.
- Hørsholt, A., Christiansen, L.H., Meyer, K., Huusom, J.K., and Jørgensen, J.B. (2019a). Spatial Discretization and Kalman Filtering for Ideal Packed-Bed Chromatography. *18th European Control Conference (ECC)*, 2356–2361.
- Hørsholt, A., Christiansen, L.H., Ritschel, T.K., Meyer, K., Huusom, J.K., and Jørgensen, J.B. (2019b). State and Input Estimation of Nonlinear Chromatographic Processes. *Conference on Control Technology and Applications (CCTA)*, 1030–1035.
- Meyer, K. (2020). *Advanced simulation of preparative chromatography processes*. Ph.D. thesis, Technical University of Denmark.
- Meyer, K., Leweke, S., von Lieres, E., Huusom, J.K., and Abildskov, J. (2020). Chromatech: A discontinuous Galerkin spectral element simulator for preparative liquid chromatography. *Computers and Chemical Engineering*, 141, 107012.
- Ryde, T.E., Wahlgreen, M.R., Nielsen, M.K., Hørsholt, S., Jørgensen, S.B., and Jørgensen, J.B. (2021). Optimal Feed Trajectories for Fedbatch Fermentation with Substrate Inhibition Kinetics. *IFAC-PapersOnline*, 54(3), 318–323.
- Walsh, G. (2018). Biopharmaceutical benchmarks 2018. *Nature Biotechnology*, 36(12), 1136–1145.

APPENDIX D

Paper IV: FOCAPO/CPC 2023

Model Predictive Control Tuning by Monte Carlo Simulation and Controller Matching

Authors:

Morten Ryberg Wahlgreen, John Bagterp Jørgensen, Mario Zanon

To be published in:

Proceedings of Foundations of Computer Aided Process Operations / Chemical Process Control (FOCAPO/CPC), San Antonio, Texas, USA, January 8–12, 2023.
arXiv: <https://doi.org/10.48550/arXiv.2212.02142>

MODEL PREDICTIVE CONTROL TUNING BY MONTE CARLO SIMULATION AND CONTROLLER MATCHING

Morten Ryberg Wahlgreen ^a, John Bagterp Jørgensen ^{a,1} and Mario Zanon ^b

^a Department of Applied Mathematics and Computer Science, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark

^b IMT School for Advanced Studies Lucca, IT-55100 Lucca, Italy

Abstract

This paper presents a systematic method for the selection of the Model Predictive Control (MPC) stage cost. We match the MPC feedback law to a proportional-integral (PI) controller, which we efficiently tune by high-performance Monte Carlo (MC) simulation. The PI tuning offers a wide range of tuning possibilities that is then inherited by the MPC design. The MC simulation tuning of the PI controller is based on the minimization of two different objectives; 1) the 2-norm tracking error, and 2) a bi-objective consisting of the 2-norm tracking error and a 2-norm input rate of movement penalty. We apply the method to design MPC for an exothermic chemical reaction conducted in an adiabatic continuous stirred tank reactor (CSTR). The process is of interest as the nonlinear dynamics result in a desired operating point very close to a constraint. Our MPC design includes stage costs automatically designed to match the tuned PI controllers, hard input constraints, and a soft output constraint. Stochastic simulation results show that both the PI controller and the MPC can track the desired operating point. However, the MPC shows reduced output constraint violation compared to the PI controller. As such, the MPC design method successfully combines the efficient tuning of the PI controller with the constraint handling properties of MPC.

Keywords

Model Predictive Control, Controller Matching, Automatic Monte Carlo Simulation Tuning.

Introduction

Model Predictive Control (MPC) is an advanced control technology, that is widely applied in the industry (Qin and Badgwell, 2003). MPC offers direct methods to handle constraints in the specific system. However, the lack of systematic tuning methods can make the selection of the MPC parameters cumbersome. On the contrary, simple linear controllers, such as proportional–integral (PI) controllers, offer systematic tuning options. Therefore, it has previously been proposed to match the MPC stage cost to tuned linear controllers (Di Cairano and Bemporad, 2009, 2010; Zanon and Bemporad, 2021). As such, one benefits from the systematic tuning of linear controllers in combination with the MPC ability to handle constraints.

There exist a variety of tuning methods for linear controllers, see, e.g., (Bansal et al., 2012) for proportional–integral–derivative (PID) controllers. Also, we have previously proposed an automatic tuning method for PID controllers based on high-performance Monte Carlo (MC) simulations of stochastic closed-loop systems (Wahlgreen et al., 2021). This tuning method offers a systematic ap-

proach, where one can select any tuning objective, e.g., target tracking, input usages, etc. The method is even applicable to advanced controllers like MPC. However, the tuning process requires a large number of closed-loop simulations, which can be computationally expensive for MPC. The computational cost of MC simulations for direct MPC tuning will be significantly higher compared to MC simulation for tuning linear controllers.

In this paper, we propose an MPC design method based on matching the MPC feedback to a PI controller tuned by MC simulations. As such, the method offers a systematic and efficient approach to design the MPC stage cost. We demonstrate the design method on a simulation case study, where we consider an exothermic chemical reaction conducted in an adiabatic continuous stirred tank reactor (CSTR) (Wahlgreen et al., 2020; Jørgensen et al., 2020). We apply a nonlinear stochastic differential equation (SDE) model for the CSTR dynamics, which results in an operating point close to an output constraint. We track the optimal operating point with a PI controller and an MPC controller, where the PI controller is tuned based on MC simulation and the MPC is matched to the tuned PI controller. The MPC formulation includes hard input constraints and a soft output constraint to avoid infeasible Optimal Control Problems (OCPs) in the MPC. Our

¹ Corresponding author: J. B. Jørgensen (E-mail: jbj@dtu.dk).

results show that MPC reduces output constraint violation compared to the PI controller while maintaining the tuned PI performance when output constraints are inactive.

The remaining parts of the paper are organized as follows. First, we introduce the simulation model for the CSTR. Next, we present the discrete PI controller with anti-windup mechanism and the hard input and soft output constrained MPC formulation for stabilization of the CSTR. Then, we present the controller matching problem, show how to represent the PI controller as a linear controller for the system, and provide a short description of the MC simulation tuning. Finally, we present the tuning and simulation results and end with our conclusions.

Simulation Model for the Adiabatic CSTR

We consider an exothermic reaction conducted in an adiabatic CSTR (Wahlgreen et al., 2020; Jørgensen et al., 2020).

General ODE model for the CSTR

A general ordinary differential equation (ODE) model for a non-constant volume CSTR is (Wahlgreen et al., 2022)

$$\frac{dV}{dt} = e^\top F_{\text{in}} - F_{\text{out}}, \quad (1a)$$

$$\frac{dn}{dt} = C_{\text{in}} F_{\text{in}} - c F_{\text{out}} + RV, \quad (1b)$$

where n is a vector of mole numbers, $c = n/V$ is a vector of concentrations, C_{in} is a matrix of inlet concentrations, R is the production rate, V is the volume of the CSTR, F_{in} is a vector of inlet stream flow rates, F_{out} is a scalar with the outlet stream flow rate, and e is a vector of ones of proper dimension. The production rate is given as,

$$R = S^\top r, \quad (2)$$

where S is the stoichiometric matrix and $r = r(c)$ is the reaction rate.

Exothermic reaction conducted in a constant volume CSTR

The medium in the constant volume CSTR consists of two components, A and B , and have temperature, T , which we treat as an additional chemical component. As such, the vector n consists of mole numbers (for A and B) and the total internal energy (for T). The vector c consists of the concentrations of A and B , and the temperature $T = c_T$. The CSTR has constant volume and one inlet stream, i.e., $e^\top F_{\text{in}} = F_{\text{in}} = F_{\text{out}} = F$. The stoichiometric matrix is,

$$S = \begin{bmatrix} -1.0 & -2.0 & \beta \end{bmatrix}, \quad (3)$$

where $\beta = -\Delta H_r / (\rho c_p)$, ΔH_r is the enthalpy of reaction, ρ is the density of the mixture, and c_p is the specific heat capacity. The rate of reaction is,

$$r(c) = k(c_T) c_A c_B, \quad (4)$$

with

$$k(c_T) = k_0 \exp\left(-\frac{E_a}{R} \frac{1}{c_T}\right), \quad (5)$$

and E_a/R denoting the activation energy. The inlet stream has the concentrations

$$C_{\text{in}} = \begin{bmatrix} c_{A,\text{in}} \\ c_{B,\text{in}} \\ c_{T,\text{in}} \end{bmatrix} = \begin{bmatrix} 1.6/2 \\ 2.4/2 \\ 273.65 \end{bmatrix}. \quad (6)$$

Together, Eq. (1)-(6) forms a three-state ODE model, where the states are n_A , n_B , and n_T . We refer to (Wahlgreen et al., 2020) for the system parameters.

One-state model

At steady-state, the three-state model is exactly represented by a one-state model, where $c_A(c_T)$ and $c_B(c_T)$ are functions of the temperature given as (Wahlgreen et al., 2020),

$$c_A(c_T) = c_{A,\text{in}} + \frac{1}{\beta}(c_{T,\text{in}} - c_T), \quad (7a)$$

$$c_B(c_T) = c_{B,\text{in}} + \frac{2}{\beta}(c_{T,\text{in}} - c_T). \quad (7b)$$

The resulting inlet matrix and stoichiometric matrix are,

$$C_{\text{in}} = [c_{T,\text{in}}], \quad S = [\beta], \quad (8)$$

and the state is n_T .

Stochastic differential equations

We extend the general ODE formulation, Eq. (1), to an SDE formulation with a stochastic diffusion term and disregard the volume equation since the volume is constant. The SDE is,

$$dn(t) = (C_{\text{in}} F - c F + RV) dt + F \bar{\sigma} d\omega(t), \quad (9)$$

where $d\omega(t) \sim N_{\text{idd}}(0, Idt)$ is a standard Wiener process and the diffusion function, $F \bar{\sigma}$ with $\bar{\sigma} = \text{diag}([0; 0; \sigma_T])$, models inlet temperature variations (Wahlgreen et al., 2020).

Stochastic continuous-discrete system

Let $x = n$ be the states, $u = F$ be the inputs, $y_k = y(t_k)$ be discrete measurements corrupted by noise, z be the output, p be the parameters, and $v_k = v(t_k) \sim N_{\text{idd}}(0, R_v)$ be measurement noise. Then, we formulate the system as a stochastic continuous-discrete system,

$$dx(t) = f(t, x(t), u(t), p) dt + \sigma(t, x(t), u(t), p) d\omega(t), \quad (10a)$$

$$y(t_k) = g(t_k, x(t_k)) + v(t_k), \quad (10b)$$

$$z(t) = h(t, x(t)), \quad (10c)$$

where

$$f(t, x(t), u(t), p) = C_{\text{in}} F - c F + RV, \quad (11a)$$

$$\sigma(t, x(t), u(t), p) = F \bar{\sigma}, \quad (11b)$$

$$g(t_k, x(t_k)) = n_T / V = c_T, \quad (11c)$$

$$h(t, x(t)) = n_T / V = c_T. \quad (11d)$$

We assume that discrete measurements are available with sampling time, T_s .

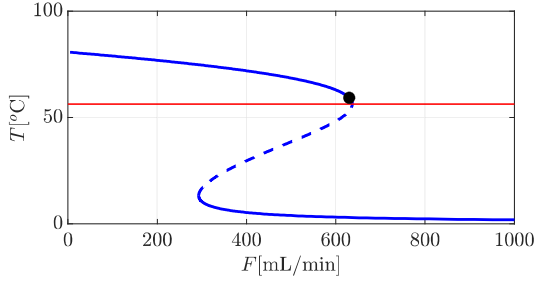


Figure 1: Steady state plot for the CSTR. Black dot: desired operation point. Red line: lower temperature constraint.

Operation of the CSTR

Figure 1 presents the steady-states of the CSTR within the flow rate limits, $F_{\min} = 0$ [mL/min] and $F_{\max} = 1000$ [mL/min] (Wahlgreen et al., 2020). The desired operating point has the temperature steady-state $\bar{T} = 59.30$ [°C] achieved at a flow rate of $\bar{F} = 630$ [mL/min]. The lower temperature constraint is $T_{\min} = 57.26$ [°C].

PI Controller

We stabilize the CSTR at the reference temperature, \bar{T} , with a PI controller. The discrete PI controller with anti-windup mechanism is given as,

$$e_k = \bar{y}_k - y_k, \quad (12a)$$

$$P_k = k_P e_k, \quad (12b)$$

$$I_k = \hat{I}_{k-1} + T_s k_I e_k, \quad (12c)$$

$$\hat{u}_k = \bar{u} + P + I, \quad (12d)$$

$$u_k = \max(u_{\min}, \min(u_{\max}, \hat{u}_k)), \quad (12e)$$

$$I_{aw,k} = T_s k_{aw} (u_k - \hat{u}_k), \quad (12f)$$

$$\hat{I}_k = I_k + I_{aw,k}. \quad (12g)$$

The anti-windup mechanism, Eq. (12f), ensures reasonable integrator behavior, when the PI response saturates the input, u_k , at u_{\min} or u_{\max} .

Model Predictive Controller

We also stabilize the CSTR with a linear MPC (LMPC). At time t_j , the LMPC solves the hard input and soft output constrained OCP,

$$\min_{u,x} \phi_j(u,x) + \phi_{\varepsilon,j}(u,x), \quad (13a)$$

$$s.t. \quad x_j = \hat{x}_{j|j} \quad (13b)$$

$$x_{j+k+1} = Ax_{j+k} + Bu_{j+k}, \quad (13c)$$

$$z_{j+k} = C_z x_{j+k}, \quad (13d)$$

$$u_{\min,j+k} \leq u_{j+k} \leq u_{\max,j+k}, \quad (13e)$$

$$z_{j+k} \geq z_{\min,j+k} - \varepsilon_{l,j+k}, \quad (13f)$$

$$z_{j+k} \leq z_{\max,j+k} + \varepsilon_{u,j+k}, \quad (13g)$$

where $\hat{x}_{j|j}$ is an estimate of the states x_j and

$$\phi_j(u,x) = \sum_{k=0}^{N-1} \begin{bmatrix} x_{j+k} \\ u_{j+k} \end{bmatrix}^\top \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} x_{j+k} \\ u_{j+k} \end{bmatrix} + x_{j+N}^\top P x_{j+N}, \quad (14a)$$

$$\phi_{\varepsilon,j}(u,x) = \sum_{k=0}^N \left(\begin{bmatrix} \varepsilon_{l,j+k} \\ \varepsilon_{u,j+k} \end{bmatrix}^\top \begin{bmatrix} Q_{\varepsilon_l} & \\ & Q_{\varepsilon_u} \end{bmatrix} \begin{bmatrix} \varepsilon_{l,j+k} \\ \varepsilon_{u,j+k} \end{bmatrix} + \begin{bmatrix} q_{\varepsilon_l} \\ q_{\varepsilon_u} \end{bmatrix}^\top \begin{bmatrix} \varepsilon_{l,j+k} \\ \varepsilon_{u,j+k} \end{bmatrix} \right). \quad (14b)$$

MPC Stage Cost Design

We design the MPC stage cost, Eq. (14a), by controller matching to a well-tuned stabilizing PI controller.

Controller matching problem

The controller matching problem is formulated as the semi-definite programming (SDP) (Zanon and Bemporad, 2021),

$$\min_{\Gamma, P, \beta} \beta, \quad (15a)$$

$$s.t. \quad \beta I \succeq H_\Gamma + H_P \succeq I, \quad (15b)$$

where

$$H_\Gamma = \begin{bmatrix} \hat{K}^\top \Gamma \hat{K}^\top & \hat{K}^\top \Gamma \\ \Gamma \hat{K} & \Gamma \end{bmatrix}, \quad (16a)$$

$$H_P = - \begin{bmatrix} A^\top P A - P & A^\top P B \\ B^\top P A & B^\top P B \end{bmatrix}, \quad (16b)$$

and \hat{K} is the stabilizing linear controller feedback matrix to be matched such that,

$$u = -\hat{K}x. \quad (17)$$

The OCP, Eq. (13), with stage cost matrices Q , R , and S ,

$$Q = \hat{K}^\top \Gamma \hat{K} + P - A^\top P A, \quad (18a)$$

$$R = \Gamma - B^\top P B, \quad (18b)$$

$$S = \Gamma \hat{K} - B^\top P A, \quad (18c)$$

produces the same response as the linear controller, Eq. (17), when inequality constraints are inactive.

Linearization of the one-state model

The matching problem, Eq. (15), requires a model in linear discrete state-space form,

$$x_{k+1}^l = A x_k^l + B u_k^l, \quad (19a)$$

$$y_k^l = C x_k^l. \quad (19b)$$

We apply linearization of the continuous model at the operation steady state, (x_s, u_s) , to obtain the continuous state-space matrices,

$$A^c = \frac{\partial f}{\partial x}(x_s, u_s), \quad B^c = \frac{\partial f}{\partial u}(x_s, u_s), \quad (20)$$

and exact discretization to obtain the discrete state-space matrices,

$$\begin{bmatrix} A & B \\ 0 & I \end{bmatrix} = \exp\left(\begin{bmatrix} A^c & B^c \\ 0 & 0 \end{bmatrix} T_s\right). \quad (21)$$

We base the MPC on the one-state model, as it is exact at steady-state. Linearization and discretization of the one-state CSTR model at the operating point, $x_s = \bar{y} = \bar{T} = 59.30$ [°C] and $u_s = \bar{F} = 630$ [mL/min], results in the following discrete state-space matrices,

$$A = [0.9572], \quad B = [-57.5381]. \quad (22)$$

Additionally, the measurement function, Eq. (11c), and output function, Eq. (11d), are linear and we get,

$$C = C_z = [1/V]. \quad (23)$$

The linear-discrete variables, x_k^l , u_k^l , and y_k^l , are deviation variables, i.e., $x_k^l = x(t_k) - x_s$, $u_k^l = u(t_k) - u_s$, and $y_k^l = y(t_k) - y_s$.

PI controller with anti-windup as linear control law

We write the PI controller, Eq. (12), in the linear form, Eq. (17). We include the integral state of the PI controller as a state in the discrete state-space model. As such, we define,

$$\tilde{x}_k = \begin{bmatrix} x_k^l \\ I_{k-1} \end{bmatrix}, \quad \tilde{u}_k = u_k^l. \quad (24)$$

The corresponding state-space matrices are,

$$\hat{A} = \begin{bmatrix} A & 0 \\ -T_s k_I C & 1 \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}. \quad (25)$$

The proportional and integral part of the PI controller is linearly expressed as,

$$\hat{K}_P = [k_P C \ 0], \quad \hat{K}_I = [T_s k_I C \ -1], \quad (26a)$$

where we point out that the reference is $\bar{y}^l = \bar{y} - y_s = 0$. The linear control law is a linear combination of the proportional and integral parts,

$$\hat{K} = \hat{K}_P + \hat{K}_I. \quad (27)$$

We express the anti-windup mechanism, Eq. (12f), in terms of additional discrete state-space matrices for the integral state,

$$A_{aw} = \begin{bmatrix} 0 \\ T_s k_{aw} \hat{K} \end{bmatrix}, \quad B_{aw} = \begin{bmatrix} 0 \\ T_s k_{aw} \end{bmatrix}, \quad (28)$$

such that the state-space matrices for \tilde{x}_k and \tilde{u}_k are,

$$\tilde{A} = \hat{A} + A_{aw}, \quad \tilde{B} = \hat{B} + B_{aw}. \quad (29a)$$

The discrete state-space model is,

$$\tilde{x}_{k+1} = \tilde{A}\tilde{x}_k + \tilde{B}\tilde{u}_k, \quad (30)$$

and the PI response, Eq. (12), is expressed as the linear control response,

$$\tilde{u}_k = -\tilde{K}\tilde{x}. \quad (31)$$

Monte Carlo based Tuning

We apply a MC simulation based method to tune the PI controller, Eq. (12), in the stochastic system, Eq. (10) (Wahlgreen et al., 2021).

Objectives for tuning

We consider two tuning objectives,

$$\Phi_1^{(n)} = \sum_{k=0}^N \|z_k^{(n)} - \bar{z}_k\|_{Q_z}^2, \quad (32a)$$

$$\Phi_2^{(n)} = \sum_{k=0}^N \|z_k^{(n)} - \bar{z}_k\|_{Q_z}^2 + \sum_{n=1}^N \|\Delta u_k^{(n)}\|_{Q_{\Delta u}}^2, \quad (32b)$$

where \bar{z}_k is the output target at sampling time t_k , $z_k^{(n)}$ is the output of the n 'th closed-loop simulation at sampling time t_k , $\Delta u_k^{(n)} = u_k^{(n)} - u_{k-1}^{(n)}$ is the change in input from sampling time t_{k-1} to t_k , Q_z is an output weight matrix, and $Q_{\Delta u}$ is an input rate of change weight matrix.

Results

This section presents results for PI tuning, MPC design by controller matching, and closed-loop performance for PI controllers and MPCs. We simulate the stochastic closed-loop system with the three-state CSTR model for $t \in [t_0, t_f]$, where $t_0 = 0$ [s], $t_f = 300$ [s], $R_v = 0.1$, and $\sigma_T = 5$. The sampling time of the system is $T_s = 1$ [s]. We solve the SDE, Eq. (10), between sampling times with an explicit Euler-Maruyama scheme with $N = 10$ intermediate steps.

We perform the closed-loop simulations on a 6 core Intel(R) Xeon(R) W-2235 CPU with frequency 3.80GHz.

Initial PI controller and MPC matching

We initially consider a non-tuned PI controller and demonstrate MPC matching. The PI controller has gains $k_P = -5.0 \cdot 10^{-4}$, $k_I = -5.0 \cdot 10^{-4}$, and $k_{aw} = 1.0 \cdot 10^{-1}$. We obtain the MPC stage cost matrices by solution of the SDP, Eq. (15). We solve the SDP through *cvx* with MOSEK in MATLAB (Grant and Boyd, 2008, 2014; MOSEK ApS, 2022). Figure 2 shows that the PI and MPC responses are identical as expected.

Tuning of PI controller

We base our tuning of each component of the PI gain on 1.000.000 closed-loop simulation. Thus, tuning of the three PI gains, k_P , k_I , and k_{aw} requires 3.000.000 simulations, which we perform in ≈ 90 [s]. We tune each gain by selecting 100 equidistant values of the gain in a selected range. The tuned gain value minimizes the average objective value over 10.000 closed-loop simulations with different process noise. We initialize the system at the operating point for the tuning of k_P and k_I , and we initialize the system far from the operating point to tune k_{aw} , in order to trigger input saturation. We use $Q_z = 1.0$ and $Q_{\Delta u} = 5 \cdot 10^3$.

Figure 3 presents tuning plots for the gains k_P , k_I , and k_{aw} with the Φ_1 tuning objective, Eq. (32a). The tuned PI con-

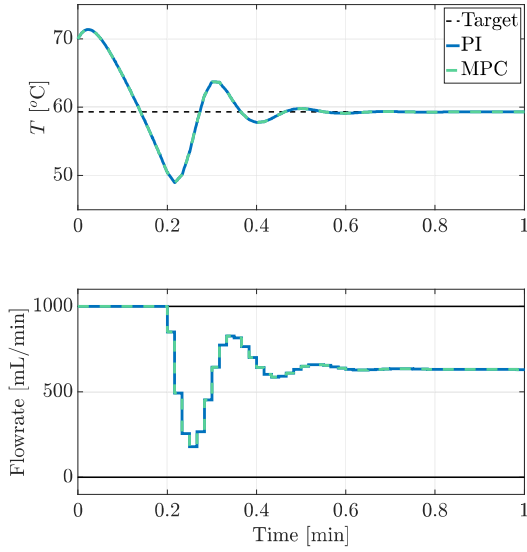


Figure 2: Non-tuned PI controller and matched MPC. The MPC response is identical to the PI response.

troller has gains,

$$k_P = -2.0455 \cdot 10^{-3}, \quad k_I = -2.0909 \cdot 10^{-4}, \quad (33a)$$

$$k_{aw} = 1.1111 \cdot 10^{-1}. \quad (33b)$$

Figure 4 presents probability density functions for a P controller, a non-tuned PI controller, and the tuned PI controller based on 30.000 closed-loop simulations each. We observe that, as expected, the tuned PI controller delivers the best performance both in terms of mean and variance.

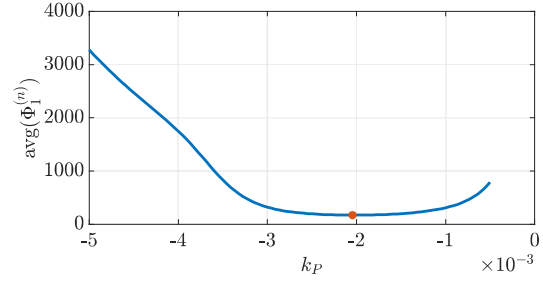
We apply the same procedure for the the Φ_2 tuning objective, Eq. (32b), and obtain the optimal PI gains,

$$k_P = -4.0000 \cdot 10^{-4}, \quad k_I = -4.9091 \cdot 10^{-5}, \quad (34a)$$

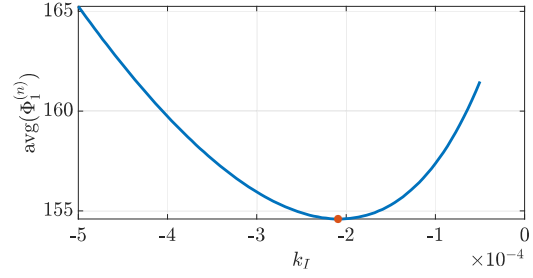
$$k_{aw} = 1.1636 \cdot 10^{-1}. \quad (34b)$$

Closed-loop simulation with PI controller and MPC

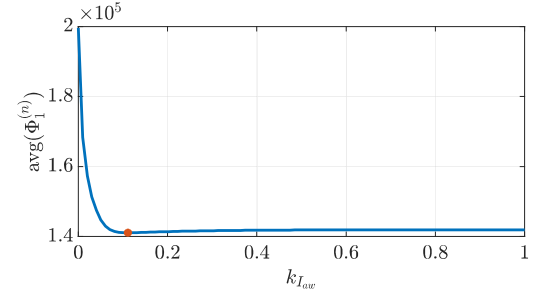
We consider the MPC formulation, Eq. (13), with stage costs matched to the tuned PI controllers, Eq. (33) and Eq. (34). The input constraints are $u_{\min} = 0$ [mL/min] and $u_{\max} = 1000$ [mL/min]. We impose a lower soft output constraint $z_{\min} = 59.0$ [°C], with $Q_{\varepsilon_l} = 10^6$, and $q_{\varepsilon_l} = 10^6$. Notice that the soft constraint is placed above the critical constraint, T_{\min} . This allows the MPC to take action before violation of the critical constraint, while ensuring feasible OCPs. Figure 5 presents simulation results for both the Φ_1 and Φ_2 tuning objective. The results are based on 100 simulations of the closed-loop system with different noise realizations. The Φ_1 objective leads to large input variance and small output variance compared to the Φ_2 objective. We consider the Φ_2 simulations, where we observe that the increased output variance causes likely constraint violation for the PI controller. The MPC reduces the constraint violation and has an average time out of range of 0.044% compared to the PI controller with 12.18%. The Φ_1 values are $1.7079 \cdot 10^3$ and $1.7462 \cdot 10^3$ for the PI controller and MPC respectively, and similarly the Φ_2 values are $2.7160 \cdot 10^3$ and $3.3178 \cdot 10^3$. Thus, the MPC



(a) Tuning of k_P . The optimum is $k_P = -2.0455 \cdot 10^{-3}$.



(b) Tuning of k_I . The optimum is $k_I = -2.0909 \cdot 10^{-4}$.



(c) Tuning of k_{aw} . The optimum is $k_{aw} = 1.1111 \cdot 10^{-1}$.

Figure 3: Φ_1 PI gain tuning. Each objective average is computed from 10.000 closed-loop simulations.

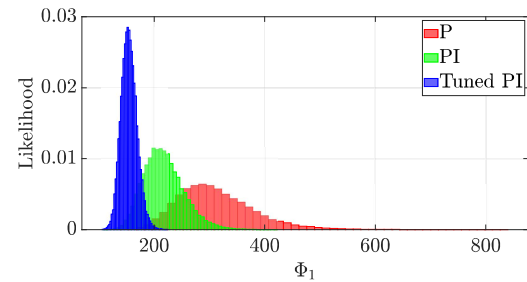
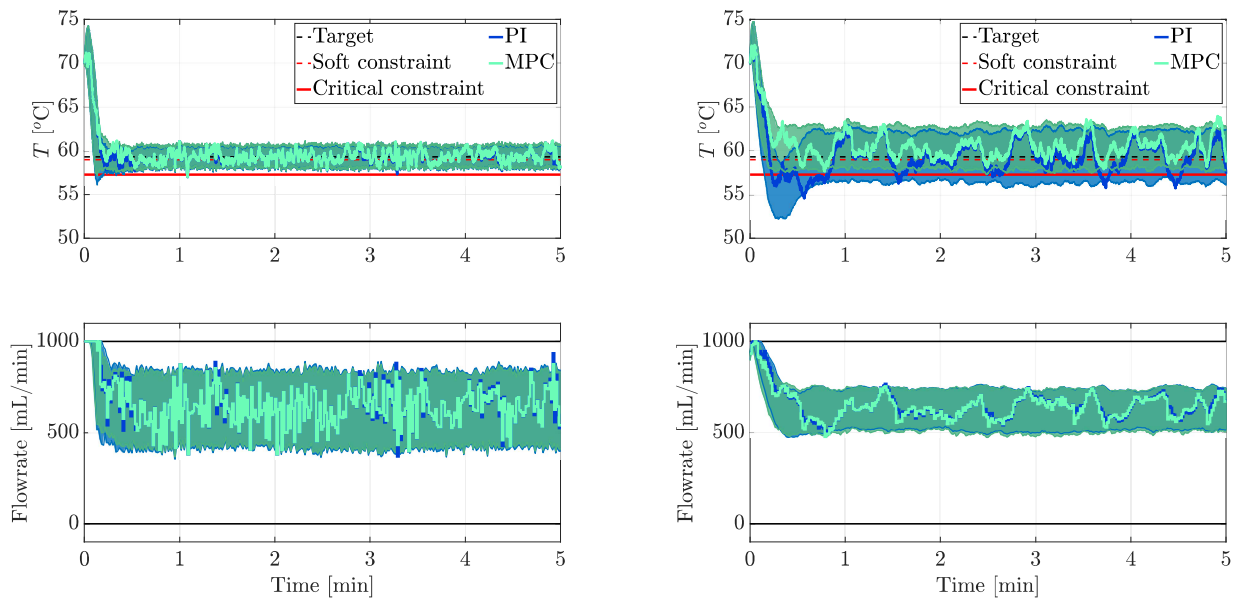


Figure 4: Probability density function for three controllers. P: $k_P = -1.0 \cdot 10^{-3}$. PI: $k_P = -1.0 \cdot 10^{-3}$ and $k_I = -1.0 \cdot 10^{-4}$. Tuned PI: gains given in Eq. (33).

successfully reduces the constraint violation while maintaining the tuned performance of the PI controller at the cost of slightly increased input changes.

Conclusion

The paper presents a systematic method to design MPC. The method matches the MPC stage cost to a high-performance MC simulation tuned PI controller. As such,



(a) Φ_1 tuned controllers with chattering input responses.

(b) Φ_2 tuned controllers with less chattering input responses.

Figure 5: Highlighted closed-loop trajectories for a single noise realization with PI controllers and matched MPCs and shaded 95% confidence intervals based on 100 simulations. MPC reduces constraint violation compared to the PI controllers.

the MPC design combines efficient and systematic tuning of a linear controller and advanced MPC properties such as constraint handling.

We apply the method to design MPC for an exothermic chemical reaction conducted in an adiabatic CSTR, where the operation point is close to a constraint. Our results show, that MPC is successfully matched to the MC simulation tuned PI controller. With the introduction of a soft output constraint, MPC is able to reduce constraint violation compared to the PI controller, while maintaining the tuned performance from the PI controller.

References

- Bansal, H. O., R. Sharma, and P. R. Shreeraman (2012). PID Controller Tuning Techniques: A Review. *Journal of Control Engineering and Technology*, 733–764.
- Di Cairano, S. and A. Bemporad (2009, August). Model Predictive Controller Matching: Can MPC Enjoy Small Signal Properties of My Favorite Linear Controller? *European Control Conference (ECC)*, 2217–2222.
- Di Cairano, S. and A. Bemporad (2010, January). Model Predictive Control Tuning by Controller Matching. *IEEE Transactions on Automatic Control* 55(1), 185–190.
- Grant, M. and S. Boyd (2008). Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura (Eds.), *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pp. 95–110. Springer-Verlag Limited.
- Grant, M. and S. Boyd (2014, March). CVX: Matlab software for disciplined convex programming, version 2.1.
- Jørgensen, J. B., T. K. S. Ritschel, D. Boiroux, E. Schroll-Fleischer, M. R. Wahlgreen, M. K. Nielsen, H. Wu, and J. K. Huusom (2020). Simulation of NMPC for a Laboratory Adiabatic CSTR with an Exothermic Reaction. *Proceedings of 2020 European Control Conference*, 202–207.
- MOSEK ApS (2022). *The MOSEK optimization toolbox for MATLAB manual. Version 9.3.18.*
- Qin, S. J. and T. A. Badgwell (2003). A survey of industrial model predictive control technology. *Control Engineering Practice* 11, 733–764.
- Wahlgreen, M. R., K. Meyer, T. K. S. Ritschel, A. P. Engsig-Karup, K. V. Gernaey, and J. B. Jørgensen (2022, June 14-17). Modeling and Simulation of Upstream and Downstream Processes for Monoclonal Antibody Production. *The 13th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS)*, Busan, Republic of Korea.
- Wahlgreen, M. R., A. T. Reenberg, M. K. Nielsen, A. Rydahl, T. K. S. Ritschel, B. Dammann, and J. B. Jørgensen (2021, December 13-17). A High-Performance Monte Carlo Simulation Toolbox for Uncertainty Quantification of Closed-loop Systems. *IEEE Conference on Decision and Control (CDC)*, Austin, Texas, USA.
- Wahlgreen, M. R., E. Schroll-Fleischer, D. Boiroux, T. K. S. Ritschel, H. Wu, J. K. Huusom, and J. B. Jørgensen (2020, February 16-19). Nonlinear Model Predictive Control for an Exothermic Reaction in an Adiabatic CSTR. *6th Conference on Advances in Control and Optimization of Dynamical Systems (ACODS)*, Chennai, India.
- Zanon, M. and A. Bemporad (2021). Constrained Control and Observer Design by Inverse Optimality. *IEEE Transactions on Automatic Control*, Accepted.

APPENDIX E

Paper V: ECC 2023

Performance Quantification of a Nonlinear Model Predictive Controller by Parallel Monte Carlo Simulations of a Closed-loop System

Authors:

Morten Wahlgreen Kaysfeld, Mario Zanon, John Bagterp Jørgensen

To be published in:

Proceedings of the European Control Conference (ECC), Bucharest, Romania, June
13-16, 2023.

arXiv: <https://doi.org/10.48550/arXiv.2212.02197>

Performance Quantification of a Nonlinear Model Predictive Controller by Parallel Monte Carlo Simulations of a Closed-loop System

Morten Wahlgreen Kaysfeld, Mario Zanon, John Bagterp Jørgensen

Abstract—This paper presents a parallel Monte Carlo simulation based performance quantification method for nonlinear model predictive control (NMPC) in closed-loop. The method provides distributions for the controller performance in stochastic systems enabling performance quantification. We perform high-performance Monte Carlo simulations in C enabled by a new thread-safe NMPC implementation in combination with an existing high-performance Monte Carlo simulation toolbox in C. We express the NMPC regulator as an optimal control problem (OCP), which we solve with the new thread-safe sequential quadratic programming software NLPSQP. Our results show almost linear scale-up for the NMPC closed-loop on a 32 core CPU. In particular, we get approximately 27 times speed-up on 32 cores. We demonstrate the performance quantification method on a simple continuous stirred tank reactor (CSTR), where we perform 30,000 closed-loop simulations with both an NMPC and a reference proportional-integral (PI) controller. Performance quantification of the stochastic closed-loop system shows that the NMPC outperforms the PI controller in both mean and variance.

I. INTRODUCTION

In closed-loop systems, there exist many unknown or uncertain quantities, such as parameters, measurement noise, and process noise, even when simple linear controllers like proportional–integral–derivative (PID) controllers are applied. As such, achieving useful closed-loop performance quantification can be difficult. Previous work has focused on development of a high-performance Monte Carlo simulation toolbox for parallel computing on shared memory architectures. The Monte Carlo simulation toolbox has previously enabled tuning of PID controllers in closed-loop systems [1], tuning of a model predictive controller (MPC) through controller matching [2], and been applied for PID closed-loop insulin dosing in a virtual clinical trial with 1,000,000 participants for people with type 1 diabetes [3]. Similar results have not yet been obtained for more advanced controllers such as nonlinear MPC (NMPC).

Monte Carlo approaches have previously been applied in relation to NMPC. Sequential Monte Carlo (SMC) has been applied as a method to find global optimizers in NMPC [4]. Additionally, an SMC filter has been applied as an alternative to Kalman filtering and moving horizon estimation (MHE) for state estimation [5]. However, application of Monte Carlo simulation to quantify the closed-loop performance of NMPC is novel, likely due to the difficulty of running

sufficiently many closed-loop simulations with NMPC. We propose to apply the existing Monte Carlo simulation toolbox as a performance quantification technique for NMPC closed-loop systems. Computational feasibility is achieved by full utilization of multi-core CPUs [6]. To this end, the approach requires a thread-safe NMPC implementation.

In this paper, we apply parallel Monte Carlo simulation as a method for performance quantification of NMPC in closed-loop. The method provides performance distributions of the stochastic closed-loop and enables performance quantification. We achieve parallel scaling by implementation of a new thread-safe NMPC featuring a continuous-discrete extended Kalman filter (CD-EKF) for state estimation and a regulator expressed as an optimal control problem (OCP). We solve the OCP with a new thread-safe sequential quadratic programming (SQP) software, NLPSQP (nonlinear-programming-sequential-quadratic-programming), required to achieve parallel scaling. Due to thread-safety of the NMPC, we achieve almost linear parallel scaling and approximately 27 times speed-up on 32 cores. The efficient parallel framework enables computationally feasible Monte Carlo simulations of closed-loop systems with NMPC, which enables novel performance quantification of NMPC. We consider a well-known continuous stirred tank reactor (CSTR) example as a case study [7], [8]. We demonstrate the performance quantification method by comparing NMPC performance to a reference proportional-integral (PI) controller. The resulting performance distributions show that the NMPC outperforms the PI controller in both mean and variance.

The remaining parts of the paper are organized as follows. Section II introduces the continuous-discrete system. Section III introduces our NMPC formulation including the estimator and regulator. Section IV presents the SQP software NLPSQP. Section V presents the PI controller. Section VI introduces our case study model. Section VII presents our results. Section VIII presents our conclusions.

II. CONTINUOUS-DISCRETE SYSTEM

In our closed-loop simulations, we consider stochastic continuous-discrete systems in the form [1],

$$dx(t) = f(t, x(t), u(t), d(t), p)dt + \sigma(t, x(t), u(t), d(t), p)d\omega(t), \quad (1a)$$

$$y(t_i) = g(t_i, x(t_i), p) + v(t_i, p), \quad (1b)$$

$$z(t) = h(t, x(t), p), \quad (1c)$$

where $x(t)$ are states, $u(t)$ are inputs, $d(t)$ are disturbances, p are parameters, $y(t_i)$ are measurements at discrete time, $z(t)$

*M. W. Kaysfeld and J. B. Jørgensen are with the Department of Applied Mathematics and Computer Science, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark. Mario Zanon is with IMT School for Advanced Studies Lucca, IT-55100 Lucca, Italy.

Corresponding author: J. B. Jørgensen (E-mail: jbj@dtu.dk).

are outputs, $\omega(t)$ is a standard Wiener process, and $v(t_i, p)$ is normally distributed measurement noise at discrete time, i.e.,

$$d\omega(t) \sim N_{iid}(0, Idt), \quad (2a)$$

$$v(t_i, p) \sim N_{iid}(0, R(t_i, p)), \quad (2b)$$

where R is the measurement covariance. Measurements, $y(t_i)$, are assumed available with sampling time, T_s . We apply models in the stochastic continuous-discrete form, (1), for both simulation of the system and in the NMPC.

