Dinesh Krishnamoorthy

Novel Approaches to Online Process Optimization Under Uncertainty

Addressing the limitations of current industrial practice

Thesis for the degree of Philosophiae Doctor
Trondheim, Nov 2019

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

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Natural Sciences
Department of Chemical Engineering

ISBN 978-82-326-4198-7 (printed version)
ISBN 978-82-326-4199-4 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2019:301

Printed by NTNU Graphic Center

To my parents

Abstract

In the face of growing competition, and increased necessity to focus on sustainability and energy efficiency, there is a clear need to optimize the day-to-day operation of many industrial processes. One strategy for online process optimization is to use model-based real-time optimization (RTO). Despite the motivation and the potential, real-time optimization is not as commonly used in practice as one would expect. This thesis takes a detailed look at the different challenges that impede practical implementation of real-time optimization and aims to address some of these challenges. In brief, this thesis presents novel algorithms and methods ranging from simple control structures, to model-free and model-based optimization and more complex scenario-based economic model predictive control.

One of the fundamental limiting factors of traditional steady-state RTO is the steady-state wait time. This essentially discards transient measurements, which otherwise contains useful information. In part I of this thesis, we propose different approaches to use transient measurements for steady-state optimization, with the goal of minimizing the steady-state wait time. Moreover, different algorithms to real-time optimization that do not require the need to solve numerical optimization problems are proposed, thus alleviating many of the computational challenges which impede practical application of traditional RTO approaches.

First, we propose a "hybrid" approach, where the model adaptation is done with transient measurements and dynamic models, and the optimization is performed using steady-state models. To further simplify the steady-state optimization, we then convert the hybrid RTO approach into a feedback RTO approach. Here, the transient measurements are used to estimate the steady-state gradient, which is controlled to a constant setpoint of zero using feedback controllers. The steady-state gradient is estimated using a novel method based on linearizing the nonlinear dynamic model around the current operating point.

To address the cost of developing models, we demonstrate the use of classical controllers where the economic objectives are translated into control objectives. We also provide a systematic approach to switch between different active constraint regions using selectors. For the unconstrained degrees of freedom, we then propose a novel extremum seeking scheme using transient measurements for a class of Hammerstein systems, where the linear dynamics are fixed. We show that the proposed approach converges significantly faster than the classical extremum seeking scheme, and provide robust stability margins. Part I concludes by showing that the different methods work in different time scales, and by hierarchically combining the different approaches, one can handle a wider class of uncertainties.

In the second part of the thesis, we take a different approach, and focus on addressing the computational cost of solving dynamic optimization problems under uncertainty. Part II begins with discussing what is a good optimization problem formulation in the presence of uncertainty. We then consider the multistage scenario-based formulation, where the evolution of uncertainty in the prediction horizon is represented via a discrete scenario tree. We show that the resulting large-scale optimization problem can be decomposed into several smaller subproblems, and argue in favor of using primal decomposition over dual decomposition. Since the different scenarios differ only in the uncertain parameters, we show that the distributed multistage scenario optimization problem can be cast as a parametric nonlinear programming (NLP) problem. By using the NLP sensitivity, we can reduce the number of NLPs required to to be solved. By doing so, we show that the computational cost of solving the multistage scenario-based problem can be significantly reduced. Finally, the thesis is concluded by providing an overall understanding of the different approaches to online process optimization in terms of the degree of complexity and flexibility, the different timescales they work in and the different kinds of uncertainty each method can handle, which are summarized in Table 10.2.

To summarize, this thesis shows that there is no one single approach to RTO that addresses all the challenges. The different approaches to RTO have their own advantages and disadvantages. The key to addressing the limiting factors of current industrial practice is to have a clear overview and understanding of the different RTO approaches in order to select the "*right*" tool for the problem at hand.

Contents

List of figures

List of tables

Preface

The main motivation behind the past three years of my research, culminating in this thesis, is the realization that real-time optimization is not used as much in practice as one would expect.

After completing my masters degree in Control Systems from Imperial College London in 2012, I switched to process control field when I started working at Statoil Research Center[1] in Porsgrunn, Norway. This experience gave me valuable insight into the research-to-practice workflow and the human aspects involved in successful industrial use of control and optimization. During my time at Statoil, I noticed that a huge chunk of production data containing valuable information is often discarded in practice. Naturally, one of the research questions that became apparent was "*how can we use this data efficiently to incorporate this information in the decision-making process and improve production optimization?*"

With four years of industrial experience and some well-motivated research questions, I decided to pursue a PhD degree and I was very happy when both Prof. Sigurd Skogestad (Department of Chemical Engineering, NTNU) and Prof. Bjarne Foss (Department of Engineering Cybernetics, NTNU) were eager to take me on as their PhD Student. Finally, in late July 2016, I started my PhD studies. Over the past three years, I have been working on various approaches to online process optimization under uncertainty, including new approaches to real-time optimization, extremum seeking control and robust multistage economic NMPC, which are reported in this thesis. The results from Part I of this thesis was also the content of the 2019 IFAC DYCOPS half-day pre-symposium tutorial workshop organized in Florianopolis, Brazil.