III. NONLINEAR MODEL PREDICTIVE CONTROLLER

We design an NMPC scheme to regulate continuous-discrete systems in the form (1). Our NMPC includes a CD-EKF for state estimation [1], [9] and a regulator expressed as an OCP.

A. State estimator

The CD-EKF receives a measurement, y_i , at time t_i . It computes the state-covariance one-step prediction, $\hat{x}_{i|i-1}$ and $P_{i|i-1}$, from the previous state-covariance estimate, $\hat{x}_{i-1|i-1}$ and $P_{i-1|i-1}$, and applies the measurement and the one-step prediction to compute the new filtered state-covariance estimate, $\hat{x}_{i|i}$ and $P_{i|i}$.

1) *Prediction*: The state and covariance one-step prediction is,

$$\hat{x}_{i|i-1} = \hat{x}_{i-1}(t_i), \quad P_{i|i-1} = P_{i-1}(t_i), \quad (3)$$

obtained as the solution to,

$$\frac{d}{dt} \hat{x}_{i-1}(t) = f(t, \hat{x}_{i-1}(t), u_{i-1}, d_{i-1}, p), \quad (4a)$$

$$\begin{aligned} \frac{d}{dt} P_{i-1}(t) &= A_{i-1}(t)P_{i-1}(t) + P_{i-1}(t)A_{i-1}(t)^\top \\ &+ \sigma_{i-1}(t)\sigma_{i-1}(t)^\top, \end{aligned} \quad (4b)$$

for $t_{i-1} \leq t \leq t_i$, where

$$A_{i-1}(t) = \frac{\partial}{\partial x} f(t, \hat{x}_{i-1}(t), u_{i-1}, d_{i-1}, p), \quad (5a)$$

$$\sigma_{i-1}(t) = \sigma(t, \hat{x}_{i-1}(t), u_{i-1}, d_{i-1}, p). \quad (5b)$$

The initial condition of (4) is the previous filtered state-covariance pair,

$$\hat{x}_{i-1}(t_{i-1}) = \hat{x}_{i-1|i-1}, \quad P_{i-1}(t_{i-1}) = P_{i-1|i-1}. \quad (6a)$$

2) *Filtering*: Given the measurement, y_i , and the state-covariance one-step prediction, $\hat{x}_{i|i-1}$ and $P_{i|i-1}$, the CD-EKF computes the filtered state estimate, $\hat{x}_{i|i}$, as

$$\hat{y}_{i|i-1} = g(\hat{x}_{i|i-1}, p), \quad C_i = \frac{\partial}{\partial x} g(\hat{x}_{i|i-1}, p), \quad (7a)$$

$$e_i = y_i - \hat{y}_{i|i-1}, \quad R_{e,i} = R_i + C_i P_{i|i-1} C_i^\top, \quad (7b)$$

$$\hat{x}_{i|i} = \hat{x}_{i|i-1} + K_i e_i, \quad K_i = P_{i|i-1} C_i^\top R_{e,i}^{-1}, \quad (7c)$$

where $R_i = R(t_i, p)$ is the measurement covariance. The filtered covariance estimate, $P_{i|i}$, is

$$\begin{aligned} P_{i|i} &= P_{i|i-1} - K_i R_{e,i} K_i^\top \\ &= (I - K_i C_i) P_{i|i-1} (I - K_i C_i)^\top + K_i R_i K_i^\top, \end{aligned} \quad (8)$$

where (8) is the Joseph stabilizing form [10].

B. Regulator

We express the NMPC regulator in terms of an OCP, which the NMPC solves at time t_i once the filtered state estimate, $\hat{x}_{i|i}$, is provided by the estimator. The solution to the OCP is the input and state trajectories in the finite horizon. However, the regulator only implements the first input, u_i , and resolves the OCP once the next state estimate is available from the estimator. Let T be the prediction and control horizon, which is split into N control intervals of size T_s . As such, $T = NT_s$. We assume zero-order hold parameterization of inputs, u , and disturbances, d , in each control interval,

$$u(t) = u_{i+k}, \quad t_{i+k} \leq t < t_{i+k+1}, \quad (9a)$$

$$d(t) = d_{i+k}, \quad t_{i+k} \leq t < t_{i+k+1}, \quad (9b)$$

where $t_{i+k} = t_i + kT_s$. Let $\mathcal{N} = \{0, 1, \dots, N-1\}$, then the regulator OCP is,

$$\min_{x, u} \varphi_i, \quad (10a)$$

$$\text{s.t. } x(t_i) = \hat{x}_{i|i}, \quad (10b)$$

$$\dot{x}(t) = f(t, x, u, d, p), \quad t_i \leq t \leq t_i + T, \quad (10c)$$

$$u(t) = u_{i+k}, \quad k \in \mathcal{N}, \quad t_{i+k} \leq t \leq t_{i+k+1}, \quad (10d)$$

$$d(t) = d_{i+k}, \quad k \in \mathcal{N}, \quad t_{i+k} \leq t \leq t_{i+k+1}, \quad (10e)$$

$$u_{\min} \leq u_{i+k} \leq u_{\max}, \quad k \in \mathcal{N}, \quad (10f)$$

where $f(t, x, u, d, p) = f(t, x(t), u(t), d(t), p)$ and $\varphi_i = \varphi_i(x(t), u(t))$. We apply a direct multiple-shooting discretization to solve the OCP, (10), which yields a nonlinear programming (NLP) in the form,

$$\min_{\xi} \varphi, \quad (11a)$$

$$\text{s.t. } x_{k+1} - F(t_k, x_k, u_k, d_k, p) = 0, \quad (11b)$$

$$u_{\min} \leq u_k \leq u_{\max}, \quad (11c)$$

where $k \in \mathcal{N}$ is relative in time to t_i , $x_0 = \hat{x}_{0|0}$ is a parameter, $F(\cdot)$ is a numerical state integration scheme, and the decision variables are,

$$\xi = [u_0 \quad x_1 \quad \cdots \quad u_{N-1} \quad x_N]^\top. \quad (12)$$

Note that x_0 is not required as a decision variable, but can be included without loss of generality. We assume that the NLP objective, $\varphi = \varphi(\xi)$, is partially separable locally in time with respect to the decision variables. Additionally, we denote the number of equality constraints, m_e , the number of lower bounds, m_l , and the number of upper bounds, m_u .

IV. SEQUENTIAL QUADRATIC PROGRAMMING

We solve the NLP, (11), with our SQP software, NLPSQP. The NLPSQP implementation is dedicated to solve multiple similar NLPs in parallel applications. The iterative SQP algorithm performs three steps in each iteration, 1) Obtain a search direction by solution of a Quadratic Programming (QP) subproblem, 2) Obtain a step-size by a backtracking line-search algorithm, and 3) Lagrangian Hessian approximation with a block Broyden–Fletcher–Goldfarb–Shanno (BFGS) update.

In the following, we let $g(\xi)$ denote the vectorized constraint evaluation of (11b) together with λ , π_l , and π_u denoting Lagrange multipliers for equality constraint, lower input bound constraints, and upper input bound constraints respectively. Additionally, we let $f(\xi) = \varphi(\xi)$ for simplicity and apply $[l]$ as superscript to denote the l 'th iteration of the algorithm.

A. Quadratic Programming subproblem

In iteration l , the QP-subproblem solved in NLPSQP is

$$\min_{\Delta\xi} \bar{l}_0(\Delta u_0) + \sum_{k=1}^{N-1} \bar{l}_k(\Delta x_k, \Delta u_k) + \bar{l}_N(\Delta x_N), \quad (13a)$$

$$\text{s.t. } \Delta x_{k+1} = A_k^\top \Delta x_k + B_k^\top \Delta u_k + b_k, \quad (13b)$$

$$u_{\min} - u_k \leq \Delta u_k \leq u_{\max} - u_k, \quad (13c)$$

where $k \in \mathcal{N}$, $\Delta x_0 = 0$ is a parameter, and

$$\bar{l}_0(\Delta u_0) = \frac{1}{2} \Delta u_0^\top R_0 \Delta u_0 + r_0^\top \Delta u_0 + \rho_0, \quad (14a)$$

$$\begin{aligned} \bar{l}_k(\Delta x_k, \Delta u_k) &= \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix}^\top \begin{bmatrix} Q_k & M_k \\ M_k^\top & R_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \\ &+ \begin{bmatrix} q_k \\ r_k \end{bmatrix}^\top \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \rho_k, \end{aligned} \quad (14b)$$

$$\bar{l}_N(\Delta x_N) = \frac{1}{2} \Delta x_N^\top P_N \Delta x_N + p_N^\top \Delta x_N + \rho_N. \quad (14c)$$

Due to partial separability of the Lagrangian function,

$$\begin{aligned} \mathcal{L}(\xi, \lambda, \pi_l, \pi_u) &= \mathcal{L}_0(u_0, \lambda, \pi_l, \pi_u) \\ &+ \sum_{k=1}^{N-1} \mathcal{L}_k(x_k, u_k, \lambda, \pi_l, \pi_u) + \mathcal{L}_N(x_N, \lambda), \end{aligned} \quad (15)$$

the Lagrangian Hessian is block diagonal with blocks, W_k , defined as,

$$W_0 = R_0, \quad W_k = \begin{bmatrix} Q_k & M_k \\ M_k^\top & R_k \end{bmatrix}, \quad W_N = P_N, \quad (16)$$

for $k = 1, \dots, N-1$. The matrices, Q_k , R_k , M_k , and P_N , are second order derivatives of the Lagrangian function. However, NLPSQP applies a BFGS type approximation for the blocks, W_k [11]. The remaining matrices and vectors in the QP-subproblem, (13), are given as,

$$r_k = \nabla_{u_k} \mathcal{L}_k, \quad k = 0, \dots, N-1, \quad (17a)$$

$$q_k = \nabla_{x_k} \mathcal{L}_k, \quad k = 1, \dots, N-1, \quad (17b)$$

$$p_N = \nabla_{x_N} \mathcal{L}_N, \quad (17c)$$

$$A_k = \nabla_{x_k} F_k, \quad k = 1, \dots, N-1, \quad (17d)$$

$$B_k = \nabla_{u_k} F_k, \quad k = 0, \dots, N-1, \quad (17e)$$

$$b_k = F_k - x_{k+1}, \quad k = 0, \dots, N-1, \quad (17f)$$

where $F_k = F(t_k, x_k, u_k, d_k, p)$. The solution to the QP-subproblem, (14), is the data $(\Delta\xi, \mu, \nu_l, \nu_u)^{[l]}$, where

$$\mu^{[l]} = \lambda^{[l]} + \Delta\lambda^{[l]}, \quad (18a)$$

$$\nu_l^{[l]} = \pi_l^{[l]} + \Delta\pi_l^{[l]}, \quad (18b)$$

$$\nu_u^{[l]} = \pi_u^{[l]} + \Delta\pi_u^{[l]}. \quad (18c)$$

Notice that the search direction, $(\Delta\xi, \Delta\lambda, \Delta\pi_l, \Delta\pi_u)^{[l]}$, ensures satisfaction of the linear bound constraints, (11c), if the initial guess is feasible. NLPSQP solves the structured QP-subproblem, (13), with a Riccati recursion based primal-dual interior point algorithm [12]–[14].

B. Line-search

Given the search direction, $(\Delta\xi, \Delta\lambda, \Delta\pi_l, \Delta\pi_u)^{[l]}$, NLPSQP performs the step,

$$\begin{aligned} (\xi, \lambda, \pi_l, \pi_u)^{[l+1]} &= (\xi, \lambda, \pi_l, \pi_u)^{[l]} \\ &+ \alpha(\Delta\xi, \Delta\lambda, \Delta\pi_l, \Delta\pi_u)^{[l]}, \end{aligned} \quad (19)$$

where α is a step-size. NLPSQP applies a backtracking line-search algorithm to select a step-size, α , ensuring sufficient decrease in Powell's l_1 -merit function [13], [15],

$$P(\xi, \sigma) = f(\xi) + \sigma^\top |g(\xi)|, \quad (20)$$

where

$$\sigma_i = \max\left(|\mu_i|, \frac{1}{2}(\sigma_i + |\mu_i|)\right), \quad i = 1, \dots, m, \quad (21)$$

with $\sigma_i = |\mu_i|$ in the first iteration. We define

$$T(\alpha) = P(\xi^{[l+1]}, \sigma) = P(\xi^{[l]} + \alpha\Delta\xi^{[l]}, \sigma), \quad (22)$$

and let sufficient decrease be defined from the Armijo condition,

$$T(\alpha) \leq T(0) + c_1 \alpha D_{\Delta\xi} T(0), \quad (23)$$

where

$$T(\alpha) = f(\xi^{[l]} + \alpha\Delta\xi^{[l]}) + \sigma^\top |g(\xi^{[l]} + \alpha\Delta\xi^{[l]})|, \quad (24)$$

$$T(0) = f(\xi^{[l]}) + \sigma^\top |g(\xi^{[l]})|, \quad (25)$$

$$D_{\Delta\xi} T(0) = \nabla f(\xi^{[l]})^\top \Delta\xi^{[l]} - \sigma^\top |g(\xi^{[l]})|. \quad (26)$$

The backtracking line-search algorithm is,

- 1) Set $\alpha = 1$
- 2) Evaluate (23). If satisfied, **break** with α as output
- 3) Compute $\alpha = \beta\alpha$
- 4) Go to 2)

where $0 < \beta < 1$. We use $c_1 = 10^{-4}$ and $\beta = 0.5$ (similarly to IPOPT [16]).

C. Block BFGS update

NLPSQP estimates the block matrices, W_k , with a block damped BFGS update [11]. Define

$$s = \xi^{[l+1]} - \xi^{[l]}, \quad (27a)$$

$$y = \nabla_{\xi} \mathcal{L}^{[l+1]} - \nabla_{\xi} \mathcal{L}^{[l]}, \quad (27b)$$

where $\nabla_{\xi} \mathcal{L}^{[l]} = \nabla_{\xi} \mathcal{L}(\xi^{[l]}, \lambda^{[l+1]}, \pi_l^{[l+1]}, \pi_u^{[l+1]})$, $\nabla_{\xi} \mathcal{L}^{[l+1]} = \nabla_{\xi} \mathcal{L}(\xi^{[l+1]}, \lambda^{[l+1]}, \pi_l^{[l+1]}, \pi_u^{[l+1]})$, and let s_k and y_k be the elements of s and y corresponding to W_k , respectively. Let

$$r_k = \theta_k y_k + (1 - \theta_k) W_k s_k, \quad (28)$$

where

$$\theta_k = \begin{cases} 1 & s_k^\top y_k \geq 0.2 s_k^\top W_k s_k \\ \frac{0.8 s_k^\top W_k s_k}{s_k^\top W_k s_k - s_k^\top y_k} & \text{else} \end{cases} \quad (29)$$

Then the block BFGS update is given by

$$W_{k+1} = \begin{cases} W_k - \frac{(W_k s_k)(W_k s_k)^\top}{s_k^\top (W_k s_k)} + \frac{r_k r_k^\top}{s_k^\top r_k} & \kappa > \epsilon_m \\ W_k & \text{else} \end{cases} \quad (30)$$

where ϵ_m is the machine precision, $\kappa = \min(\kappa_1, \kappa_2)$ with $\kappa_1 = s_k^\top W_k s_k$ and $\kappa_2 = s_k^\top r_k$. The update safeguard is required as some blocks might converge faster than others resulting in zero-division. NLPSQP initializes the Hessian update as identity, $W^{[0]} = I$. Numerical rounding errors might cause indefinite BFGS block updates. In this case, NLPSQP applies the simple strategy to reset the entire Hessian to identity.

D. Convergence

NLPSQP converges when the KKT conditions are satisfied to the user-specified tolerance ϵ . In practice, we apply a scaled convergence condition,

$$\|\nabla \mathcal{L}^{[l]}/s_d\|_\infty \leq \epsilon, \quad \|g^{[l]}\|_\infty \leq \epsilon, \quad (31)$$

where $\nabla \mathcal{L}^{[l]} = \nabla \mathcal{L}(\xi, \lambda, \pi_l, \pi_u)^{[l]}$, $g^{[l]} = g(\xi)^{[l]}$, and

$$s_d = \max\left(s_{\max}, \frac{\|\lambda\|_1 + \|\pi_l\|_1 + \|\pi_u\|_1}{m_e + m_l + m_u}\right) / s_{\max}, \quad (32)$$

with $s_{\max} = 100$ (similar to IPOPT [16]).

E. Implementation

NLPSQP is implemented thread-safe in C for parallel applications, specifically intended for closed-loop Monte Carlo simulation. NLPSQP is BLAS dependent. In particular, we apply OpenBLAS [17], [18]. To ensure thread-safety of OpenBLAS, we compile a single threaded version by setting `USE_THREAD=0` and `USE_LOCKING=1`. However, we have not been able to achieve parallel scaling for the functions `dpotrf`, `dpotrs`, and `dgemm`. Therefore, we have implemented our own versions of these functions only intended for small matrices. In future work, we will consider other thread-safe BLAS libraries.

V. PROPORTIONAL-INTEGRAL CONTROLLER

We consider a PI controller with input bounds, u_{\min} and u_{\max} , and anti-windup mechanism to ensure proper integrator behavior when the PI response is saturated,

$$e_k = \bar{y}_k - y_k, \quad (33a)$$

$$P_k = k_P e_k, \quad (33b)$$

$$I_k = \hat{I}_{k-1} + T_s k_I e_k, \quad (33c)$$

$$\hat{u}_k = \bar{u} + P_k + I_k, \quad (33d)$$

$$u_k = \max(u_{\min}, \min(u_{\max}, \hat{u}_k)), \quad (33e)$$

$$I_{aw,k} = T_s k_{aw} (\hat{u}_k - u_k), \quad (33f)$$

$$\hat{I}_k = I_k + I_{aw,k}. \quad (33g)$$

We apply the PI controller (33) as a reference for the NMPC.

VI. MODEL

As a simulation case study, we consider an exothermic chemical reaction conducted in an adiabatic CSTR [7], [8]. The example is simple yet effective due to the non-trivial dynamics causing a branch of unstable steady-states in the operating window [7]. In addition, the model is well-approximated by a one-state model, well-suited for NMPC. The stochastic model for the constant volume CSTR is compactly written as the SDE [2],

$$dn(t) = (C_{\text{in}}F - cF + RV) dt + F\bar{\sigma}d\omega(t), \quad (34)$$

where

$$c = \frac{n}{V}, \quad R = S^\top r(c), \quad (35)$$

and

$$r(c) = k(c_T)c_A c_B, \quad k(c_T) = k_0 \exp\left(-\frac{E_a}{R} \frac{1}{c_T}\right), \quad (36)$$

with E_a/R denoting the activation energy. In the three-state model, the stoichiometric matrix and inlet stream concentration matrix is,

$$C_{\text{in}} = \begin{bmatrix} c_{A,\text{in}} \\ c_{B,\text{in}} \\ c_{T,\text{in}} \end{bmatrix}, \quad S = \begin{bmatrix} -1.0 & -2.0 & \beta \end{bmatrix}, \quad (37)$$

and in the one-state model they are,

$$C_{\text{in}} = [c_{T,\text{in}}], \quad S = [\beta]. \quad (38)$$

The stochastic diffusion term of (34) models inlet concentration variations and we apply

$$\bar{\sigma} = \begin{bmatrix} \sigma_A & & \\ & \sigma_B & \\ & & \sigma_T \end{bmatrix}, \quad \bar{\sigma} = [\sigma_T], \quad (39)$$

in the three-state and one-state models, respectively. We refer to [7] for more details and the parameters of the model. The output of the model is the temperature c_T , i.e., $z(t) = c_T(t)$. Additionally, we assume the temperature to be measured at discrete times, i.e., $y(t_i) = c_T(t_i)$.

VII. RESULTS

This section presents our simulation results. The simulations are conducted on a dual-socket Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz system. See TABLE I for CPU details.

A. Closed-loop simulation

We simulate the CSTR in closed-loop with the NMPC. We apply the three-state stochastic model for simulation of the system and apply the one-state model in the NMPC. We select a variable set-point, \bar{z} , together with, $t_0 = 0.0$ s, $t_f = 600.0$ s, $T_s = 1.0$ s, and $N_s = 20$, where N_s is the number of Euler-Maruyama steps to integrate the state equations from t_i to t_{i+1} . We initialize the system at

$$x_0 = n_0 = \begin{bmatrix} c_{A,\text{in}} \\ c_{B,\text{in}} \\ c_{T,\text{in}} \end{bmatrix} V. \quad (40)$$

TABLE I
CPU INFORMATION.

Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit
CPU(s):	32
Thread(s) per core:	1
Core(s) per socket:	16
Socket(s):	2
NUMA node(s):	2
Model name:	Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz
CPU MHz:	2900.000
L1d cache:	32 kB
L1i cache:	32 kB
L2 cache:	1024 kB
L3 cache:	22528 kB
RAM:	384 GB

TABLE II
STATISTICS FOR MONTE CARLO SIMULATION OF NMPC CLOSED-LOOP.

MC simulation time	≈ 55	[min]
Number of MC simulations	30,000	[-]
Total number of OCPs	18,000,000	[-]
Successful OCPs	17,999,576	[-]
Failed OCPs	424	[-]
Percentage success	99.9976	[%]
Percentage fails	0.0024	[%]

The NMPC has the discrete prediction and control horizon, $N = 60$, applies $N_c = 5$ classical Runge-Kutta steps to integrate the state dynamics in each control interval in the OCP, and applies a point-wise weighted least-squares output objective in the OCP,

$$\varphi_i = \sum_{k=1}^N \|z(t_{i+k}) - \bar{z}(t_{i+k})\|_{Q_z}^2 T_s, \quad (41)$$

where $Q_z = 1.0$. We initialize the CD-EKF states as,

$$\hat{x}_{-1|-1} = c_{T,\text{in}} V, \quad P_{-1|-1} = 10^{-6}, \quad (42)$$

and apply the input bounds $u_{\min} = 0.0$ mL/min and $u_{\max} = 1000.0$ mL/min. The PI controller with anti-windup mechanism, (33), has the hand-tuned gains,

$$k_P = -10^{-3}, \quad k_I = -10^{-4}, \quad k_{aw} = -10^{-1}. \quad (43a)$$

We perform a single closed-loop simulation with the PI controller and the NMPC. Fig. 1 presents the result. We observe that both the PI controller and the NMPC are able to track set-points at both stable and unstable steady-states. However, the NMPC has better tracking performance at set-point changes due to its anticipatory action.

B. Parallel scalability

We apply the Monte Carlo simulation toolbox to perform 100 simulations with different process noise in the system. We compute the 100 simulations with the NMPC on different numbers of cores to get scale-up data. Fig. 2 presents a scale-up plot for the simulations. We observe almost linear scale-up and approximately 27 times speed-up on 32 cores. In previous work, we showed similar scale-up results for a PID controller [1].

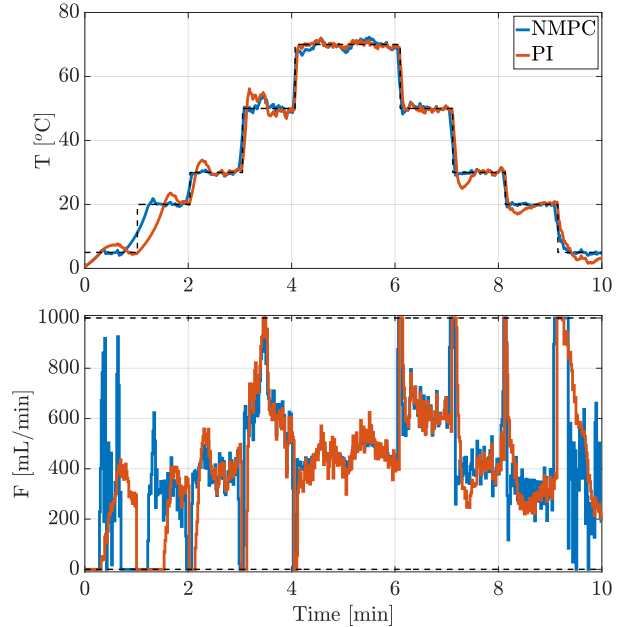


Fig. 1. Stochastic closed-loop simulation with PI controller and NMPC. Both controllers are able to track the set-points at both stable and unstable steady-states. The NMPC has better tracking performance at set-point changes due to its anticipatory action.

C. Monte Carlo Simulations

We perform Monte Carlo simulations to quantify the closed-loop performance of the NMPC in presences of process noise. In particular, we perform 30,000 simulations with varying process noise for both the PI controller and the NMPC. We apply a scaled point-wise squared-2-norm metric to evaluate the closed-loop performance,

$$\Phi = \frac{1}{\bar{N} + 1} \sum_{i=0}^{\bar{N}} \|z(t_i) - \bar{z}(t_i)\|_2^2, \quad (44)$$

where \bar{N} is the number of samplings over the full simulation, i.e., $\bar{N} = \frac{t_f - t_0}{T_s} = 600$ in our simulations. Fig. 3 shows histograms of the distribution of Φ , (44), over the 30,000 Monte Carlo simulations. The results show that the NMPC outperforms the PI controller in both mean and variance with respect to the Φ -metric. However, due to Fig. 1, we expect the better NMPC performance to be mainly due to set-point changes and anticipatory action from the NMPC.

TABLE II shows simulation statistics for the NMPC closed-loop Monte Carlo simulations. We observe that NLPSQP successfully solves 99.9976% of the 18,000,000 OCPs in the 30,000 closed-loop simulations. In the remaining 0.0024%, NLPSQP reaches the maximum number of iterations set to 100. We suspect that NLPSQP locates an almost optimal point, but has trouble detecting it. Therefore, we implement the detected solution in the NMPC. In future work, we will further investigate the convergence detection of NLPSQP.

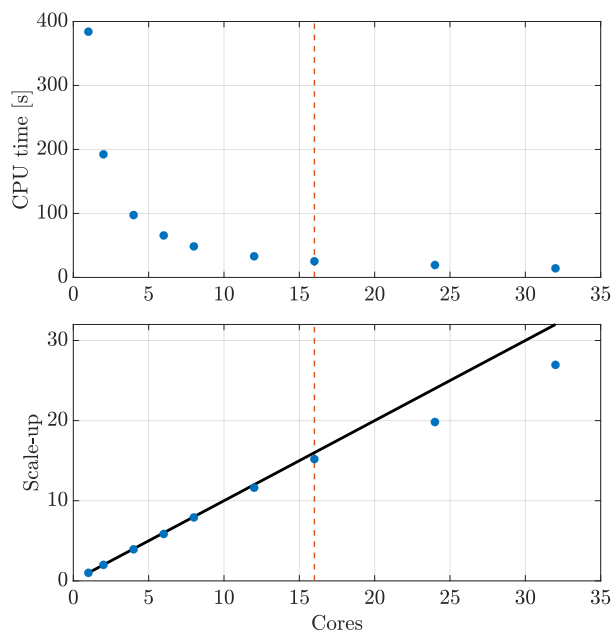


Fig. 2. Scaling plot for Monte Carlo simulations of NMPC closed loop in parallel. Scaling on 32 cores is approximately 27 times.

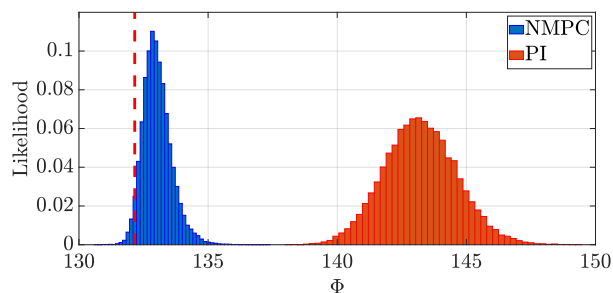


Fig. 3. Histograms of the distribution of the Φ -metric, (44), for the PI controller and the NMPC in closed-loop. The red dashed line indicates the performance of the NMPC in the deterministic case. Computation time for 30,000 closed-loop simulations with NMPC is approximately 55 min.

VIII. CONCLUSION

The paper presented a parallel Monte Carlo simulation based performance quantification method for NMPC in closed-loop. The method is made computationally feasible by combining a new thread-safe NMPC implementation with an existing implementation of a high-performance Monte Carlo simulation toolbox in C. The toolbox showed almost linear scale-up for the closed-loop simulations with an NMPC. We considered a simple CSTR model to demonstrate the performance quantification method. We performed 30,000 Monte Carlo simulations of the closed-loop with NMPC in approximately 55 min and applied the simulations to quantify the performance of the NMPC in presence of process noise. We compared the NMPC performance to a hand-tuned PI

controller. Performance distributions provided by the Monte Carlo simulations showed that the NMPC outperforms the PI controller in both mean and variance. In addition, the performance quantification method is well-suited for NMPC tuning, e.g., tuning of stage cost or constraint back-off.

REFERENCES

- [1] M. R. Wahlgreen, A. Thode Reenberg, M. K. Nielsen, A. Rydahl, T. K. S. Ritschel, B. Dammann, and J. B. Jørgensen, "A High-Performance Monte Carlo Simulation Toolbox for Uncertainty Quantification of Closed-loop Systems," *Proceedings of the 60th IEEE Conference on Decision and Control (CDC)*, pp. 6755–6761, 2021.
- [2] M. R. Wahlgreen, J. B. Jørgensen, and M. Zanon, "Model Predictive Control Tuning by Monte Carlo Simulation and Controller Matching," *Proceedings of Foundations of Computer Aided Process Operations / Chemical Process Control (FOCAPO / CPC)*, accepted, 2023.
- [3] A. T. Reenberg, T. K. S. Ritschel, B. Dammann, and J. B. Jørgensen, "High-performance Uncertainty Quantification in Large-scale Virtual Clinical Trials of Closed-loop Diabetes Treatment," *Proceedings of the American Control Conference (ACC)*, pp. 1367–1372, 2022.
- [4] N. Kantas, J. M. Maciejowski, and A. Lecchini-Visintini, "Sequential Monte Carlo for Model Predictive Control," *Lecture Notes in Control and Information Sciences*, vol. 384, pp. 263–273, 2009.
- [5] S. K. Botchu and S. Ungarala, "Nonlinear Model Predictive Control Based on Sequential Monte Carlo State Estimation," *Ifac Proceedings Volumes (ifac-papersonline)*, vol. 40, no. 5, pp. 29–34, 2007.
- [6] P. Ross, "Why CPU Frequency Stalled," *IEEE Spectrum*, vol. 45, no. 4, p. 72, 2008.
- [7] M. R. Wahlgreen, E. Schroll-Fleischer, D. Boiroux, T. K. S. Ritschel, H. Wu, J. K. Huusom, and J. B. Jørgensen, "Nonlinear Model Predictive Control for an Exothermic Reaction in an adiabatic CSTR," *6th Conference on Advances in Control and Optimization of Dynamical Systems ACODS, Chennai, India*, February 16–19 2020.
- [8] J. B. Jørgensen, T. K. S. Ritschel, D. Boiroux, E. Schroll-Fleischer, M. R. Wahlgreen, M. K. Nielsen, H. Wu, and J. K. Huusom, "Simulation of NMPC for a Laboratory Adiabatic CSTR with an Exothermic Reaction," *Proceedings of 2020 European Control Conference (ECC)*, pp. 202–207, 2020.
- [9] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, 2006.
- [10] R. Schneider and C. Georgakis, "How To NOT Make the Extended Kalman Filter Fail," *Industrial & Engineering Chemistry Research*, vol. 52, p. 3354–3362, 2013.
- [11] H. Bock and K. Plitt, "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.
- [12] G. Frison and J. B. Jørgensen, "Efficient Implementation of the Riccati Recursion for Solving Linear-Quadratic Control Problems," *IEEE International Conference on Control Applications (CCA), Hyderabad, India*, pp. 1117–1122, 2013.
- [13] J. B. Jørgensen, "Moving Horizon Estimation and Control," Ph.D. dissertation, Technical University of Denmark, 2004.
- [14] M. R. Wahlgreen and J. B. Jørgensen, "On the Implementation of a Preconditioned Riccati Recursion based Primal-Dual Interior-Point Algorithm for Input Constrained Optimal Control Problems," *IFAC-PapersOnLine*, vol. 55, no. 7, pp. 346–351, 2022, 13th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS).
- [15] M. J. D. Powell, "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," *Proceedings of the Biennial Conference on Numerical Analysis*, pp. 144–157, 1978.
- [16] 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.
- [17] Z. Xianyi, W. Qian, and Z. Yunquan, "Model-driven Level 3 BLAS Performance Optimization on Loongson 3A Processor," *2012 IEEE 18th International Conference on Parallel and Distributed Systems*, pp. 684–691, 2012.
- [18] Q. Wang, X. Zhang, Y. Zhang, and Q. Yi, "AUGEM: Automatically Generate High Performance Dense Linear Algebra Kernels on x86 CPUs," *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, 2013.

APPENDIX F

Paper VI: IFAC WC 2023

Dynamic Optimization for Monoclonal Antibody Production

Authors:

Morten Wahlgreen Kaysfeld, Deepak Kumar, Marcus Krogh Nielsen, John Bagterp
Jørgensen

To be published in:

Proceedings of the 22nd World Congress of the International Federation of Automatic
Control (IFAC WC), Yokohama, Japan, July 9-14, 2023.
arXiv: <https://doi.org/10.48550/arXiv.2302.09932>

Dynamic Optimization for Monoclonal Antibody Production

Morten Wahlgreen Kaysfeld, Deepak Kumar,
Marcus Krogh Nielsen, John Bagterp Jørgensen

*Department of Applied Mathematics and Computer Science,
Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark*

Abstract:

This paper presents a dynamic optimization numerical case study for Monoclonal Antibody (mAb) production. The fermentation is conducted in a continuous perfusion reactor. We represent the existing model in terms of a general modeling methodology well-suited for simulation and optimization. The model consists of six ordinary differential equations (ODEs) for the non-constant volume and the five components in the reactor. We extend the model with a glucose inhibition term to make the model feasible for optimization case studies. We formulate an optimization problem in terms of an optimal control problem (OCP) and consider four different setups for optimization. Compared to the base case, the optimal operation of the perfusion reactor increases the mAb yield with 44% when samples are taken from the reactor and with 52% without sampling. Additionally, our results show that multiple optimal feeding trajectories exist and that full glucose utilization can be forced without loss of mAb formation.

Keywords: Monoclonal antibody production, optimal control, process modeling, perfusion reactor, fermentation.

1. INTRODUCTION

Monoclonal antibody (mAb) production in mammalian cells is a well-known technique for synthesising identical antibodies. These antibodies are proteins with significance in medical applications. In 2017, mAbs represented the 6 top-selling biopharmaceutical products and has an expected yearly sale of 130-200 billion US dollars in 2022 (Walsh, 2018; Grilo and Mantalaris, 2019). This has resulted in significant efforts to increase the synthesis of mAbs in bioreactors with mammalian cell cultures. Biopharmaceutical companies increasingly seek for novel protein production methods to accommodate for an increasing number of protein drug candidates that enter various phases of research. In today's competitive market, it is challenging to retain desirable quality attributes while shortening time to market, maintaining cost efficiency, and enabling manufacturing flexibility.

High cell density and productivity at large scale are key factors to achieve high process yield. There exist multiple bioreactor designs to achieve a balance between cell growth and volumetric productivity (Blunt et al., 2018; Mitra and Murthy, 2022; Carvalho et al., 2017). Step-wise bolus injections of the feed solution to the production bioreactor is the most frequently applied method due to its simplicity (Maria, 2020). Recently, continuous bioreactor systems have received increasing attention due to its easy scale-up, waste minimization, and non-sterile cultivation (Blunt et al., 2018).

Medium development for a bioreactor system consists of multiple parts including optimization of feeding strategies,

and production of both batch medium and feed concentrates. Thus, medium development with statistical design of experiments type methods requires lots of time, effort, and cost (Wohlenberg et al., 2022; Luna and Martínez, 2014; Rendón-Castrillón et al., 2021). Therefore, process models can lead to valuable insights and optimization of bioreactors without the need of performing expensive and time consuming experiments. As an example, mechanistic modeling and optimization has been applied for a U-loop reactor for single-cell protein production (Ritschel et al., 2019).

Mechanistic models are extensively implemented for bioreactor optimization, since the models provide a deeper understanding of the growth and production of mammalian cells (Glen et al., 2018; Sha et al., 2018). Glucose concentration, lactate concentration, and temperature of the cellular environment impact their metabolic pathway (Sissolak et al., 2019; Fan et al., 2015). A higher glucose concentration results in decreased cellular growth rate and increased product formation (Vergara et al., 2018), whereas a higher lactate concentration deteriorates both cell growth and productivity (Li et al., 2012).

In this paper, we consider an existing mechanistic model for mAb production and apply advanced optimization techniques to compute novel optimal feeding strategies for the process. The model describes a fermentation process for mAb production conducted in a continuous perfusion bioreactor (Kumar et al., 2022). We present the model with a general modeling methodology well-suited for simulation and optimization (Wahlgreen et al., 2022), and extend the model with a glucose inhibition term to ensure that optimization does not exceed physical glucose inhibi-

* Corresponding author: J.B. Jørgensen (E-mail: jbjo@dtu.dk).

tion limits. The model consists of six ordinary differential equations (ODEs) for the non-constant volume and five components in the bioreactor. We present an optimization problem expressed as an optimal control problem (OCP). The OCP maximizes the final amount of mAb in the reactor while satisfying a set of operational constraints. Our results show that optimal process operation improves the mAb productivity with up to 52%. Additionally, our results show that there exists multiple feeding trajectories resulting in the same mAb production. Similar results have been obtained for fed-batch fermentation with Haldane growth kinetics (Ryde et al., 2021).

The remaining part of the paper is organized as follows. Section 2 presents the model for mAb production with a general modeling methodology. Section 3 introduces the considered optimization problem formulated as an OCP. Section 4 presents our optimization and simulation results. Section 5 presents our conclusions.

2. MONOCLONAL ANTIBODY PRODUCTION

We consider a biotechnological process for mAb production (Kumar et al., 2022) and reformulate the model in terms of a general modeling methodology for chemical reacting systems (Wahlgreen et al., 2022).

2.1 General model for a continuous perfusion reactor

The biotechnological process is conducted in a continuous perfusion reactor. We apply a general ODE model,

$$\frac{dV}{dt} = e^\top F_{\text{in}} - F_{\text{out}} - F_{\text{per}}, \quad (1a)$$

$$\frac{dm}{dt} = C_{\text{in}}F_{\text{in}} - cF_{\text{out}} - C_{\text{per}}cF_{\text{per}} + RV, \quad (1b)$$

where V is the volume, F_{in} is a vector with inlet flow rates, e is a vector of ones of proper dimension, F_{out} is a scalar outlet flow rate, F_{per} is a scalar perfusion flow rate, m is a vector of masses for each component, C_{in} is a matrix with inlet concentrations, $c = m/v$ is a vector with concentrations (densities), C_{per} is a diagonal matrix with elements between 0 and 1 describing the percentage of each component removed from the bioreactor by the perfusion stream, and R is a vector with production rates,

$$R = S^\top r(c), \quad (2)$$

with S being the stoichiometric matrix and $r(c)$ being a vector with reaction rates.