The PhD research was carried out at the Department of Chemical Engineering at the Norwegian University of Science and Technology (NTNU) with Prof. Sigurd Skogestad as my main supervisor and Prof. Bjarne Foss and Assoc. Prof. Johannes Jäschke as my co-supervisors in the period from July 2016 to July 2019. The work was carried out as a part of the Center for Research-based Innovation within subsea production and processing (SFI SUBPRO), which is financed by the Research Council of Norway, NTNU and major industry partners (Project number 237893).

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

[1] Re-named as Equinor since May 2018

Acknowledgments

First and foremost I would like to thank my PhD supervisor Professor Sigurd Skogestad for his unfaltering belief and trust in me. I would like to thank him for taking me on as his PhD student, for his guidance and advice, for his dedication and patience, and for all the intellectual freedom that he has given me. I must also thank him for allowing me to travel around the world and introducing me to many leading researchers at several conferences. I will always cherish our often long and stimulating discussions. His passion for process control, his motivation to solve real-world problems, and his ability to see the big picture has not only been very inspiring, but also very valuable in shaping my own research interests. I also thank him for his constant support and encouragement, and for facilitating my personal growth, both as a researcher and as a person. Apart from being a fantastic advisor, he has also been a great mentor and friend.

I thank my co-supervisor Professor Bjarne Foss, who was also eager to take me on as his PhD student. I am so glad that I did not have to choose between Bjarne and Sigurd, but instead have Bjarne as a co-advisor along with Sigurd. I am very grateful that he was always available for discussions, even after he had taken on his new role as Pro Rector for Research. I always felt very optimistic and motivated after our discussions. I would also like to thank my co-supervisor, Assoc. Prof. Johannes Jäschke for engaging discussions on economic MPC and for introducing me to NLP sensitivities. I would also like to express my profound gratitude to him for facilitating my research stay at Carnegie Mellon University.

I would like to thank Prof. Lorenz T. Biegler for graciously hosting me at Carnegie Mellon University during spring of 2019. I absolutely enjoyed working on the adaptive horizon economic NMPC framework and I learned a lot during my stay there. I would also like to thank the graduate students at the PSE group at CMU for the welcoming atmosphere and making my stay in Pittsburgh even more enjoyable. Special thanks goes to Dr. Esmaeil Jahanshahi, who was instrumental in developing the feedback RTO approach.

I would also like to thank Professor Mauricio B. de Souza and Professor Argimiro R. Secchi for hosting me during my brief visits to Federal University of Rio de Janeiro (UFRJ), Brazil. In particular, I am very grateful to them for inviting me to give a course on Numerical Optimal Control and Process Optimization in 2019, which I thoroughly enjoyed. I would also like to thank Professor Eduardo Camponogara, Dr. Thiago L. Silva and Marco Aguiar for interesting discussions on distributed optimization. Financial support from the INTPART funding is gratefully acknowledged for enabling collaborations with Brazil.

I am also grateful for the opportunity to co-supervise several talented master thesis students: Jeongrim Ryu, Simen Bjorvand and Ingvild Sørlie from NTNU, Carlo Valli from Politecnio di Milano, Harro Bonnowitz from TU Berlin, and Otavio Ivo, Allyne dos Santos and Rodrigo Curvelo from UFRJ. I hope that I was able to impart at least some of the outstanding supervision that I have received.

I gratefully acknowledge the financial support from SFI SUBPRO, which is financed by the Research Council of Norway, major industry partners and NTNU. The several social activities organized by SUBPRO was also very enjoyable.

I would also like to thank my former colleagues from Equinor research center Kjetil Fjalestad, Elvira M Bergheim, Alexey Pavlov, Qin Li, Robert Aasheim and Pål Kittilsen, for many interesting discussions on the practical aspects of control and optimization. Working together with such exceptional practitioners was very important in gaining valuable experience and industrial perspective, which I hope is reflected in this thesis. I would also like to acknowledge Equinor for giving me the opportunity to work 20% part-time between May 2018 and July 2019 alongside my PhD research.

To all my friends and colleagues in the systems group; thank you for creating such a fun-filled and relaxed workplace. I am also grateful to Cristina Zotica and Haakon Holck for proofreading parts of the thesis. Special thanks to my office-mates Adriaen Verheyleweghen, Eka Suwartadi, Christoph Backi, and Tamal Das for the great times together. Cheers to Julian, Mandar, Adriana, Tobias, Cansu, Timur, Jose, David, Andrea, Allyne, and the many other fellow colleagues at the department for the many good memories (and the several moving-in parties!)

I would also like to thank the PhD evaluation committee: Professor Manfred Morari, Professor Jay H Lee, and Professor Sebastian Gros, for taking the time to read through my thesis and for their valuable feedback. It is a great honor to be evaluated by the most distinguished group of researchers!

Finally, I would like to express my heart-felt gratitude to my family. I thank my fiancè, Shreya, for her constant love and support. I am also fortunate to have grown up with my brother, Rajesh, and my cousin, Divya, who always set the bar very high for me. This thesis is dedicated to my parents, whom I cannot thank enough for everything that they have done for me.

Dinesh Krishnamoorthy
Trondheim, Nov 2019

Chapter 1

Introduction

The safe and optimal operation of large and complex industrial processes requires meeting goals and objectives in different time scales ranging from long-term planning and scheduling to fast corrective actions for stable operation. Realizing all the goals and constraints as a whole can be a very challenging and unrealistic task, especially if formulated as a single centralized optimization and control problem. Thus, the operation of any process is typically decomposed into various decision making layers [168, Ch.10], [46]. Such a hierarchical implementation is a widely accepted industry standard [34] and is also well studied in academic literature under the context of plantwide control, see for e.g. [165], [167],[110] and [141] to name a few.

A typical control system hierarchy is shown in Fig.1.1, where the time horizon for the decisions are clearly shown for each layer. The information flow in this control hierarchy is such that the upper layers provide setpoints to the layer below, which reports back any problems in achieving this [165].

The long term decisions involve selecting an investment strategy, operation model, infrastructure etc., which is typically known as *Asset management*. Then there are decisions taken on a horizon of days such as plantwide scheduling. This is followed by decisions that have to be taken on decision horizon in the timescale of hours known as *Real-time Optimization* (RTO). This decision making step is the main focus of this thesis. It aims to maximize the revenue and minimize the operational costs of hour-by-hour operations, thereby optimizing the economics of the process. This is followed by a faster control and automation layer that accounts for fast corrective actions. The control layer could be broadly divided into supervisory and regulatory layers, where the objective of the supervisory layer (such as model predictive control) is to track the reference trajectory provided by the RTO layer and to look after other variables and constraints. On the other hand, the primary objective of the regulatory layer is to stabilize and avoid drift in the variables.

The upper three layers in Fig.1.1 explicitly deals with the optimal economic operation of the process. Generally, there is more multivariable coordination as we move upwards in the hierarchy [168]. The economic optimization of any process performance in the context of real-time optimization is becoming more crucial in

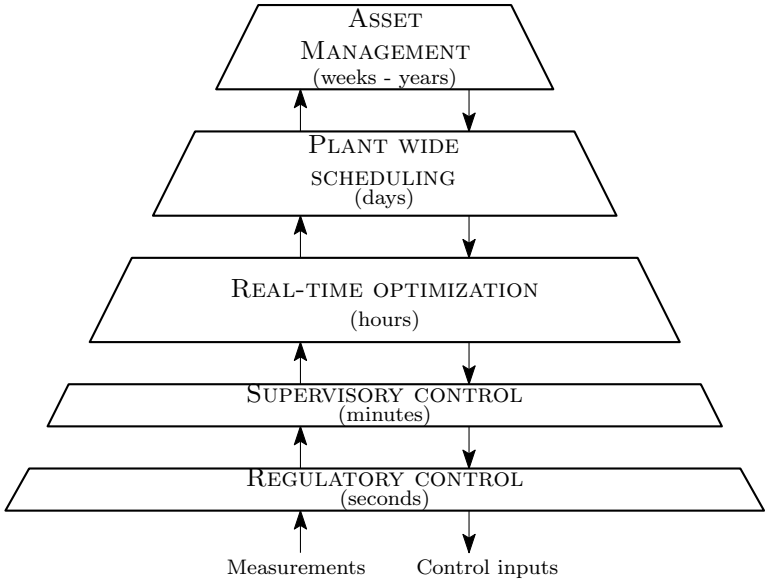


Figure 1.1: Typical control hierarchy in process control

the face of growing competition, increasing demands, and the necessity to focus on sustainability and energy efficiency. Process optimization directly enables safe operation, cost reduction, improving product quality and meeting environmental regulations and this is the main focus of the RTO layer.

A widely accepted definition of real-time optimization is that it is a work flow where the *decision variables are optimized using the system model and the economic model* along with the *process constraints* by solving some kind of mathematical optimization [63]. In order to account for process disturbances and plant-model mismatch, there has been advancements in measurement-based optimization that adjust the optimal inputs in real-time, hence defining RTO as a workflow that optimizes process performance by iteratively adjusting the decision variables using measurement data [28]. A good overview and classification of the different RTO methods can be found in [28] and [170].

In many process control applications, real-time optimization uses *nonlinear steady-state* process models to compute the optimal setpoint at steady-state operation [158]. The justification for using steady-state models is twofold; 1) the economic operation of the plant often occurs at steady-state operation, 2) steady-state models are more easily available and can be much simpler [158]. RTO is also provided with constraints such as process and equipment constraints, storage and capacity constraints, product quality constraints etc. In addition, RTO uses an economic model that constitutes the cost of raw material, value of the products, operational costs, environmental regulations etc. to evaluate the economics of operation.