2.2 Reaction stoichiometry and kinetics

The process consists of five components,

$$\mathcal{C} = \{X_v, X_d, G, L, P\}, \quad (3)$$

where X_v are viable cells, X_d are dead cells, G is glucose, L is lactate, and P is mAb (product). We express the process in terms of six stoichiometric reactions,

1. Cell division, $\alpha_{1,G}G + X_v \longrightarrow 2X_v + \alpha_{1,P}P, r_1,$
2. Cell death, $X_v \longrightarrow X_d, r_2,$
3. Maintenance 1, $\alpha_{3,G}G + X_v \longrightarrow X_v + \alpha_{3,P}P, r_3,$
4. Maintenance 2, $X_v \longrightarrow X_v + \alpha_{4,L}L, r_4,$
5. Lactate production 1, $X_v \longrightarrow X_v + \alpha_{5,L}L, r_5,$
6. Lactate production 2, $X_v \longrightarrow X_v + \alpha_{6,L}L, r_6,$

which are compactly written in the stoichiometric matrix,

$$S = \begin{matrix} & X_v & X_d & G & L & P \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{bmatrix} 1 & 0 & -\alpha_{1,G} & 0 & \alpha_{1,P} \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_{3,G} & 0 & \alpha_{3,P} \\ 0 & 0 & 0 & \alpha_{4,L} & 0 \\ 0 & 0 & 0 & \alpha_{5,L} & 0 \\ 0 & 0 & 0 & \alpha_{6,L} & 0 \end{bmatrix} \end{matrix} \quad (4)$$

The six reaction rates are given as

$$r_1 = \mu_X(c, T)c_{X_v}, \quad r_2 = \mu_D(T)c_{X_v}, \quad (5a)$$

$$r_3 = \mu_{m_1}c_{X_v}, \quad r_4 = \mu_{m_2}c_{X_v}, \quad (5b)$$

$$r_5 = \mu_{L,p_1}(c, T)c_{X_v}, \quad r_6 = \mu_{L,p_2}(c)c_{X_v}, \quad (5c)$$

where the different rate functions are

$$\mu_X = \mu_{X,max}f_{lim}f_{inh}f_{temp}, \quad (6a)$$

$$\mu_D = \mu_{D,max}f_{D,temp}, \quad (6b)$$

$$\mu_{m_1} = \bar{\mu}_{m_1}, \quad (6c)$$

$$\mu_{m_2} = \bar{\mu}_{m_2} \frac{L_{max,2} - c_L}{L_{max,2}}, \quad (6d)$$

$$\mu_{L,p_1} = \mu_X \frac{L_{max,1} - c_L}{L_{max,1}}, \quad (6e)$$

$$\mu_{L,p_2} = \bar{\mu}_{L,p_2} \frac{L_{max,1} - c_L}{L_{max,1}}, \quad (6f)$$

and

$$f_{lim} = \frac{c_G}{K_G c_{X_v} + c_G}, \quad (7a)$$

$$f_{inh} = \frac{K_{IL}}{K_{IL} + c_L} (1 - K_{IPC_P}), \quad (7b)$$

$$f_{temp} = \exp\left(-\frac{K_1}{T}\right), \quad (7c)$$

$$f_{D,temp} = \exp\left(-\frac{K_2}{T}\right). \quad (7d)$$

The model, (1)-(7), is identical to the model presented by Kumar et al. (2022).

2.3 Model extension for optimization

We point out that the model, (1)-(7), does not include glucose inhibition for cell growth. In addition, the inhibition term, f_{inh} , becomes negative for high product concentrations. As such, the model is not directly suitable for optimization purposes. However, small model extensions lead to a feasible model well-suited for optimization. We extend the model with a simple glucose inhibition term,

$$f_{G,inh} = 1 - s_\gamma(c_G, \bar{c}_G), \quad (8)$$

where s_γ is a Sigmoid function given as

$$s_\gamma(x, \bar{x}) = \frac{1}{1 + \exp(-\gamma(x - \bar{x}))}. \quad (9)$$

Note that $f_{G,inh} = 1$ for small glucose concentrations and rapidly converges to 0 when $c_G > \bar{c}_G$. In addition, we remove the possibility of negative growth rate by adding a smooth maximum approximation to the product inhibition term. By combining the smooth maximum approximation with the glucose inhibition term, we get the final inhibition term,

$$f_{inh} = \frac{K_{IL}}{K_{IL} + c_L} \max_\alpha(0, 1 - K_{IPC_P}) f_{G,inh}, \quad (10)$$

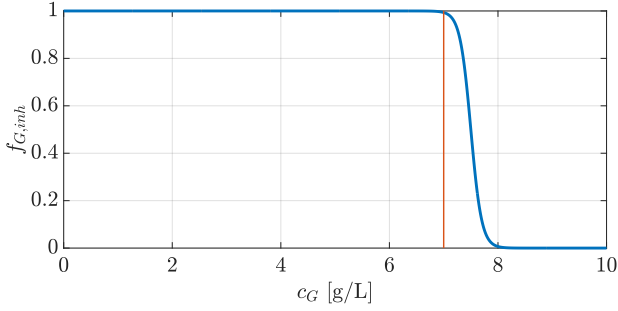


Fig. 1. Glucose inhibition term, $f_{G,inh}$. Red line: 7.0 g/L.

where \max_α is a smooth maximum approximation given as

$$\max_\alpha(x_1, \dots, x_n) = \frac{\sum_{i=1}^n x_i \exp(\alpha x_i)}{\sum_{i=1}^n \exp(\alpha x_i)}. \quad (11)$$

We design our glucose inhibition term such that the growth rate decreases for $c_G > 7$ [g/L] (Vergara et al., 2018). As such, we select $\bar{c}_G = 7.5$ [g/L] together with $\gamma = 10.0$. Fig. 1 shows the activation of the glucose inhibition term, $f_{G,inh}$. Fig. 2 presents the smooth maximum approximation of the product inhibition term for $\alpha = 100.0$. We point out that alternative glucose inhibition terms can be applied without loss of generality.

2.4 Operation of the continuous perfusion reactor

We operate the reactor in continuous perfusion mode. We feed glucose through an inlet stream with flow rate, F_G , and glucose concentration, $c_{G,in}$. In addition, we apply a pure water inlet stream with flow rate, F_W , resulting in the inlet flow vector,

$$F_{in} = \begin{bmatrix} F_W \\ F_G \end{bmatrix}. \quad (12)$$

As such, the inlet concentration matrix becomes

$$C_{in} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & c_{G,in} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \quad (13)$$

The perfusion outlet has a filter that only lets spend media pass, i.e., glucose and lactate. As such, the perfusion matrix becomes,

$$C_{per} = \begin{bmatrix} 0 & & & & \\ & 0 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 0 \end{bmatrix}. \quad (14)$$

We apply the outlet stream for sampling.

2.5 General notation

We formulate the model, (1), in terms of the general ODE system,

$$\dot{x}(t) = f(t, x(t), u(t), p), \quad x(t_0) = x_0, \quad (15)$$

where p are the parameters, and the states, x , and the inputs, u , are given as

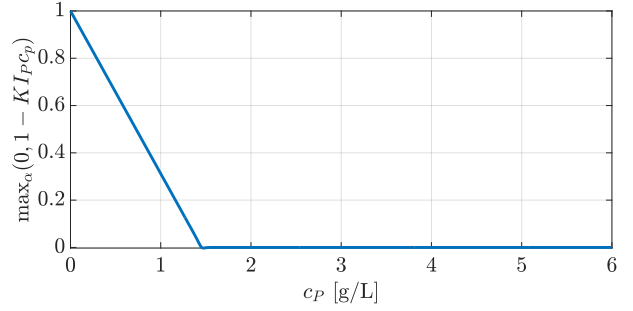


Fig. 2. Smooth maximum function for product inhibition.

$$x = \begin{bmatrix} V \\ m \end{bmatrix}, \quad u = \begin{bmatrix} F_{in} \\ F_{per} \\ F_{out} \\ T \end{bmatrix}. \quad (16)$$

We apply the general ODE notation, (15), to formulate optimization problems.

3. OPTIMIZATION

We formulate an optimization problem in terms of an OCP. The solution is the state and input trajectories in the finite horizon, T_h . We denote the initial time t_0 and the final time $t_f = t_0 + T_h$. We split the prediction and control horizon, T_h , into N control intervals of equal size T_s . As such, $T_h = NT_s$. We assume zero order hold parameterization of the inputs,

$$u(t) = u_k, \quad k = 0, \dots, N-1, \quad (17)$$

and consider the following OCP formulation,

$$\min_{\xi} \varphi, \quad (18a)$$

$$\text{s.t. } x(t_0) = x_0, \quad (18b)$$

$$\dot{x}(t) = f(t, x, u, d, p), \quad t_0 \leq t \leq t_0 + T_h, \quad (18c)$$

$$x_{\min} \leq x_k \leq x_{\max}, \quad k = 1, \dots, N, \quad (18d)$$

$$u_{\min} \leq u_k \leq u_{\max}, \quad k = 0, \dots, N-1, \quad (18e)$$

where $\xi = \{u_k, x_{k+1}\}_{k=0}^{N-1}$ are the decision variables and $\varphi = \varphi(\xi)$ is the objective function. We intent to maximize the final mAb production while satisfying a set of operational constraints. This can be formulated in terms of the following OCP,

$$\min_{\xi} \varphi = -m_P(t_f), \quad (19a)$$

$$\text{s.t. } x(t_0) = x_0, \quad (19b)$$

$$\dot{x}(t) = f(t, x, u, d, p), \quad t_0 \leq t \leq t_0 + T_h, \quad (19c)$$

$$V_{\min} \leq V_k \leq V_{\max}, \quad k = 1, \dots, N, \quad (19d)$$

$$m_{G,\min} \leq m_{G,k} \quad k = 1, \dots, N, \quad (19e)$$

$$m_{L,\min} \leq m_{L,k} \quad k = 1, \dots, N, \quad (19f)$$

$$F_{\min} \leq F_{W,k} \leq F_{\max}, \quad k = 0, \dots, N-1, \quad (19g)$$

$$F_{\min} \leq F_{G,k} \leq F_{\max}, \quad k = 0, \dots, N-1, \quad (19h)$$

$$F_{\min} \leq F_{per,k} \leq F_{\max}, \quad k = 0, \dots, N-1, \quad (19i)$$

$$T_{\min} \leq T_k \leq T_{\max}, \quad k = 0, \dots, N-1. \quad (19j)$$

We point out that additional non-negativity constraints could be applied, but have not been required for meaningful optimization.

Table 1. Model parameters.

Parameter	Value	Unit
$\mu_{X,max}$	0.153	[min ⁻¹]
$\mu_{D,max}$	$3.955 \cdot 10^{-5}$	[min ⁻¹]
$\bar{\mu}_{m_1}$	1.0	[min ⁻¹]
$\bar{\mu}_{m_2}$	1.0	[min ⁻¹]
$\bar{\mu}_{L,p_2}$	1.0	[min ⁻¹]
K_1	1689	[K]
K_2	524	[K]
K_G	0.85	[g/(cells $\times 10^9$)]
KI_L	344	[g/L]
KI_P	6.88×10^{-1}	[L/g]
$L_{max,1}$	628	[g/L]
$L_{max,2}$	0.5	[g/L]
\bar{c}_G	7.5	[g/L]
$\alpha_{1,G}$	0.4876	[g/(cells $\times 10^9$)]
$\alpha_{1,P}$	6.62×10^{-8}	[g/(cells $\times 10^9$)]
$\alpha_{3,G}$	1.102×10^{-4}	[g/(cells $\times 10^9$)]
$\alpha_{3,P}$	1.2×10^{-5}	[g/(cells $\times 10^9$)]
$\alpha_{4,L}$	1.89×10^{-5}	[g/(cells $\times 10^9$)]
$\alpha_{5,L}$	0.5504	[g/(cells $\times 10^9$)]
$\alpha_{6,L}$	1.0249×10^{-5}	[g/(cells $\times 10^9$)]

4. RESULTS

This section presents our results. We perform a base case simulation that reproduces the results from Kumar et al. (2022). Additionally, we perform four different optimizations and compare these to the base case. In all simulations, we consider a 14 days fermentation.

4.1 Base case simulation

We simulate the model to reproduce the results presented by Kumar et al. (2022). As such, we operate the reactor in three phases, 1) batch phase, 2) fed-batch phase, and 3) perfusion phase. In the batch phase, no inlets or outlets are active. In the fed-batch phase, both water and glucose inlet streams are active in boluses once a day in 30 min intervals to achieve a total inlet flow rate $F_{tot} = F_W + F_G = 0.018$ L/min with glucose concentration 32.0 g/L. In particular, this requires $F_W = 2.7692 \cdot 10^{-4}$ L/min and $F_G = 0.0177$ L/min. In the perfusion phase, the perfusion outlet is active with $F_{per} = 0.0015$ L/min and the inlets are active at $F_W = 0.0011$ L/min and $F_G = 3.9923 \cdot 10^{-4}$ L/min to achieve a total inlet flow rate of $F_{tot} = 0.0015$ L/min at a glucose concentration of 8.65 g/L. Table 1 presents the list of model parameters adapted from Kumar et al. (2022) and Table 2 presents the operation parameters including initial conditions. Fig. 3 presents the base case simulation reproducing the results by Kumar et al. (2022).

4.2 Optimization

We solve the OCP, (19), for the full horizon of $T_h = 14$ day with $T_s = 30$ min. As such, we get the discrete horizon, $N = 672$. We consider four different optimization setups. We either apply no sampling or the sampling strategy from the base case. Additionally, we test the effect of enforcing almost full glucose utilization in the end of the fermentation. Table 3 presents the four optimization setups including the additional OCP constraints in each setup. We apply a direct multiple shooting discretization approach of the OCP and solve the resulting nonlinear programming (NLP) in CasADi (Andersson et al., 2019).

Table 2. Operation parameters.

Parameter	Value	Unit
V_0	5.650	[L]
$m_{X_v,0}$	3.955	[cells $\times 10^9$]
$m_{X_d,0}$	0.0	[cells $\times 10^9$]
$m_{G,0}$	34.18	[g]
$m_{L,0}$	0.678	[g]
$m_{P,0}$	0.0	[g]
$c_{G,in}$	32.5	[g/L]
F_{min}	0.0	[L/min]
F_{max}	0.02	[L/min]
T_{min}	308.15	[K]
T_{max}	310.15	[K]
V_{min}	4.0	[L]
V_{max}	8.0	[L]
$m_{G,min}$	0.0	[g]
$m_{L,min}$	0.0	[g]

Table 3. Optimization setups.

Setup	Description	Constraints in OCP
(1)	× sampling	• $F_{out} = 0.0$ [L/min]
	× glucose utilization	• $m_G(t_f) \leq \infty$ [g]
(2)	× sampling	• $F_{out} = 0.0$ [L/min]
	✓ glucose utilization	• $m_G(t_f) \leq 1.0$ [g]
(3)	✓ sampling	• $F_{out} = \bar{F}_{out}$ [L/min]
	× glucose utilization	• $m_G(t_f) \leq \infty$ [g]
(4)	✓ sampling	• $F_{out} = \bar{F}_{out}$ [L/min]
	✓ glucose utilization	• $m_G(t_f) \leq 1.0$ [g]

Table 4. mAb production and improvement relative to the base case.

Simulation	mAb [g]	Improvement [%]
Base case	15.57	-
Opt. (1)	23.63	52
Opt. (2)	23.63	52
Opt. (3)	22.47	44
Opt. (4)	22.47	44

Fig. 4 compares the base case simulation to the four optimal simulations. We observe that optimal cell growth is achieved by maintaining a constant glucose concentration of around $c_G = 7$ g/L to avoid glucose inhibition. If sampling is required, the optimal feeding trajectories compensates for the loss of medium to maintain the optimal glucose concentration. Once the product concentration fully inhibits the cell growth, the temperature is decreased to increase product formation.

Fig. 5 presents the time evolution of mAb in the bioreactor for the base case simulation and the optimal simulations. Table 4 presents the total mAb production in the end of the 14 day fermentation for all five simulations. We observe that all optimal simulations increase the mAb production compared to the base case. In particular, the optimal simulations produces 44% more mAb with sampling and 52% without sampling. We notice that forcing full glucose utilization does not affect the final production of mAb. This shows that there are multiple optimal solutions all resulting in the same mAb production (Ryde et al., 2021). As such, optimization setup (2) and (4) are preferred under the assumption that glucose has a cost. This cost can be included in the optimization problem resulting in an economic OCP.

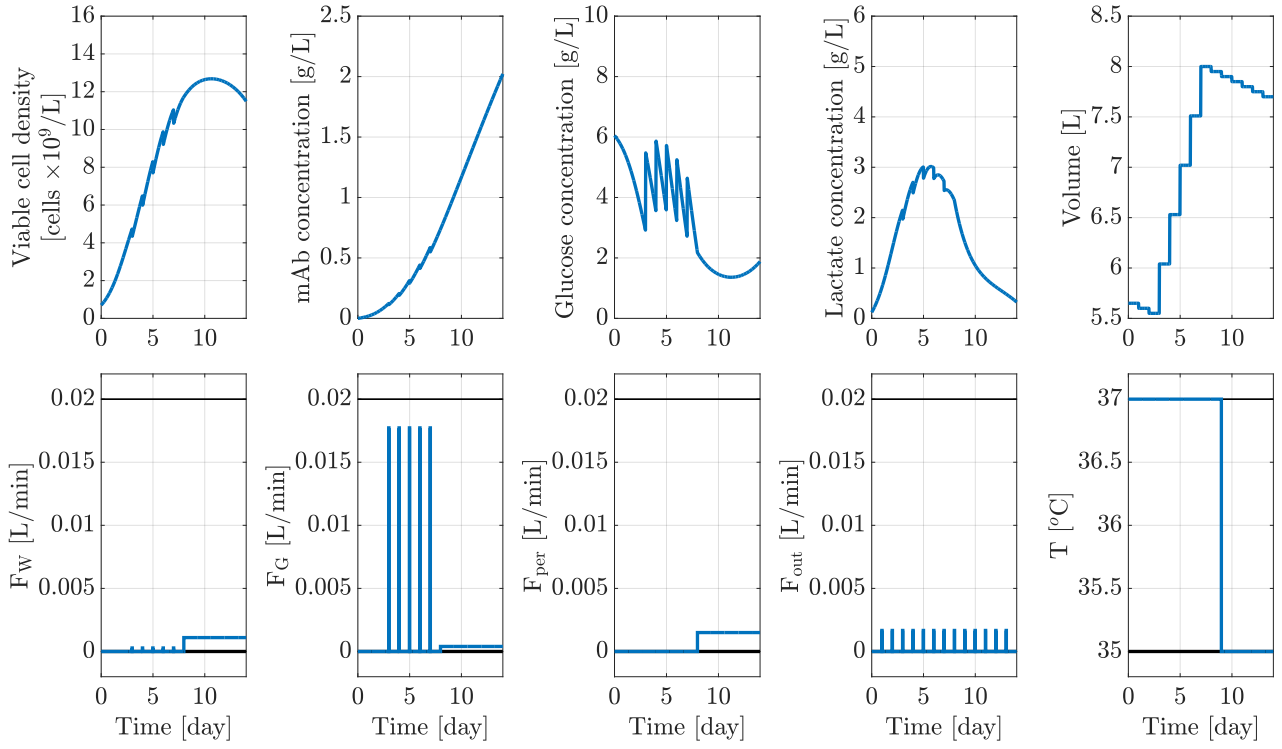


Fig. 3. Base case simulation reproducing the results from Kumar et al. (2022).

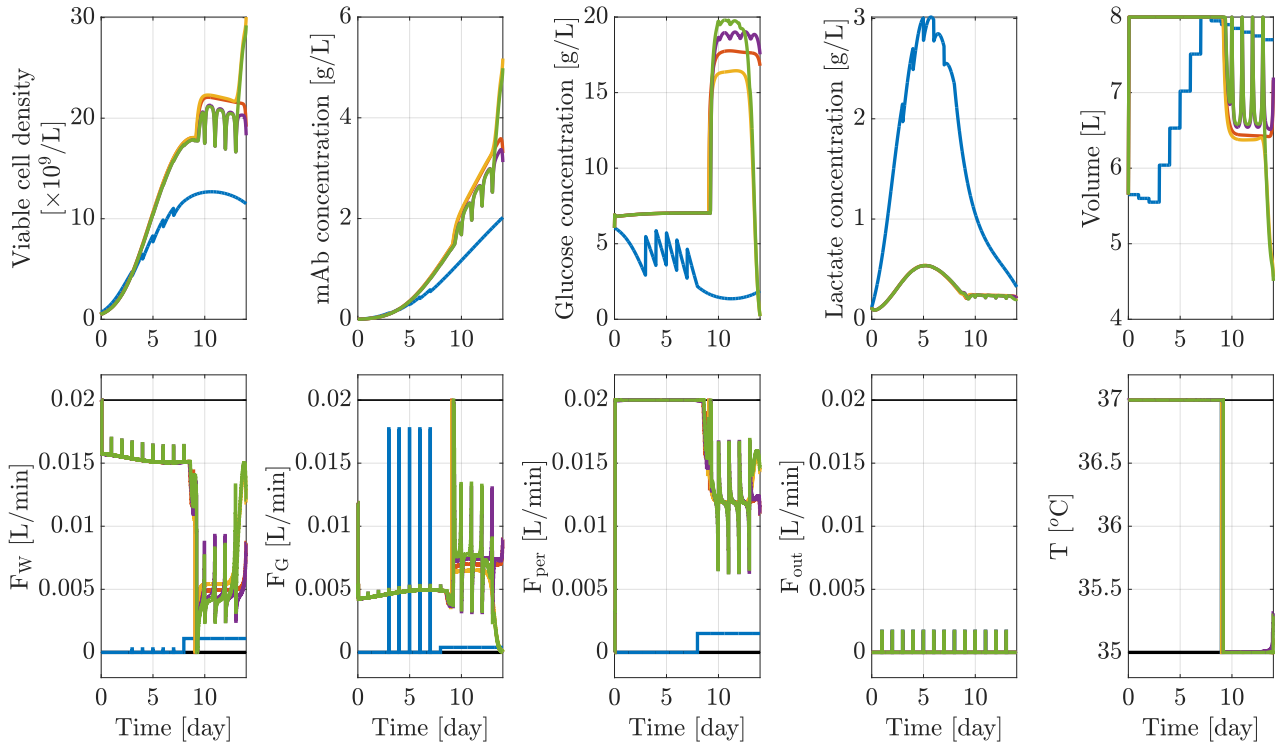


Fig. 4. The base case simulation and simulations for four different optimization setups. The blue line is the base case, the red line is setup (1), the yellow line is setup (2), the purple line is setup (3), and the green line is setup (4).

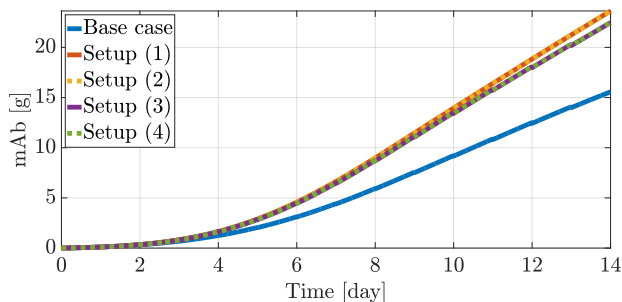


Fig. 5. Final mAb production for the base case simulation and the four optimal simulations.

5. CONCLUSION

The paper presented a dynamic optimization numerical case study for mAb production. We applied a general modeling methodology to present an existing fermentation model for mAb production conducted in a continuous perfusion reactor. We expressed an optimization problem in terms of an OCP for maximization of mAb production in the end of the fermentation. Our results showed that optimal operation of the continuous perfusion reactor improves the mAb production by up to 52% compared to the base case. Additionally, our results showed that there exist multiple optimal solutions producing the same amount of mAb. Therefore, a full glucose utilization constraint in the OCP is preferred to reduce glucose loss.

REFERENCES

- Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.
- Blunt, W., Levin, D.B., and Cicek, N. (2018). Bioreactor Operating Strategies for Improved Polyhydroxyalkanoate (PHA) Productivity. *Polymers*, 10(11).
- Carvalho, L.S., da Silva, O.B., de Almeida, G.C., de Oliveira, J.D., Parachin, N.S., and Carmo, T.S. (2017). Production Processes for Monoclonal Antibodies. In A.F. Jozala (ed.), *Fermentation Processes*, chapter 10. IntechOpen.
- Fan, Y., Jimenez Del Val, I., Müller, C., Wagtberg Sen, J., Rasmussen, S.K., Kontoravdi, C., Weilguny, D., and Andersen, M.R. (2015). Amino Acid and Glucose Metabolism in Fed-Batch CHO Cell Culture Affects Antibody Production and Glycosylation. *Biotechnology and Bioengineering*, 112(3), 521–535.
- Glen, K.E., Cheeseman, E.A., Stacey, A.J., and Thomas, R.J. (2018). A mechanistic model of erythroblast growth inhibition providing a framework for optimisation of cell therapy manufacturing. *Biochemical Engineering Journal*, 133, 28–38.
- Grilo, A.L. and Mantalaris, A. (2019). The Increasingly Human and Profitable Monoclonal Antibody Market. *Trends in Biotechnology*, 37(1), 9–16.
- Kumar, D., Gangwar, N., Rathore, A.S., and Ramteke, M. (2022). Multi-objective optimization of monoclonal antibody production in bioreactor. *Chemical Engineering and Processing - Process Intensification*, 180.
- Li, J., Wong, C.L., Vijayasankaran, N., Hudson, T., and Amanullah, A. (2012). Feeding Lactate for CHO Cell Culture Processes: Impact on Culture Metabolism and Performance. *Biotechnology and Bioengineering*, 109(5), 1173–1186.
- Luna, M. and Martínez, E. (2014). A Bayesian Approach to Run-to-Run Optimization of Animal Cell Bioreactors Using Probabilistic Tendency Models. *Industrial & Engineering Chemistry Research*, 53(44), 17252–17266.
- Maria, G. (2020). Model-Based Optimization of a Fed-Batch Bioreactor for mAb Production Using a Hybridoma Cell Culture. *Molecules*, 25(23).
- Mitra, S. and Murthy, G.S. (2022). Bioreactor control systems in the biopharmaceutical industry: a critical perspective. *Systems Microbiology and Biomanufacturing*, 2, 91–112.
- Rendón-Castrillón, L., Ramírez-Carmona, M., Ocampo-López, C., and Gómez-Arroyave, L. (2021). Mathematical Model for Scaling up Bioprocesses Using Experiment Design Combined with Buckingham Pi Theorem. *Applied Sciences*, 11(23).
- Ritschel, T.K.S., Boiroux, D., Nielsen, M.K., Huusom, J.K., Jørgensen, S.B., and Jørgensen, J.B. (2019). Economic Optimal Control of a U-loop Bioreactor using Simultaneous Collocation-based Approaches. In *Proceedings of the IEEE Conference on Control Technology and Applications (CCTA)*, 933–938.
- Ryde, T.E., Wahlgreen, M.R., Nielsen, M.K., Hørsholt, S., Jørgensen, S.B., and Jørgensen, J.B. (2021). Optimal Feed Trajectories for Fedbatch Fermentation with Substrate Inhibition Kinetics. *IFAC-PapersOnLine*, 54(3), 318–323.
- Sha, S., Huang, Z., Wang, Z., and Yoon, S. (2018). Mechanistic modeling and applications for CHO cell culture development and production. *Current Opinion in Chemical Engineering*, 22, 54–61.
- Sissolak, B., Lingg, N., Sommeregger, W., Striedner, G., and Vorauer-Uhl, K. (2019). Impact of mammalian cell culture conditions on monoclonal antibody charge heterogeneity: an accessory monitoring tool for process development. *Journal of Industrial Microbiology and Biotechnology*, 46(8), 1167–1178.
- Vergara, M., Torres, M., Müller, A., Avello, V., Acevedo, C., Berrios, J., Reyes, J.G., Valdez-Cruz, N.A., and Altamirano, C. (2018). High glucose and low specific cell growth but not mild hypothermia improve specific r-protein productivity in chemostat culture of CHO cells. *PLOS ONE*, 13(8), 1–22.
- Wahlgreen, M.R., Meyer, K., Ritschel, T.K.S., Engsig-Karup, A.P., Gernaey, K.V., and Jørgensen, J.B. (2022). Modeling and Simulation of Upstream and Downstream Processes for Monoclonal Antibody Production. *The 13th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS)*, Busan, Republic of Korea.
- Walsh, G. (2018). Biopharmaceutical benchmarks 2018. *Nature Biotechnology*, 36(12), 1136–1145.
- Wohlenberg, O.J., Kortmann, C., Meyer, K.V., Schellenberg, J., Dahlmann, K., Bahnemann, J., Scheper, T., and Solle, D. (2022). Optimization of a mAb production process with regard to robustness and product quality using quality by design principles. *Engineering in Life Sciences*, 22(7), 484–494.

APPENDIX G

Paper VII: JPC 2023

Uncertainty Quantification of an Economic Nonlinear Model Predictive Controller for Monoclonal Antibody Production

Authors:

Morten Wahlgreen Kaysfeld, John Bagterp Jørgensen

Submitted to:

Journal of Process Control, 2023.

Note: This version includes minor corrections compared to the submitted manuscript.

Uncertainty Quantification of an Economic Nonlinear Model Predictive Controller for Monoclonal Antibody Production

Morten Wahlgreen Kaysfeld^a, John Bagterp Jørgensen^{a,*}

^aDepartment of Applied Mathematics and Computer Science, Technical University of Denmark, Kgs. Lyngby, DK-2800, Denmark

ARTICLE INFO

Keywords:

Uncertainty Quantification
Monoclonal Antibody production
Economic NMPC
Modeling
Closed-loop simulation
Monte Carlo simulation

ABSTRACT

In this paper, we develop an economic nonlinear model predictive controller (ENMPC) for optimal operation of a monoclonal antibody (mAb) fermentation process. We apply Monte Carlo simulation for uncertainty quantification of the operation with the ENMPC. The fermentation is conducted in a perfusion reactor with a pure water inlet stream, a glucose inlet stream, a perfusion stream, an outlet stream, and a temperature regulation system. We consider an existing mechanistic model for the fermentation process, which we extend to a stochastic differential equation (SDE) model for simulation and design of the ENMPC. The ENMPC consists of a continuous-discrete extended Kalman filter (CD-EKF) for state estimation and an economic regulator based on mAb and glucose prices. A stochastic simulation of a 14 days fermentation in closed-loop shows that the reactor is operated in two phases, which we denote the *growth phase* and the *production phase*. Additionally, a Monte Carlo simulation study with 10.000 closed-loop simulations shows that the fermentation process operated by the ENMPC has a mean mAb production of 23.89 g with a 95% confidence interval [20.78, 27.04] g. Compared to a base case strategy, the ENMPC increases the mean mAb production with 52% at the cost of a 56% range (uncertainty) increase. The uncertainty increase is likely due to the ENMPC operating the perfusion reactor close to a point of glucose inhibition. The study showed no overlap in the 95% confidence intervals, which indicate that there is a high statistical probability for the ENMPC to produce more mAb. These results provide valuable uncertainty measures for the mAb fermentation process, which can be obtained prior to experiments. Finally, we apply insights from the simulation study to design a simple controller, which has practically identical performance to the ENMPC.

1. Introduction

The first monoclonal antibody (mAb) was developed in 1975 [1], and the first therapeutic mAb, muromonab-CD3 (Orthoclone OKT3), was fully licensed and approved by the United States Food and Drug Administration (FDA) in 1986 [2, 3]. The production of therapeutic mAbs has drastically increased since then, and mAbs are now valuable pharmaceuticals with a market share of 185.50 billion USD in 2021 and an expected share of 494.53 billion USD in 2030 [4]. Treatments involving mAbs have been applied for various diseases including cancer, autoimmune disorders, and infectious diseases [5–7]. Additionally, during the COVID-19 pandemic [8], mAbs were proposed for both treatment and prevention of COVID-19 [9–11]. Clearly, efficient, reliable, and cheap production of mAbs is essential for affordable treatment of a number of diseases.


Large-scale production of mAbs is usually conducted in bioreactors. The biotechnological fermentation process involves the growth of specific cells, which produce mAbs. The bioreactor forms a controlled environment, where factors like temperature, pH, dissolved oxygen concentration, and available nutrients can be controlled to improve cell growth and mAb production [12]. Bioreactors are commonly run in batch or fed-batch modes, where pulse injection of feed solutions is a frequently applied operational strategy [13]. However, there has been increasing attention towards

continuous operation to reduce manufacturing cost, increase flexibility, and increase product quality [14–16].

Due to the huge value of mAbs, efforts to optimize mAb fermentation processes have received much attention, e.g., by exploring new cell culture systems, optimizing bioreactor designs and operation, and developing advanced modeling and control strategies [17–20]. These techniques all strive to increase cell growth, increase mAb yield, minimize production time, and reduce uncertainties in the production process. One promising control strategy is model predictive control (MPC), which has been applied as an advanced model-based process control technique for biotechnological processes [21–28]. Development of mathematical models enables improved process understanding through simulation studies and can be applied for optimization purposes [29–34]. Simulation studies in both open-loop and closed-loop provide valuable insights to the process, but single deterministic or stochastic simulations provide only limited information about the process uncertainty. Furthermore, misleading or even wrong conclusions about the performance can be made based on single simulations [35, 36].

There exist various techniques for quantification of mAb fermentation uncertainties. Techniques such as design of experiments (DoE) and multivariate statistical analysis (MSA) have been applied for uncertainty quantification of mAb fermentation processes [37]. Also, valuable online monitoring and continuous feedback have become more available for biotechnological processes through process analytical technology (PAT) proposed by the FDA [38]. However, DoE and MSA require time consuming and expensive experiments [39–41], and PAT does not provide uncertainty measures

*Corresponding author

 jbj@dtu.dk (J.B. Jørgensen)
ORCID(s):

prior to experiments. We propose a large-scale Monte Carlo simulation approach to achieve uncertainty quantification of the mAb fermentation process prior to experiments. The approach combines mechanistic modeling of the mAb fermentation process, noise sampling, and parallel simulation of the system [36]. The method has previously been applied for tuning of simple controllers [36, 42], in a large-scale virtual clinical trial with 1,000,000 participants for people with type 1 diabetes [43], and for uncertainty quantification of a nonlinear model predictive controller (NMPC) in closed-loop [44].

In this work, we consider a previously developed and verified mechanistic model for mAb fermentation conducted in a perfusion reactor [34], which was later extended for optimization and control purposes [45]. The model includes the influence of temperature, glucose, lactate, and the product (mAb) on the metabolic pathway [46–49]. Factors like oxygen and pH are not included in the model, but are part of the reactor operation and kept at their specified values in experiments [34]. We present the model with a generic modeling methodology, where mass balances in the form of ordinary differential equations (ODEs) represent the reactor. Specification of stoichiometry and kinetic models for a specific process completes the model. The generic modeling methodology has previously been applied for a range of reactors including fed-batch reactors (FBRs) and continuous stirred tank reactors (CSTRs) [35, 42, 50]. In later work, we applied the methodology for the considered mAb fermentation process conducted in a perfusion reactor [45]. The mAb fermentation model consists of six ODEs, which we in this work extend with diffusion terms to express uncertainties in the number of viable cells, glucose mass, and lactate mass. The result is a stochastic differential equation (SDE) model for the mAb fermentation process conducted in a perfusion reactor. The measured variables are static functions of the states, corrupted by additive noise, and obtained at discrete times. The SDE model for the dynamics and the static model for the observations constitute a continuous-discrete model. We develop an economic nonlinear model predictive controller (ENMPC) based on this continuous-discrete model [28, 51–57]. The ENMPC consists of a continuous-discrete extended Kalman filter (CD-EKF) for state estimation and an economic regulator based on mAb and glucose prices. The study consists of 10,000 Monte Carlo simulations of the closed-loop system. We apply these simulations to quantify the uncertainty of the fermentation process. Finally, based on key insights from the ENMPC, we apply the 'from-simple-via-complex-to-lucid' approach to develop a simple controller design with a proportional-integral (PI) controller that has the same performance as the ENMPC [58].

The remaining parts of the paper are organized as follows. Section 2 introduces the model for mAb fermentation in a perfusion reactor. Section 3 presents the developed ENMPC including the CD-EKF and the regulator. Section 4 introduces the discrete closed-loop system for simulation, and Section 5 presents a short description of the Monte

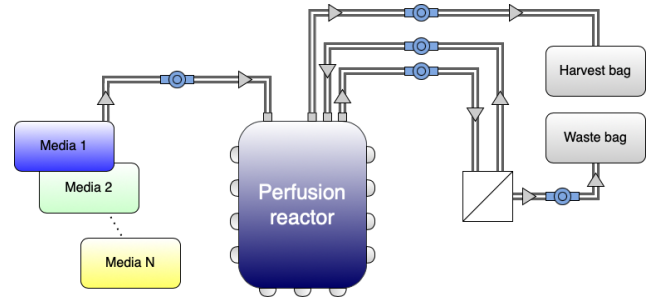


Figure 1: Illustration of the perfusion reactor with N media inlet streams, a perfusion stream, and an outlet stream.

Carlo simulation framework. In Section 6, we discuss the simulation results, and conclusions are made in Section 7.

2. Model for mAb fermentation

We consider a perfusion reactor for mAb fermentation. The modeled perfusion reactor has a set of inlet streams, a perfusion stream, an outlet stream, and adjustable reactor temperature (the reactor has an ideal temperature controller). The perfusion stream has a filter that only lets spend media pass such that cells and mAbs are kept in the reactor. Factors like dissolved oxygen concentration and pH are not included in the model, but are part of the reactor operation and kept at their specified values in experiments [34]. Figure 1 presents an overview of the reactor components included in the model. The considered mAb fermentation model was originally presented by Kumar et al. [34], and later the model was introduced using a generic modeling methodology and adapted for optimization purposes [45]. In this section, we introduce the mAb fermentation model in terms of the generic modeling methodology and extend the model with diffusion terms.

2.1. Model of the perfusion reactor

We let \mathcal{C} denote the set of components, \mathcal{S} denote the set of inlet streams, and \mathcal{R} denote the set of reactions. We assume that the density of the well-stirred reactor content is constant. Then, the mass balances of the perfusion reactor are [45],

$$\frac{dV}{dt} = e^T F_{in} - F_{out} - F_{per}, \quad (1a)$$

$$\frac{dm}{dt} = C_{in} F_{in} - c F_{out} - C_{per} c F_{per} + R V. \quad (1b)$$

$V \in \mathbb{R}$ is the volume, $e \in \mathbb{R}^{|\mathcal{S}|}$ is a vector of ones, $F_{in} \in \mathbb{R}^{|\mathcal{S}|}$ are the inlet flow rates, $F_{out} \in \mathbb{R}$ is the outlet flow rate, $F_{per} \in \mathbb{R}$ is the perfusion flow rate, $m \in \mathbb{R}^{|\mathcal{C}|}$ are the component masses, $C_{in} \in \mathbb{R}^{|\mathcal{C}| \times |\mathcal{S}|}$ contains the inlet stream concentrations, $c = m/V \in \mathbb{R}^{|\mathcal{C}|}$ are the component concentrations, $C_{per} \in \mathbb{R}^{|\mathcal{C}| \times |\mathcal{C}|}$ is a diagonal matrix with perfusion removal percentages, $R \in \mathbb{R}^{|\mathcal{C}|}$ are the production rates given as

$$R = S^T r, \quad (2)$$

Table 1

The six stoichiometric reactions in the mAb fermentation model.

1. Cell division,	$\alpha_{1,G}G + X_v \longrightarrow 2X_v + \alpha_{1,P}P,$	$r_1,$
2. Cell death,	$X_v \longrightarrow X_d,$	$r_2,$
3. Maintenance 1,	$\alpha_{3,G}G + X_v \longrightarrow X_v + \alpha_{3,P}P,$	$r_3,$
4. Maintenance 2,	$X_v \longrightarrow X_v + \alpha_{4,L}L,$	$r_4,$
5. Lactate production 1,	$X_v \longrightarrow X_v + \alpha_{5,L}L,$	$r_5,$
6. Lactate production 2,	$X_v \longrightarrow X_v + \alpha_{6,L}L,$	$r_6.$

with $S \in \mathbb{R}^{|\mathcal{R}| \times |\mathcal{C}|}$ being the stoichiometric matrix, and $r = r(c) \in \mathbb{R}^{|\mathcal{R}|}$ being the reaction rates.

The system of ODEs (1) represents mass balances of the perfusion reactor. This generic perfusion reactor model is completed for a specific reaction system by providing the kinetic and stoichiometric information, r and S . In addition, to conduct a simulation with the model, we must provide the operational information, C_{in} , C_{per} , and open- or closed-loop profiles for the flow rates (F_{in} , F_{per} , and F_{out}) and the temperature (T).

2.2. Stoichiometry and kinetics for mAb fermentation

The reactor contains five modeled components,

$$C = \{X_v, X_d, G, L, P\}, \quad (3)$$

where X_v are viable cells, X_d are dead cells, G is glucose, L is lactate, and P is the product (mAb). The process consists of six reactions,

$$\mathcal{R} = \{1, 2, 3, 4, 5, 6\}. \quad (4)$$

Table 1 presents the six reactions, which are represented by the stoichiometric matrix,

$$S = \begin{bmatrix} X_v & X_d & G & L & P \\ 1 & 0 & -\alpha_{1,G} & 0 & \alpha_{1,P} \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_{3,G} & 0 & \alpha_{3,P} \\ 0 & 0 & 0 & \alpha_{4,L} & 0 \\ 0 & 0 & 0 & \alpha_{5,L} & 0 \\ 0 & 0 & 0 & \alpha_{6,L} & 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix}. \quad (5)$$

The reaction rates are given as

$$r_1 = \mu_X(c, T)c_{X_v}, \quad r_2 = \mu_D(T)c_{X_v}, \quad (6a)$$

$$r_3 = \mu_{m_1}c_{X_v}, \quad r_4 = \mu_{m_2}c_{X_v}, \quad (6b)$$

$$r_5 = \mu_{L,p_1}(c, T)c_{X_v}, \quad r_6 = \mu_{L,p_2}(c)c_{X_v}. \quad (6c)$$

We define a sigmoid function, s_γ , and a smooth maximum approximation function, \max_α , applied in the specific rate

functions,

$$s_\gamma(c_G, \bar{c}_G) = \frac{1}{1 + \exp(-\gamma(c_G - \bar{c}_G))}, \quad (7a)$$

$$\max_\alpha(x_1, x_2) = \frac{x_1 + x_2 + \sqrt{(x_1 - x_2)^2 + \alpha^2}}{2}. \quad (7b)$$

\bar{c}_G is the value of half activation in the sigmoid function, γ determines the smoothness of the sigmoid function, and α determines the smoothness of the maximum approximation. The specific growth rate, μ_X , is

$$\mu_X = \mu_{X,\max} f_{lim} f_{L,inh} f_{G,inh} f_{P,inh} f_{temp}, \quad (8)$$

where the limiting growth term, f_{lim} , depends on the glucose concentration and the cell density, $f_{L,inh}$ is a lactate inhibition term, $f_{G,inh}$ is a glucose inhibition term, $f_{P,inh}$ is a product inhibition term, and f_{temp} models the temperature dependency of the specific growth rate. The terms are

$$f_{lim} = \frac{c_G}{K_G c_{X_v} + c_G}, \quad (9a)$$

$$f_{L,inh} = \frac{K_{I,L}}{K_{I,L} + c_L}, \quad (9b)$$

$$f_{G,inh} = 1 - s_\gamma(c_G, \bar{c}_G), \quad (9c)$$

$$f_{P,inh} = \max_\alpha(0, 1 - K_{I,P}c_P), \quad (9d)$$

$$f_{temp} = \exp\left(-\frac{K_1}{T}\right). \quad (9e)$$

The specific death rate, μ_D , is

$$\mu_D = \mu_{D,\max} f_{D,temp}, \quad (10)$$

where $f_{D,temp}$ is the specific death rate dependency on the temperature given as

$$f_{D,temp} = \exp\left(-\frac{K_2}{T}\right). \quad (11)$$

The remaining specific rate functions are

$$\mu_{m_1} = \bar{\mu}_{m_1}, \quad (12a)$$

$$\mu_{m_2} = \bar{\mu}_{m_2} \frac{L_{\max,2} - c_L}{L_{\max,2}}, \quad (12b)$$

$$\mu_{L,p_1} = \mu_X \frac{L_{\max,1} - c_L}{L_{\max,1}}, \quad (12c)$$

$$\mu_{L,p_2} = \bar{\mu}_{L,p_2} \frac{L_{\max,1} - c_L}{L_{\max,1}}. \quad (12d)$$

The rate functions (12) are the result of cell metabolism considerations [34].

Remark 1. In previous work [45], we applied the following maximum approximation,

$$\max_{\alpha}(x_1, \dots, x_n) = \frac{\sum_{i=1}^n x_i \exp(\alpha x_i)}{\sum_{i=1}^n \exp(\alpha x_i)}. \quad (13)$$

However, we observed that the approximation lead to slight negativity of $\max_{\alpha}(0, 1 - K_{I,P}c_P)$, when $1 - K_{I,P}c_P$ approaches 0. Consequently, we apply (7b) in this work rather than (13) as in previous work.

2.3. Operation of the perfusion reactor

We operate the perfusion reactor with two inlet flow streams, a perfusion stream, and a single outlet stream. Additionally, the perfusion reactor temperature, T , can be regulated between a lower and an upper bound. We let S_W denote a pure water inlet stream and S_G denote a glucose inlet stream. Thus, we define the set of inlet streams as

$$S = \{S_W, S_G\}, \quad (14)$$

and define the inlet stream flow rate vector as

$$F_{\text{in}} = \begin{bmatrix} F_W \\ F_G \end{bmatrix}. \quad (15)$$

The water stream contains pure water and the glucose stream contains dissolved glucose with concentration, $c_{G,\text{in}}$. Accordingly, the inlet concentration matrix is

$$C_{\text{in}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & c_{G,\text{in}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}. \quad (16)$$

The perfusion stream only removes spend media, i.e., glucose and lactate. We assume complete filtration of cells and product, and no filtration of glucose and lactate. Therefore, the perfusion matrix is

$$C_{\text{per}} = \text{diag}([0 \ 0 \ 1 \ 1 \ 0]). \quad (17)$$

The two inlet flow rates, F_W and F_G , the perfusion flow rate, F_{per} , the outlet flow rate, F_{out} , and the temperature of the reactor, T , are considered as manipulated variables.

2.4. Stochastic continuous-discrete system

We express stochastic variations in the reactor model with a diffusion term. The result is an SDE model for the reactor. We formulate the system as a stochastic continuous-discrete system in the form

$$x(t_0) = x_0, \quad (18a)$$

$$dx(t) = \underbrace{f(t, x(t), u(t), p)}_{\text{drift term}} dt + \underbrace{\sigma(t, x(t), u(t), p)}_{\text{diffusion term}} d\omega(t), \quad (18b)$$

$$y(t_i) = g(t_i, x(t_i), p) + v(t_i). \quad (18c)$$

$x \in \mathbb{R}^{n_x}$ are the states, $x_0 \in \mathbb{R}^{n_x}$ is the initial condition, $u \in \mathbb{R}^{n_u}$ are the manipulated variables, $p \in \mathbb{R}^{n_p}$ are the parameters, $\sigma \in \mathbb{R}^{n_x \times n_w}$ is the diffusion matrix function, $\omega(t) \in \mathbb{R}^{n_w}$ is a standard Wiener process, i.e., $d\omega(t) \sim N_{iid}(0, Idt)$, $y_i = y(t_i) \in \mathbb{R}^{n_y}$ are measurements at discrete times, and $v_i = v(t_i) \sim N_{iid}(0, R_v(t_i))$ is measurement noise. $R_v(t_i)$ is the covariance matrix for the measurement noise.

The drift function, $f(\cdot)$, is the right hand side of the ODE model (1) and the diffusion function, $\sigma(\cdot)$, models the effect of random variations on the states. The SDE formulation of the reactor model is

$$dV(t) = (e^{\top} F_{\text{in}} - F_{\text{out}} - F_{\text{per}})dt + \sigma_V d\omega_V(t), \quad (19a)$$

$$dm(t) = (C_{\text{in}} F_{\text{in}} - c F_{\text{out}} - C_{\text{per}} c F_{\text{per}} + RV)dt + \sigma_m d\omega_m(t). \quad (19b)$$

$\omega_V(t)$ and $\omega_m(t)$ are standard Wiener processes such that $d\omega_V(t) \sim N_{iid}(0, dt)$ and $d\omega_m(t) \sim N_{iid}(0, Idt)$. Therefore, the states and manipulated variables for the mAb fermentation process are

$$x = \begin{bmatrix} V \\ m_{X_v} \\ m_{X_d} \\ m_G \\ m_L \\ m_P \end{bmatrix}, \quad u = \begin{bmatrix} F_W \\ F_G \\ F_{\text{per}} \\ F_{\text{out}} \\ T \end{bmatrix}. \quad (20)$$

The units are as follows: V [L], m_{X_v} [cells $\times 10^9$], m_{X_d} [cells $\times 10^9$], m_G [g], m_L [g], m_P [g], F_W [L/min], F_G [L/min], F_{per} [L/min], F_{out} [L/min], and T [K].

We assume that the volume, V , the glucose concentration, c_G , and the lactate concentration, c_L , are measured with sampling time T_s , i.e., the measurement vector is

$$y = \begin{bmatrix} V \\ c_G \\ c_L \end{bmatrix} = \begin{bmatrix} V \\ m_G/V \\ m_L/V \end{bmatrix}, \quad (21)$$

with diagonal measurement covariance, R_v , given as

$$R_v = \text{diag} \left(\begin{bmatrix} \sigma_{y,V}^2 & \sigma_{y,G}^2 & \sigma_{y,L}^2 \end{bmatrix} \right). \quad (22)$$

Similarly, we assume independent state diffusion for the viable cells, m_{X_v} , the glucose mass, m_G , and the lactate mass,

Table 2

Overview of the model equations for the deterministic and stochastic perfusion reactor and the mAb kinetics and stoichiometry.

ODE and SDE model for perfusion reactor	
ODE	$\frac{dV}{dt} = e^T F_{in} - F_{out} - F_{per}$
	$\frac{dm}{dt} = C_{in} F_{in} - c F_{out} - C_{per} c F_{per} + RV$
SDE	$dV(t) = (e^T F_{in} - F_{out} - F_{per})dt + \sigma_V d\omega_V(t)$
	$dm(t) = (C_{in} F_{in} - c F_{out} - C_{per} c F_{per} + RV)dt + \sigma_m d\omega_m(t)$
Production rate	$R = S^T r$
Kinetics and stoichiometry for monoclonal antibody fermentation	
States	$x = [V \quad m_{X_v} \quad m_{X_d} \quad m_G \quad m_L \quad m_P]^T$
Stoichiometry	$S = \begin{bmatrix} 1 & 0 & -\alpha_{1,G} & 0 & \alpha_{1,P} \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_{3,G} & 0 & \alpha_{3,P} \\ 0 & 0 & 0 & \alpha_{4,L} & 0 \\ 0 & 0 & 0 & \alpha_{5,L} & 0 \\ 0 & 0 & 0 & \alpha_{6,L} & 0 \end{bmatrix}$
Kinetics	$r = [\mu_X(c, T)c_{X_v} \quad \mu_D(T)c_{X_v} \quad \mu_{m_1}c_{X_v} \quad \mu_{m_2}c_{X_v} \quad \mu_{L,p_1}(c, T)c_{X_v} \quad \mu_{L,p_2}(c)c_{X_v}]^T$
Operation	$u = [F_W \quad F_G \quad F_{per} \quad F_{out} \quad T]^T$
	$C_{in} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_{G,in} & 0 & 0 \end{bmatrix}^T$
	$C_{per} = \text{diag}([0 \quad 0 \quad 1 \quad 1 \quad 0])$
Diffusion	$\sigma = \begin{bmatrix} 0 & \sigma_{X_v} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_G & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_L & 0 \end{bmatrix}^T$

m_L , resulting in the following diffusion matrix

$$\sigma = \begin{bmatrix} 0 & 0 & 0 \\ \sigma_{X_v} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \sigma_G & 0 \\ 0 & 0 & \sigma_L \\ 0 & 0 & 0 \end{bmatrix}, \quad (23)$$

where

$$\sigma_{X_v} = \bar{\sigma}_{X_v} c_{X_v}, \quad \sigma_G = \bar{\sigma}_G c_G, \quad \sigma_L = \bar{\sigma}_L c_L, \quad (24)$$

with $\bar{\sigma}_{X_v}$, $\bar{\sigma}_G$, and $\bar{\sigma}_L$ being parameters for estimation.

2.5. Model overview

Table 2 presents an overview of the complete model for the perfusion reactor. The model consists of mass balances expressed as ODEs and SDEs, stoichiometry, and a kinetic model.

3. Economic nonlinear model predictive controller

We develop an ENMPC for closed-loop control of the mAb fermentation process [27, 28]. The ENMPC consists

of a CD-EKF for state estimation [36, 44, 54, 59, 60] and a regulator expressed as an economic optimal control problem (OCP) based on mAb and glucose prices.

3.1. State estimator

At time t_i , the CD-EKF receives the measurement vector, y_i , the previous actual vector of manipulated variables, u_{i-1} , and the previous mean-covariance pair, $(\hat{x}_{i-1|i-1}, P_{i-1|i-1})$. The CD-EKF consists of a one-step prediction step and a filtering step. The result of the CD-EKF is the filtered state estimate, $\hat{x}_{i|i}$, and its covariance, $P_{i|i}$. In this way, we use the actually implemented u_{i-1} for computation of the one-step prediction, $\hat{x}_{i|i-1}$, and the filtered state, $\hat{x}_{i|i}$. In the first iteration, we specify $\hat{x}_{-1|-1}$, $P_{-1|-1}$, and u_{-1} .

3.1.1. Prediction

Given the previous mean-covariance pair, $\hat{x}_{i-1|i-1}$ and $P_{i-1|i-1}$, the CD-EKF computes the state-covariance one-step prediction,

$$\hat{x}_{i|i-1} = \hat{x}_{i-1}(t_i), \quad P_{i|i-1} = P_{i-1}(t_i). \quad (25)$$

The one-step prediction is obtained by numerical solution of

$$\frac{d}{dt} \hat{x}_{i-1}(t) = f(t, \hat{x}_{i-1}(t), u_{i-1}, p), \quad (26a)$$

$$\frac{d}{dt} P_{i-1}(t) = A_{i-1}(t)P_{i-1}(t) + P_{i-1}(t)A_{i-1}(t)^\top + \sigma_{i-1}(t)\sigma_{i-1}(t)^\top, \quad (26b)$$

for $t_{i-1} \leq t \leq t_i$ and

$$A_{i-1}(t) = \frac{\partial}{\partial x} f(t, \hat{x}_{i-1}(t), u_{i-1}, p), \quad (27a)$$

$$\sigma_{i-1}(t) = \sigma(t, \hat{x}_{i-1}(t), u_{i-1}, p), \quad (27b)$$

with initial condition,

$$\hat{x}_{i-1}(t_{i-1}) = \hat{x}_{i-1|i-1}, \quad P_{i-1}(t_{i-1}) = P_{i-1|i-1}. \quad (28a)$$

3.1.2. Filtering

From the measurement, y_i , the CD-EKF computes the filtered state estimate, $\hat{x}_{i|i}$, as

$$\hat{y}_{i|i-1} = g(\hat{x}_{i|i-1}, p), \quad C_i = \frac{\partial}{\partial x} g(\hat{x}_{i|i-1}, p), \quad (29a)$$

$$e_i = y_i - \hat{y}_{i|i-1}, \quad R_{e,i} = R_i + C_i P_{i|i-1} C_i^\top, \quad (29b)$$

$$\hat{x}_{i|i} = \hat{x}_{i|i-1} + K_i e_i, \quad K_i = P_{i|i-1} C_i^\top R_{e,i}^{-1}, \quad (29c)$$

where $R_i = R_v(t_i)$ is the covariance matrix for the measurement noise. The filtered covariance estimate, $P_{i|i}$, is then computed as

$$P_{i|i} = P_{i|i-1} - K_i R_{e,i} K_i^\top \quad (30a)$$

$$= (I - K_i C_i) P_{i|i-1} (I - K_i C_i)^\top + K_i R_i K_i^\top, \quad (30b)$$

where (30b) is the Joseph stabilizing form [61]. The form (30b) is preferred, since it ensures symmetry and positive semi-definiteness of the filtered covariance.

3.2. Regulator

The ENMPC solves the regulator OCP at time t_i using the filtered state estimate, $\hat{x}_{i|i}$, computed by the CD-EKF. The solution of the OCP is the optimal predicted state and input trajectories in a finite horizon. The manipulated variables, $u(t)$, corresponding to the first period, $t \in [t_i; t_i + T_s]$, is implemented. The ENMPC resolves the OCP once the estimator provides the next state estimate. We denote the control and prediction horizon, T_h , and let $t_{f,i} = t_i + T_h$. The regulator OCP is,

$$\min_{[u(t); x(t)]_{t_i}^{t_{f,i}}} \varphi_i = \varphi_{P,i} + \varphi_{G,i}, \quad (31a)$$

$$\text{s.t.} \quad x(t_i) = \hat{x}_{i|i}, \quad (31b)$$

$$\dot{x}(t) = f(t, x(t), u(t), p), \quad t_i \leq t \leq t_{f,i}, \quad (31c)$$

$$x_{\min} \leq x(t) \leq x_{\max}, \quad t_i \leq t \leq t_{f,i}, \quad (31d)$$

$$u_{\min} \leq u(t) \leq u_{\max}, \quad t_i \leq t \leq t_{f,i}. \quad (31e)$$

The terms in the economic objective function are

$$\varphi_{P,i} = -P_P (m_P(t_{f,i}) - m_P(t_i)), \quad (32)$$

$$\varphi_{G,i} = P_G \int_{t_i}^{t_{f,i}} c_{G,\text{in}} F_G(t) dt, \quad (33)$$

where P_P [USD/g] is the price of product (mAb) and P_G [USD/g] is the price of glucose. The state and input bound constraints are given as,

$$x_{\min} = \begin{bmatrix} V_{\min} \\ m_{X_v, \min} \\ m_{X_d, \min} \\ m_{G, \min} \\ m_{L, \min} \\ m_{P, \min} \end{bmatrix}, \quad x_{\max} = \begin{bmatrix} V_{\max} \\ m_{X_v, \max} \\ m_{X_d, \max} \\ m_{G, \max} \\ m_{L, \max} \\ m_{P, \max} \end{bmatrix}, \quad (34a)$$

$$u_{\min} = \begin{bmatrix} F_{W, \min} \\ F_{G, \min} \\ F_{\text{per}, \min} \\ F_{\text{out}, \min} \\ T_{\min} \end{bmatrix}, \quad u_{\max} = \begin{bmatrix} F_{W, \max} \\ F_{G, \max} \\ F_{\text{per}, \max} \\ F_{\text{out}, \max} \\ T_{\max} \end{bmatrix}. \quad (34b)$$

We split the horizon in N equidistant control intervals, i.e., $T_h = NT_s$, and assume zero-order-hold parameterization of the manipulated variables,

$$u(t) = u_k, \quad k = 0, \dots, N-1. \quad (35)$$

We discretize the OCP (31) with a direct multiple shooting approach, where we apply an explicit classical Runge-Kutta numerical scheme with N_c steps for state integration [62]. To solve the discretized OCP, we apply a Riccati recursion based sequential quadratic programming (SQP) algorithm implemented in C. The algorithm is thread-safe. Consequently, it can be applied efficiently in parallel Monte Carlo simulations [44, 63–66]. Thread-safety of the SQP algorithm ensures that the algorithm can be called in parallel to solve multiple problems with scaling as observed in previous work [36, 44]. Lack of thread-safety would result in poor parallel scaling and make parallel Monte Carlo simulations using the SQP algorithm computationally inefficient (slow). We ensure thread-safety by avoiding any parallelism inside the SQP algorithm and by careful distribution of memory allocated prior to calling the optimization software. Additionally, the SQP algorithm is BLAS dependent, and we link to BLASFEO that is a thread-safe BLAS library [67, 68].

4. Closed-loop simulation procedure

In this section, we describe the procedure for closed-loop simulation. We consider a sample time, T_s , a simulation period, $T_{\text{sim}} = N_{\text{sim}} T_s$, the initial time, t_0 , and the final time, $t_f = t_0 + T_{\text{sim}}$. In each control interval, $[t_i, t_{i+1} = t_i + T_s]$ with $i = 0, 1, \dots, N_{\text{sim}} - 1$, we apply the Euler-Maruyama method with N_s steps of equal size, Δt , for integration of the SDE system (18b) [69]. The discretized stochastic continuous-discrete system (18) is written as

$$x_{i+1} = \Phi(t_i, x_i, u_i, w_i, p), \quad (36a)$$

$$y_i = g(t_i, x_i, p) + v_i. \quad (36b)$$

$x_i = x(t_i)$, $u_i = u(t_i)$, $v_i = v(t_i)$, and the process noise, w_i , is N_s realizations of $\Delta\omega_i \sim N_{\text{iid}}(0, I\Delta t)$. The operator, $\Phi(t_i, x_i, u_i, w_i, p)$, is defined by the procedure:

1. Let $w_i = [\Delta\omega_{i,0}; \Delta\omega_{i,1}; \dots; \Delta\omega_{i,N_s-1}]$, where $\Delta\omega_{i,k} \sim N_{iid}(0, I\Delta t)$ for $k = 0, 1, \dots, N_s - 1$.
2. Set the initial conditions:

$$t_{i,0} = t_i, \quad (37a)$$

$$x_{i,0} = x_i. \quad (37b)$$

3. Compute the next states using the Euler-Maruyama method for $k = 0, 1, \dots, N_s - 1$:

$$t_{i,k+1} = t_{i,k} + \Delta t, \quad (38a)$$

$$x_{i,k+1} = x_{i,k} + f(t_{i,k}, x_{i,k}, u_i, p)\Delta t + \sigma(t_{i,k}, x_{i,k}, u_i, p)\Delta\omega_{i,k}. \quad (38b)$$

4. Compute the next time, t_{i+1} , and state, x_{i+1} , by

$$t_{i+1} = t_{i,N_s}, \quad (39a)$$

$$x_{i+1} = x_{i,N_s}. \quad (39b)$$

The ENMPC can similarly be expressed as [36]

$$x_{i+1}^c = \kappa(t_i, x_i^c, y_{i+1}, u_i, p_c), \quad (40a)$$

$$u_i = \lambda(t_i, x_i^c, p_c), \quad (40b)$$

where x_i^c are estimated states, p_c are the controller parameters, $\kappa(\cdot)$ is the CD-EKF for state estimation, and $\lambda(\cdot)$ is the regulator. A closed-loop simulation can be written as [36]

$$y_i = g(t_i, x_i, p) + v_i, \quad (\text{Measurement}) \quad (41a)$$

$$x_i^c = \kappa(t_{i-1}, x_{i-1}^c, y_i, u_{i-1}, p_c), \quad (\text{Estimation}) \quad (41b)$$

$$u_i = \lambda(t_i, x_i^c, p_c), \quad (\text{Regulation}) \quad (41c)$$

$$x_{i+1} = \Phi(t_i, x_i, u_i, w_i, p). \quad (\text{Simulation}) \quad (41d)$$

5. Monte Carlo simulation

We apply a parallelized Monte Carlo simulation toolbox implemented in C to quantify uncertainties in the mAb fermentation process [36, 44]. The toolbox is designed for parallel Monte Carlo simulation of closed-loop systems on shared memory architectures. We evaluate the performance of the closed-loop fermentation process by two performance indicators

$$\Phi_1 = m_P(t_f) - m_P(t_0), \quad (42a)$$

$$\Phi_2 = P_P (m_P(t_f) - m_P(t_0)) - P_G \sum_{k=0}^{N_{\text{sim}}-1} c_{G,\text{in}} F_{G,k} T_s, \quad (42b)$$

where Φ_1 is the mAb production of the fermentation and Φ_2 is the profit of the fermentation. We measure the uncertainty of the fermentation by the range computed by

$$\text{Range} = x_{\text{max}} - x_{\text{min}}. \quad (43)$$

x_{max} is the largest value in the data set and x_{min} is the smallest value.

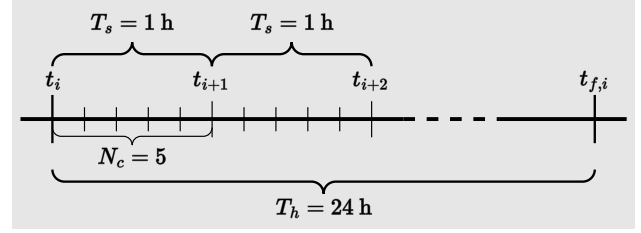


Figure 2: Illustration of the ENMPC horizon and internal steps. The control and prediction horizon is 24 h. The ENMPC applies $N_c = 5$ classical Runge-Kutta steps for state integration in every control interval of size $T_s = 1$ h.

6. Results

This section presents the results of the simulation case study. We present a stochastic closed-loop simulation with the ENMPC and an uncertainty quantification study based on Monte Carlo simulation for the fermentation process. The uncertainty quantification study includes a comparison to a base case strategy [34]. We consider 14 days fermentation in a small-scale perfusion reactor with an upper volume limit of 8 L. We assume that measurements are available with sampling time $T_s = 60$ min, and we apply $N_s = 20$ Euler-Maruyama steps for simulation between samples. Table 3 presents a complete overview of the model parameters, operational parameters, and the simulation parameters for all simulations conducted in this case study. In the Monte Carlo simulation study, we generate 10.000 realizations of the process noise sequence and 1 realization of the measurement noise. These realizations are used to compare the performance of different controllers.

We conduct the simulations using a standard Linux workstation with a 6 core Intel® Xeon® W-2235 CPU with frequency 3.80 GHz.

6.1. Stochastic closed-loop simulations with the ENMPC

The ENMPC has a discrete control and prediction horizon $N = 24$, and it applies $N_c = 5$ classical Runge-Kutta internal steps for state integration in each control interval of size T_s . Figure 2 presents an overview of the ENMPC horizon and internal steps. Since $T_s = 60.0$ min, the control and prediction horizon of the ENMPC is $T_h = 24.0$ h. In the ENMPC objective function, we apply the glucose price $P_G = 5.8 \times 10^{-4}$ USD/g [70] and the price for the mAb, Remicade (infiximab), $P_P = 1.239 \times 10^4$ USD/g [71]. We point out that these prices can be updated without loss of generality. We initialize the CD-EKF state information, $\hat{x}_{-1|-1}$ and $P_{-1|-1}$, using

$$\hat{x}_{-1|-1} = \begin{bmatrix} 5.1150 & 5.5149 & 1.0636 \\ & 36.0652 & 0.3416 & 0.1720 \end{bmatrix}^T, \quad (44a)$$

$$P_{-1|-1} = \text{diag}([1.0; 1.0; 1.0; 10.0; 0.1; 0.1]). \quad (44b)$$

$\hat{x}_{-1|-1}$ is a sample from the distribution $N(x_0, P_{-1,-1})$, where $P_{-1,-1}$ is manually selected. Similarly, we choose

Table 3

Model parameters, operational parameters, and simulation parameters for the mAb fermentation process in closed-loop.

Model parameters			Operational parameters			Simulation parameters		
Parameters	Value	Unit	Parameters	Value	Unit	Parameters	Value	Unit
$\mu_{X,max}$	0.153	[min ⁻¹]	$c_{G,in}$	32.5	[g/L]	t_0	0.0	[min]
$\mu_{D,max}$	$3.955 \cdot 10^{-5}$	[min ⁻¹]	F_{min}	0.0	[L/min]	t_f	20, 160.0	[min]
$\bar{\mu}_{m_1}$	1.0	[min ⁻¹]	F_{max}	0.02	[L/min]	T_s	60.0	[min]
$\bar{\mu}_{m_2}$	1.0	[min ⁻¹]	T_{min}	308.15	[K]	N_s	20	[-]
$\bar{\mu}_{L,p_2}$	1.0	[min ⁻¹]	T_{max}	310.15	[K]	V_0	5.650	[L]
K_1	1689	[K]	V_{min}	4.0	[L]	$m_{X_v,0}$	3.955	[cells×10 ⁹]
K_2	524	[K]	V_{max}	8.0	[L]	$m_{X_d,0}$	0.0	[cells×10 ⁹]
K_G	0.85	[g/(cells×10 ⁹)]	$m_{X_v,min}$	−∞	[g]	$m_{G,0}$	34.18	[g]
$K_{I,L}$	344	[g/L]	$m_{X_v,max}$	∞	[g]	$m_{L,0}$	0.678	[g]
$K_{I,P}$	6.88×10^{-1}	[L/g]	$m_{X_d,min}$	−∞	[g]	$m_{P,0}$	0.0	[g]
$L_{max,1}$	628	[g/L]	$m_{X_d,max}$	∞	[g]	P_P	1.239×10^4	[USD/g]
$L_{max,2}$	0.5	[g/L]	$m_{G,min}$	0.0	[g]	P_G	5.8×10^{-4}	[USD/g]
\bar{c}_G	7.5	[g/L]	$m_{G,max}$	∞	[g]	$\sigma_{y,V}^2$	3.0×10^{-3}	[-]
γ	10.0	[-]	$m_{L,min}$	0.0	[g]	$\sigma_{y,G}^2$	2.0×10^{-2}	[-]
α	0.1	[-]	$m_{L,max}$	∞	[g]	$\sigma_{y,L}^2$	2.0×10^{-3}	[-]
$\alpha_{1,G}$	0.4876	[g/(cells×10 ⁹)]	$m_{P,min}$	−∞	[g]	σ_{X_v}	1.0×10^{-2}	[-]
$\alpha_{1,P}$	6.62×10^{-8}	[g/(cells×10 ⁹)]	$m_{P,max}$	∞	[g]	σ_G	7.0×10^{-3}	[-]
$\alpha_{3,G}$	1.102×10^{-4}	[g/(cells×10 ⁹)]				σ_L	3.0×10^{-2}	[-]
$\alpha_{3,P}$	1.2×10^{-5}	[g/(cells×10 ⁹)]				N	24	[-]
$\alpha_{4,L}$	1.89×10^{-5}	[g/(cells×10 ⁹)]				N_c	5	[-]
$\alpha_{5,L}$	5.504×10^{-1}	[g/(cells×10 ⁹)]				N_{sim}	336	[-]
$\alpha_{6,L}$	1.0249×10^{-5}	[g/(cells×10 ⁹)]						

$u_{-1} = [F_W; F_G; F_{per}; F_{out}; T] = [0; 0; 0; 0; 310.15]$. We use this initialization in all simulations.

Figure 3 presents a single stochastic simulation of the perfusion reactor in closed-loop. The ENMPC operates the reactor in two phases, which we denote the *growth phase* and the *production phase*. The production phase is initialized when the ENMPC reduces the temperature to the lower limit. Throughout the fermentation, the ENMPC operates the perfusion reactor at the upper volume limit with no outlet flow, i.e., $F_{out} = 0$ L/min. In the growth phase, the perfusion flow rate, F_{per} , is kept at its maximum for almost the entire phase. The ENMPC selects a combination of the glucose flow rate, F_G , and the water flow rate, F_W , to keep the volume at the upper limit while maintaining an almost constant glucose concentration. The temperature is kept constant at the upper limit in the growth phase. The production phase is initialized after approximately 10 days when the ENMPC lowers the temperature. Slightly before the production phase, the flow rates, F_W , F_G , and F_{per} , are decreased to a level sufficient to maintain the optimal glucose concentration. This results in an increased lactate concentration. Figure 4 shows the produced mAb over time. We observe that the fermentation produces 24.82 g mAb during the 14 days.

The closed-loop operation of the 14 days fermentation requires 336 calls to the ENMPC. Figure 5 presents the CPU time for each of the 336 ENMPC calls. We observe that the worst case CPU time is approximately 0.12 s, which makes

the computation time negligible compared to the sampling time of 60.0 min for real-time applications. However, the CPU time is important because large-scale Monte Carlo simulation studies consist of thousands of simulations that are only tractable if the the controller computations are sufficiently fast and efficient.

6.2. Productivity analysis

We analyze the productivity of the mAb fermentation process. The production of mAb is directly related to the production of cells due to (5) and (6). Therefore, the governing reaction rate for mAb production is $r_1 = r_1(c_{X_v}, c_G, c_L, c_P, T)$, which is a function of the viable cell density, c_{X_v} , the glucose concentration, c_G , the lactate concentration, c_L , the product concentration, c_P , and the temperature, T . Figure 6 shows the growth rate, r_1 , for the simulation with the ENMPC. We observe that the ENMPC initializes the production phase when the growth rate is almost zeroed.

According to (8) and (9), the cell productivity is maximized for low c_L and c_P , and for high T . Additionally, there exist a maximizing trade-off between c_{X_v} and c_G . We analyze this trade-off by considering the c_{X_v} and c_G dependent terms of r_1 . We define the function

$$f(c_{X_v}, c_G) = f_{lim} f_{G,inh} c_{X_v}. \quad (45)$$

Figure 7 shows a heatmap of $f(c_{X_v}, c_G)$. We observe that there exists an optimal glucose trajectory to maximize

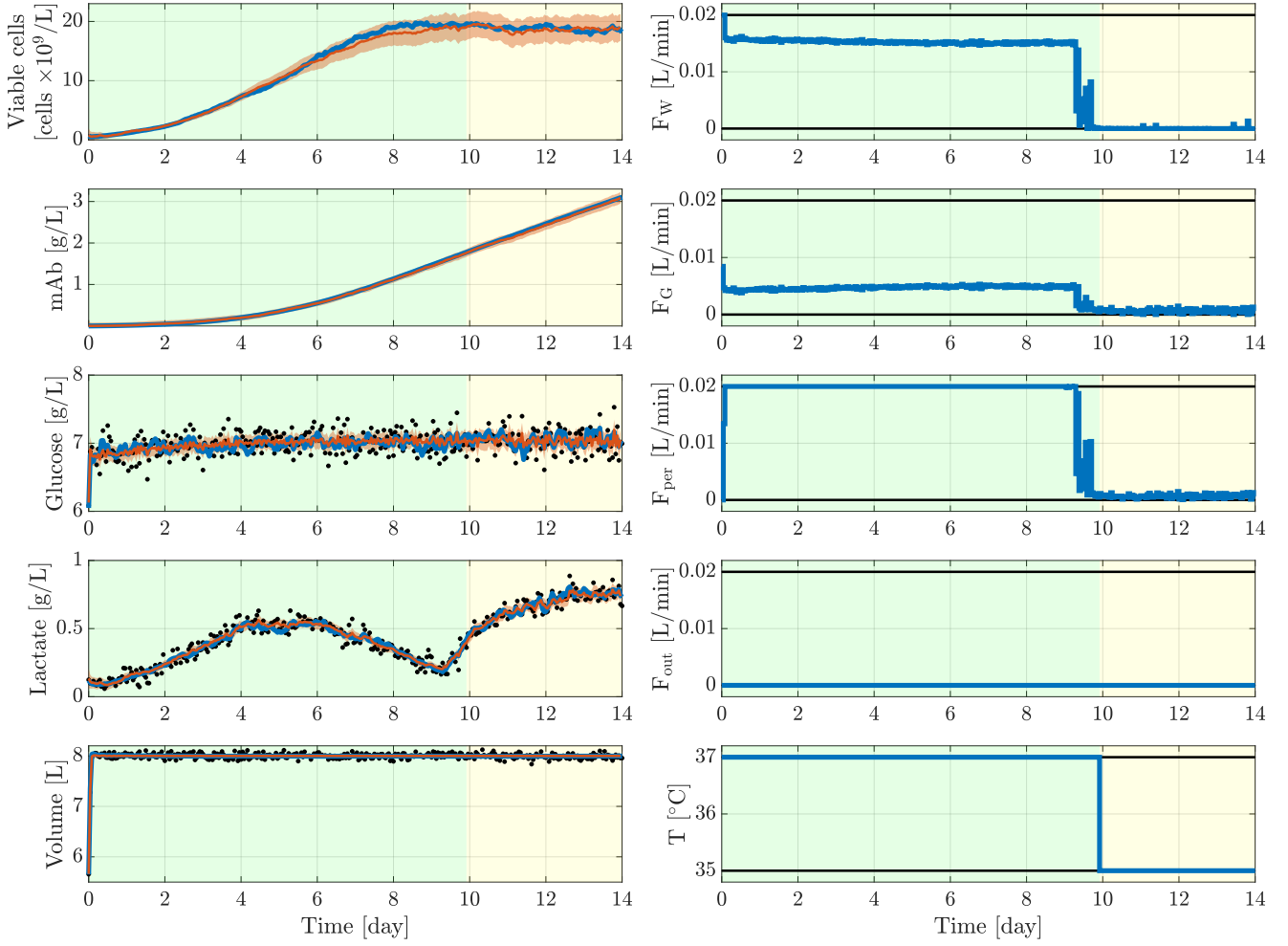


Figure 3: A single closed-loop simulation with the ENMPC. The blue lines are the simulation of the perfusion reactor, the red lines are estimates from the CD-EKF together with the red shaded 95% confidence interval for the estimates, and the black dots are measurements. We observe that the ENMPC operates the reactor in two phases: Growth phase (shaded green) and production phase (shaded yellow). The production phase is initialized after approximately 10 days, where the controller decreases the temperature.

$f(c_G, c_{X_p})$ depending on the viable cell density, and that the ENMPC tracks the optimal glucose trajectory throughout the fermentation.

Figure 3, Figure 6, and Figure 7 provide information about the operational strategy computed by the ENMPC. In the growth phase, the ENMPC maintains a high volume to lower the lactate and mAb concentration. A high perfusion flow rate also contributes to a low lactate concentration. Additionally, the ENMPC tracks the optimal glucose concentration based on the current viable cell density. In the production phase, the ENMPC decreases the temperature to decrease cell death. Since the growth rate, r_1 , is zeroed by product inhibition, the ENMPC lowers F_G , F_W , and F_{per} to reduce the glucose consumption.

6.3. Uncertainty quantification for mAb production

We perform an uncertainty quantification study for mAb fermentation conducted in the perfusion reactor and compare the ENMPC results to a base case open-loop strategy [34]. The uncertainty quantification study is based on Monte Carlo simulations using 10.000 fermentations for each of the operating strategies, i.e. ENMPC and the base case.

Each of the 10.000 fermentations has different realization of the process noise sequence, i.e., different standard Wiener process realizations for the process noise, $d\omega(t) \sim N_{iid}(0, Idt)$.

Figure 8 presents distributions for Φ_1 and Φ_2 , (42), based on 10.000 Monte Carlo simulations. We observe that the distributions for the mAb production and the profit are very similar because the glucose price is much smaller than the price of mAb. Table 4 presents statistical data for

Table 4

Statistical data for the mAb production and the profit based on 10.000 closed-loop Monte Carlo simulations. The profit numbers (except percentages) are $\times 10^4$. The increase in percentage is relative to the base case, i.e., $(x_{\text{ENMPC}} - x_{\text{base case}})/x_{\text{base case}}$.

mAb production												
	Mean		min		max		Range		Std		95% CI	
ENMPC	23.89	[g]	18.83	[g]	31.13	[g]	12.29	[g]	1.59	[g]	[20.78, 27.04]	[g]
Base case	15.68	[g]	12.69	[g]	20.56	[g]	7.87	[g]	0.99	[g]	[13.75, 17.61]	[g]
Increase	52	[%]	48	[%]	51	[%]	56	[%]	62	[%]	[51, 53]	[%]

Profit $\times 10^4$												
	Mean		min		max		Range		Std		95% CI	
ENMPC	10.18	[USD]	8.02	[USD]	13.26	[USD]	5.24	[USD]	0.68	[USD]	[8.85, 11.51]	[USD]
Base case	6.68	[USD]	5.41	[USD]	8.76	[USD]	3.35	[USD]	0.42	[USD]	[5.86, 7.50]	[USD]
Increase	52	[%]	48	[%]	51	[%]	56	[%]	62	[%]	[51, 53]	[%]

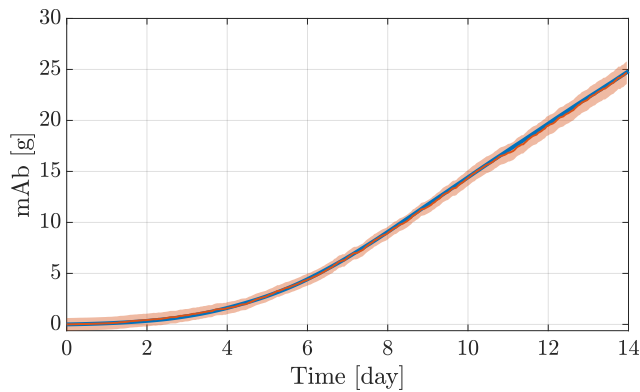


Figure 4: The mAb production over time for one closed-loop simulation sample. The blue line is the simulation, the red line is the CD-EKF estimate, and the red shaded area is a 95% confidence interval for the CD-EKF estimation. The perfusion reactor produces 24.82 g mAb in 14 days.