Traditionally, RTO implementation is based on steady-state nonlinear models that are parameterized by a set of unknown or uncertain parameters, which are updated using measurement data. The updated model is then used to compute the

optimal set of decision variables by solving a numerical optimization problem. The repeated identification and optimization scheme using steady-state models is used in many commercial RTO software packages [22].

Despite the economic benefits and promises, traditional real-time optimization is not used commonly in practice. Consequently, the full potential of RTO is not exploited in process industries. Therefore, the main research question that this thesis deals with is,

"Why is traditional real-time optimization not commonly used in industry, and how can these challenges be addressed?"

1.1 Challenges with traditional RTO

The main challenges which limits the industrial use of RTO include:

- **Challenge 1** - Cost of developing the model (offline).
- **Challenge 2** - Model uncertainty, including wrong values of disturbances and parameters (online update of the model).
- **Challenge 3** - Numerical robustness, including computational issues of solving optimization problems.
- **Challenge 4** - Frequent grade changes, which makes steady-state optimization less relevant.
- **Challenge 5** - Dynamic limitations, including infeasibility due to (dynamic) constraint violation.
- **Challenge 6** - Problem formulation - choosing the right formulation for the right problem.

The different challenges, along with the solutions proposed in this thesis for each challenge is described below.

Challenge 1: Cost of developing model

The cost of developing a model is the biggest bottleneck in the traditional RTO paradigm. Developing good first principle-based models is often challenging and expensive, especially for new application areas with limited domain knowledge. In addition, lack of knowledge or model simplification lead to mismatch between the physical models used in the optimizer and the real system. With increasing complexity of many industrial processes, simplified first-principle models are insufficient to accurately capture the system behavior.

Proposed solution: Model-free optimization approaches such as extremum seeking control as proposed in Chapter 5 may be used to circumvent the need for developing complex models. In addition, Chapter 4 proposes a systematic approach to using classical feedback controllers and simple logic structures to switch between active constraint regions. Machine learning approaches may also be used to address this challenge as briefly discussed in Appendix M.

Challenge 2: Online update of the model

Since traditional RTO uses steady-state models, the model adaptation step must be carried out using measurements that corresponds to steady-state operation. A steady-state detection algorithm is used to detect if the process is operating at steady-state conditions. This is known as steady-state wait time. In a recent review paper on current practices of RTO, Darby et al. [34] concludes that a fundamental limiting factor of RTO implementation is the steady-state wait-time associated with the online update of the model. If the process is frequently subject to disturbances, or if the settling times are rather long, this can lead to the plant being operated in transients for significant periods of time. With the inadequacy of steady-state conditions, the model is not updated frequently, leading to wrong values of disturbances and parameters in the model. Consequently the plant is operated suboptimally for long periods of time.

Proposed solution: To address the problem of steady-state wait time, several different approaches are proposed in this thesis. In Chapter 2, we propose a "hybrid" combination of steady-state and dynamic RTO approach, where transient measurements are used in the traditional two-step steady-state RTO paradigm. In addition, this thesis also proposes different alternative approaches to RTO that use transient measurements. For example, a novel model-based gradient estimation scheme using transient measurements is proposed in Chapter 3, that does not require the steady-state wait time. Chapter 5 proposes a novel dynamic extremum seeking scheme using transient measurements that results in a significantly faster convergence to the optimum, since it does not require the *static map* assumption (unlike most extremum seeking schemes). Additionally, the different methods proposed in Chapters 4 and 6 also use transient measurements, hence eliminating the need for steady-state wait time.

Challenge 3: Computational issues and numerical robustness

Solving numerical optimization problem to compute the optimal setpoints, leads to high computational effort. Although the computational cost is considerably less for solving steady-state optimization problems than dynamic optimization problems, the optimization problem may still fail to converge for large-scale processes (numerical robustness). Therefore, there is a clear need to develop alternative approaches to RTO that does not require solving numerical optimization problems online.

Proposed solution: To avoid solving numerical optimization problems online, we propose to convert the steady-state optimization problem in to a feedback control problem, where the inputs are directly manipulated based on the feedback measurements. The Hybrid RTO approach in Chapter 2 is converted to a feedback problem in Chapter 3. A systematic approach for using classical feedback controllers along with advanced control elements such as selectors is proposed in Chapter 4, which avoids the need for a separate optimization layer. Extremum

seeking control proposed in Chapter 5 and the combination of extremum seeking control and self-optimizing control in Chapter 6 are also based on feedback control.

Challenge 4: Frequent grade changes, which make steady-state optimization less relevant

Processes with frequent changes in feed, product specifications, market disturbances, frequent grade transitions, cyclic operations and batch processes etc. make traditional steady-state RTO less relevant. Such cases require dynamic optimization methods (e.g. Dynamic RTO or economic NMPC). However, solving dynamic optimization problems are computationally intensive, even with today's computing power (cf. Challenge 3). This challenge grows even more in the presence of uncertainty. In this case, one cannot avoid solving numerical optimization problems. Hence, there is a need to address the computational cost of solving dynamic optimization problems [25, 49].