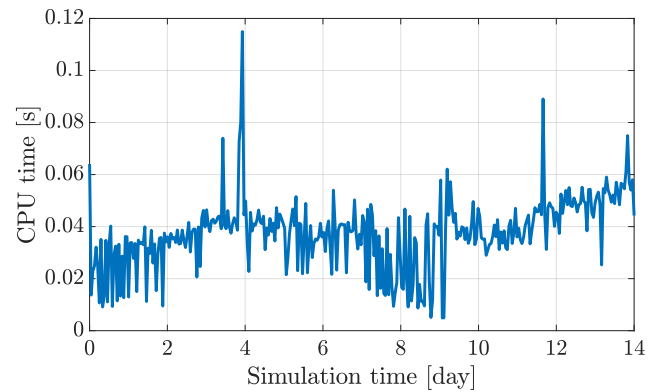


Figure 5: CPU time for ENMPC calls in one closed-loop simulation sample.

the uncertainty quantification study. We observe that the ENMPC increases the mean mAb production and profit by 52% at the cost of a 56% range (uncertainty) increase. We expect the uncertainty increase to be a result of the ENMPC operating the perfusion reactor close to the point of glucose inhibition. We observe no overlap in the 95% confidence intervals indicating that there are a high statistical probability for the ENMPC to produce more mAb than the base case operational strategy. We point out that the fermentation has multiple economic factors that are not taken into account in the current ENMPC, e.g., the price of other components than glucose in the feed medium. Figure 9 shows the CPU time distribution for all calls to the ENMPC during the 10.000 Monte Carlo simulation, i.e., $336 \cdot 10.000 = 3.36 \cdot 10^6$ calls. We observe that 0.1% of the CPU times are above 0.35 s and that the worst case CPU time in 10.000 simulations is 3.2 s.

Figure 10 shows the reaction rates r_1 and r_3 together with the product production rate, R_p , and the produced mAb over

time for the 10.000 Monte Carlo simulations. We observe a period between approximately day 9 and day 11, where some simulations are in the growth phase and some are in the production phase. This is explained by r_1 approaching 0 at different points in time. In almost all simulations, r_1 is almost zeroed after 10 days. We observe uncertainties in the production rate, R_p , which results in mAb uncertainties in the end of the fermentation as observed in Figure 8.

Figure 11 presents distributions for the mAb production with three different ENMPC horizons, $N = 12$, $N = 24$, and $N = 48$. We observe that the horizon does not have a notable impact on the mAb production. This is explained by the optimal operation strategy, computed by the ENMPC, following a few simple operational insights. In the next section, we elaborate on these insights.

6.4. Key insights

The case study with ENMPC provides a number of key insights for optimal operation of the perfusion reactor. Table 5 provides an overview of the insights, where we point out that the optimal glucose concentration significantly depends

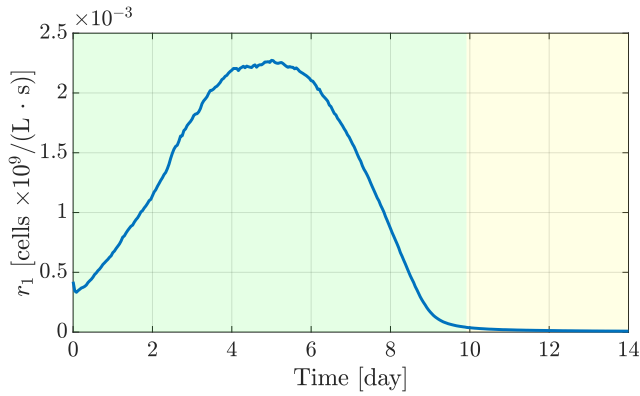


Figure 6: The growth rate, r_1 , for one ENMPC closed-loop simulation sample. The shaded green area shows the growth phase and the shaded yellow area shows the production phase. We observe that the ENMPC initializes the production phase when the growth rate is almost zero.

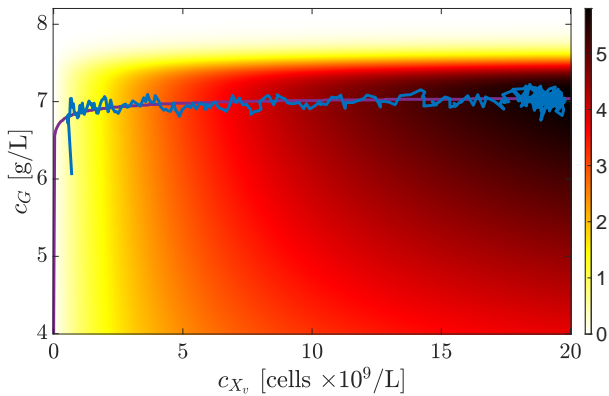


Figure 7: Heatmap of $f(c_{x_v}, c_G) = f_{lim} f_{G,inh} c_{x_v}$. The purple line is the (c_{x_v}, c_G) -trajectory that maximizes $f(c_{x_v}, c_G)$. The blue line is one ENMPC closed-loop simulation sample. We observe that the ENMPC attempts to track the optimal c_G curve as c_{x_v} increases throughout the fermentation.

on \bar{c}_G and γ in the glucose inhibition term (9c). The observed key insights have similarities to those observed by Srinivasan et al. [72]. We apply the ‘from-simple-via-complex-to-lucid’ approach and use the key insights to develop a simple controller design [58]. We assume that the production phase is always initialized after 10 days, which is motivated by Figure 10. A PI controller selects the glucose inlet flow rate, F_G , to track $c_G = 7.0$ g/L. In the growth phase, we select $F_W = F_{per} - F_G$ such that the volume is kept constant, and similarly in the production phase, we select $F_{per} = F_G$ to ensure a constant volume. Alternatively, a PI controller could compute F_W and F_{per} to reduce possible variations in the volume.

We apply the Monte Carlo simulation framework to quantify the performance of the simple controller. Figure 12 presents distributions for the mAb production in closed-loop with the ENMPC and the simple controller. We observe that the two controllers have practically identical performance.

Table 5

Manipulated variables key insights for optimal operation of the perfusion reactor.

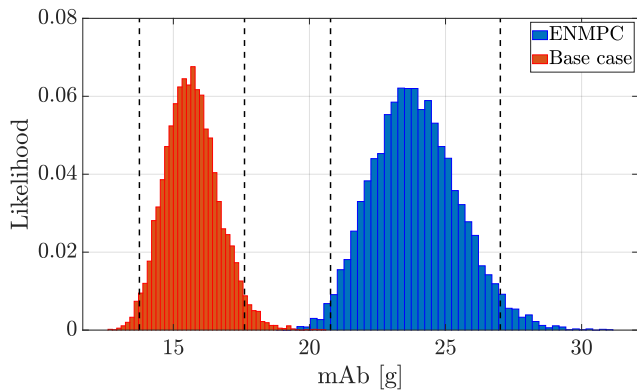
Input	Growth phase	Production phase
F_W	Selected based on F_G and F_{per} such that the volume is constant at the upper limit.	Set to zero.
F_G	Selected such that the glucose concentration, c_G , is almost constant at 7.0 g/L.	Same operation as in the growth phase.
F_{per}	Fixed at the upper limit.	Selected equal to F_G such that the volume is constant at the upper limit.
F_{out}	Set to zero.	Set to zero.
T	Fixed at the upper limit.	Fixed at the lower limit.

However, the computation time on 6 cores for the two uncertainty quantification studies are approximately 6.0 h for the ENMPC and approximately 1.0 s for the simple controller. To point out the importance of parallel computing for the Monte Carlo simulation study, we mention that an identical Monte Carlo simulation study with ENMPC took approximately 11.9 h on 3 cores.

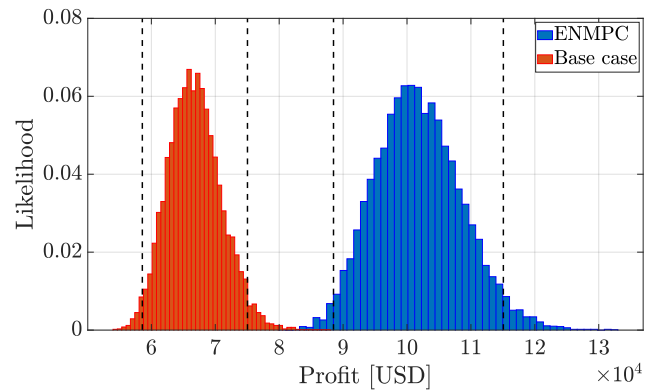
7. Conclusion

This paper presented a simulation case study for production of monoclonal antibodies (mAbs) conducted in a perfusion reactor. We considered an existing mechanistic model for the mAb fermentation process. The model consisted of six ordinary differential equations (ODEs) for the non-constant volume and five components in the reactor. We extended the model with a diffusion term to model stochastic variations, which resulted in a stochastic differential equation (SDE) model. We developed an economic nonlinear model predictive controller (ENMPC) for optimal operation of the reactor. The ENMPC consisted of a continuous-discrete extended Kalman filter (CD-EKF) and an economic regulator based on mAb and glucose prices. We conducted a simulation case study showing that the ENMPC operates the perfusion reactor in two phases, which we denoted the *growth phase* and the *production phase*. The production phase was initialized when product inhibition zeroed the cell growth rate and the ENMPC reduced the temperature to the lower limit. The worst case ENMPC computation time was approximately 0.12 s, which is negligible compared to the sampling time of 60.0 min in real-time application. The low computation time is important for computational tractability of large-scale Monte Carlo simulation studies. We presented a Monte Carlo simulation study based on 10,000 closed-loop simulations, which was conducted in approximately 6.0 h using a standard Linux workstation. This performance is

<Uncertainty Quantification for mAb Production>



(a) Distribution of mAb production for the ENMPC and the base case. The mean production is 23.89 g for the ENMPC and 15.68 g for the base case.



(b) Distribution of profit for the ENMPC and the base case. The mean profit is 10.2×10^4 USD for the ENMPC and 6.68×10^4 USD for the base case.

Figure 8: Distributions for two performance measures, (42), based on 10,000 Monte Carlo simulations of the ENMPC closed-loop and the base case open-loop. The black dashed lines show 95% confidence intervals.

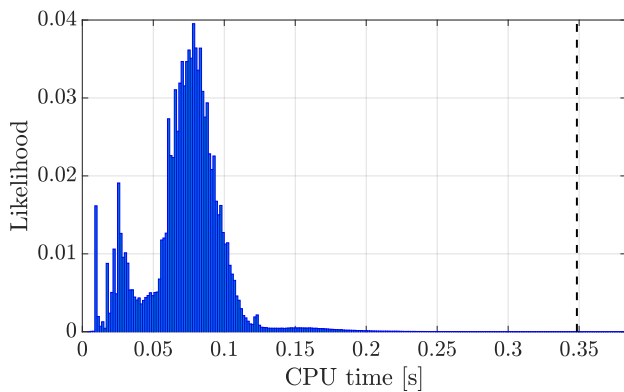


Figure 9: Distribution of CPU time for ENMPC calls in 10,000 Monte Carlo simulations. The black dashed line shows the 99.9% quantile, i.e., 99.9% of the CPU times are below 0.35 s. The worst case CPU time is 3.2 s.

due to high-performance parallel computing and advances in scientific computing. The results showed that the fermentation process, operated by the ENMPC, has a mean mAb production of 23.89 g, a range of 12.29 g between the highest and lowest mAb production, and a 95% confidence interval [20.78, 27.04] g. Compared to a base case strategy, the ENMPC had a 52% increased mean production at the cost of a 56% range (uncertainty) increase. The uncertainty increase is likely due to the ENMPC operating the reactor close to a point of glucose inhibition. We observed no overlap between the 95% confidence intervals for the ENMPC and the base case. Therefore, there is a high statistical probability that the ENMPC has a larger mAb production compared to the base case. The results provide valuable uncertainty quantification of the mAb production, which can be obtained prior to experiments. Finally, we applied the *'from-simple-via-complex-to-lucid'* approach to design a simple controller

from key insights obtained from the ENMPC. An uncertainty quantification study showed that the simple controller and the ENMPC had practically identical performance.

References

- [1] G. Köhler, C. Milstein, Continuous cultures of fused cells secreting antibody of predefined specificity, *Nature* 256 (1975) 495–497.
- [2] D. M. Ecker, S. D. Jones, H. L. Levine, The therapeutic monoclonal antibody market, *mAbs* 7 (2015) 9–14.
- [3] R.-M. Lu, Y.-C. Hwang, I.-J. Liu, C.-C. Lee, H.-Z. Tsai, H.-J. Li, H.-C. Wu, Development of therapeutic antibodies for the treatment of diseases, *Journal of Biomedical Science* 27 (2020).
- [4] Grand View Research, Monoclonal Antibodies Market Size, Share & Trends Analysis Report By Source Type (Chimeric, Murine, Humanized), By Production Type (In Vivo, In Vitro), By Application, By End-use, By Region, And Segment Forecasts, 2022 - 2030, 2022. URL: <https://www.grandviewresearch.com/industry-analysis/monoclonal-antibodies-market>, Accessed on 2023-02-09.
- [5] I. Melero, S. Hervas-Stubbs, M. Glennie, D. M. Pardoll, L. Chen, Immunostimulatory monoclonal antibodies for cancer therapy, *Nature Reviews Cancer* 7 (2007) 95–106.
- [6] U. Hafeez, H. K. Gan, A. M. Scott, Monoclonal antibodies as immunomodulatory therapy against cancer and autoimmune diseases, *Current Opinion in Pharmacology* 41 (2018) 114–121.
- [7] R. Otsubo, T. Yasui, Monoclonal antibody therapeutics for infectious diseases: Beyond normal human immunoglobulin, *Pharmacology & Therapeutics* 240 (2022).
- [8] P. Zhou, X. L. Yang, X. G. Wang, B. Hu, L. Zhang, W. Zhang, H. R. Si, Y. Zhu, B. Li, C. L. Huang, H. D. Chen, J. Chen, Y. Luo, H. Guo, R. D. Jiang, M. Q. Liu, Y. Chen, X. R. Shen, X. Wang, X. S. Zheng, K. Zhao, Q. J. Chen, F. Deng, L. L. Liu, B. Yan, F. X. Zhan, Y. Y. Wang, G. F. Xiao, Z. L. Shi, A pneumonia outbreak associated with a new coronavirus of probable bat origin, *Nature* 579 (2020) 270–273.
- [9] M. Marovich, J. R. Mascola, M. S. Cohen, Monoclonal Antibodies for Prevention and Treatment of COVID-19, *JAMA* 324 (2020) 131–132.
- [10] L. Jahanshahlu, N. Rezaei, Monoclonal antibody as a potential anti-covid-19, *Biomedicine & Pharmacotherapy* 129 (2020).
- [11] C. Wang, W. Li, D. Drabek, N. M. A. Okba, R. van Haperen, A. D. M. E. Osterhaus, F. J. M. van Kuppeveld, B. L. Haagmans, F. Grosveld, B.-J. Bosch, A human monoclonal antibody blocking SARS-CoV-2 infection, *Nature Communications* 11 (2020) 2251.

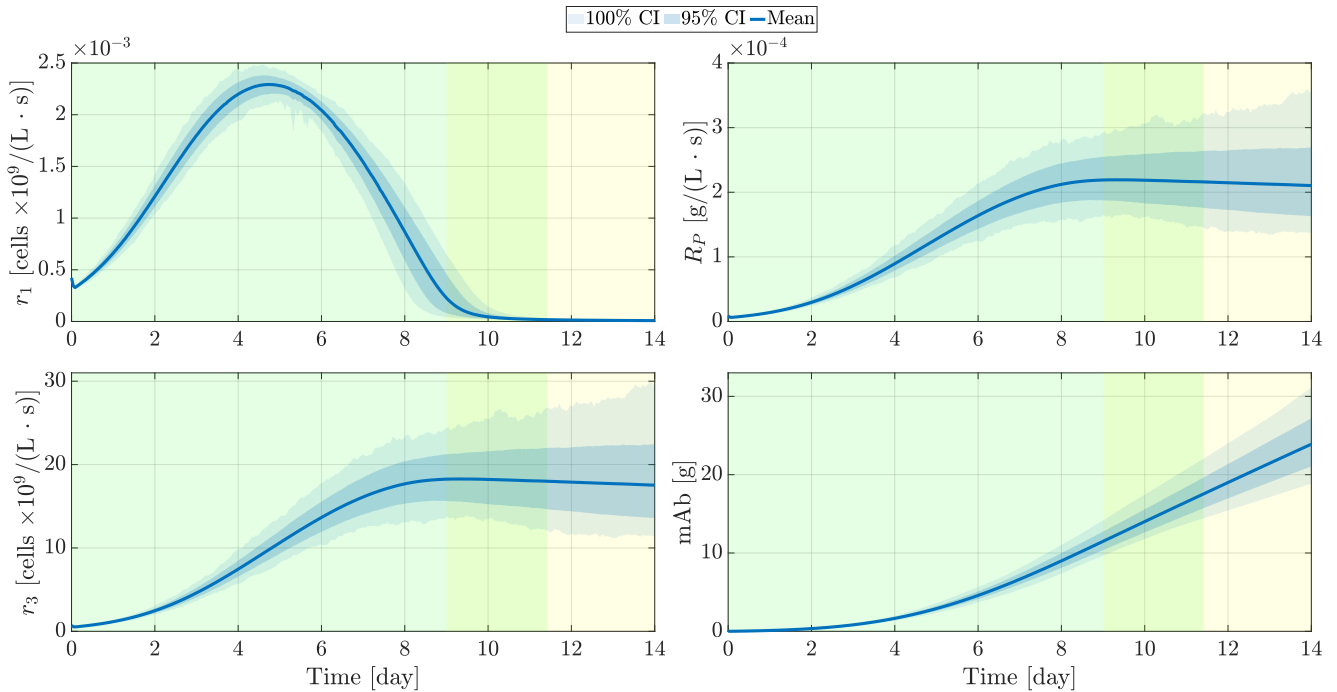


Figure 10: The growth rates r_1 and r_3 together with the product production rate, R_p , and the produced product (mAb) for 10,000 Monte Carlo simulations. In the green shaded area, all simulations are in the growth phase. In the yellow shaded area, all simulations are in the production phase. In the yellow-green shaded area, some simulations are in the growth phase and some are in the production phase. The reaction rate, r_1 , is almost zeroed after 10 days in almost all simulations.

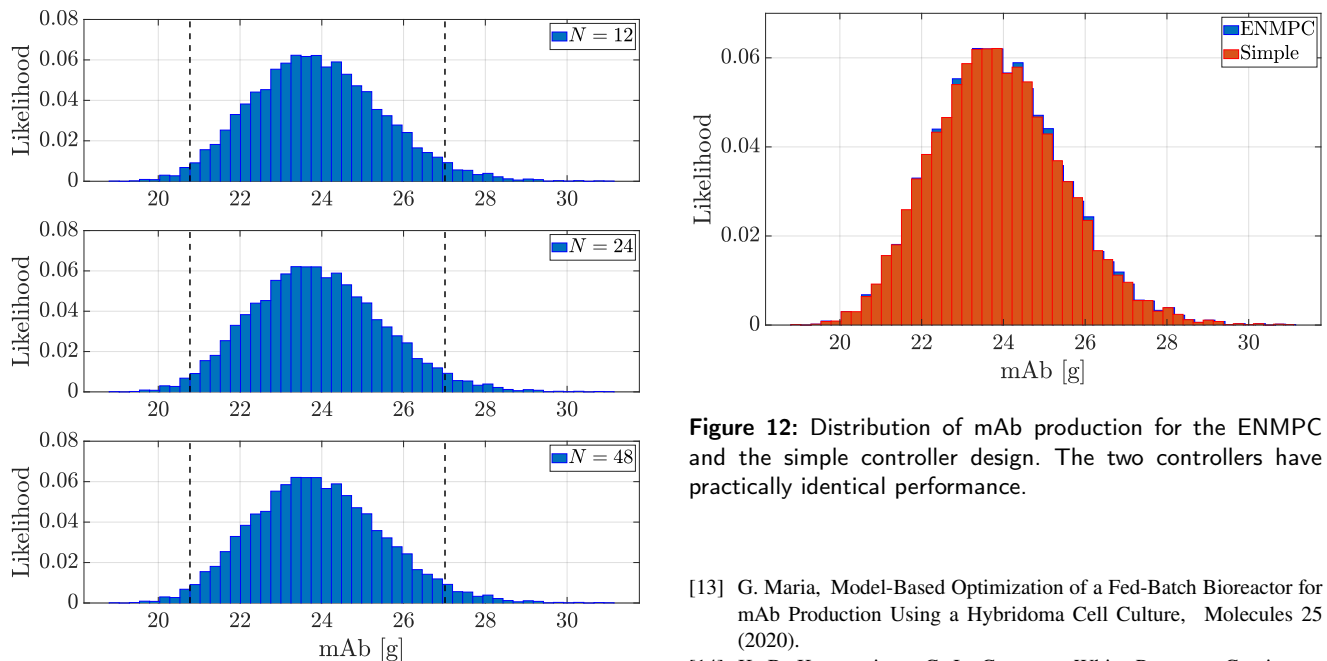


Figure 11: MAb production with different ENMPC horizons.

Figure 12: Distribution of mAb production for the ENMPC and the simple controller design. The two controllers have practically identical performance.

[12] L. Erickson, Bioreactors, in: M. Schaechter (Ed.), Encyclopedia of Microbiology (3rd ed.), Elsevier, 2009, pp. 206–211.

[13] G. Maria, Model-Based Optimization of a Fed-Batch Bioreactor for mAb Production Using a Hybridoma Cell Culture, *Molecules* 25 (2020).
 [14] K. B. Konstantinov, C. L. Cooney, White Paper on Continuous Bioprocessing, *Journal of Pharmaceutical Sciences* 104 (2015) 813–820.
 [15] M. Jiang, R. D. Braatz, Integrated control of continuous (bio)pharmaceutical manufacturing, *American Pharmaceutical Review* 19 (2016) 110–115.
 [16] W. Blunt, D. B. Levin, N. Cicek, Bioreactor Operating Strategies for Improved Polyhydroxyalkanoate (PHA) Productivity, *Polymers* 10 (2018).

- [17] W. Fong, Y. Zhang, P. Yung, Optimization of monoclonal antibody production: Combined effects of potassium acetate and perfusion in a stirred tank bioreactor, *Cytotechnology* 24 (1997) 47–54.
- [18] M. Butler, Animal cell cultures: recent achievements and perspectives in the production of biopharmaceuticals, *Applied Microbiology and Biotechnology* 68 (2005) 283–291.
- [19] F. Li, N. Vijayasankaran, A. Y. Shen, R. Kiss, A. Amanullah, Cell culture processes for monoclonal antibody production, *mAbs* 2 (2010) 466–479.
- [20] M. S. Hong, K. A. Severson, M. Jiang, A. E. Lu, J. C. Love, R. D. Braatz, Challenges and opportunities in biopharmaceutical manufacturing control, *Computers & Chemical Engineering* 110 (2018) 106–114.
- [21] M. Diehl, H. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, F. Allgöwer, Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations, *Journal of Process Control* 12 (2002) 577–585.
- [22] Z. K. Nagy, R. D. Braatz, Robust nonlinear model predictive control of batch processes, *AIChE Journal* 49 (2003) 1776–1786.
- [23] M. Aehle, K. Bork, S. Schaepe, A. Kuprijanov, R. Horstkorte, R. Simutis, A. Lübbert, Increasing batch-to-batch reproducibility of CHO-cell cultures using a model predictive control approach, *Cytotechnology* 64 (2012) 623–634.
- [24] S. Craven, J. Whelan, B. Glennon, Glucose concentration control of a fed-batch mammalian cell bioprocess using a nonlinear model predictive controller, *Journal of Process Control* 24 (2014) 344–357.
- [25] B. Huyck, J. De Brabanter, B. De Moor, J. F. Van Impe, F. Logist, Online model predictive control of industrial processes using low level control hardware: A pilot-scale distillation column case study, *Control Engineering Practice* 28 (2014) 34–48.
- [26] L. Dewasme, S. Fernandes, Z. Amribt, L. Santos, P. Bogaerts, A. Vande Wouwer, State estimation and predictive control of fed-batch cultures of hybridoma cells, *Journal of Process Control* 30 (2015) 50–57.
- [27] A. Drejer, T. Ritschel, S. B. Jørgensen, J. B. Jørgensen, Economic Optimizing Control for Single-Cell Protein Production in a U-Loop Reactor, 27th European Symposium on Computer Aided Process Engineering (ESCAPE), Barcelona, Spain 40 (2017) 1759–1764.
- [28] T. K. S. Ritschel, D. Boiroux, M. K. Nielsen, J. K. Huusom, S. B. Jørgensen, J. B. Jørgensen, Economic Optimal Control of a U-loop Bioreactor using Simultaneous Collocation-based Approaches, IEEE Conference on Control Technology and Applications (CCTA), Hong Kong, China (2019) 933–938.
- [29] J. E. Bailey, *Mathematical Modeling and Analysis in Biochemical Engineering: Past Accomplishments and Future Opportunities*, *Biotechnology Progress* 14 (1998) 8–20.
- [30] A. Provost, G. Bastin, Dynamic metabolic modelling under the balanced growth condition, *Journal of Process Control* 14 (2004) 717–728.
- [31] Y. H. Liu, J. X. Bi, A. P. Zeng, J. Q. Yuan, A simple kinetic model for myeloma cell culture with consideration of lysine limitation, *Bioprocess and Biosystems Engineering* 31 (2008) 569–577.
- [32] S. Craven, N. Shirsat, J. Whelan, B. Glennon, Process model comparison and transferability across bioreactor scales and modes of operation for a mammalian cell bioprocess, *Biotechnology Progress* 29 (2012) 186–196.
- [33] S. Kyriakopoulos, K. S. Ang, M. Lakshmanan, Z. Huang, S. Yoon, R. Gunawan, D.-Y. Lee, Kinetic Modeling of Mammalian Cell Culture Bioprocessing: The Quest to Advance Biomanufacturing, *Biotechnology Journal* 13 (2018) 1700229.
- [34] D. Kumar, N. Gangwar, A. S. Rathore, M. Ramteke, Multi-objective optimization of monoclonal antibody production in bioreactor, *Chemical Engineering and Processing - Process Intensification* 180 (2022).
- [35] T. E. Ryde, M. R. Wahlgreen, M. K. Nielsen, S. Hørsholt, S. B. Jørgensen, J. B. Jørgensen, Optimal Feed Trajectories for Fedbatch Fermentation with Substrate Inhibition Kinetics, *IFAC-PapersOnLine* 54 (2021) 318–323.
- [36] M. R. Wahlgreen, A. T. Reenberg, M. K. Nielsen, A. Rydahl, T. K. S. Ritschel, B. Dammann, J. B. Jørgensen, A High-Performance Monte Carlo Simulation Toolbox for Uncertainty Quantification of Closed-loop Systems, *Proceedings of the 60th IEEE Conference on Decision and Control (CDC)* (2021) 6755–6761.
- [37] A. S. Rathore, S. K. Singh, M. Pathak, E. K. Read, K. A. Brorson, C. D. Agarabi, M. Khan, Fermentanomics: Relating quality attributes of a monoclonal antibody to cell culture process variables and raw materials using multivariate data analysis, *Biotechnology Progress* 31 (2015) 1586–1599.
- [38] US Food and Drug Administration, Guidance for industry PAT - A framework for innovative pharmaceutical development, manufacturing, and quality assurance, 2004. URL: <https://www.fda.gov/regulatory-information/search-fda-guidance-documents/pat-framework-innovative-pharmaceutical-development-manufacturing-and-quality-assurance>, Accessed on 2023-02-10.
- [39] M. Luna, E. Martínez, A Bayesian Approach to Run-to-Run Optimization of Animal Cell Bioreactors Using Probabilistic Tendency Models, *Industrial & Engineering Chemistry Research* 53 (2014) 17252–17266.
- [40] L. Rendón-Castrillón, M. Ramírez-Carmona, C. Ocampo-López, L. Gómez-Arroyave, Mathematical Model for Scaling up Bioprocesses Using Experiment Design Combined with Buckingham Pi Theorem, *Applied Sciences* 11 (2021).
- [41] O. J. Wohlenberg, C. Kortmann, K. V. Meyer, J. Schellenberg, K. Dahlmann, J. Bahnemann, T. Scheper, D. Solle, Optimization of a mAb production process with regard to robustness and product quality using quality by design principles, *Engineering in Life Sciences* 22 (2022) 484–494.
- [42] M. R. Wahlgreen, J. B. Jørgensen, M. Zanon, Model Predictive Control Tuning by Monte Carlo Simulation and Controller Matching, *Proceedings of Foundations of Computer Aided Process Operations / Chemical Process Control (FOCAPO / CPC)* (2023).
- [43] A. T. Reenberg, T. K. S. Ritschel, B. Dammann, J. B. Jørgensen, High-performance Uncertainty Quantification in Large-scale Virtual Clinical Trials of Closed-loop Diabetes Treatment, *Proceedings of the American Control Conference (ACC)* (2022) 1367–1372.
- [44] M. W. Kaysfeld, M. Zanon, J. B. Jørgensen, Performance Quantification of a Nonlinear Model Predictive Controller by Parallel Monte Carlo Simulations of a Closed-loop System, *arXiv (Preprint)* (2022). doi:10.48550/arXiv.2212.02197.
- [45] M. W. Kaysfeld, D. Kumar, M. K. Nielsen, J. B. Jørgensen, Dynamic Optimization for Monoclonal Antibody Production, *arXiv (Preprint)* (2023). doi:10.48550/arXiv.2302.09932.
- [46] J. Li, C. L. Wong, N. Vijayasankaran, T. Hudson, A. Amanullah, Feeding Lactate for CHO Cell Culture Processes: Impact on Culture Metabolism and Performance, *Biotechnology and Bioengineering* 109 (2012) 1173–1186.
- [47] Y. Fan, I. Jimenez Del Val, C. Müller, J. Wagtberg Sen, S. K. Rasmussen, C. Kontoravdi, D. Weilguny, M. R. Andersen, Amino Acid and Glucose Metabolism in Fed-Batch CHO Cell Culture Affects Antibody Production and Glycosylation, *Biotechnology and Bioengineering* 112 (2015) 521–535.
- [48] M. Vergara, M. Torres, A. Müller, V. Avello, C. Acevedo, J. Berrios, J. G. Reyes, N. A. Valdez-Cruz, C. Altamirano, High glucose and low specific cell growth but not mild hypothermia improve specific r-protein productivity in chemostat culture of CHO cells, *PLOS ONE* 13 (2018) 1–22.
- [49] B. Sissolak, N. Lingg, W. Sommeregger, G. Sriedner, K. Vorauer-Uhl, Impact of mammalian cell culture conditions on monoclonal antibody charge heterogeneity: an accessory monitoring tool for process development, *Journal of Industrial Microbiology and Biotechnology* 46 (2019) 1167–1178.
- [50] M. R. Wahlgreen, K. Meyer, T. K. S. Ritschel, A. P. Engsig-Karup, K. V. Gernaey, J. B. Jørgensen, Modeling and Simulation of Upstream and Downstream Processes for Monoclonal Antibody Production, *The 13th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS)*, Busan, Republic of Korea

- (2022).
- [51] R. Amrit, J. B. Rawlings, D. Angeli, Economic optimization using model predictive control with a terminal cost, *Annual Reviews in Control* 35 (2011) 178–186.
- [52] D. Angeli, R. Amrit, J. B. Rawlings, On Average Performance and Stability of Economic Model Predictive Control, *IEEE Transactions on Automatic Control* 57 (2012) 1615–1626.
- [53] J. B. Rawlings, D. Angeli, C. N. Bates, Fundamentals of economic model predictive control, *Proceedings of the 51st IEEE Conference on Decision and Control (CDC)* (2012) 3851–3861.
- [54] M. R. Wahlgreen, E. Schroll-Fleischer, D. Boiroux, T. K. S. Ritschel, H. Wu, J. K. Huusom, J. B. Jørgensen, Nonlinear Model Predictive Control for an Exothermic Reaction in an adiabatic CSTR, 6th Conference on Advances in Control and Optimization of Dynamical Systems ACOODS, Chennai, India (2020).
- [55] J. B. Jørgensen, T. K. S. Ritschel, D. Boiroux, E. Schroll-Fleischer, M. R. Wahlgreen, M. K. Nielsen, H. Wu, J. K. Huusom, Simulation of NMPC for a Laboratory Adiabatic CSTR with an Exothermic Reaction, *Proceedings of 2020 European Control Conference (ECC)* (2020) 202–207.
- [56] D. Boiroux, J. B. Jørgensen, Nonlinear Model Predictive Control and Artificial Pancreas Technologies, *Proceedings of the 57th IEEE Conference on Decision and Control (CDC)* (2018) 284–290.
- [57] N. L. Brok, H. Madsen, J. B. Jørgensen, Nonlinear Model Predictive Control for Stochastic Differential Equation Systems, *Proceedings of the 6th IFAC Conference on Nonlinear Model Predictive Control (NMPC)* (2018) 430–435.
- [58] J. Stoustrup, Successful industry/academia cooperation: From simple via complex to lucid solutions, *European Journal of Control* 19 (2013) 358–368.
- [59] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Dover Publication, Inc., 2007.
- [60] M. K. Nielsen, T. K. S. Ritschel, I. Christensen, J. Dragheim, J. K. Huusom, K. V. Gernaey, J. B. Jørgensen, State Estimation for Continuous-Discrete-Time Nonlinear Stochastic Systems, *arXiv (Preprint)* (2023), doi:10.48550/arXiv.2212.02139.
- [61] R. Schneider, C. Georgakis, How To NOT Make the Extended Kalman Filter Fail, *Industrial & Engineering Chemistry Research* 52 (2013) 3354–3362.
- [62] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, 1992.
- [63] J. B. Jørgensen, *Moving Horizon Estimation and Control*, Ph.D. thesis, Technical University of Denmark, 2004.
- [64] J. B. Jørgensen, G. Frison, N. F. Gade-Nielsen, B. Damman, Numerical Methods for Solution of the Extended Linear-Quadratic Control Problem, *IFAC Proceedings Volumes* 45 (2012) 187–193.
- [65] G. Frison, J. B. Jørgensen, Efficient Implementation of the Riccati Recursion for Solving Linear-Quadratic Control Problems, *IEEE International Conference on Control Applications (CCA)*, Hyderabad, India (2013) 1117–1122.
- [66] M. R. Wahlgreen, J. B. Jørgensen, On the Implementation of a Preconditioned Riccati Recursion based Primal-Dual Interior-Point Algorithm for Input Constrained Optimal Control Problems, *IFAC-PapersOnLine* 55 (2022) 346–351. 13th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems DYCOPS 2022.
- [67] G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, M. Diehl, BLASFEO: Basic Linear Algebra Subroutines for Embedded Optimization, *ACM Transactions on Mathematical Software (TOMS)* 44 (2018).
- [68] G. Frison, T. Sartor, A. Zanelli, M. Diehl, The BLAS API of BLASFEO: Optimizing Performance for Small Matrices, *ACM Transactions on Mathematical Software (TOMS)* 46 (2020).
- [69] D. J. Higham, An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations, *SIAM Review* 43 (2001) 525–546.
- [70] Selina Wamucii, US Glucose Prices, 2023. URL: <https://www.selina.wamucii.com/insights/prices/united-states-of-america/glucose/>, Accessed on 2023-02-15.
- [71] Drugs, Remicade Prices, Coupons and Patient Assistance Programs, 2023. URL: <https://www.drugs.com/price-guide/remicade>, Accessed on 2023-02-15.
- [72] B. Srinivasan, S. Palanki, D. Bonvin, Dynamic optimization of batch processes I. Characterization of the nominal solution, *Computers & Chemical Engineering* 27 (2003) 1–26.

APPENDIX H

Technical Report

Numerical optimization packages for optimal control: QPIPM and NLPSQP

Authors:

Morten Wahlgreen Kaysfeld, John Bagterp Jørgensen

Submitted to:

DTU Library 2023.

Morten Wahlgreen Kaysfeld

Numerical optimization packages for optimal control: QPMP and NLPSQP

Technical Report, July 25, 2023

MORTEN WAHLGREEN KAYSFELD

Numerical optimization packages for optimal control: QPIPM and NLPSQP

Technical Report, July 25, 2023

Supervisors:

John Bagterp Jørgensen

DTU - Technical University of Denmark, Kgs. Lyngby - 2023

Numerical optimization packages for optimal control: QPIM and NLPSQP

This report was prepared by:

Morten Wahlgreen Kaysfeld

Advisors:

John Bagterp Jørgensen

DTU Compute - Department of Applied Mathematics and Computer Science

Section for Scientific Computing

Technical University of Denmark

Matematiktorvet, Building 303B

2800 Kgs. Lyngby

Denmark

morwa@dtu.dk

Field: Numerical optimization, quadratic programming, nonlinear programming

Class: Report publicly available - Software is closed-source

Remarks: This technical report is prepared to document the numerical optimization software packages: QPIM (quadratic-programming-interior-point-method) and NLPSQP (nonlinear-programming-sequential-quadratic-programming)

Copyrights: ©Morten Wahlgreen Kaysfeld, 2023

Table of Contents

I	QIPM	1
1	Introduction	3
2	Mathematical details	5
2.1	Primal-dual interior-point algorithm	5
2.1.1	Search direction	6
2.1.2	System reduction	8
2.1.3	Fraction-to-the-boundary	10
2.1.4	Predictor-corrector algorithm	10
2.1.5	Convergence criterion	11
2.1.6	Infinity bound constraints	11
2.1.7	Algorithm	11
2.2	Riccati based factorization for optimal control problems	13
2.2.1	Search direction	14
2.2.2	System reduction	15
2.2.3	Riccati recursion algorithm	18
2.2.4	Algorithm	18
2.2.5	A note on bounds	18
3	Implementation of QIPM in Matlab and C	21
3.1	Matlab	21
3.2	C	22
3.2.1	Memory allocation	23
3.2.2	Dependencies	24
3.2.3	Gitlab	24
3.2.4	Doxygen documentation	24
3.3	Examples	24
4	Conclusion	25
II	NLPSQP	27
5	Introduction	29

6	Mathematical details	31
6.1	Sequential quadratic programming algorithm	32
6.1.1	Optimality conditions	32
6.1.2	Quadratic programming subproblem	32
6.1.3	Line-search	33
6.1.4	BFGS update	34
6.1.5	Initialization	35
6.1.6	Convergence	35
6.1.7	Algorithm	35
6.2	Riccati version for optimal control problems	35
6.2.1	Block BFGS update	38
6.2.2	Application to solve OCPs	38
6.2.3	Algorithm	40
6.2.4	A note on bounds	41
7	Implementation of NLPSQP in Matlab and C	43
7.1	Matlab	43
7.2	C	44
7.2.1	Memory allocation	47
7.2.2	Dependencies	47
7.2.3	Gitlab	47
7.2.4	Doxygen documentation	48
7.3	Examples	48
8	Conclusion	49
	Bibliography	51

Part I

QPIP

Introduction

In this part, we introduce the Riccati based primal-dual interior-point software, QPIPM (quadratic-programming-interior-point-method), for solution of quadratic programming problems (QPs). QPIPM can solve QPs with 1) equality constraints, 2) box constraints, and 3) soft constraints. We have implemented QPIPM in both a Matlab version and a C version. Due to time constraints, currently only the Matlab version of QPIPM supports QPs with soft constraints. The Matlab version provides a non-optimized and simple implementation that can be useful in a development phase. The C version is implemented thread-safe with the intention to solve multiple optimal control problems (OCPs) in parallel. The thread-safety is achieved by QPIPM having no internal memory allocations. The main purpose of QPIPM is to be included in the sequential quadratic programming algorithm, NLPSQP, introduced in the next part of this report and the integration of QPIPM and NLPSQP in a previously implemented toolbox for parallel Monte Carlo simulation of closed-loop systems (Wahlgreen et al. 2021). We also point out that the current version of QPIPM is work in progress and that the implementation can be optimized for better computational performance.

In this report, we introduce the mathematical details in the QPIPM implementation and introduce the interfaces of QPIPM in both Matlab and C. QPIPM is stored in a private gitlab-repository `QPIPM` and is part of the project `SCPproject`, which is implemented in C and contains a number of other gitlab-repositories. For the C version, we introduce the other dependencies in `SCPproject` and explain how to allocate the required memory prior to calling QPIPM.

We point out that the implementation of QPIPM is highly inspired by previous work on the topic (Rao et al. 1998, Jørgensen 2004, Wächter and Biegler 2006, Frison and Jørgensen 2013, Jørgensen et al. 2012, Wahlgreen and Jørgensen 2022).

Mathematical details

We introduce the mathematical details of the QPIPM implementation. The mathematical details of the Matlab and C implementation are identical. However, the C version does not include the option to apply soft constraints in the current version. QPIPM is a primal-dual interior-point algorithm, which can both apply an LDL-factorization and a Riccati based method to solve the system of linear equations for the Newton search direction. The Riccati based method requires a structured QP, which, e.g., occurs in optimal control applications.