Proposed solution: In the presence of uncertainty, this thesis considers the multistage scenario-based formulation in the context of economic NMPC. Algorithms proposed in Chapters 9 and 10 deals with addressing the computation time of multistage economic NMPC by using decomposition methods and parametric optimization concepts.

Challenge 5: Dynamic limitations, including infeasibility due to (dynamic) constraint violation

The optimal solutions computed by the optimization layer is often provided as setpoints to the controllers in the automation layer. It may happen that the setpoints are not feasible for the lower level controllers, and may violate the constraints dynamically. This may be due to the unmodeled effects in the optimization layer or due to the multivariable coupling between the different control loops that are not taken into account in the optimization layer.

Proposed solution : This challenge can be addressed by using a setpoint tracking NMPC in the supervisory control layer for multivariable constrained control as shown in Chapter 2.

Challenge 6: Problem formulation - choosing the right formulation for the right problem

Problem formulation is probably one of the most important, and conceptual challenges with online process optimization. With developments in different alternative approaches to process optimization, ranging from traditional model-based formulation, to economic MPC, extremum seeking control, classical feedback control, modifier adaptation etc., a proper understanding of the advantages and disadvantages of the different approaches is lacking. Often, the different approaches are seen as competing to one another. There is no single available formulation that addresses all the challenges above.

Industrial processes differ in their infrastructure (available sensors and manipulators, computational platforms etc.), value chain (which affects the objective function) and safety criticality (robustness vs. performance) to name a few. For example, in many applications, the economic gain by using dynamic optimization may be negligible, while in others it may not be. In general, there is a lack of consensus in the literature on the use of steady-state versus dynamic problem formulation. Some applications may call for fast disturbance rejection, while some other applications can tolerate disturbances for a longer period of time. In the presence of uncertainty, some applications may require hard robust constraint satisfaction at the cost of conservativeness (safety criticality), while it may not be the case in some other applications. Understanding the needs of the application at hand, and choosing the right formulation is therefore a key factor in successful industrial application of real-time optimization.

Proposed solution: The different approaches to RTO are not contradictory, but indeed complementary. This is demonstrated in Chapter 6 using self-optimizing control and extremum seeking control. Furthermore, in the conclusion section (page 185), we attempt to provide an overview of the different approaches, comparing the advantages and disadvantages of the different approaches, in the hope that this might serve as a guideline/cheat-sheet in choosing the right problem formulation. Chapter 7 particularly considers the problem formulation of economic NMPC under uncertainty and proposes the multistage scenario-based formulation as one of the promising alternatives, that provide a certain degree of flexibility in the problem formulation.

Challenges related to human aspects

When considering our research question "*Why is traditional real-time optimization not commonly used in industry?*", one cannot ignore the human aspects. Besides the different technological challenges discussed above, one of the main bottlenecks to widespread application of real-time optimization, arises from human aspects that include the end-user's ability to learn, understand, and use the technology over a prolonged period of time. A recent industrial survey published in the International Federation of Automatic Control (IFAC) newsletter [153] aptly identifies *people* and *human aspects* as one of the major components when addressing challenges related to adopting new technology. This is also pointed out by several researchers in the field of process control and optimization, see for e.g. [49, 126, 129] to name a few. Indeed, most practitioners will also point to challenges related to human aspects as the most important among all the challenges listed here. The human aspects can be broadly divided into corporate culture and technical competence.

Corporate culture : Corporate culture forms the foundation of how an organization works, and plays a vital role in adopting a new technology. The corporate culture in some organizations may be such that, major changes such as deployment of new technology are resisted. Instead, one prefers "trusted" technology in order to minimize liability [129]. "Operator confidence" is another important aspect, as they

are the end users. Failure to gain operator confidence will lead to an unsuccessful implementation of the technology.

Technical competence : Lack of competence and training is another major issue when adopting advanced optimization tools. Models and optimization tools require regular maintenance and re-tuning, in order to sustain the performance improvements. For example, changes in feed conditions, instrument and equipment degradation and changes in process equipment leads to performance degradation over time. The expected benefits from using online optimization tools are at a risk, without regular monitoring and maintenance [49]. This was also pointed out on a special report on process control in the Oil and Gas Journal [163].

Since the optimization layer is generally a multivariable and large-scale problem, the complexity and the understanding of the optimization concepts presents key challenges for the end users, as also previously pointed out by Qin and Badgwell [139] and Mayne [125]. Often, expert knowledge is required to perform the maintenance, which may be limited in the organization[1]. With increasing number of applications, there is a paucity of skilled engineers to provide maintenance and support, to sustain the benefits. As noted by Forbes et al. [49], skilled engineers involved in the initial implementation are often not available for maintenance, resulting in performance degradation, and the application being turned off by the operators.

Therefore, when addressing the different challenges listed in Section 1.1, it is imperative to take into account the human aspects. The different methods proposed in thesis are also influenced by the challenges related to human aspects. The method proposed in Chapter 4 is the perfect example of this, since it is based on classical control tools and simple logic blocks that have been in use for several decades in the process industry. In Chapter 7, we again consider the human aspects when justifying the multistage formulation as a promising approach to dynamic optimization under uncertainty.

1.2 Main contributions of the thesis

The main contributions of this thesis are the novel methods and algorithms that are proposed in the different chapters to address the challenges listed above. The key contributions[2] are now listed chapter-wise:

Chapter 2

- **Hybrid RTO approach with dynamic model update and steady-state optimization to avoid steady-state wait time.**
- *Application - Demonstrated using an oil and gas production optimization problem with 2 wells in the chapter and 6 wells in [105].*

[1] This is also my first-hand experience from Statoil.
[2] theory in bold and application in italics

Chapter 3

- **A novel gradient estimation algorithm using nonlinear models and transient measurements (Theorem 3.1).**
- *Application - Demonstrated using a CSTR process with 2 components in the chapter. This method was also successfully tested on a 3-bed ammonia reactor example [18], oil and gas production optimization with 2 wells [96] and 6 wells [100], evaporator process [103], isothermal CSTR with 4 components [89], which are all appended to this thesis.*

Chapter 4

- **Linear gradient combination (4.9) as optimal controlled variables (Theorem 4.1).**
- **Systematic approach to designing selectors for CV-CV switching (Theorem 4.2).**
- *Application - Demonstrated using a CSTR process with 2 components and an oil and gas production optimization problem with 6 wells, and an isothermal CSTR process with 4 components in the chapter. The appendix includes an experimental evaluation of this approach for optimal operation of an electrical submersible pump lifted well.*

Chapter 5

- **Novel dynamic extremum seeking scheme with fixed linear dynamics for Hammerstein systems.**
- **Bounds on neglected linear dynamics for robust stability.**
- *Application - Demonstrated using a pressure oscillation damper in lean burn combustors.*

Chapter 6

- **Hierarchical combination of extremum seeking control and self-optimizing control for improved performance.**
- *Application - Demonstrated using a 3-bed ammonia reactor example.*

Chapter 8

- *Application - Application of multistage scenario-based MPC to an oil and gas production optimization problem.*

Chapter 9

- **A Distributed multistage scenario MPC framework using primal decomposition to ensure feasibility of the non-anticipativity constraints (Algorithm 9.2).**
- **A backtracking algorithm to choose the step-length in the master problem (Algorithm 9.1).**

- *Application - Demonstrated using a CSTR process with 2 components in the chapter, in addition to an oil and gas production optimization problem [94], which is appended to the thesis.*

Chapter 10

- **A sensitivity-based distributed multistage scenario MPC to reduce the number of NLPs that needs to be solved (Corollary 10.2 and Algorithm 10.1).**
- *Application - Demonstrated using a CSTR process with 2 components.*

In addition to the different methods and algorithms proposed, perhaps one of the important contributions of this thesis is that it aims to provide an overview and a clear understanding of the different approaches to online process optimization, which is summarized in Table 10.2. Although chapter 7 does not present any novel material, it presents interesting discussions on optimization problem formulation under uncertainty.

Some other minor contributions include:

- A systematic approach to select the discrete scenarios for multistage NMPC from historical process data using principal component analysis (see Appendix K).
- Algorithm to shrink the uncertainty set online using recursive Bayesian weighting for time-invariant parametric uncertainty in the context of multistage scenario MPC (See Appendix L)

1.3 Structure of the thesis

Part I - The first part of the thesis deals with optimal steady-state operation and looks into how transient measurements can be used in order to address the steady-state wait time problem. In addition, it also presents some algorithms to achieve optimal operation without the need to explicitly solve numerical optimization problems online. To address the challenge of developing models, some model-free optimization tools are also presented in Part I. In general, Part I of this thesis deals with the use of "simple" tools for RTO, that are motivated by industrial needs.

Part II - The second part of the thesis deals with dynamic optimization problem, and in particular addresses the problem of computation cost of solving the economic NMPC problem. To handle uncertainty in the economic NMPC problem, we consider the multistage scenario-based problem formulation, which we propose to solve using primal decomposition, in order to ensure close-loop implementation.

Finally, an overview of the different RTO approaches are summarized in Table 10.2. The thesis concludes with some useful discussions and preliminary ideas that paves the way for future research directions.

Chapters 1-11, that presents the main contributions, are all based on journal publications. All other peer-reviewed publications that are not included in Chapters 1-11, are appended to this thesis.