2.1 Primal-dual interior-point algorithm

In this section, we introduce the mathematical details of the primal-dual interior-point algorithm applied in QPIPM. The algorithm solves the first order Karush–Kuhn–Tucker (KKT) conditions with Newtons' method (Karush 1939, Kuhn and Tucker 1951, Kjeldsen 2000). As such, the algorithm is iterative and in each iteration, l , a system of linear equations is solved for the Newton search direction. We apply Mehrotra's predictor-corrector method, as such QPIPM computes both a predictor and corrector step with the same factorization of the search direction matrix (Mehrotra 1992).

We design QPIPM to solve QPs with bound constraints and general soft constraints. The general soft constraints include a lower and upper soft bound with slack variables, and the slack variables are penalized with both a linear and quadratic term in the objective. As such, the QP is in the form

$$\min_{x, \epsilon_l, \epsilon_u} \frac{1}{2} x^\top H x + g^\top x + \frac{1}{2} \epsilon_l^\top Q_l \epsilon_l + q_l^\top \epsilon_l + \frac{1}{2} \epsilon_u^\top Q_u \epsilon_u + q_u^\top \epsilon_u, \quad (2.1a)$$

$$s.t. \quad A^\top x = b, \quad (2.1b)$$

$$l \leq x \leq u, \quad (2.1c)$$

$$l_s - \epsilon_l \leq S^\top x \leq u_s + \epsilon_u, \quad (2.1d)$$

$$\epsilon_l, \epsilon_u \geq 0. \quad (2.1e)$$

$H \in \mathbb{R}^{n \times n}$, $g \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times m_e}$, $b \in \mathbb{R}^{m_e}$, $l \in \mathbb{R}^n$, $u \in \mathbb{R}^n$, $S \in \mathbb{R}^{n \times m_s}$, $l_s \in \mathbb{R}^{m_s}$, $u_s \in \mathbb{R}^{m_s}$, and $x \in \mathbb{R}^n$ are the decision variables. $\epsilon_l \in \mathbb{R}^{m_s}$ are lower soft bound slack variables and $\epsilon_u \in \mathbb{R}^{m_s}$ are upper soft bound slack variables. $Q_l \in \mathbb{R}^{m_s \times m_s}$ and $Q_u \in \mathbb{R}^{m_s \times m_s}$ are (assumed) diagonal penalty matrices, and $q_l \in \mathbb{R}^{m_s}$ and $q_u \in \mathbb{R}^{m_s}$ are penalty vectors. As such, n is the number of decision variables, m_e is the number of equality constraints, and m_s is the number of upper and lower soft constraints.

2.1.1 Search direction

QPIPM computes a search direction in each iteration. First, we consider the Lagrangian function, $\mathcal{L} = \mathcal{L}(x, \epsilon_l, \epsilon_u, y, v_l, v_u, z_{s_l}, z_{s_u}, v_{\epsilon_l}, v_{\epsilon_u})$, of (2.1), which is

$$\begin{aligned} \mathcal{L} = & \frac{1}{2}x^\top Hx + g^\top x + \frac{1}{2}\epsilon_l^\top Q_l \epsilon_l + q_l^\top \epsilon_l + \frac{1}{2}\epsilon_u^\top Q_u \epsilon_u + q_u^\top \epsilon_u \\ & - y^\top (A^\top x - b) - v_l^\top (x - l) - v_u^\top (u - x) - v_{\epsilon_l}^\top \epsilon_l - v_{\epsilon_u}^\top \epsilon_u \\ & - z_{s_l}^\top (S^\top x - l_s + \epsilon_l) - z_{s_u}^\top (-S^\top x + u_s + \epsilon_u). \end{aligned} \quad (2.2)$$

y are equality constraint (2.1b) Lagrange multipliers, v_l and v_u are bound constraint (2.1c) Lagrange multipliers, z_{s_l} and z_{s_u} are soft constraint (2.1d) Lagrange multipliers, and v_{ϵ_l} and v_{ϵ_u} are ϵ -bound constraint (2.1e) Lagrange multipliers. As such, we write up the corresponding first order KKT-conditions,

$$\nabla_x \mathcal{L} = Hx + g - Ay - v_l + v_u - Sz_{s_l} + Sz_{s_u} = 0, \quad (2.3a)$$

$$\nabla_{\epsilon_l} \mathcal{L} = Q_l \epsilon_l + q_l - z_{s_l} - v_{\epsilon_l} = 0, \quad (2.3b)$$

$$\nabla_{\epsilon_u} \mathcal{L} = Q_u \epsilon_u + q_u - z_{s_u} - v_{\epsilon_u} = 0, \quad (2.3c)$$

$$b - A^\top x = 0, \quad (2.3d)$$

$$t_l + l - x = 0, \quad t_u + x - u = 0, \quad (2.3e)$$

$$t_{\epsilon_l} - \epsilon_l = 0, \quad t_{\epsilon_u} - \epsilon_u = 0, \quad (2.3f)$$

$$s_{s_l} - S^\top x + l_s - \epsilon_l = 0, \quad s_{s_u} + S^\top x - u_s - \epsilon_u = 0, \quad (2.3g)$$

$$t_{l,i} v_{l,i} = 0, \quad t_{u,i} v_{u,i} = 0, \quad (2.3h)$$

$$t_{\epsilon_l,i} v_{\epsilon_l,i} = 0, \quad t_{\epsilon_u,i} v_{\epsilon_u,i} = 0, \quad (2.3i)$$

$$s_{s_l,i} z_{s_l,i} = 0, \quad s_{s_u,i} z_{s_u,i} = 0, \quad (2.3j)$$

$$(v_l, v_u, z_{s_l}, z_{s_u}, v_{\epsilon_l}, v_{\epsilon_u}) \geq 0, \quad (t_l, t_u, s_{s_l}, s_{s_u}, t_{\epsilon_l}, t_{\epsilon_u}) \geq 0, \quad (2.3k)$$

where t_l and t_u are bound constraint (2.1c) slack variables, s_{s_l} and s_{s_u} are soft constraint (2.1d) slack variables, and t_{ϵ_l} and t_{ϵ_u} are ϵ -bound constraint (2.1e) slack variables. The slack variables are defined as

$$s_{s_l} = S^\top x - l_s + \epsilon_l, \quad s_{s_u} = -S^\top x + u_s + \epsilon_u, \quad (2.4a)$$

$$t_l = x - l, \quad t_u = u - x, \quad (2.4b)$$

$$t_{\epsilon_l} = \epsilon_l, \quad t_{\epsilon_u} = \epsilon_u. \quad (2.4c)$$

We write the KKT-conditions, (2.3), as a system of nonlinear equations in the form

$$\begin{bmatrix} r_L \\ r_{\epsilon_l} \\ r_{\epsilon_u} \\ r_A \\ r_{S_l} \\ r_{S_u} \\ r_{B_l} \\ r_{B_u} \\ r_{B_{\epsilon_l}} \\ r_{B_{\epsilon_u}} \\ r_{SZ_{s_l}} \\ r_{SZ_{s_u}} \\ r_{TV_l} \\ r_{TV_u} \\ r_{TV_{\epsilon_l}} \\ r_{TV_{\epsilon_u}} \end{bmatrix} = \begin{bmatrix} Hx + g - Ay - v_l + v_u - Sz_{s_l} + Sz_{s_u} \\ Q_l \epsilon_l + q_l - z_{s_l} - v_{\epsilon_l} \\ Q_u \epsilon_u + q_u - z_{s_u} - v_{\epsilon_u} \\ b - A^\top x \\ s_{s_l} - S^\top x + l_s - \epsilon_l \\ s_{s_u} + S^\top x - u_s - \epsilon_u \\ t_l + l - x \\ t_u + x - u \\ t_{\epsilon_l} - \epsilon_l \\ t_{\epsilon_u} - \epsilon_u \\ S_{s_l} Z_{s_l} e \\ S_{s_u} Z_{s_u} e \\ T_l V_l e \\ T_u V_u e \\ T_{\epsilon_l} V_{\epsilon_l} e \\ T_{\epsilon_u} V_{\epsilon_u} e \end{bmatrix} = 0, \quad (2.5a)$$

$$(v_l, v_u, v_{\epsilon_l}, v_{\epsilon_u}, z_{s_l}, z_{s_u}, t_l, t_u, t_{\epsilon_l}, t_{\epsilon_u}, s_{s_l}, s_{s_u}) \geq 0. \quad (2.5b)$$

$V_l = \text{diag}(v_l)$, $V_u = \text{diag}(v_u)$, $V_{\epsilon_l} = \text{diag}(v_{\epsilon_l})$, $V_{\epsilon_u} = \text{diag}(v_{\epsilon_u})$, $Z_{s_l} = \text{diag}(z_{s_l})$, $Z_{s_u} = \text{diag}(z_{s_u})$, $T_l = \text{diag}(t_l)$, $T_u = \text{diag}(t_u)$, $T_{\epsilon_l} = \text{diag}(t_{\epsilon_l})$, $T_{\epsilon_u} = \text{diag}(t_{\epsilon_u})$, $S_{s_l} = \text{diag}(s_l)$, $S_{s_u} = \text{diag}(s_u)$, and e is a vector of ones of proper dimension. We apply Newton's method to solve the nonlinear system of equations, (2.5), which results in the following linear system of equations for the Newton search direction,

$$\begin{bmatrix} H & 0 & 0 & -A & -S & S & -I & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Q_l & 0 & 0 & -I & 0 & 0 & 0 & -I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Q_u & 0 & 0 & -I & 0 & 0 & 0 & -I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline -A^\top & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline -S^\top & -I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ S^\top & 0 & -I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I \\ 0 & 0 & -I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I \\ \hline 0 & 0 & 0 & 0 & S_{s_l} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & S_{s_u} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & T_l & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & T_u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & T_{\epsilon_l} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & V_{\epsilon_l} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & T_{\epsilon_u} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & V_{\epsilon_u} \\ \hline \Delta x \\ \Delta \epsilon_l \\ \Delta \epsilon_u \\ \hline \Delta y \\ \hline \Delta z_{s_l} \\ \Delta z_{s_u} \\ \Delta v_l \\ \Delta v_u \\ \Delta v_{\epsilon_l} \\ \Delta v_{\epsilon_u} \\ \hline \Delta s_{s_l} \\ \Delta s_{s_u} \\ \Delta t_l \\ \Delta t_u \\ \Delta t_{\epsilon_l} \\ \Delta t_{\epsilon_u} \end{bmatrix} = - \begin{bmatrix} r_L \\ r_{\epsilon_l} \\ r_{\epsilon_u} \\ \hline r_A \\ \hline r_{S_l} \\ r_{S_u} \\ r_{B_l} \\ r_{B_u} \\ r_{B_{\epsilon_l}} \\ r_{B_{\epsilon_u}} \\ \hline r_{SZ_{s_l}} \\ r_{SZ_{s_u}} \\ r_{TV_l} \\ r_{TV_u} \\ r_{TV_{\epsilon_l}} \\ r_{TV_{\epsilon_u}} \end{bmatrix}. \quad (2.6)$$

The solution,

$$(\Delta x, \Delta \epsilon_l, \Delta \epsilon_u, \Delta y, \Delta z_{s_l}, \Delta z_{s_u}, \Delta v_l, \Delta v_u, \Delta v_{\epsilon_l}, \Delta v_{\epsilon_u}, \Delta s_{s_l}, \Delta s_{s_u}, \Delta t_l, \Delta t_u, \Delta t_{\epsilon_l}, \Delta t_{\epsilon_u}), \quad (2.7)$$

is the search direction applied in QPIP. We point out that the system of equations, (2.6), can be compactly written as

$$\begin{bmatrix} \hat{H} & -\hat{A} & -\hat{C} & 0 \\ -\hat{A}^\top & 0 & 0 & 0 \\ -\hat{C}^\top & 0 & 0 & I \\ 0 & 0 & \hat{S} & \hat{Z} \end{bmatrix} \begin{bmatrix} \Delta \hat{x} \\ \Delta \hat{y} \\ \Delta \hat{z} \\ \Delta \hat{s} \end{bmatrix} = - \begin{bmatrix} \hat{r}_L \\ \hat{r}_A \\ \hat{r}_C \\ \hat{r}_{SZ} \end{bmatrix}, \quad (2.8)$$

where

$$\hat{x} = \begin{bmatrix} x \\ \epsilon_l \\ \epsilon_u \end{bmatrix}, \quad \hat{y} = y, \quad \hat{z} = \begin{bmatrix} z_{s_l} \\ z_{s_u} \\ v_l \\ v_u \\ v_{\epsilon_l} \\ v_{\epsilon_u} \end{bmatrix}, \quad \hat{s} = \begin{bmatrix} s_{s_l} \\ s_{s_u} \\ t_l \\ t_u \\ t_{\epsilon_l} \\ t_{\epsilon_u} \end{bmatrix}, \quad (2.9a)$$

$$\hat{r}_L = \begin{bmatrix} r_L \\ r_{\epsilon_l} \\ r_{\epsilon_u} \end{bmatrix}, \quad \hat{r}_A = r_A, \quad \hat{r}_C = \begin{bmatrix} r_{S_l} \\ r_{S_u} \\ r_{B_l} \\ r_{B_u} \\ r_{B_{\epsilon_l}} \\ r_{B_{\epsilon_u}} \end{bmatrix}, \quad \hat{r}_{SZ} = \begin{bmatrix} r_{SZ_{s_l}} \\ r_{SZ_{s_u}} \\ r_{TV_l} \\ r_{TV_u} \\ r_{TV_{\epsilon_l}} \\ r_{TV_{\epsilon_u}} \end{bmatrix}, \quad (2.9b)$$

$$\hat{H} = \begin{bmatrix} H & & \\ & Q_l & \\ & & Q_u \end{bmatrix}, \quad \hat{A} = \begin{bmatrix} A \\ 0 \\ 0 \end{bmatrix}, \quad \hat{C} = \begin{bmatrix} S & -S & I & -I & 0 & 0 \\ I & 0 & 0 & 0 & I & 0 \\ 0 & I & 0 & 0 & 0 & I \end{bmatrix}, \quad (2.9c)$$

$$\hat{Z} = \begin{bmatrix} Z_{s_l} & & & & & & \\ & Z_{s_u} & & & & & \\ & & V_l & & & & \\ & & & V_u & & & \\ & & & & V_{\epsilon_l} & & \\ & & & & & V_{\epsilon_u} & \end{bmatrix}, \quad \hat{S} = \begin{bmatrix} S_{s_l} & & & & & & \\ & S_{s_u} & & & & & \\ & & T_l & & & & \\ & & & T_u & & & \\ & & & & T_{\epsilon_l} & & \\ & & & & & T_{\epsilon_u} & \end{bmatrix}. \quad (2.9d)$$

However, we exploit the structure of the matrices in (2.9), and elimination of Lagrange multipliers and slack variables, to reduce the size of the system (2.6) in the following section.

2.1.2 System reduction

The linear system (2.6) can be reduced in size by elimination of the inequality Lagrange multipliers and corresponding slack variables (i.e., for the lower and upper bound constraint, the soft constraints, and the ϵ -bound constraints). We define six diagonal matrices from the Lagrange multipliers and corresponding slack variables,

$$D_{s_l} = \text{diag}(z_{s_l}/s_{s_l}), \quad D_{s_u} = \text{diag}(z_{s_u}/s_{s_u}), \quad (2.10a)$$

$$D_l = \text{diag}(v_l/t_l), \quad D_u = \text{diag}(v_u/t_u), \quad (2.10b)$$

$$D_{\epsilon_l} = \text{diag}(v_{\epsilon_l}/t_{\epsilon_l}), \quad D_{\epsilon_u} = \text{diag}(v_{\epsilon_u}/t_{\epsilon_u}). \quad (2.10c)$$

By elimination of the six Lagrange multipliers and slack variables, we arrive at the following reduced system

$$\begin{bmatrix} \bar{H} & E & F & -A \\ E^\top & \bar{Q}_l & & \\ F^\top & & \bar{Q}_u & \\ -A^\top & & & \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \epsilon_l \\ \Delta \epsilon_u \\ \Delta y \end{bmatrix} = \begin{bmatrix} \bar{r}_L \\ \bar{r}_{\epsilon_l} \\ \bar{r}_{\epsilon_u} \\ \bar{r}_A \end{bmatrix} \quad (2.11)$$

where

$$\bar{H} = H + D_l + D_u + ES^\top - FS^\top, \quad (2.12a)$$

$$E = SD_{s_l}, \quad F = -SD_{s_u}, \quad (2.12b)$$

$$\bar{Q}_l = Q_l + D_{\epsilon_l} + D_{s_l}, \quad \bar{Q}_u = Q_u + D_{\epsilon_u} + D_{s_u}, \quad (2.12c)$$

and

$$\begin{aligned} \bar{r}_L = & -r_L + S(S_{s_l}^{-1}Z_{s_l}(r_{s_l} - Z_{s_l}^{-1}r_{SZ_{s_l}})) - S(S_{s_u}^{-1}Z_{s_u}(r_{s_u} - Z_{s_u}^{-1}r_{SZ_{s_u}})) \\ & + T_l^{-1}V_l(r_{B_l} - V_l^{-1}r_{TV_l}) - T_u^{-1}V_u(r_{B_u} - V_u^{-1}r_{TV_u}), \end{aligned} \quad (2.13a)$$

$$\bar{r}_{\epsilon_l} = -r_{\epsilon_l} + T_{\epsilon_l}^{-1}V_{\epsilon_l}(r_{B_{\epsilon_l}} - V_{\epsilon_l}^{-1}r_{TV_{\epsilon_l}}) + S_{s_l}^{-1}Z_{s_l}(r_{s_l} - Z_{s_l}^{-1}r_{SZ_{s_l}}), \quad (2.13b)$$

$$\bar{r}_{\epsilon_u} = -r_{\epsilon_u} + T_{\epsilon_u}^{-1}V_{\epsilon_u}(r_{B_{\epsilon_u}} - V_{\epsilon_u}^{-1}r_{TV_{\epsilon_u}}) + S_{s_u}^{-1}Z_{s_u}(r_{s_u} - Z_{s_u}^{-1}r_{SZ_{s_u}}), \quad (2.13c)$$

$$\bar{r}_A = -r_A. \quad (2.13d)$$

The eliminated Lagrange multipliers and slack variables are

$$\Delta v_l = T_l^{-1}V_l(r_{B_l} - V_l^{-1}r_{TV_l}) - T_l^{-1}V_l\Delta x, \quad (2.14a)$$

$$\Delta v_u = T_u^{-1}V_u(r_{B_u} - V_u^{-1}r_{TV_u}) + T_u^{-1}V_u\Delta x, \quad (2.14b)$$

$$\Delta v_{\epsilon_l} = T_{\epsilon_l}^{-1}V_{\epsilon_l}(r_{B_{\epsilon_l}} - V_{\epsilon_l}^{-1}r_{TV_{\epsilon_l}}) - T_{\epsilon_l}^{-1}V_{\epsilon_l}\Delta \epsilon_l, \quad (2.14c)$$

$$\Delta v_{\epsilon_u} = T_{\epsilon_u}^{-1}V_{\epsilon_u}(r_{B_{\epsilon_u}} - V_{\epsilon_u}^{-1}r_{TV_{\epsilon_u}}) - T_{\epsilon_u}^{-1}V_{\epsilon_u}\Delta \epsilon_u, \quad (2.14d)$$

$$\Delta z_{s_l} = S_{s_l}^{-1}Z_{s_l}(r_{s_l} - Z_{s_l}^{-1}r_{SZ_{s_l}}) - S_{s_l}^{-1}Z_{s_l}(S^\top \Delta x + \Delta \epsilon_l), \quad (2.14e)$$

$$\Delta z_{s_u} = S_{s_u}^{-1}Z_{s_u}(r_{s_u} - Z_{s_u}^{-1}r_{SZ_{s_u}}) - S_{s_u}^{-1}Z_{s_u}(-S^\top \Delta x + \Delta \epsilon_u), \quad (2.14f)$$

$$\Delta t_l = -V_l^{-1}r_{TV_l} - V_l^{-1}T_l\Delta v_l, \quad (2.14g)$$

$$\Delta t_u = -V_u^{-1}r_{TV_u} - V_u^{-1}T_u\Delta v_u, \quad (2.14h)$$

$$\Delta t_{\epsilon_l} = -V_{\epsilon_l}^{-1}r_{TV_{\epsilon_l}} - V_{\epsilon_l}^{-1}T_{\epsilon_l}\Delta v_{\epsilon_l}, \quad (2.14i)$$

$$\Delta t_{\epsilon_u} = -V_{\epsilon_u}^{-1}r_{TV_{\epsilon_u}} - V_{\epsilon_u}^{-1}T_{\epsilon_u}\Delta v_{\epsilon_u}, \quad (2.14j)$$

$$\Delta s_{s_l} = -Z_{s_l}^{-1}r_{SZ_{s_l}} - Z_{s_l}^{-1}S_{s_l}\Delta z_{s_l}, \quad (2.14k)$$

$$\Delta s_{s_u} = -Z_{s_u}^{-1}r_{SZ_{s_u}} - Z_{s_u}^{-1}S_{s_u}\Delta z_{s_u}. \quad (2.14l)$$

In addition, we eliminate the soft constraint slack variables, ϵ_l and ϵ_u , from the system (2.11) to further reduce the size. The resulting system of linear equations is

$$\begin{bmatrix} \tilde{H} & -A \\ -A^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \tilde{r}_L \\ \tilde{r}_A \end{bmatrix}, \quad (2.15)$$

where

$$\tilde{H} = \bar{H} - E\bar{Q}_l^{-1}E^\top - F\bar{Q}_u^{-1}F^\top, \quad (2.16a)$$

$$\tilde{r}_L = \bar{r}_L - E\bar{Q}_l^{-1}\bar{r}_{\epsilon_l} - F\bar{Q}_u^{-1}\bar{r}_{\epsilon_u}, \quad (2.16b)$$

$$\tilde{r}_A = \bar{r}_A. \quad (2.16c)$$

The eliminated slack variables are given as

$$\Delta\epsilon_l = \bar{Q}_l^{-1}(\bar{r}_{\epsilon_l} - E^\top \Delta x), \quad (2.17a)$$

$$\Delta\epsilon_u = \bar{Q}_u^{-1}(\bar{r}_{\epsilon_u} - F^\top \Delta x). \quad (2.17b)$$

The search direction (2.7) can be obtained by solution of the system of linear equations, (2.15), to obtain $(\Delta x, \Delta y)$ and computing first the soft constraint slack variables from (2.17) and finally the remaining Lagrange multipliers and slack variables from (2.14). QPIPM solves (2.15) with an LDL-factorization and back substitution.

Applying the compact notation in (2.9a), we define the QPIPM step as

$$(\hat{x}, \hat{y}, \hat{z}, \hat{s}) = (\hat{x}, \hat{y}, \hat{z}, \hat{s}) + \eta\alpha(\Delta\hat{x}, \Delta\hat{y}, \Delta\hat{z}, \Delta\hat{s}), \quad (2.18)$$

where $\eta = 0.995$ and the step-size, α , ensures $(\hat{z}, \hat{s}) \geq 0$.

2.1.3 Fraction-to-the-boundary

QPIPM applies a fraction-to-the-boundary rule to avoid the QPIPM step zeroing the Lagrange multipliers or slack variables (Wahlgreen and Jørgensen 2022). The rule is

$$\begin{bmatrix} \hat{z} \\ \hat{s} \end{bmatrix} + \alpha \begin{bmatrix} \Delta\hat{z} \\ \Delta\hat{s} \end{bmatrix} \geq \kappa \begin{bmatrix} \hat{z} \\ \hat{s} \end{bmatrix}, \quad (2.19)$$

where $0 \leq \kappa \ll 1$ and $\kappa \rightarrow 0$ as the iteration number of QPIPM, l , increases. The rule (2.19) implements a proportional step-back from the zero-boundary. In the predictor phase, QPIPM uses $\kappa = 0$ to compute α^{aff} , and in the corrector phase QPIPM uses $\kappa = \min(1 - \eta, \mu^{aff})$ to compute α . The rule (2.19) is similar to the rule applied in IPOPT (Wächter and Biegler 2006).

2.1.4 Predictor-corrector algorithm

QPIPM applies Mehrotra's predictor-corrector algorithm (Mehrotra 1992), i.e., QPIPM applies the factorization of (2.15) twice: 1) in the predictor step and 2) in the corrector step. In the predictor phase, we solve

$$\begin{bmatrix} \tilde{H} & -A \\ -A^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta y^{aff} \end{bmatrix} = \begin{bmatrix} \tilde{r}_L \\ \tilde{r}_A \end{bmatrix}, \quad (2.20)$$

and compute the remaining part of the affine search direction from (2.17) and (2.14). From the affine search direction, we compute the duality gap, μ , and the centering parameter, σ as

$$\mu^{aff} = \frac{(\hat{z} + \alpha^{aff} \Delta\hat{z}^{aff})^\top (\hat{s} + \alpha^{aff} \Delta\hat{s}^{aff})}{\bar{m}}, \quad \mu = \frac{\hat{s}^\top \hat{z}}{\bar{m}}, \quad \sigma = \left(\frac{\mu^{aff}}{\mu} \right)^3, \quad (2.21)$$

where we apply the notation in (2.9a) for simplicity and \bar{m} is the total number of inequality constraints (bound constraints, soft constraints, and ϵ -bound constraints). In the corrector step, we adapt the right hand side of (2.15) and consider the system

$$\begin{bmatrix} \tilde{H} & -A \\ -A^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \tilde{r}_L \\ \tilde{r}_A \end{bmatrix}, \quad (2.22)$$

where \tilde{r}_L is computed according to (2.13a) and (2.16b), with the terms, r_{TV_l} , r_{TV_u} , $r_{TV_{\epsilon_l}}$, $r_{TV_{\epsilon_u}}$, $r_{SZ_{s_l}}$, and $r_{SZ_{s_u}}$ being defined as

$$r_{TV_l} \leftarrow r_{TV_l} + \Delta T_l^{aff} \Delta V_l^{aff} - \sigma \mu e, \quad r_{TV_u} \leftarrow r_{TV_u} + \Delta T_u^{aff} \Delta V_u^{aff} - \sigma \mu e, \quad (2.23a)$$

$$r_{TV_{\epsilon_l}} \leftarrow r_{TV_{\epsilon_l}} + \Delta T_{\epsilon_l}^{aff} \Delta V_{\epsilon_l}^{aff} - \sigma \mu e, \quad r_{TV_{\epsilon_u}} \leftarrow r_{TV_{\epsilon_u}} + \Delta T_{\epsilon_u}^{aff} \Delta V_{\epsilon_u}^{aff} - \sigma \mu e, \quad (2.23b)$$

$$r_{SZ_{s_l}} \leftarrow r_{SZ_{s_l}} + \Delta S_{s_l}^{aff} \Delta Z_{s_l}^{aff} - \sigma \mu e, \quad r_{SZ_{s_u}} \leftarrow r_{SZ_{s_u}} + \Delta S_{s_u}^{aff} \Delta Z_{s_u}^{aff} - \sigma \mu e. \quad (2.23c)$$

Then the QPIPM search direction is the solution to (2.22) with the remaining part being computed from (2.17) and (2.14).

We point out that system matrix in the predictor and corrector phase is identical. Therefore, QPIPM reuses the factorization from the predictor phase in the corrector phase.

2.1.5 Convergence criterion

QPIPM converges once the KKT-conditions (2.3) are satisfied. In practice, we consider a scaled violation, ξ , and define convergence as $\xi < \epsilon$, where $\epsilon > 0$ is a user-selected convergence tolerance. The scaled violation is

$$\xi = \max \left(s_H \|r_L, r_{\epsilon_l}, r_{\epsilon_u}\|_{\infty}, s_A \|r_A\|_{\infty}, s_S \|r_{S_l}, r_{S_u}\|_{\infty}, s_B \|r_{B_l}, r_{B_u}\|_{\infty}, \|r_{B_{\epsilon_l}}, r_{B_{\epsilon_u}}\|_{\infty}, \|r_{SZ_{s_l}}, r_{SZ_{s_u}}, r_{TV_l}, r_{TV_u}, r_{TV_{\epsilon_l}}, r_{TV_{\epsilon_u}}\|_{\infty} \right), \quad (2.24)$$

where

$$s_H = \max(1, \|H\|_{\infty}, \|g\|_{\infty}, \|A\|_{\infty}, \|Q_l\|_{\infty}, \|Q_u\|_{\infty}, \|q_l, q_u\|_{\infty}, \|S_l\|_{\infty}, \|S_u\|_{\infty})^{-1}, \quad (2.25a)$$

$$s_A = \max(1, \|A^T\|_{\infty}, \|b\|_{\infty})^{-1}, \quad (2.25b)$$

$$s_S = \max(1, \|S_l^T\|_{\infty}, \|S_u^T\|_{\infty}, \|l_s\|_{\infty}, \|u_s\|_{\infty})^{-1}, \quad (2.25c)$$

$$s_B = \max(1, \|l\|_{\infty}, \|u\|_{\infty})^{-1}. \quad (2.25d)$$

QPIPM computes ξ after taking the step (2.18) in the end of the corrector phase.

2.1.6 Infinity bound constraints

QPIPM eliminates all infinity bounds, i.e., bounds set to $-\infty$ or ∞ , before starting the loop. As such, columns of S are not accessed if both l_s and u_s are infinity.

2.1.7 Algorithm

Algorithm 1 presents a detailed implementation guide for QPIPM.

Algorithm 1: QPIM pseudo code**Input:** Initial guess, x_0 , and soft constrained QP,

$$\begin{aligned} \min_{x, \epsilon_l, \epsilon_u} \quad & \frac{1}{2} x^\top H x + g^\top x + \frac{1}{2} \epsilon_l^\top Q_l \epsilon_l + q_l^\top \epsilon_l + \frac{1}{2} \epsilon_u^\top Q_u \epsilon_u + q_u^\top \epsilon_u, \\ \text{s.t.} \quad & A^\top x = b, \\ & l \leq x \leq u, \\ & l_s - \epsilon_l \leq S^\top x \leq u_s + \epsilon_u, \\ & \epsilon_l, \epsilon_u \geq 0, \end{aligned}$$

i.e. the matrices and vectors: $H, Q_l, Q_u, g, q_l, q_u, A, b, l, u, S, l_s$, and l_u .

- Initialize:

$$x = x_0, \quad \epsilon_l = \epsilon_u = 0, \quad y = 0, \quad \hat{z} = 1, \quad \hat{s} = 1.$$

- Compute scaling factors,

$$\begin{aligned} \tilde{r}_L &= \max(1, \|H, Q_l, Q_u\|_\infty, \|g, q_l, q_u\|_\infty, \|A\|_\infty, \|S\|_\infty)^{-1}, & \tilde{r}_A &= \max(1, \|A^\top\|_\infty, \|b\|_\infty)^{-1}, \\ \tilde{r}_B &= \max(1, \|l\|_\infty, \|u\|_\infty)^{-1}, & \tilde{r}_S &= \max(1, \|S^\top\|_\infty, \|l_s\|_\infty, \|l_u\|_\infty)^{-1} \end{aligned}$$

- Compute scaled KKT-violation, ξ ,

$$\begin{aligned} \xi &= \max(\tilde{r}_L \|r_L, r_{\epsilon_l}, r_{\epsilon_u}\|_\infty, \tilde{r}_A \|r_A\|_\infty, \tilde{r}_S \|r_{S_l}, r_{S_u}\|_\infty, \tilde{r}_B \|r_{B_l}, r_{B_u}\|_\infty, \|r_{\epsilon_l}, r_{\epsilon_u}\|_\infty, \\ & \quad \|r_{S_{s_l}}, r_{S_{s_u}}, r_{TV_l}, r_{TV_u}, r_{TV_{\epsilon_l}}, r_{TV_{\epsilon_u}}\|_\infty) \end{aligned}$$

while $\xi > \epsilon$ **do****1. Predictor phase:**

- Setup augmented system,

$$\underbrace{\begin{bmatrix} \tilde{H} & -A \\ -A^\top & 0 \end{bmatrix}}_M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \tilde{r}_L \\ \tilde{r}_A \end{bmatrix} \quad (2.26)$$

- LDL factorize: $[L, D] = \text{ldl}(M)$

- Solve the system (2.26) to get the affine direction, $\Delta x = \Delta x^{aff}$ and $\Delta y = \Delta y^{aff}$

- Compute $\Delta \epsilon_l$ and $\Delta \epsilon_u$,

$$\Delta \epsilon_l = \bar{Q}_l^{-1} (\tilde{r}_{\epsilon_l} - E^\top \Delta x), \quad \Delta \epsilon_u = \bar{Q}_u^{-1} (\tilde{r}_{\epsilon_u} - F^\top \Delta x)$$

- Compute $\Delta z_{s_l}, \Delta z_{s_u}, \Delta v_l, \Delta v_u, \Delta \epsilon_l, \Delta \epsilon_u, \Delta s_{s_l}, \Delta s_{s_u}, \Delta t_l, \Delta t_u, \Delta t_{\epsilon_l}$, and Δt_{ϵ_u} ,

$$\begin{aligned} \Delta z_{s_l} &= S_{s_l}^{-1} Z_{s_l} (r_{S_l} - Z_{s_l}^{-1} r_{S Z_{s_l}}) - S_{s_l}^{-1} Z_{s_l} (S^\top \Delta x + \Delta \epsilon_l), & \Delta s_{s_l} &= -Z_{s_l}^{-1} r_{S Z_{s_l}} - Z_{s_l}^{-1} S_{s_l} \Delta z_{s_l}, \\ \Delta z_{s_u} &= S_{s_u}^{-1} Z_{s_u} (r_{S_u} - Z_{s_u}^{-1} r_{S Z_{s_u}}) - S_{s_u}^{-1} Z_{s_u} (-S^\top \Delta x + \Delta \epsilon_u), & \Delta s_{s_u} &= -Z_{s_u}^{-1} r_{S Z_{s_u}} - Z_{s_u}^{-1} S_{s_u} \Delta z_{s_u}, \\ \Delta v_l &= T_l^{-1} V_l (r_{B_l} - V_l^{-1} r_{TV_l}) - T_l^{-1} V_l \Delta x, & \Delta t_l &= -V_l^{-1} r_{TV_l} - V_l^{-1} T_l \Delta v_l, \\ \Delta v_u &= T_u^{-1} V_u (r_{B_u} - V_u^{-1} r_{TV_u}) + T_u^{-1} V_u \Delta x, & \Delta t_u &= -V_u^{-1} r_{TV_u} - V_u^{-1} T_u \Delta v_u, \\ \Delta v_{\epsilon_l} &= T_{\epsilon_l}^{-1} V_{\epsilon_l} (r_{B_{\epsilon_l}} - V_{\epsilon_l}^{-1} r_{TV_{\epsilon_l}}) - T_{\epsilon_l}^{-1} V_{\epsilon_l} \Delta \epsilon_l, & \Delta t_{\epsilon_l} &= -V_{\epsilon_l}^{-1} r_{TV_{\epsilon_l}} - V_{\epsilon_l}^{-1} T_{\epsilon_l} \Delta v_{\epsilon_l}, \\ \Delta v_{\epsilon_u} &= T_{\epsilon_u}^{-1} V_{\epsilon_u} (r_{B_{\epsilon_u}} - V_{\epsilon_u}^{-1} r_{TV_{\epsilon_u}}) - T_{\epsilon_u}^{-1} V_{\epsilon_u} \Delta \epsilon_u, & \Delta t_{\epsilon_u} &= -V_{\epsilon_u}^{-1} r_{TV_{\epsilon_u}} - V_{\epsilon_u}^{-1} T_{\epsilon_u} \Delta v_{\epsilon_u}, \end{aligned}$$

- Find α^{aff} such that $(\hat{z}, \hat{s}) + \alpha^{aff} \Delta(\hat{z}, \hat{s}) \geq 0$, where $\hat{z} = (z_{s_l}, z_{s_u}, v_l, v_u, v_{\epsilon_l}, v_{\epsilon_u})$ and $\hat{s} = (s_{s_l}, s_{s_u}, t_l, t_u, t_{\epsilon_l}, t_{\epsilon_u})$

- Compute the duality gap, μ , and the centering parameter, σ (with \bar{m} being the total number of inequality constraints including soft constraints)

$$\mu^{aff} = \frac{(\hat{z} + \alpha^{aff} \Delta \hat{z})^\top (\hat{s} + \alpha^{aff} \Delta \hat{s})}{\bar{m}}, \quad \mu = \frac{\hat{s}^\top \hat{z}}{\bar{m}}, \quad \sigma = \left(\frac{\mu^{aff}}{\mu} \right)^3$$

2. Corrector phase:

- Recompute \tilde{r}_L with the following definitions

$$\begin{aligned} r_{S Z_{s_l}} &\leftarrow r_{S Z_{s_l}} + \Delta S_{s_l}^{aff} \Delta Z_{s_l}^{aff} - \sigma \mu e, & r_{S Z_{s_u}} &\leftarrow r_{S Z_{s_u}} + \Delta S_{s_u}^{aff} \Delta Z_{s_u}^{aff} - \sigma \mu e, \\ r_{TV_l} &\leftarrow r_{TV_l} + \Delta T_l^{aff} \Delta V_l^{aff} - \sigma \mu e, & r_{TV_u} &\leftarrow r_{TV_u} + \Delta T_u^{aff} \Delta V_u^{aff} - \sigma \mu e, \\ r_{TV_{\epsilon_l}} &\leftarrow r_{TV_{\epsilon_l}} + \Delta T_{\epsilon_l}^{aff} \Delta V_{\epsilon_l}^{aff} - \sigma \mu e, & r_{TV_{\epsilon_u}} &\leftarrow r_{TV_{\epsilon_u}} + \Delta T_{\epsilon_u}^{aff} \Delta V_{\epsilon_u}^{aff} - \sigma \mu e. \end{aligned}$$

- Repeat step 1ii-1v from the predictor phase (reapply LDL factorization from predictor phase)

- Compute the step size, α , such that $(\hat{z}, \hat{s}) + \alpha^{aff} \Delta(\hat{z}, \hat{s}) \geq \kappa(\hat{z}, \hat{s})$, for $\kappa = \min(1 - \eta, \mu^{aff})$

- Take step: $\chi = \hat{\chi} + \eta \alpha \Delta \hat{\chi}$, where $\chi = (x, \epsilon_l, \epsilon_u, y, z_{s_l}, z_{s_u}, v_l, v_u, v_{\epsilon_l}, v_{\epsilon_u}, s_{s_l}, s_{s_u}, t_l, t_u, t_{\epsilon_l}, t_{\epsilon_u})$ and $\eta = 0.995$

- Compute scaled KKT-violation,

$$\begin{aligned} \xi &= \max(\tilde{r}_L \|r_L, r_{\epsilon_l}, r_{\epsilon_u}\|_\infty, \tilde{r}_A \|r_A\|_\infty, \tilde{r}_S \|r_{S_l}, r_{S_u}\|_\infty, \tilde{r}_B \|r_{B_l}, r_{B_u}\|_\infty, \|r_{\epsilon_l}, r_{\epsilon_u}\|_\infty, \\ & \quad \|r_{S_{s_l}}, r_{S_{s_u}}, r_{TV_l}, r_{TV_u}, r_{TV_{\epsilon_l}}, r_{TV_{\epsilon_u}}\|_\infty) \end{aligned}$$

Return: $\hat{\chi} = (x, \epsilon_l, \epsilon_u, y, z_{s_l}, z_{s_u}, v_l, v_u, v_{\epsilon_l}, v_{\epsilon_u}, s_{s_l}, s_{s_u}, t_l, t_u, t_{\epsilon_l}, t_{\epsilon_u})$

$$Q_u = \begin{bmatrix} 0 & & & & & & \\ & Q_{\epsilon_u,1} & & & & & \\ & & 0 & & & & \\ & & & Q_{\epsilon_u,2} & & & \\ & & & & \ddots & & \\ & & & & & 0 & \\ & & & & & & Q_{\epsilon_u,N} \end{bmatrix}, \quad (2.29f)$$

$$g = \begin{bmatrix} r_0 & q_1 & r_1 & \cdots & q_{N-1} & r_{N-1} & q_N \end{bmatrix}^\top, \quad (2.29g)$$

$$q_l = \begin{bmatrix} 0 & q_{\epsilon_l,1} & 0 & q_{\epsilon_l,2} & \cdots & 0 & q_{\epsilon_l,N} \end{bmatrix}^\top, \quad (2.29h)$$

$$q_u = \begin{bmatrix} 0 & q_{\epsilon_u,1} & 0 & q_{\epsilon_u,2} & \cdots & 0 & q_{\epsilon_u,N} \end{bmatrix}^\top, \quad (2.29i)$$

$$A = \begin{bmatrix} -B_0^\top & I & & & & & \\ & -A_1^\top & -B_1^\top & I & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -A_{N-1}^\top & -B_{N-1}^\top & I & \end{bmatrix}^\top, \quad (2.29j)$$

$$b = \begin{bmatrix} \tilde{b}_0 & b_1 & \cdots & b_{N-1} \end{bmatrix}^\top, \quad (2.29k)$$

$$l = \begin{bmatrix} u_{\min,0} & -\infty & u_{\min,1} & -\infty & \cdots & u_{\min,N-1} & -\infty \end{bmatrix}^\top, \quad (2.29l)$$

$$u = \begin{bmatrix} u_{\max,0} & \infty & u_{\max,1} & \infty & \cdots & u_{\max,N-1} & \infty \end{bmatrix}^\top, \quad (2.29m)$$

$$S = \begin{bmatrix} 0 & & & & & & \\ & S_1 & & & & & \\ & & 0 & & & & \\ & & & S_2 & & & \\ & & & & \ddots & & \\ & & & & & 0 & \\ & & & & & & S_N \end{bmatrix}, \quad (2.29n)$$

$$l_s = \begin{bmatrix} -\infty & x_{\min,1} & -\infty & x_{\min,2} & \cdots & -\infty & x_{\min,N} \end{bmatrix}^\top, \quad (2.29o)$$

$$u_s = \begin{bmatrix} \infty & x_{\max,1} & \infty & x_{\max,2} & \cdots & \infty & x_{\max,N} \end{bmatrix}^\top, \quad (2.29p)$$

where $\tilde{b}_0 = b_0 + A_0^\top x_0$. We point out that QPIP can solve the OCP (2.27) by applying the definitions (2.29). However, the Riccati based version utilizes the structure, which will result in better computational performance.

In the Riccati version, QPIP utilizes the structure of the QP (2.27) to compute the search direction. As such, QPIP does not apply a standard LDL factorization to solve (2.6), but rather a dedicated structure-utilizing Riccati algorithm.

2.2.1 Search direction

In the Riccati mode, the Newton search direction is on the form (2.6) with the provided matrices in (2.29). Due to space restrictions, we do not write out the full system matrix. The right hand side of the linear

system is

$$r_L = \begin{bmatrix} r_{L,u_0} & r_{L,x_1} & r_{L,u_1} & r_{L,x_2} & \cdots & r_{L,u_{N-1}} & r_{L,x_N} \end{bmatrix}^\top, \quad (2.30a)$$

$$r_{\epsilon_l} = \begin{bmatrix} r_{\epsilon_l,1} & r_{\epsilon_l,2} & \cdots & r_{\epsilon_l,N} \end{bmatrix}^\top, \quad (2.30b)$$

$$r_{\epsilon_u} = \begin{bmatrix} r_{\epsilon_u,1} & r_{\epsilon_u,2} & \cdots & r_{\epsilon_u,N} \end{bmatrix}^\top, \quad (2.30c)$$

$$r_A = \begin{bmatrix} r_{A,0} & r_{A,1} & \cdots & r_{A,N-1} \end{bmatrix}^\top, \quad (2.30d)$$

$$r_{S_l} = \begin{bmatrix} r_{S_l,1} & r_{S_l,2} & \cdots & r_{S_l,N} \end{bmatrix}^\top, \quad (2.30e)$$

$$r_{S_u} = \begin{bmatrix} r_{S_u,1} & r_{S_u,2} & \cdots & r_{S_u,N} \end{bmatrix}^\top, \quad (2.30f)$$

$$r_{B_l} = \begin{bmatrix} r_{B_l,0} & r_{B_l,1} & \cdots & r_{B_l,N-1} \end{bmatrix}^\top, \quad (2.30g)$$

$$r_{B_u} = \begin{bmatrix} r_{B_u,0} & r_{B_u,1} & \cdots & r_{B_u,N-1} \end{bmatrix}^\top, \quad (2.30h)$$

$$r_{\epsilon_l} = \begin{bmatrix} r_{\epsilon_l,1} & r_{\epsilon_l,2} & \cdots & r_{\epsilon_l,N} \end{bmatrix}^\top, \quad (2.30i)$$

$$r_{\epsilon_u} = \begin{bmatrix} r_{\epsilon_u,1} & r_{\epsilon_u,2} & \cdots & r_{\epsilon_u,N} \end{bmatrix}^\top, \quad (2.30j)$$

$$r_{SZ_{s_l}} = \begin{bmatrix} r_{SZ_{s_l},1} & r_{SZ_{s_l},2} & \cdots & r_{SZ_{s_l},N} \end{bmatrix}^\top, \quad (2.30k)$$

$$r_{SZ_{s_u}} = \begin{bmatrix} r_{SZ_{s_u},1} & r_{SZ_{s_u},2} & \cdots & r_{SZ_{s_u},N} \end{bmatrix}^\top, \quad (2.30l)$$

$$r_{TV_l} = \begin{bmatrix} r_{TV_l,0} & r_{TV_l,1} & \cdots & r_{TV_l,N-1} \end{bmatrix}^\top, \quad (2.30m)$$

$$r_{TV_u} = \begin{bmatrix} r_{TV_u,0} & r_{TV_u,1} & \cdots & r_{TV_u,N-1} \end{bmatrix}^\top, \quad (2.30n)$$

$$r_{TV_{\epsilon_l}} = \begin{bmatrix} r_{TV_{\epsilon_l},1} & r_{TV_{\epsilon_l},2} & \cdots & r_{TV_{\epsilon_l},N} \end{bmatrix}^\top, \quad (2.30o)$$

$$r_{TV_{\epsilon_u}} = \begin{bmatrix} r_{TV_{\epsilon_u},1} & r_{TV_{\epsilon_u},2} & \cdots & r_{TV_{\epsilon_u},N} \end{bmatrix}^\top, \quad (2.30p)$$

Currently, QIPM computes the right hand side (2.30) directly from (2.5). However, the algorithm can be improved further by exploiting the structure of the problem and compute individual elements separately.

2.2.2 System reduction

Similarly as in the general case, we eliminate Lagrange multipliers and slack variables. Diagonal matrices are defined as in (2.10) and submatrices are defined with k as subscript. By elimination of Lagrange multipliers and slack variables for inequality constraints and rearranging decision variables, we arrive at the

KKT system (for $N = 3$)

$$\left[\begin{array}{cccc|cccc|ccc} \bar{R}_0 & & & & B_1 & & & & \Delta u_0 & & \bar{r}_{L,u_0} \\ \bar{Q}_1 & E_1 & F_1 & M_1 & -I & A_1 & & & \Delta x_1 & & \bar{r}_{L,x_1} \\ E_1^\top & \bar{Q}_{\epsilon_l,1} & & & & & & & \Delta \epsilon_{l,1} & & \bar{r}_{\epsilon_l,1} \\ F_1^\top & & \bar{Q}_{\epsilon_u,1} & & & & & & \Delta \epsilon_{u,1} & & \bar{r}_{\epsilon_u,1} \\ M_1^\top & & & \bar{R}_1 & & & & & \Delta u_1 & & \bar{r}_{L,u_1} \\ & & & & & & & & \Delta x_2 & & \bar{r}_{L,x_2} \\ & & & \bar{Q}_2 & E_2 & F_2 & M_2 & & \Delta \epsilon_{l,2} & & \bar{r}_{\epsilon_l,2} \\ & & & E_2^\top & \bar{Q}_{\epsilon_l,2} & & & & \Delta \epsilon_{u,2} & & \bar{r}_{\epsilon_u,2} \\ & & & F_2^\top & & \bar{Q}_{\epsilon_u,2} & & & \Delta u_2 & & \bar{r}_{L,u_2} \\ & & & M_2^\top & & & \bar{R}_2 & & \Delta x_3 & & \bar{r}_{L,x_3} \\ & & & & & & & & \Delta \epsilon_{l,3} & & \bar{r}_{\epsilon_l,3} \\ & & & & & & \bar{Q}_3 & E_3 & F_3 & & \bar{r}_{\epsilon_u,3} \\ & & & & & & E_3^\top & \bar{Q}_{\epsilon_l,3} & & & \bar{r}_{A,0} \\ & & & & & & F_3^\top & & \bar{Q}_{\epsilon_u,3} & & \bar{r}_{A,1} \\ \hline B_0^\top & -I & & & & & & & \Delta y_0 & & \bar{r}_{A,2} \\ & A_1^\top & & B_1^\top & -I & & & & \Delta y_1 & & \\ & & & A_2^\top & & B_2^\top & -I & & \Delta y_2 & & \end{array} \right] = \left[\begin{array}{c} \bar{r}_{L,u_0} \\ \bar{r}_{L,x_1} \\ \bar{r}_{\epsilon_l,1} \\ \bar{r}_{\epsilon_u,1} \\ \bar{r}_{L,u_1} \\ \bar{r}_{L,x_2} \\ \bar{r}_{\epsilon_l,2} \\ \bar{r}_{\epsilon_u,2} \\ \bar{r}_{L,u_2} \\ \bar{r}_{L,x_3} \\ \bar{r}_{\epsilon_l,3} \\ \bar{r}_{\epsilon_u,3} \\ \bar{r}_{A,0} \\ \bar{r}_{A,1} \\ \bar{r}_{A,2} \end{array} \right], \quad (2.31)$$

where

$$E_k = S_k D_{s_l,k}, \quad k = 1, \dots, N, \quad (2.32a)$$

$$F_k = -S_k D_{s_u,k}, \quad k = 1, \dots, N, \quad (2.32b)$$

$$\bar{Q}_k = Q_k + E_k S_k^\top - F_k S_k^\top, \quad k = 1, \dots, N, \quad (2.32c)$$

$$\bar{Q}_{\epsilon_l,k} = Q_{\epsilon_l,k} + D_{\epsilon_l,k} + D_{s_l,k}, \quad k = 1, \dots, N, \quad (2.32d)$$

$$\bar{Q}_{\epsilon_u,k} = Q_{\epsilon_u,k} + D_{\epsilon_u,k} + D_{s_u,k}, \quad k = 1, \dots, N, \quad (2.32e)$$

$$\bar{R}_k = R_k + D_{l,k} + D_{u,k}, \quad k = 0, \dots, N - 1. \quad (2.32f)$$

and

$$\bar{r}_{L,x_k} = -r_{L,x_k} + S_k (S_{s_l,k}^{-1} Z_{s_l,k} (r_{s_l,k} - Z_{s_l,k}^{-1} r_{S Z_{s_l,k}})) - S_k (S_{s_u,k}^{-1} Z_{s_u,k} (r_{s_u,k} - Z_{s_u,k}^{-1} r_{S Z_{s_u,k}})), \quad k = 1, \dots, N, \quad (2.33a)$$

$$\bar{r}_{\epsilon_l,k} = -r_{\epsilon_l,k} + T_{\epsilon_l,k}^{-1} V_{\epsilon_l,k} (r_{B_{\epsilon_l,k}} - V_{\epsilon_l,k}^{-1} r_{TV_{\epsilon_l,k}}) + S_{s_l,k}^{-1} Z_{s_l,k} (r_{s_l,k} - Z_{s_l,k}^{-1} r_{S Z_{s_l,k}}), \quad k = 1, \dots, N, \quad (2.33b)$$

$$\bar{r}_{\epsilon_u,k} = -r_{\epsilon_u,k} + T_{\epsilon_u,k}^{-1} V_{\epsilon_u,k} (r_{B_{\epsilon_u,k}} - V_{\epsilon_u,k}^{-1} r_{TV_{\epsilon_u,k}}) + S_{s_u,k}^{-1} Z_{s_u,k} (r_{s_u,k} - Z_{s_u,k}^{-1} r_{S Z_{s_u,k}}), \quad k = 1, \dots, N, \quad (2.33c)$$

$$\bar{r}_{L,u_k} = -r_{L,u_k} + T_{l,k} V_{l,k} (r_{B_{l,k}} - V_{l,k}^{-1} r_{TV_{l,k}}) - T_{u,k}^{-1} V_{u,k} (r_{B_{u,k}} - V_{u,k}^{-1} r_{TV_{u,k}}), \quad k = 0, \dots, N - 1, \quad (2.33d)$$

$$\bar{r}_{A,k} = -r_{A,k}, \quad k = 0, \dots, N - 1. \quad (2.33e)$$

The eliminated Lagrange multipliers and slack variables are

$$\Delta v_{l,k} = T_{l,k}^{-1} V_{l,k} (r_{B_{l,k}} - V_{l,k}^{-1} r_{TV_{l,k}}) - T_{l,k}^{-1} V_{l,k} \Delta x_k, \quad k = 0, \dots, N-1, \quad (2.34a)$$

$$\Delta v_{u,k} = T_{u,k}^{-1} V_{u,k} (r_{B_{u,k}} - V_{u,k}^{-1} r_{TV_{u,k}}) + T_{u,k}^{-1} V_{u,k} \Delta x_k, \quad k = 0, \dots, N-1, \quad (2.34b)$$

$$\Delta v_{\epsilon_l,k} = T_{\epsilon_l,k}^{-1} V_{\epsilon_l,k} (r_{B_{\epsilon_l,k}} - V_{\epsilon_l,k}^{-1} r_{TV_{\epsilon_l,k}}) - T_{\epsilon_l,k}^{-1} V_{\epsilon_l,k} \Delta \epsilon_{l,k}, \quad k = 1, \dots, N, \quad (2.34c)$$

$$\Delta v_{\epsilon_u,k} = T_{\epsilon_u,k}^{-1} V_{\epsilon_u,k} (r_{B_{\epsilon_u,k}} - V_{\epsilon_u,k}^{-1} r_{TV_{\epsilon_u,k}}) - T_{\epsilon_u,k}^{-1} V_{\epsilon_u,k} \Delta \epsilon_{u,k}, \quad k = 1, \dots, N, \quad (2.34d)$$

$$\begin{aligned} \Delta z_{s_l,k} &= S_{s_l,k}^{-1} Z_{s_l,k} (r_{S_{l,k}} - Z_{s_l,k}^{-1} r_{SZ_{s_l,k}}) \\ &\quad - S_{s_l,k}^{-1} Z_{s_l,k} (S_k^\top \Delta x_k + \Delta \epsilon_{l,k}), \end{aligned} \quad k = 1, \dots, N, \quad (2.34e)$$

$$\begin{aligned} \Delta z_{s_u,k} &= S_{s_u,k}^{-1} Z_{s_u,k} (r_{S_{u,k}} - Z_{s_u,k}^{-1} r_{SZ_{s_u,k}}) \\ &\quad - S_{s_u,k}^{-1} Z_{s_u,k} (-S_k^\top \Delta x_k + \Delta \epsilon_{u,k}), \end{aligned} \quad k = 1, \dots, N, \quad (2.34f)$$

$$\Delta t_{l,k} = -V_{l,k}^{-1} r_{TV_{l,k}} - V_{l,k}^{-1} T_{l,k} \Delta v_{l,k}, \quad k = 0, \dots, N-1, \quad (2.34g)$$

$$\Delta t_{u,k} = -V_{u,k}^{-1} r_{TV_{u,k}} - V_{u,k}^{-1} T_{u,k} \Delta v_{u,k}, \quad k = 0, \dots, N-1, \quad (2.34h)$$

$$\Delta t_{\epsilon_l,k} = -V_{\epsilon_l,k}^{-1} r_{TV_{\epsilon_l,k}} - V_{\epsilon_l,k}^{-1} T_{\epsilon_l,k} \Delta v_{\epsilon_l,k}, \quad k = 1, \dots, N, \quad (2.34i)$$

$$\Delta t_{\epsilon_u,k} = -V_{\epsilon_u,k}^{-1} r_{TV_{\epsilon_u,k}} - V_{\epsilon_u,k}^{-1} T_{\epsilon_u,k} \Delta v_{\epsilon_u,k}, \quad k = 1, \dots, N, \quad (2.34j)$$

$$\Delta s_{s_l,k} = -Z_{s_l,k}^{-1} r_{SZ_{s_l,k}} - Z_{s_l,k}^{-1} S_{s_l,k} \Delta z_{s_l,k}, \quad k = 1, \dots, N, \quad (2.34k)$$

$$\Delta s_{s_u,k} = -Z_{s_u,k}^{-1} r_{SZ_{s_u,k}} - Z_{s_u,k}^{-1} S_{s_u,k} \Delta z_{s_u,k}, \quad k = 1, \dots, N. \quad (2.34l)$$

We eliminate the soft constraint slack variables. The resulting system is (for $N = 3$)

$$\left[\begin{array}{ccc|cc} \tilde{R}_0 & & & B_0 & \\ & \tilde{Q}_1 & M_1 & -I & A_1 \\ & M_1^\top & \tilde{R}_1 & & B_1 \\ & & & -I & A_2 \\ & & \tilde{Q}_2 & M_2 & \\ & & M_2^\top & \tilde{R}_2 & B_2 \\ & & & & -I \\ & & & & \tilde{Q}_3 \\ \hline B_0^\top & -I & & & \\ & A_1^\top & B_1^\top & -I & \\ & & A_2^\top & B_2^\top & -I \end{array} \right] \left[\begin{array}{c} \Delta u_0 \\ \Delta x_1 \\ \Delta u_1 \\ \Delta x_2 \\ \Delta u_2 \\ \Delta x_3 \\ \Delta y_0 \\ \Delta y_1 \\ \Delta y_2 \end{array} \right] = \left[\begin{array}{c} \tilde{r}_{L,u_0} \\ \tilde{r}_{L,x_1} \\ \tilde{r}_{L,u_1} \\ \tilde{r}_{L,x_2} \\ \tilde{r}_{L,u_2} \\ \tilde{r}_{L,x_3} \\ \tilde{r}_{A,0} \\ \tilde{r}_{A,1} \\ \tilde{r}_{A,2} \end{array} \right], \quad (2.35)$$

where

$$\tilde{Q}_k = \bar{Q}_k - E_k \bar{Q}_{\epsilon_l,k}^{-1} E_k^\top - F_k \bar{Q}_{\epsilon_u,k}^{-1} F_k^\top, \quad k = 1, \dots, N, \quad (2.36a)$$

$$\tilde{R}_k = \bar{R}_k, \quad k = 0, \dots, N-1, \quad (2.36b)$$

$$\tilde{r}_{L,x_k} = \bar{r}_{L,x_k} + E_k \bar{Q}_{\epsilon_l,k}^{-1} \bar{r}_{\epsilon_l,k} + F_k \bar{Q}_{\epsilon_u,k}^{-1} \bar{r}_{\epsilon_u,k}, \quad k = 1, \dots, N, \quad (2.36c)$$

$$\tilde{r}_{L,u_k} = \bar{r}_{L,u_k}, \quad k = 0, \dots, N-1, \quad (2.36d)$$

$$\tilde{r}_A = \bar{r}_A, \quad k = 0, \dots, N-1. \quad (2.36e)$$

and the eliminated slack variables are

$$\Delta \epsilon_{l,k} = \bar{Q}_{\epsilon_l,k}^{-1} (\bar{r}_{\epsilon_l,k} - E_k^\top \Delta x_k), \quad (2.37a)$$

$$\Delta \epsilon_{u,k} = \bar{Q}_{\epsilon_u,k}^{-1} (\bar{r}_{\epsilon_u,k} - F_k^\top \Delta x_k). \quad (2.37b)$$

The KKT-system (2.35) can be solved with Riccati recursion, and finally the remaining part of the search direction can be compute from (2.37) and (2.34).

2.2.3 Riccati recursion algorithm

We apply a Riccati recursion based algorithm to solve structured systems of linear equations in the form (2.35). For simplicity of notation, we write the system (2.35) as

$$\left[\begin{array}{ccc|cc} R_0 & & & B_0 & \\ & Q_1 & M_1 & -I & A_1 \\ & M_1^\top & R_1 & & B_1 \\ & & & -I & A_2 \\ & & Q_2 & M_2 & \\ & & M_2^\top & R_2 & B_2 \\ & & & & -I \\ \hline & & & P_3 & \\ \hline B_0^\top & -I & & & \\ & A_1^\top & B_1^\top & -I & \\ & & A_2^\top & B_2^\top & -I \end{array} \right] \begin{bmatrix} \Delta u_0 \\ \Delta x_1 \\ \Delta u_1 \\ \Delta x_2 \\ \Delta u_2 \\ \Delta x_3 \\ \Delta y_0 \\ \Delta y_1 \\ \Delta y_2 \end{bmatrix} = - \begin{bmatrix} r_0 \\ q_1 \\ r_1 \\ q_2 \\ r_2 \\ p_3 \\ b_0 \\ b_1 \\ b_2 \end{bmatrix}, \quad (2.38)$$

We point out that the data in (2.38) should not be confused with variables with similar names previously introduced. Algorithm 2 and 3 introduce the factorization and solution phase of the Riccati recursion algorithm (Jørgensen 2004, Wahlgreen and Jørgensen 2022). We point out that the Algorithm 2 returns the cholesky factorization of $R_{e,k}$, which QPIPM applies in Algorithm 3 to solve the linear systems involving $R_{e,k}$.

Note the negation on the right hand side in (2.38). Before calling the Riccati algorithm to solve (2.35), QPIPM negates the right hand side of the system (2.35) such that it is in the form (2.38).

2.2.4 Algorithm

In Riccati mode, QPIPM follows the steps in Algorithm 1, where the LDL-factorization step and LDL-solve step are replaced with the the Riccati factorization and Riccati solve algorithms in Algorithm 2 and 3.

2.2.5 A note on bounds

The Riccati recursion part of QPIPM does allow for hard output constraints, i.e., box constraints on x_k (we have not provided these equations here, but they are easily included based on the input, u_k , box constraints). Therefore, if elements corresponding to x_k in l and/or u are not set to infinity, QPIPM does include the bound. On the other hand, QPIPM does not support soft input constraints in Riccati mode. Therefore, elements corresponding to the inputs, u_k , in S will never be accessed even if the corresponding values of l_s and/or l_u are not set to infinity. QPIPM does however require the entries in S corresponding to u_k to be set. We have implemented QPIPM in this way such that one can turn Riccati mode on and off without changing the provided QP formulation.

Algorithm 2: Riccati factorization

Input: $\{R_k, Q_k, M_k, A_k, B_k\}_{k=0}^{N-1}, P_N$.

1. Compute,

$$\begin{aligned} R_{e,k} &= R_k + B_k P_{k+1} B_k^\top, \\ K_k &= -R_{e,k}^{-1} (M_k^\top + B_k P_{k+1} A_k^\top), \\ P_k &= Q_k + A_k P_{k+1} A_k^\top - K_k^\top R_{e,k} K_k, \end{aligned}$$

for $k = N - 1, N - 2, \dots, 1$ and

$$R_{e,0} = R_0 + B_0 P_1 B_0^\top.$$

Return: $\{R_{e,k}, \text{chol}(R_{e,k}), P_{k+1}\}_{k=0}^{N-1}, \{K_k\}_{k=1}^{N-1}$.

Algorithm 3: Riccati solution

Input: $\{Q_k, M_k, A_k, B_k, R_{e,k}, \text{chol}(R_{e,k}), P_{k+1}\}_{k=0}^{N-1}, \{K_k\}_{k=1}^{N-1}$.

1. Compute,

$$\begin{aligned} a_k &= -R_{e,k}^{-1} (r_k + B_k (P_{k+1} b_k + p_{k+1})), \\ p_k &= q_k + A_k (P_{k+1} b_k + p_{k+1}) + K_k^\top (r_k + B_k (P_{k+1} b_k + p_{k+1})), \end{aligned}$$

for $k = N - 1, N - 2, \dots, 1$ and

$$a_0 = -R_{e,0}^{-1} (r_0 + B_0 (P_1 \tilde{b}_0 + p_1)).$$

2. Compute the solution, $\{\Delta u_k, \Delta x_{k+1}\}_{k=0}^{N-1}$,

$$\begin{aligned} \Delta u_0 &= a_0, \\ \Delta x_1 &= B_0^\top \Delta u_0 + \tilde{b}_0, \end{aligned}$$

and

$$\begin{aligned} \Delta u_k &= K_k \Delta x_k + a_k, \\ \Delta x_{k+1} &= A_k^\top \Delta x_k + B_k^\top \Delta u_k + b_k, \end{aligned}$$

for $k = 1, 2, \dots, N - 1$.3. Compute the Lagrange multipliers, $\{\Delta y_k\}_{k=0}^{N-1}$,

$$\begin{aligned} \Delta y_{N-1} &= P_N \Delta x_N + p_N, \\ \Delta y_{k-1} &= A_k \Delta y_k + Q_k \Delta x_k + M_k \Delta u_k + q_k, \end{aligned}$$

for $k = N - 1, N - 2, \dots, 1$.**Return:** $\{\Delta u_k, \Delta x_{k+1}, \Delta y_k\}_{k=0}^{N-1}$.

Implementation of QIPM in Matlab and C

In this chapter, we introduce how QIPM can be called in both Matlab and in C. Both versions are part of a gitlab-repository, which can be cloned with the command line command

```
git clone https://gitlab.gbar.dtu.dk/SCGroup/QIPM.git
```

3.1 Matlab

QIPM in Matlab has the following interface:

```
1 function [x, stat] = QIPM(H, g, A, b, C, d, l, u, options, ls, S, us, Ql, Qu, ql, qu)
```

Inputs:

The inputs $H, g, A, b, l, u, ls, us, S, Ql, Qu, ql,$ and qu are as in (2.1). The inputs C and d implements general inequality constraints in the form

$$C^T x \geq d. \quad (3.1)$$

The general inequality constraints (3.1) have only been included in QIPM for testing purposes and are ignored in Riccati mode. The inputs $ls - qu$ can be left empty in which case QIPM solves a problem without soft constraints. The `options` input is a structure with the following fields

<code>print</code>	0 or 1 to print iteration information	Default: 1
<code>tol</code>	Convergence tolerance	Default: 10^{-8}
<code>maxit</code>	Maximum iterations	Default: 100
<code>riccati</code>	0 or 1 to turn Riccati mode off/on	Default: 0
<code>N</code>	Horizon. Required if <code>riccati=1</code>	Default: NaN

We point out that QIPM takes the same inputs and have the same outputs when Riccati mode is off and on. When applying Riccati mode, QIPM assumes that the provided matrices are structured as described in section 2.2. QIPM will not check that this is the case. Therefore, Riccati mode can be applied for a non-structured QP, but the result will likely be wrong.

Outputs:

The output x is the solution at convergence or after maximum iterations are reached. QIPM prints a warning message in the case that maximum iterations are reached. The `stat` output is a structure with the

following fields

obj	Objective value at solution
conv	0 (not converged) or 1 (converged)
iter	Number of iterations
lamEq	Lagrange multipliers for equality constraints
lamIneq	Lagrange multipliers for inequality constraints
lamBn	Lagrange multipliers for bound constraints
lamZs	Lagrange multipliers for soft constraints
lamEpsBn	Lagrange multipliers for ϵ -bound constraints
eps	ϵ -slack variables

3.2 C

As previously mentioned, the C version of QIPM does currently not have the option to include soft constraints. QIPM in C has the following interface:

```

1 void QIPM(
2     // Inputs
3     struct mat *H      ,
4     struct vec *g      ,
5     struct mat *A      ,
6     struct vec *b      ,
7     struct mat *C      ,
8     struct vec *d      ,
9     struct vec *l      ,
10    struct vec *u      ,
11    void *optionsIn    ,
12    mem *memory        ,
13
14    // Outputs
15    struct vec *x      ,
16    void *statIn
17 )

```

The structures `vec`, `mat`, and `mem` are vector, matrix, and memory structures, respectively. These structures are defined in the dependency `SCInterface`, which is shortly introduced in section 3.2.2. In the following, we introduce the inputs and outputs of the C version.

Inputs:

The inputs H , g , A , b , C , d , l , and u are as in the Matlab version. The `optionsIn` input is a `options` structure of type `optionsQPIPM_t`, which has the fields

<code>print</code>	0 or 1 to print iteration information	Default: 1
<code>tol</code>	Convergence tolerance	Default: 10^{-8}
<code>maxit</code>	Maximum iterations	Default: 100
<code>riccati</code>	0 or 1 to turn Riccati mode off/on	Default: 0
<code>N</code>	Horizon. Required if <code>riccati=1</code>	Default: NaN
<code>bigN</code>	Numbers above treated as infinity	Default: 10^{20}

The input `memory` is a structure of type `mem`, which contains sufficient integer and double memory for QPIPM (see section 3.2.1).

Outputs:

The output x is the solution at convergence or after maximum iterations are reached. Similarly to the Matlab version, QPIPM in C prints a warning message if the maximum number of iterations are reached. The `stat` structure is of type `statQPIPM_t` and has the following fields

<code>obj</code>	Objective value at solution
<code>conv</code>	0 (not converged) or 1 (converged)
<code>iter</code>	Number of iterations
<code>lamEq</code>	Lagrange multipliers for equality constraints
<code>lamIneq</code>	Lagrange multipliers for inequality constraints
<code>lamBn</code>	Lagrange multipliers for bound constraints

3.2.1 Memory allocation

QPIPM requires both integer and double workspace, which should be allocated in the input memory structure. QPIPM features the function

```
1 void workspaceQPIPM( int n, int me, int mi, int *iwork, int *dwork )
```

which given the dimensions of the QP, n (decision variables), me (equality constraints), and mi (inequality constraints), computes the required workspace for QPIPM. Then the amount of integer workspace, `iwork`, and double workspace, `dwork`, can be used to initialize the `memory` input with sufficient memory. Additionally, the `stat` structure for the output is required to be initialized, which can be done with the function

```
1 void createStatQPIPM( const int n, const int me, const int mi, statQPIPM_t
  *const stat )
```

`createStatQPIPM` allocates the required memory for the output `stat` structure. Note that when finished using the `stat` structure, the memory can be freed with the function

```
1 void destroyStatQPIPM( statQPIPM_t *stat )
```

3.2.2 Dependencies

QPIPM is a part of the private gitlib-repository `SCProject`, which is a project containing a series of git repositories. QPIPM is dependent on the following two repositories in `SCProject`

<code>SCInterface</code>	A set of structure and function definitions
<code>linalg</code>	A set of vector and matrix linear algebra functions

Additionally, `linalg` is BLAS dependent and requires linking to a BLAS installation on the system.

3.2.3 Gitlab

The private Gitlab group `SCGroup` grants access to all projects contained in `SCProject`. Therefore, the three projects, `QPIPM`, `SCInterface`, and `linalg` are also included in `SCGroup`. When access is granted to `SCGroup`, one can clone the whole `SCProject` or parts of it. To apply QPIPM, one has to clone `QPIPM`, `SCInterface`, and `linalg` (and install a version of BLAS). The C version of QPIPM includes a `settings.mk` file where the dependency paths can be set. The three git repositories can be cloned with the following command line commands (accompanied with a username and password):

```
git clone https://gitlab.gbar.dtu.dk/SCGroup/SCInterface.git
git clone https://gitlab.gbar.dtu.dk/SCGroup/linalg.git
git clone https://gitlab.gbar.dtu.dk/SCGroup/QPIPM.git
```

3.2.4 Doxygen documentation

The C version of QPIPM is documented with Doxygen. The Doxygen documentation is available in `QPIPM/C/docs`, which can be compiled by typing `doxygen` in the command line. Afterwards, the documentation is available in `QPIPM/C/docs/results/html/index.html`, which will open in a browser. The documentation includes descriptions of all QPIPM functions and their inputs and outputs. Note, this requires an installation of Doxygen on the system.

3.3 Examples

Both the Matlab and C version of QPIPM has a few test examples. The Matlab version has a driver to test the implementation on a linearized four tank system. The C version includes a simple test example and a few examples showing that the algorithm can be called in parallel to solve multiple QPs. The examples can be found in the `examples` folder in the Matlab and C version of QPIPM.

Note: The C version of QPIPM is thread-safe such that it can be called in parallel to solve multiple QPs. This feature requires linking to a thread-safe BLAS library, e.g., BLASFEO (Frison et al. 2018, 2020).

Conclusion

In this part, we introduced the Riccati based primal-dual interior-point software, QPIPM, to solve structured quadratic programming problems (QPs). QPIPM is a software package that is stored in a private gitlab-repository `QPIPM`, which is part of the project `SCPproject`. QPIPM has a Matlab version and a C version, where the Matlab version is intended for testing purposes and have not been implemented for computational speed. The C version is thread-safe due to internal distribution of memory allocated prior to calling QPIPM. QPIPM can solve QPs with equality constraints, box constraints, and soft constraints. However, currently only the Matlab version supports soft constraints. We have provided the mathematical details of QPIPM and introduced the implementation of QPIPM in both Matlab and C. We have provided the interfaces of the implementations and described the inputs and outputs. In the C version, we have elaborated on how to allocate the needed memory and how to link to the introduced dependencies.

Part II

NLPSQP

Introduction

We introduce the sequential quadratic programming (SQP) software, NLPSQP (nonlinear-programming-sequential-quadratic-programming), for solution of nonlinear programming problems (NLPs). NLPSQP applies an iterative sequential quadratic programming (SQP) algorithm. In each iteration, NLPSQP performs three major steps, 1) solve a quadratic programming problem (QP) subproblem with QPIPM, 2) apply a line-search algorithm to ensure sufficient decrease in a merit function, and 3) perform a Broyden–Fletcher–Goldfarb–Shanno (BFGS) update to avoid the need of evaluating second order derivatives. NLPSQP supports a Riccati mode for solution of structured problems arising in optimal control problems (OCPs). NLPSQP is intended for use in nonlinear model predictive control (NMPC) and economic NMPC (ENMPC) applications. We have implemented NLPSQP in both a Matlab version and a C version. The Matlab version is intended for testing purposes, while the C version is intended for uncertainty quantification studies of closed-loop systems with Monte Carlo simulation. To that end, the C version of NLPSQP is implemented thread-safe to enable parallel scaling in Monte Carlo simulations, i.e., NLPSQP can be called in parallel to solve different NLPs. The thread-safety of NLPSQP is ensured by internally distributing memory allocated prior to calling NLPSQP. Similarly to QPIPM, the Matlab version supports soft constraints, while the C version lacks this feature due to time constraints. The current implementation of NLPSQP is still work in progress and can likely be optimized for better performance. However, the most computational work is done in QPIPM.

In this report, we introduce the mathematical details in the NLPSQP implementation and introduce the interfaces of NLPSQP in both Matlab and C. NLPSQP is stored in a private gitlab-repository `NLPSQP` and is part of the project `SCPproject`, which is implemented in C and contains a number of other gitlab-repositories. For the C version, we introduce the other dependencies in `SCPproject` and explain how to allocate the required memory prior to calling NLPSQP.

We point out that the implementation of NLPSQP is highly inspired by previous work (Wächter and Biegler 2006, Kaysfeld et al. 2023).

Mathematical details

We have developed NLPSQP to solve NLPs with equality constraints, box constraints, and soft constraints in the form,

$$\min_{x, \epsilon_l, \epsilon_u} f(x) + Q(\epsilon_l, \epsilon_u), \quad (6.1a)$$

$$s.t. \quad g(x) = 0, \quad (6.1b)$$

$$l \leq x \leq u, \quad (6.1c)$$

$$l_s - \epsilon_l \leq s(x) \leq u_s + \epsilon_u, \quad (6.1d)$$

$$\epsilon_l, \epsilon_u \geq 0, \quad (6.1e)$$

where $x \in \mathbb{R}^n$, $\epsilon_l \in \mathbb{R}^{m_s}$, $\epsilon_u \in \mathbb{R}^{m_s}$, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $Q : \mathbb{R}^{2m_s} \rightarrow \mathbb{R}$, $g : \mathbb{R}^n \rightarrow \mathbb{R}^{m_e}$, $l \in \mathbb{R}^n$, $u \in \mathbb{R}^n$, $s : \mathbb{R}^n \rightarrow \mathbb{R}^{m_s}$, $l_s \in \mathbb{R}^{m_s}$, and $u_s \in \mathbb{R}^{m_s}$. We point out that l and u can have elements set to $-\infty$ and ∞ in which case NLPSQP eliminates these constraints in a pre-computing phase. We let m_l and m_u denote the actual number of lower bounds and upper bounds after elimination of ∞ -bounds, respectively.

The penalty function, $Q(\cdot)$, is a combination of quadratic and linear terms similarly to the penalty term the QP solved in QPIPM,

$$Q(\epsilon_l, \epsilon_u) = \frac{1}{2} \epsilon_l^\top Q_l \epsilon_l + q_l^\top \epsilon_l + \frac{1}{2} \epsilon_u^\top Q_u \epsilon_u + q_u^\top \epsilon_u, \quad (6.2)$$

where we assume that $Q_l \in \mathbb{R}^{m_s \times m_s}$ and $Q_u \in \mathbb{R}^{m_s \times m_s}$ are diagonal matrices.

NLPSQP is an iterative algorithm that in each iteration goes through the following three major steps,

- Compute the search-direction by solving a QP-subproblem,
- A line-search algorithm to ensure sufficient decrease in a merit function,
- A BFGS update for Lagrangian Hessian approximation.

We let the superscript $[l]$ denote the l 'th iteration of NLPSQP. In each iteration NLPSQP takes the step

$$x^{[l+1]} = x^{[l]} + \alpha^{[l]} \Delta x^{[l]}, \quad (6.3)$$

where $\alpha^{[l]}$ is a step-size computed by the line-search algorithm to ensure sufficient decrease in a merit function and $\Delta x^{[l]}$ is the search direction computed as the solution to the QP-subproblem.

NLPSQP features a version to solve the general NLP (6.1) and a Riccati recursion based version to solve structured NLPs arising in OCPs. The non-Riccati version is not optimized and primarily implemented for testing purposes.

6.1 Sequential quadratic programming algorithm

In this section, we introduce the SQP algorithm implemented in NLPSQP to solve soft constrained NLPs in the form (6.1).

6.1.1 Optimality conditions

We define the Lagrangian function for (6.1), which is

$$\begin{aligned} \mathcal{L}(x, \lambda, \pi_l, \pi_u, \pi_{\epsilon_l}, \pi_{\epsilon_u}, \pi_{l_s}, \pi_{u_s}) = & f(x) + Q(\epsilon_l, \epsilon_u) - \lambda^\top g(x) - \pi_l^\top (x - l) - \pi_u^\top (u - x) \\ & - \pi_{\epsilon_l}^\top \epsilon_l - \pi_{\epsilon_u}^\top \epsilon_u - \pi_{l_s}^\top (s(x) - l_s + \epsilon_l) - \pi_{u_s}^\top (u_s + \epsilon_u - s(x)). \end{aligned} \quad (6.4)$$

$\lambda \in \mathbb{R}^m$ are equality constraint Lagrange multipliers, $\pi_l \in \mathbb{R}^n$ are lower bound Lagrange multipliers, $\pi_u \in \mathbb{R}^n$ are upper bound Lagrange multipliers, $\pi_{\epsilon_l} \in \mathbb{R}^{m_s}$ are ϵ_l -non-negativity Lagrange multipliers, $\pi_{\epsilon_u} \in \mathbb{R}^{m_s}$ are ϵ_u -non-negativity Lagrange multipliers, $\pi_{l_s} \in \mathbb{R}^{m_s}$ are lower soft constraint Lagrange multipliers, and $\pi_{u_s} \in \mathbb{R}^{m_s}$ are upper soft constraint Lagrange multipliers. The Lagrangian gradient with respect to the decision variables, x , and the soft constraint Lagrange multipliers, ϵ_l and ϵ_u , is then

$$\nabla_x \mathcal{L} = \nabla f(x) - \nabla g(x) \lambda - \pi_l + \pi_u - \nabla s(x) \pi_{l_s} + \nabla s(x) \pi_{u_s}, \quad (6.5a)$$

$$\nabla_{\epsilon_l} \mathcal{L} = \nabla_{\epsilon_l} Q(\epsilon_l, \epsilon_u) - \pi_{\epsilon_l} - \pi_{l_s}, \quad (6.5b)$$

$$\nabla_{\epsilon_u} \mathcal{L} = \nabla_{\epsilon_u} Q(\epsilon_u, \epsilon_u) - \pi_{\epsilon_u} - \pi_{u_s}, \quad (6.5c)$$

where $\mathcal{L}(x, \lambda, \pi_l, \pi_u, \pi_{\epsilon_l}, \pi_{\epsilon_u}, \pi_{l_s}, \pi_{u_s})$. The first order KKT-conditions for (6.1) are given as

$$\nabla_x \mathcal{L} = 0, \quad (6.6a)$$

$$\nabla_{\epsilon_l} \mathcal{L} = 0, \quad (6.6b)$$

$$\nabla_{\epsilon_u} \mathcal{L} = 0, \quad (6.6c)$$

$$g(x) = 0, \quad (6.6d)$$

$$x - l \geq 0, \quad u - x \geq 0, \quad (6.6e)$$

$$s(x) - l_s + \epsilon_l \geq 0, \quad u_s + \epsilon_u - s(x) \geq 0, \quad (6.6f)$$

$$\epsilon_l \geq 0, \quad \epsilon_u \geq 0. \quad (6.6g)$$

6.1.2 Quadratic programming subproblem

NLPSQP solves a QP-subproblem in each iteration to get the search direction. For simplicity of notation, we disregard the iteration superscript $[l]$ in this section ($x = x^{[l]}$, $\Delta x = \Delta x^{[l]}$, $\epsilon_l = \epsilon_l^{[l]}$, $\epsilon_u = \epsilon_u^{[l]}$, $W = W^{[l]}$). The QP-subproblem solved in NLPSQP is

$$\min_{\Delta x, \epsilon_l, \epsilon_u} \quad \frac{1}{2} \Delta x^\top W \Delta x + \nabla f(x)^\top \Delta x + Q(\epsilon_l, \epsilon_u), \quad (6.7a)$$

$$s.t. \quad \nabla g(x)^\top \Delta x = -g(x), \quad (6.7b)$$

$$l - x \leq \Delta x \leq u - x, \quad (6.7c)$$

$$l_s - s(x) - \epsilon_l \leq \nabla s(x)^\top \Delta x \leq u_s - s(x) + \epsilon_u, \quad (6.7d)$$

$$\epsilon_l, \epsilon_u \geq 0. \quad (6.7e)$$

W is a BFGS approximation of the second order derivative of the Lagrangian. We denote the Lagrange multipliers of the QP-subproblem (6.7) as: μ for equality constrain, τ_l and τ_u for bound constraints, and τ_{l_s} and τ_{u_s} for soft constraints. We point out that the slack variables ϵ_l and ϵ_u in the QP-subproblem (6.7)

are identical to those in the original NLP (6.1). Therefore, the Lagrange multipliers for the ϵ -bounds in the QP-subproblem are exactly π_{ϵ_l} and π_{ϵ_u} , i.e., the Lagrange multipliers from the original NLP (6.1).