1.4 Industrial relevance and impact

As mentioned earlier, the main research focus of this thesis is motivated by the realization that real-time optimization is not used as much in practice as one would expect. We consider the different challenges in detail and provide various solutions to address the different challenges listed above. Furthermore, the method proposed in Chapter 4 may be immediately applicable in practice, since this is based on classical feedback controllers and simple logic blocks that are used widely in process industries.

The different approaches and algorithms proposed in this thesis are based on control technologies that have a high impact on industry. In April 2019, the industrial committee of the International Federation of Automatic Control (IFAC) published a list of control technologies along with its current and future impact [153] (A survey article with very similar conclusion was also published in the IEEE Control Systems Magazine [152]). Comparing the survey results in [152, 153], it can be seen that the different methods and algorithms proposed in this thesis are in fact based on the top five control technologies listed in this survey. This is shown in Table. 1.1, which is indicative of the industrial relevance and impact of this thesis, now and in the future.

Table 1.1: Control technologies used in this thesis, along with its impact based on the industrial survey from the 2019 IFAC newsletter[153]

Control Technology	Current Impact	Future Impact	Used in Chapter
PID control	91%	78%	3,4,6
System Identification	65%	72%	5
Estimation and filtering	64%	63%	2,3,8
Model Predictive Control	62%	85%	2,7–10
Process Data Analytics	51%	70%	App K,L
Fault detection	48%	8%	–
Decentralized/coordinated control	29%	54%	9,10
Robust Control	26%	42%	–
Intelligent Control	24%	59%	–
Nonlinear Control	21%	42%	–
Discrete-event systems	24%	39%	–
Adaptive Control	18%	44%	–
Repetitive Control	12%	17%	–
Other advanced control	11%	25%	–
hybrid dynamical systems	11%	33%	–
Game theory	5%	17%	–

1.5 List of publications

Over the past three years, my PhD work has resulted in 11 journal papers and 19 peer-reviewed conference papers and more than 17 abstract-only/invited presentations at workshops and meetings.

1.5.1 Journal papers

1. Krishnamoorthy, D., Foss, B. and Skogestad, S., 2018. Steady-State Real-time Optimization using Transient Measurements. *Comput. & Chem. Eng.*, Vol.115, p.34-45 - **Chapter 2**
2. Krishnamoorthy, D., Jahanshahi, E. and Skogestad, S., 2019. Feedback Real-Time Optimization Strategy Using a Novel Steady-state Gradient Estimate and Transient Measurements, *Ind. Eng. Chem. Res.* Vol.58 (1), p. 207-216 - **Chapter 3**
3. Krishnamoorthy, D., and Skogestad, S., 2019. Online process optimization with changes in active constraint sets using simple feedback control structures, *Ind. Eng. Chem. Res.*, Vol.58 (30), p. 13555-13567 - **Chapter 4**
4. Krishnamoorthy, D., Fjalestad, K. and Skogestad, S., 2019. Optimal operation of offshore oil and gas production using simple control structures, *Control Engineering Practice*, Vol.91 - **Chapter 4**.
5. Krishnamoorthy, D., and Skogestad, S., 2019. A Fast Robust Extremum Seeking Scheme using Transient Measurements, *Automatica*. (Under review) - **Chapter 5**
6. Straus, J., Krishnamoorthy, D. and Skogestad, S., 2019. Combining self-optimizing control and extremum seeking control - Applied to ammonia reactor case study, *J. Proc. Control.* Vol.78, p. 78-87 - **Chapter 6**
7. Krishnamoorthy, D., Foss, B. and Skogestad, S., 2016. Real-Time Optimization under Uncertainty Applied to a Gas Lifted Well Network. *Processes*, Vol.4(4), p.52 - **Chapter 8**
8. Krishnamoorthy, D., Foss, B. and Skogestad, S., 2018. A Primal decomposition algorithm for distributed multistage scenario model predictive control. *J. Proc. Control.*, Vol.81, p.162-171- **Chapter 9**
9. Krishnamoorthy, D., Foss, B. Suwartadi, E., Jäschke, J. and Skogestad, S., 2018. Improving Scenario Decomposition for Multistage NMPC using a Sensitivity-based Path-following Algorithm, *IEEE Control System Letters*, Vol.2(4), p.581-586 - **Chapter 10**
10. Krishnamoorthy, D., Biegler, L. T. and Jäschke, J., 2019. Adaptive Horizon Economic Model Predictive Control using turnpike properties, *J. Proc. Control.* (In-preparation)
11. Jahanshahi, E., Krishnamoorthy, D., Codas, A., Foss, B. and Skogestad, S., 2019. Plantwide control of an oil production network, *Comput. & Chem. Eng.* (Under review) - **Appendix G**