The Lagrange multipliers of the QP-subproblem (6.7) are related to the Lagrange multipliers of the original NLP (6.1) as

$$\mu = \lambda + \Delta\lambda, \quad (6.8a)$$

$$\tau_l = \pi_l + \Delta\pi_l, \quad \tau_u = \pi_u + \Delta\pi_u, \quad (6.8b)$$

$$\tau_{l_s} = \pi_{l_s} + \Delta\pi_{l_s}, \quad \tau_{u_s} = \pi_{u_s} + \Delta\pi_{u_s}. \quad (6.8c)$$

Using the relation (6.8), we can compute the search direction for the Lagrange multipliers,

$$(\Delta\lambda, \Delta\pi_l, \Delta\pi_u, \Delta\pi_{l_s}, \Delta\pi_{u_s}). \quad (6.9)$$

We point out that the solution to the QP-subproblem (6.7) ensures to satisfy the linear constraints in the original NLP (6.1), i.e., the bound constraints (6.1c) and the ϵ -non-negativity constraints (6.1e). Note also that the QP-subproblem (6.7) is in the form (2.1) and can be solved with QPIP.

6.1.3 Line-search

NLPSQP applies a backtracking line-search algorithm to compute the step-size, α , that ensures sufficient degrees in Powell's l_1 -merit function (Powell 1978, Jørgensen 2004). We have adapted the merit function to include soft constraint

$$P(x) = f(x) + \sigma^\top |g(x)| + \kappa_l^\top |\min(0, s(x) - l_s + \epsilon_l)| + \kappa_u^\top |\max(0, s(x) - u_s - \epsilon_u)|. \quad (6.10)$$

The j 'th element of the vectors, σ , κ_l , and κ_u , are defined as

$$\sigma_j = \max\left(|\mu_j|, \frac{1}{2}(\sigma_j + |\mu_j|)\right), \quad j = 1, \dots, m, \quad (6.11a)$$

$$\kappa_{l,j} = \max\left(|\tau_{l_s,j}|, \frac{1}{2}(\kappa_{l,j} + |\tau_{l_s,j}|)\right), \quad j = 1, \dots, m_s, \quad (6.11b)$$

$$\kappa_{u,j} = \max\left(|\tau_{u_s,j}|, \frac{1}{2}(\kappa_{u,j} + |\tau_{u_s,j}|)\right), \quad j = 1, \dots, m_s, \quad (6.11c)$$

where $\sigma_j = |\mu_j|$, $\kappa_{l,j} = |\tau_{l_s,j}|$, and $\kappa_{u,j} = |\tau_{u_s,j}|$ in the first iteration ($l = 0$). Note, linear constraints are not included in the merit function since these are satisfied by construction of the QP-subproblem (6.7). Also, the penalty function, $Q(\epsilon_l, \epsilon_u)$, is not included in the merit function since ϵ_l and ϵ_u are not affected by changes in the step-size, α . We define the following function

$$T(\alpha) = P(x^{l+1}) = P(x^{[l]} + \alpha\Delta x^{[l]}). \quad (6.12)$$

We define sufficient decrease with the Armijo condition as

$$T(\alpha) \leq T(0) + c_1 \alpha D_{\Delta x} T(0), \quad (6.13)$$

where

$$\begin{aligned} T(\alpha) &= f(x^{[l]} + \alpha\Delta x^{[l]}) + \sigma^\top |g(x^{[l]} + \alpha\Delta x^{[l]})| \\ &\quad + \kappa_l^\top |\min(0, s(x^{[l]} + \alpha\Delta x^{[l]}) - l_s + \epsilon_l^{[l]})| \\ &\quad + \kappa_u^\top |\max(0, s(x^{[l]} + \alpha\Delta x^{[l]}) - u_s - \epsilon_u^{[l]})| \end{aligned} \quad (6.14a)$$

$$\begin{aligned} T(0) &= f(x^{[l]}) + \sigma^\top |g(x^{[l]})| \\ &\quad + \kappa_l^\top |\min(0, s(x^{[l]}) - l_s + \epsilon_l^{[l]})| \\ &\quad + \kappa_u^\top |\max(0, s(x^{[l]}) - u_s - \epsilon_u^{[l]})| \end{aligned} \quad (6.14b)$$

$$\begin{aligned} D_{\Delta x}T(0) &= \nabla f(x^{[l]})^\top \Delta x^{[l]} - \sigma^\top |g(x^{[l]})| \\ &\quad - \kappa_l^\top |\min(0, s(x^{[l]}) - l_s + \epsilon_l^{[l]})| \\ &\quad - \kappa_u^\top |\max(0, s(x^{[l]}) - u_s - \epsilon_u^{[l]})| \end{aligned} \quad (6.14c)$$

The backtracking line-search algorithm is (Kaysfeld et al. 2023)

1. Set $\alpha = 1$
2. Check the Armijo condition (6.13) and if satisfied **break** with $\alpha^{[l]} = \alpha$ as output
3. Reduce step $\alpha \leftarrow \beta\alpha$
4. Go to 2.

We apply $c_1 = 10^{-4}$ and $\beta = 0.5$, which are similar values as chosen in IPOPT (Wächter and Biegler 2006).

Once the step-size, $\alpha^{[l]}$, is computed by the line-search algorithm, NLPSQP performs the step

$$x^{[l+1]} = x^{[l]} + \alpha^{[l]} \Delta x^{[l]}, \quad \lambda^{[l+1]} = \lambda^{[l]} + \alpha^{[l]} \Delta \lambda^{[l]}, \quad (6.15a)$$

$$\pi_l^{[l+1]} = \pi_l^{[l]} + \alpha^{[l]} \Delta \pi_l^{[l]}, \quad \pi_u^{[l+1]} = \pi_u^{[l]} + \alpha^{[l]} \Delta \pi_u^{[l]}, \quad (6.15b)$$

$$\pi_{l_s}^{[l+1]} = \pi_{l_s}^{[l]} + \alpha^{[l]} \Delta \pi_{l_s}^{[l]}, \quad \pi_{u_s}^{[l+1]} = \pi_{u_s}^{[l]} + \alpha^{[l]} \Delta \pi_{u_s}^{[l]}. \quad (6.15c)$$

6.1.4 BFGS update

NLPSQP requires only gradient information. Thus, no second order derivatives are required to apply NLPSQP. In NLPSQP, we apply a BFGS update for the Lagrange Hessian. Specifically, we apply a damped version of the BFGS update to ensure positive definiteness of the update (Powell 1978). In the remainder of this section, we apply the following definitions to ease notation: $W = W^{[l]}$ and $\bar{W} = W^{[l+1]}$.

We define the following two vectors

$$s = x^{[l+1]} - x^{[l]}, \quad (6.16a)$$

$$y = \nabla_x \mathcal{L}_+ - \nabla_x \mathcal{L}_-, \quad (6.16b)$$

where

$$\begin{aligned} \nabla_x \mathcal{L}_- &= \nabla_x \mathcal{L}(x^{[l]}, \lambda^{[l+1]}, \pi_l^{[l+1]}, \pi_u^{[l+1]}, \pi_{\epsilon_l}^{[l+1]}, \pi_{\epsilon_u}^{[l+1]}, \pi_{l_s}^{[l+1]}, \pi_{u_s}^{[l+1]}) \\ &= \nabla f(x^{[l]}) - \nabla g(x^{[l]}) \lambda^{[l+1]} - \pi_l^{[l+1]} + \pi_u^{[l+1]} - \nabla s(x^{[l]}) \pi_{l_s}^{[l+1]} + \nabla s(x^{[l]}) \pi_{u_s}^{[l+1]}, \end{aligned} \quad (6.17a)$$

$$\begin{aligned} \nabla_x \mathcal{L}_+ &= \nabla_x \mathcal{L}(x^{[l+1]}, \lambda^{[l+1]}, \pi_l^{[l+1]}, \pi_u^{[l+1]}, \pi_{\epsilon_l}^{[l+1]}, \pi_{\epsilon_u}^{[l+1]}, \pi_{l_s}^{[l+1]}, \pi_{u_s}^{[l+1]}) \\ &= \nabla f(x^{[l+1]}) - \nabla g(x^{[l+1]}) \lambda^{[l+1]} - \pi_l^{[l+1]} + \pi_u^{[l+1]} - \nabla s(x^{[l+1]}) \pi_{l_s}^{[l+1]} + \nabla s(x^{[l+1]}) \pi_{u_s}^{[l+1]}. \end{aligned} \quad (6.17b)$$

We point out that the π_l and π_u contributions in \mathcal{L}_- and \mathcal{L}_+ can be ignored as these are eliminated in (6.16b). Now let

$$r = \theta y + (1 - \theta)Ws, \quad (6.18)$$

where

$$\theta = \begin{cases} 1 & s^\top y \geq 0.2s^\top Ws \\ \frac{0.8s^\top Ws}{s^\top Ws - s^\top y} & s^\top y < 0.2s^\top Ws \end{cases} \quad (6.19)$$

The damped BFGS update is then

$$\bar{W} = W - \frac{(Ws)(Ws)^\top}{s^\top(Ws)} + \frac{rr^\top}{s^\top r}. \quad (6.20)$$

NLPSQP applies $W^{[0]} = I$, where I is an identity matrix of proper dimensions.

6.1.5 Initialization

NLPSQP requires an initial guess on the decision variables $x^{[0]}$, which the user has to provide. The soft constraint slack variables, ϵ_l and ϵ_u , and all Lagrange multipliers are initialized by NLPSQP to 0.

6.1.6 Convergence

NLPSQP converges when the KKT-conditions (6.6) are satisfied, i.e., a local optimum is located. In practice, NLPSQP evaluates a scaled convergence criterion based on a user-specified convergence tolerance $\epsilon > 0$

$$\|\nabla_x \mathcal{L}/s_d\|_\infty \leq \epsilon, \quad (6.21a)$$

$$\|\nabla_{\epsilon_l} \mathcal{L}\|_\infty \leq \epsilon, \quad (6.21b)$$

$$\|\nabla_{\epsilon_u} \mathcal{L}\|_\infty \leq \epsilon, \quad (6.21c)$$

$$\|g(x)\|_\infty \leq \epsilon, \quad (6.21d)$$

$$\|\min(0, s(x) - l_s + \epsilon_l)\|_\infty \leq \epsilon, \quad (6.21e)$$

$$\|\max(0, s(x) - u_s - \epsilon_u)\|_\infty \leq \epsilon, \quad (6.21f)$$

where

$$s_d = \max \left(s_{\max}, \frac{\|\lambda\|_1 + \|\pi_l\|_1 + \|\pi_u\|_1}{m + m_l + m_u} \right) / s_{\max}. \quad (6.22)$$

We apply $s_{\max} = 100$ similarly to IPOPT (Wächter and Biegler 2006). NLPSQP evaluates the criterion (6.21) after the step (6.15) is computed.

6.1.7 Algorithm

Algorithm 4 presents a detailed implementation guide for NLPSQP.

6.2 Riccati version for optimal control problems

In this section, we introduce the Riccati recursion option for NLPSQP. In this mode, NLPSQP assumes that the NLP has a specific structure, where the QP-subproblem is in the form (2.27) such that QPIPM can apply Riccati mode. Therefore, the following is required of the NLP for NLPSQP to apply Riccati mode,

Algorithm 4: NLPSQP pseudo code**Input:** Initial guess, x_0 , and soft constrained NLP,

$$\begin{aligned} \min_{x, \epsilon_l, \epsilon_u} \quad & f(x) + Q(\epsilon_l, \epsilon_u), \\ \text{s.t.} \quad & g(x) = 0, \\ & l \leq x \leq u, \\ & l_s - \epsilon_l \leq s(x) \leq u_s + \epsilon_u, \\ & \epsilon_l, \epsilon_u \geq 0, \end{aligned}$$

i.e. the functions: $f(x)$, $g(x)$, $s(x)$ and the matrices and vectors: Q_l , Q_u , g_l , g_u , l , u , l_s , l_u .

- Initialize ($l = 0$):

$$x^{[0]} = x_0, \quad \epsilon_l^{[0]} = \epsilon_u^{[0]} = 0, \quad \lambda^{[0]} = \pi_l^{[0]} = \pi_u^{[0]} = \pi_{\epsilon_l}^{[0]} = \pi_{\epsilon_u}^{[0]} = \pi_{l_s}^{[0]} = \pi_{u_s}^{[0]} = 0, \quad W^{[0]} = I.$$

- Check convergence (6.21).

while not converged **do**

1. Update iteration counter: $l \leftarrow l + 1$.

2. Apply QPIPM to solve the QP-subproblem for $\Delta x^{[l]} = \Delta x$, $\epsilon_l^{[l]} = \epsilon_l$, and $\epsilon_u^{[l]} = \epsilon_u$,

$$\begin{aligned} \min_{\Delta x, \epsilon_l, \epsilon_u} \quad & \frac{1}{2} \Delta x^\top W \Delta x + \nabla f(x)^\top \Delta x + Q(\epsilon_l, \epsilon_u), \\ \text{s.t.} \quad & \nabla g(x)^\top \Delta x = -g(x), \\ & l - x \leq \Delta x \leq u - x, \\ & l_s - s(x) - \epsilon_l \leq \nabla s(x)^\top \Delta x \leq u_s - s(x) + \epsilon_u, \\ & \epsilon_l, \epsilon_u \geq 0, \end{aligned}$$

where $W = W^{[l]}$ and $x = x^{[l]}$. Note, the Lagrange multipliers for the QP-subproblem are

$$\begin{aligned} \mu &= \lambda^{[l]} + \Delta \lambda^{[l]}, \\ \tau_l &= \pi_l^{[l]} + \Delta \pi_l^{[l]}, & \tau_u &= \pi_u^{[l]} + \Delta \pi_u^{[l]}, \\ \tau_{l_s} &= \pi_{l_s}^{[l]} + \Delta \pi_{l_s}^{[l]}, & \tau_{u_s} &= \pi_{u_s}^{[l]} + \Delta \pi_{u_s}^{[l]}, \\ \pi_{\epsilon_l} &= \pi_{\epsilon_l}^{[l]}, & \pi_{\epsilon_u} &= \pi_{\epsilon_u}^{[l]}. \end{aligned}$$

3. Compute the step-size, $\alpha^{[l]}$, with the line-search algorithm as described in section 6.1.3.

4. Compute the step

$$\begin{aligned} x^{[l+1]} &= x^{[l]} + \alpha^{[l]} \Delta x^{[l]}, & \lambda^{[l+1]} &= \lambda^{[l]} + \alpha^{[l]} \Delta \lambda^{[l]}, \\ \pi_l^{[l+1]} &= \pi_l^{[l]} + \alpha^{[l]} \Delta \pi_l^{[l]}, & \pi_u^{[l+1]} &= \pi_u^{[l]} + \alpha^{[l]} \Delta \pi_u^{[l]}, \\ \pi_{l_s}^{[l+1]} &= \pi_{l_s}^{[l]} + \alpha^{[l]} \Delta \pi_{l_s}^{[l]}, & \pi_{u_s}^{[l+1]} &= \pi_{u_s}^{[l]} + \alpha^{[l]} \Delta \pi_{u_s}^{[l]}. \end{aligned}$$

5. Check convergence (6.21) - **break** if criterion is satisfied.

6. Apply the BFGS update as described in section 6.2.1

$$W^{[l+1]} = W^{[l]} - \frac{(W^{[l]}s)(W^{[l]}s)^\top}{s^\top(W^{[l]}s)} + \frac{rr^\top}{s^\top r}.$$

Return: x , y , π_l , π_u , π_{ϵ_l} , π_{ϵ_u} , π_{l_s} , and π_{u_s} .

6.2.1 Block BFGS update

In Riccati mode, NLPSQP applies a block BFGS update to maintain a block diagonal Hessian structure for the QP-subproblem. A usual BFGS update would result in a dense matrix and would therefore not produce the structure required to apply Riccati recursion in the QP-subproblem. In the remainder of this section, we apply $W_k = W_k^{[l]}$ and $\bar{W}_k = W_k^{[l+1]}$ for simplicity of notation.

We define the vectors, s and y , similar to (6.16),

$$s = \xi^{[l+1]} - \xi^{[l]}, \quad (6.28a)$$

$$y = \nabla_{\xi} \mathcal{L}_+ - \nabla_{\xi} \mathcal{L}_-, \quad (6.28b)$$

where \mathcal{L}_- and \mathcal{L}_+ is defined as in (6.17). We let s_k and y_k be sub-vectors in s and y corresponding to the diagonal block matrices, W_k , in (6.26). Similarly to the normal damped BFGS update, we define

$$r_k = \theta_k y_k + (1 - \theta_k) W_k s_k, \quad (6.29)$$

where

$$\theta_k = \begin{cases} 1 & s_k^{\top} y_k \geq 0.2 s_k^{\top} W_k s_k \\ \frac{0.8 s_k^{\top} W_k s_k}{s_k^{\top} W_k s_k - s_k^{\top} y_k} & \text{else} \end{cases} \quad (6.30)$$

Finally, the BFGS update of each block is

$$\bar{W}_k = \begin{cases} W_k - \frac{(W_k s_k)(W_k s_k)^{\top}}{s_k^{\top} (W_k s_k)} + \frac{r_k r_k^{\top}}{s_k^{\top} r_k} & \kappa > \epsilon_m \\ W_k & \text{else} \end{cases} \quad (6.31)$$

ϵ_m is the machine precision of the computer and $\kappa = \min(\kappa_1, \kappa_2)$ with $\kappa_1 = s_k^{\top} W_k s_k$ and $\kappa_2 = s_k^{\top} r_k$. These update safeguards are implemented to avoid zero-division if some blocks converge faster than others. NLPSQP initializes the full block diagonal structured Hessian approximation as $W^{[0]} = I$, where I is an identity matrix of proper dimensions. Numerical tests have shown that numerical errors might cause indefinite BFGS block updates. NLPSQP applies the simple strategy to reset the entire Hessian approximation to identity if an indefinite update is detected.

6.2.2 Application to solve OCPs

In this section, we introduce an OCP and demonstrate that direct multiple shooting discretization transcribes the OCP to an NLP in the form required for NLPSQP to apply Riccati mode.

We consider continuous OCPs in the form

$$\min_{[x(t); u(t)]_{t_0}^{t_f}} \phi = \int_{t_0}^{t_f} l(t, x(t), u(t), p) dt + \hat{l}(x(t_f), p), \quad (6.32a)$$

$$\text{s.t.} \quad x(t_0) = x_0, \quad (6.32b)$$

$$\dot{x}(t) = f(t, x(t), u(t), d(t), p), \quad t_0 \leq t \leq t_f, \quad (6.32c)$$

$$u_{\min}(t) \leq u(t) \leq u_{\max}(t), \quad t_0 \leq t \leq t_f. \quad (6.32d)$$

By direct multiple shooting discretization, we transcribe the continuous OCP (6.32) to the following NLP,

$$\min_{\{u_k, x_{k+1}\}_{k=0}^{N-1}} \phi = \hat{\Phi}(\{u_k, x_{k+1}\}_{k=0}^{N-1}), \quad (6.33a)$$

$$\text{s.t.} \quad R_k = x_{k+1} - F(t_k, x_k, u_k, d_k, p) = 0, \quad k = 0, \dots, N-1, \quad (6.33b)$$

$$u_{\min, k} \leq u_k \leq u_{\max, k}, \quad k = 0, \dots, N-1, \quad (6.33c)$$

where $F(\cdot)$ is a numerical integration scheme and

$$\hat{\Phi}(\{u_k, x_{k+1}\}_{k=0}^{N-1}) = \left\{ \begin{array}{l} \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} l(x_k(t), u_k, d_k, p) dt + \hat{l}(x_N, p) : \\ x_0(t_0) = x_0, \\ x_k(t_k) = x_k, \quad k = 1, \dots, N-1, \\ \dot{x}_k(t) = f(t, x_k(t), u_k, d_k, p), \quad t_k \leq t \leq t_{k+1} \end{array} \right\}. \quad (6.34)$$

We add soft constraints to the discretized OCP and get a soft constrained OCP in the form

$$\min_{\{u_k, x_{k+1}, \epsilon_{l,k+1}, \epsilon_{u,k+1}\}_{k=0}^{N-1}} \phi = \hat{\Phi}(\{u_k, x_{k+1}\}_{k=0}^{N-1}) + Q(\epsilon_l, \epsilon_u), \quad (6.35a)$$

$$\text{s.t. } R_k = x_{k+1} - F(t_k, x_k, u_k, d_k, p) = 0, \quad k = 0, \dots, N-1, \quad (6.35b)$$

$$u_{\min,k} \leq u_k \leq u_{\max,k}, \quad k = 0, \dots, N-1, \quad (6.35c)$$

$$x_{\min,k} - \epsilon_{l,k} \leq s_k(x_k) \leq x_{\max,k} + \epsilon_{u,k}, \quad k = 1, \dots, N, \quad (6.35d)$$

where $\epsilon_l = [\epsilon_{l,1} \ \epsilon_{l,2} \ \dots \ \epsilon_{l,N}]^\top$, $\epsilon_u = [\epsilon_{u,1} \ \epsilon_{u,2} \ \dots \ \epsilon_{u,N}]^\top$, and $Q(\epsilon_l, \epsilon_u)$ is as in (6.2). In the following, we demonstrate that the NLP (6.35) satisfies the requirements for NLPSQP to be called in Riccati mode.

Equality constraints

We write the equality constraints (6.35b) as

$$g(\xi) = [R_0 \ R_1 \ \dots \ R_{N-1}]^\top, \quad (6.36)$$

and observe that the gradient has the required form (6.24),

$$\nabla g(\xi) = \begin{bmatrix} -\nabla_{u_0} F \\ I & -\nabla_{x_1} F \\ & -\nabla_{u_1} F \\ & I & -\nabla_{x_2} F \\ & & -\nabla_{u_2} F \\ & & I & \dots & -\nabla_{x_{N-1}} F \\ & & & & -\nabla_{u_{N-1}} F \\ & & & & I \end{bmatrix}, \quad (6.37)$$

where we define

$$A_k = \nabla_{x_k} F = \nabla_{x_k} F(t_k, x_k, u_k, d_k, p), \quad k = 1, \dots, N-1, \quad (6.38)$$

$$B_k = \nabla_{u_k} F = \nabla_{u_k} F(t_k, x_k, u_k, d_k, p), \quad k = 0, \dots, N-1. \quad (6.39)$$

Soft constraints

Similarly, we write the soft constraints (6.35d) as

$$s(\xi) = [s_1(x_1) \ s_2(x_2) \ \dots \ s_N(x_N)]^\top \quad (6.40)$$

6.2.4 A note on bounds

Even though we have not included hard output constraints in the problem (6.32), NLPSQP does have the option to include these in the OCP.

Implementation of NLPSQP in Matlab and C

In this chapter, we introduce how to call NLPSQP in both Matlab and in C. Both versions are part of a private gitlab-repository, which can be cloned with the command line command

```
git clone https://gitlab.gbar.dtu.dk/SCGroup/NLPSQP.git
```

7.1 Matlab

NLPSQP in Matlab has the following interface:

```
1 function [x, stat] = NLPSQP(ffun, x0, gfun, hfun, l, u, options, varargin)
```

Inputs:

The inputs are as follows: `ffun` is the objective function, $f(x)$, `x0` is the user-provided initial condition, `x0`, `gfun` is the equality constraint function, $g(x)$, `l` is the lower bound vector, `u` is the upper bound vector, `options` is an options structure, and `varargin` contains a set of variable input arguments for $f(x)$, $g(x)$, and $h(x)$.

The input `hfun` is for general inequality constraints,

$$h(x) \geq 0, \quad (7.1)$$

which is only implemented for testing purposes and not optimized in any way. In Riccati mode, general inequality constraints are not supported and `hfun` has to be left empty.

The input `options` contains a number of options and the possibility to enable soft constraints. The options structure has the following fields

<code>print</code>	0 or 1 to print iteration information	Default: 1
<code>tol</code>	Convergence tolerance	Default: 10^{-8}
<code>tolStep</code>	Minimum allowed step-size	Default: 10^{-8}
<code>maxit</code>	Maximum iterations	Default: 100
<code>riccati</code>	0 or 1 to turn Riccati mode off/on	Default: 0
<code>N</code>	Horizon. Required if <code>riccati=1</code>	Default: NaN
<code>printQP</code>	0 or 1 to print QPIPM iteration information	Default: 0
<code>tolQP</code>	QPIPM convergence tolerance	Default: $10^{-2} \cdot \text{tol}$
<code>maxitQP</code>	QPIPM maximum iterations	Default: 100

<code>subWarn</code>	0 or 1 to suppress Matlab warnings	Default: 0
<code>softLin</code>	0 or 1 to specify linear soft constraints	Default: 0
<code>softNonlin</code>	0 or 1 to specify nonlinear soft constraints	Default: 0
<code>softProblemLin</code>	Structure with linear soft constraints	Default: empty
<code>softProblemNonlin</code>	Structure with nonlinear soft constraints	Default: empty

The two soft constrained problem structures `softProblemLin` and `softProblemNonlin` are required to be set if `softLin=1` and `softNonlin=1` respectively. Note also that `softLin` and `softNonlin` cannot be set to 1 at the same time. The fields of `softProblemLin` and `softProblemNonlin` are

<code>ls</code>	Lower soft bound
<code>S/sfun</code>	Linear case: Soft constraint matrix - Nonlinear case: Soft constraint function
<code>us</code>	Upper soft bound
<code>Ql</code>	Quadratic penalty matrix for lower soft bound - assumed diagonal
<code>Qu</code>	Quadratic penalty matrix for upper soft bound - assumed diagonal
<code>ql</code>	Linear penalty vector for lower soft bound
<code>qu</code>	Linear penalty vector for upper soft bound

We point out that NLPSQP takes the same inputs and have the same outputs when Riccati mode is off and on. When applying Riccati mode, NLPSQP assumes that the provided NLP has the required structure as described in section 6.2. NLPSQP does not check that this is the case. Riccati mode can be applied for a non-structured NLP, but NLPSQP makes assumptions about the structure in the QP-subproblem, which likely leads to poor search directions. This can cause bad convergence properties of NLPSQP and might ultimately prevent convergence.

Outputs:

The output `x` is the solution vector after 1) convergence, i.e., `x` is a local optimum, 2) the maximum number of iterations are reached in which case NLPSQP prints a warning, and 3) the computed step-size is smaller than `tolStep` in which case NLPSQP prints a warning. The `stat` output is a structure with the following fields

<code>obj</code>	Objective value at solution
<code>conv</code>	0 (not converged) or 1 (converged)
<code>iter</code>	Number of iterations
<code>lamEq</code>	Lagrange multipliers for equality constraints
<code>lamIneq</code>	Lagrange multipliers for inequality constraints
<code>lamBn</code>	Lagrange multipliers for bound constraints
<code>eps</code>	ϵ -slack variables

7.2 C

The C version of NLPSQP does not include soft constraints as previously mentioned. The interface for NLPSQP in C is

```

1 void NLPSQP (
2     // Inputs
3     objectiveFunctionNLPSQP_t      *ffun      ,
4     struct vec                    *x0        ,
5     void                          *varargin  ,
6     equalityConstraintFunctionNLPSQP_t *gfun    ,
7     inequalityConstraintFunctionNLPSQP_t *hfun    ,
8     struct vec                    *l         ,
9     struct vec                    *u         ,
10    optionsNLPSQP_t               *options   ,
11    mem                            *memory    ,
12
13    // Outputs
14    struct vec                    *x         ,
15    statNLPSQP_t                 *stat      ,
16 )

```

Inputs:

NLPSQP takes three function inputs *ffun*, *gfun*, and *hfun* similarly to the Matlab version. The *varargin* input is a set of variable input arguments required by the three input functions. The vectors *l* and *u* are the lower and upper bounds, respectively. The *memory* input contains both integer and double workspace required by NLPSQP (see section 7.2.1). The *options* inputs is a structure of type *optionsNLPSQP_t* which has the following fields

<i>print</i>	0 or 1 to print iteration information	Default: 1
<i>tol</i>	Convergence tolerance	Default: 10^{-8}
<i>tolStep</i>	Minimum allowed step-size	Default: 10^{-8}
<i>maxit</i>	Maximum iterations	Default: 100
<i>riccati</i>	0 or 1 to turn Riccati mode off/on	Default: 0
<i>N</i>	Horizon. Required if <i>riccati</i> =1	Default: <i>idummy</i>
<i>printQP</i>	0 or 1 to print QPIPM iteration information	Default: 0
<i>tolQP</i>	QPIPM convergence tolerance	Default: $10^{-2} \cdot \text{tol}$
<i>maxitQP</i>	QPIPM maximum iterations	Default: 100
<i>bigN</i>	Numbers above treated as infinity	Default: 10^{20}

where *idummy* = -11111 is an integer dummy variable defined in NLPSQP. The function types of *ffun*, *gfun*, and *hfun* are

```

1 typedef void objectiveFunctionNLPSQP_t (
2     // Inputs
3     struct vec *x      ,
4     void      *varargin ,
5     int       nargout  ,
6
7     // Outputs

```



```

8     double      *f          ,
9     struct vec  *df
10 );

```

```

1 typedef void equalityConstraintFunctionNLPSQP_t (
2     // Inputs
3     struct vec  *x          ,
4     void        *varargin  ,
5     int         nargout    ,
6
7     // Outputs
8     struct vec  *g          ,
9     struct mat  *dg
10 );

```

```

1 typedef void inequalityConstraintFunctionNLPSQP_t (
2     // Inputs
3     struct vec  *x          ,
4     void        *varargin  ,
5     int         nargout    ,
6
7     // Outputs
8     struct vec  *h          ,
9     struct mat  *dh
10 );

```

The three function types have the same inputs, which are

x	Decision variables
varargin	A set of variable input arguments
nargout	Number of outputs to evaluate

and their outputs are

f	Objective value
df	Gradient for objective function
g	Equality constraints vector
dg	Gradient for equality constraints
h	Inequality constraints vector
dh	Gradient for inequality constraints

Outputs:

The output `x` is the solution vector after 1) convergence, i.e., `x` is a local optimum, 2) the maximum number of iterations are reached in which case NLPSQP prints a warning, and 3) the computed step-size is smaller than `tolStep` in which case NLPSQP prints a warning. The `stat` output is a structure of type

`statNLPSQP_t` with the following fields

<code>obj</code>	Objective value at solution
<code>conv</code>	0 (not converged) or 1 (converged)
<code>iter</code>	Number of iterations
<code>lamBn</code>	Lagrange multipliers for bound constraints
<code>lamEq</code>	Lagrange multipliers for equality constraints
<code>lamIneq</code>	Lagrange multipliers for inequality constraints

7.2.1 Memory allocation

NLPSQP requires both integer and double workspace, which should be allocated in the input memory structure. NLPSQP features the function

```
1 void workspaceNLPSQP( int n, int me, int mi, int *iwork, int *dwork )
```

which given the dimensions of the NLP, `n` (decision variables), `me` (equality constraints), and `mi` (inequality constraints), computes the required workspace for NLPSQP. Then the amount of integer workspace, `iwork`, and double workspace, `dwork`, can be use to initialize the memory input with sufficient memory. Additionally, the `stat` structure for the output is required to be initialized, which can be done with the function

```
1 void createStatNLPSQP( const int n, const int me, const int mi,
    statNLPSQP_t *const stat )
```

`createStatNLPSQP` allocates the required memory for the output `stat` structure. Note that when finished using the `stat` structure, the memory can be freed with the function

```
1 void destroyStatNLPSQP( statNLPSQP_t *stat )
```

7.2.2 Dependencies

NLPSQP is a part of the private gitlib-repository `SCProject`, which is a project containing a series of git repositories. NLPSQP is dependent on the following three repositories in `SCProject`:

<code>SCInterface</code>	A set of structure and function definitions
<code>linalg</code>	A set of vector and matrix linear algebra functions
<code>util</code>	A set of utility functions
<code>QPIP</code>	A primal-dual interior-point software to solve QPs

Additionally, `linalg` is BLAS dependent and requires linking to a BLAS installation on the system.

7.2.3 Gitlab

The private Gitlab group `SCGroup` grants access to all projects contained in `SCProject`. Therefore, the four projects, `NLPSQP`, `QPIP`, `SCInterface`, and `linalg` are also included in `SCGroup`. When access is granted to `SCGroup`, one can clone the whole `SCProject` or parts of it. To apply NLPSQP, one has to clone `NLPSQP`, `QPIP`, `SCInterface`, `util`, and `linalg` (and install a version of BLAS). The C version of NLPSQP includes a `settings.mk` file, where the dependency paths can be set. The five git

repositories can be cloned with the following command line commands (accompanied with a username and password):

```
git clone https://gitlab.gbar.dtu.dk/SCGroup/SCInterface.git
git clone https://gitlab.gbar.dtu.dk/SCGroup/linalg.git
git clone https://gitlab.gbar.dtu.dk/SCGroup/util.git
git clone https://gitlab.gbar.dtu.dk/SCGroup/QIPM.git
git clone https://gitlab.gbar.dtu.dk/SCGroup/NLPSQP.git
```

7.2.4 Doxygen documentation

The C version of NLPSQP is documented with Doxygen. The Doxygen documentation is available in `NLPSQP/C/docs`, which can be compiled by typing `doxygen` in the command line. Afterwards, the documentation is available in `NLPSQP/C/docs/results/html/index.html`, which opens in a browser. The documentation includes descriptions of all NLPSQP functions and their inputs and outputs. Note, this requires an installation of Doxygen on the system.

7.3 Examples

Both the Matlab and C version of NLPSQP has a few test examples. The Matlab version includes a driver to test NLPSQP on a simple NLP and a few drivers to apply NLPSQP to solve OCPs for both a four tank system model and a continuous stirred tank reactor (CSTR) model. The drivers show how to apply NLPSQP and demonstrate NLPSQP with/without Riccati mode and with/without soft constraints.

The C version also includes a test simple test NLP. Additionally, the C version includes test examples that demonstrate that NLPSQP can solve an OCP for the CSTR model similarly to the Matlab version. Finally, the C version includes an example that demonstrates that NLPSQP can be called to solve multiple OCPs in parallel using `openMP`. This example requires linking to a thread-safe BLAS installation, e.g., BLASFEO (Frison et al. 2018, 2020).

Furthermore, we refer to previous work, where we have integrated NLPSQP in an NMPC. The NMPC was applied in large-scale closed-loop Monte Carlo simulations to quantify uncertainties in the closed-loop system (Kaysfeld et al. 2023).

Conclusion

In this part, we introduced the sequential quadratic programming (SQP) software, NLPSQP, to solve structured nonlinear programming problems (NLPs). NLPSQP is a software package that is stored in a private gitlab-repository `NLPSQP`, which is part of the project `SCPproject`. NLPSQP has a Matlab version and a C version. In the current version only NLPSQP in Matlab supports soft constraints. We have provided the mathematical details of NLPSQP and introduced the implementation of NLPSQP in both Matlab and C. We showed interfaces of the implementations and described the inputs and outputs. In the C version, we elaborated on how to allocate the needed memory for NLPSQP and how to link to the introduced the dependencies.

The C version of NLPSQP is intended for application in parallel Monte Carlo simulation of closed-loop systems containing nonlinear model predictive control (NMPC) algorithms. Due to the thread-safety of the implementation, NLPSQP can be applied to solve multiple OCPs in parallel with almost linear scaling and is therefore well-suited for the purpose.

Bibliography

- Frison, G. and Jørgensen, J. B.: 2013, Efficient implementation of the riccati recursion for solving linear-quadratic control problems, *IEEE International Conference on Control Applications (CCA), Hyderabad, India* pp. 1117–1122.
- Frison, G., Kouzoupis, D., Sartor, T., Zanelli, A. and Diehl, M.: 2018, BLASFEO: Basic linear algebra subroutines for embedded optimization, *ACM Transactions on Mathematical Software* **44**(4).
- Frison, G., Sartor, T., Zanelli, A. and Diehl, M.: 2020, The BLAS API of BLASFEO: Optimizing performance for small matrices, *ACM Transactions on Mathematical Software* **46**(2).
- Jørgensen, J. B.: 2004, *Moving Horizon Estimation and Control*, PhD thesis, Technical University of Denmark.
- Jørgensen, J. B., Frison, G., Gade-Nielsen, N. F. and Damman, B.: 2012, Numerical methods for solution of the extended linear-quadratic control problem, *IFAC Proceedings Volumes* **45**(17), 187–193.
- Karush, W.: 1939, *Minima of functions of several variables with inequalities as side constraints*, M.sc. thesis, University of Chicago, Chicago, Illinois.
- Kaysfeld, M. W., Zanon, M. and Jørgensen, J. B.: 2023, Performance quantification of a nonlinear model predictive controller by parallel Monte Carlo simulations of a closed-loop system, *Proceedings of the 21st European Control Conference (ECC), Bucharest, Romania, 2023, accepted* .
- Kjeldsen, T. H.: 2000, A contextualized historical analysis of the Kuhn–Tucker theorem in nonlinear programming: The impact of world war II, *Historia Mathematica* **27**(4), 331–361.
- Kuhn, H. W. and Tucker, A. W.: 1951, Nonlinear programming, *University of California Press* pp. 481–492.
- Mehrotra, S.: 1992, On the implementation of a primal-dual interior point method, *SIAM Journal on Optimization* **2**(4), 575–601.
- Powell, M. J. D.: 1978, A fast algorithm for nonlinearly constrained optimization calculations, *Proceedings of the Biennial Conference on Numerical Analysis* pp. 144–157.
- Rao, C. V., Wright, S. J. and Rawlings, J. B.: 1998, Application of interior-point methods to model predictive control, *Journal of Optimization Theory and Applications* **99**(3), 723–757.

- Wahlgreen, M. R. and Jørgensen, J. B.: 2022, On the implementation of a preconditioned riccati recursion based primal-dual interior-point algorithm for input constrained optimal control problems, *IFAC-PapersOnLine* **55(7)**, 346–351. 13th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS) 2022.
- Wahlgreen, M. R., Reenberg, A. T., Nielsen, M. K., Rydahl, A., Ritschel, T. K. S., Dammann, B. and Jørgensen, J. B.: 2021, A high-performance monte carlo simulation toolbox for uncertainty quantification of closed-loop systems, *Proceedings of the 60th IEEE Conference on Decision and Control (CDC)* pp. 6755–6761.
- 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.