1.5.2 Peer-reviewed conference papers

1. Krishnamoorthy, D., Thombre, M., Jäschke, J. and Skogestad, S., 2018. Data-driven scenario selection for multistage robust model predictive control, IFAC-PapersOnline, Vol.51 (20), p.462-468 (*IFAC Workshop on Nonlinear Model Predictive Control*, Madison, WI, USA) - **Appendix K**
2. Krishnamoorthy, D., Jäschke, J. and Skogestad, S., 2019. Multistage Model Predictive Control with Online Scenario Tree Update using Recursive Bayesian Weighting, *Proceedings of the 18th European Control Conference*, p.1443-1448, Naples, Italy - **Appendix L**
3. Krishnamoorthy, D., and Skogestad, S., 2019. Real-time optimization strategies using surrogate optimizers, *Foundation on Process Analytics and Machine Learning (FOPAM)*, Raleigh, NC, USA, Aug 2019 (Poster). - **Appendix M**
4. Krishnamoorthy, D., Jahanshahi, E. and Skogestad, S., 2018. Gas-lift Optimization by Controlling Marginal Gas-Oil Ratio using Transient Measurements, IFAC-PapersOnLine, Vol.51 (8), p.19-24 (*IFAC Workshop on Automatic Control in Offshore Oil and Gas Production*, Esbjerg, Denmark) - **Appendix C**
5. Bonnowitz, H., Straus, J., Krishnamoorthy, D., and Skogestad, S., 2018. Control of the Steady-State Gradient of an Ammonia Reactor using Transient Measurements, *Computer aided chemical engineering,* Vol.43, p.1111-1116 (ESCAPE 28) - **Appendix C**
6. Krishnamoorthy, D., Jahanshahi, E. and Skogestad, S., 2019. A feedback real-time optimization strategy applied to an evaporator process, *PSE Asia*, Bangkok, Thailand - **Appendix C**
7. Reyes-Lúa, A., Zotica, C., Das, T., Krishnamoorthy, D., and Skogestad, S., 2018. Changing between Active Constraint Regions for Optimal Operation: Classical Advanced Control versus Model Predictive Control, *Computer aided chemical engineering (ESCAPE 28)* Vol.43, p.1015-1020, Graz, Austria - **Appendix F**
8. Krishnamoorthy, D., Foss, B. Suwartadi, E., Jäschke, J. and Skogestad, S., 2018. Improving Scenario Decomposition for Multistage NMPC using a Sensitivity-based Path-following Algorithm, *Proceedings of the 57th IEEE Conference on Decision and Control*, Miami beach, FL, USA - **Chapter 10**
9. Krishnamoorthy, D., Foss, B. and Skogestad, S., 2018. A distributed algorithm for scenario-based model predictive control using primal decomposition, IFAC-PapersOnline Vol.51 (18), p.351-356 (*IFAC Symposium on Advanced Control of Chemical Processes*, Shenyang, China) - **Appendix J**
10. Thombre, M., Krishnamoorthy, D., and Jäschke, J., 2019. Data-driven Multistage Model Predictive Control of a Thermal Storage System with Time-Varying Uncertainty, IFAC-PapersOnline Vol.52 (1), p. 461-467 (*IFAC Symposium on Dynamic Control of Process Systems (DYCOPS-CAB)*, Florianopolis, Brazil) - **Appendix K**

11. Krishnamoorthy, D., Aguiar, M. A. M., Foss, B. and Skogestad, S., 2018. A Distributed Optimization Strategy for Large scale Oil and Gas Production Systems, *Proceedings of the 2nd IEEE International Conference on Control Technology and Applications*, p. 521-526, Copenhagen, Denmark
12. Suwartadi, E.,Krishnamoorthy, D.and Jäschke, J., 2018. Fast Economic Model Predictive Control for a Gas Lifted Well Network, IFAC-PapersOnLine, Vol.51 (8), pp.25-30 (*IFAC Workshop on Automatic Control in Offshore Oil and Gas Production*, Esbjerg, Denmark) - **Appendix N**
13. Krishnamoorthy, D., Ryu, J. and Skogestad, S., 2019. Dynamic extremum seeking control applied to a gas lifted well network, IFAC-PapersOnline, Vol.52 (1), p. 802-807 (*IFAC Symposium on Dynamic Control of Process Systems (DYCOPS-CAB)*, Florianopolis, Brazil) - **Appendix O**
14. Krishnamoorthy, D., Foss, B. and Skogestad, S., 2017. Gaslift optimization under uncertainty. *Computer Aided Chemical Engineering (ESCAPE 27)*, Vol.40, pg 1753-1758, Barcelona, Spain - **Chapter 8**.
15. Krishnamoorthy, D.,Valli, C. and Skogestad, S., 2020. Optimal resource allocation in an industrial symbiotic setting. *Submitted to American Control Conference 2020.*
16. Krishnamoorthy, D. and Skogestad, S., 2020. Multivariable extremum seeking control using fast Fourier transform. *Submitted to IFAC World Congress 2020.*

1.5.3 Conference papers published during my PhD studies, but not included in the thesis

1. Delou, P., Azevedo, J., Krishnamoorthy, D., de Souza Jr, M. and Secchi, A., 2019. Model Predictive Control with Reconfiguration Strategy applied to an Electric Submersible Pump in a subsea environment, IFAC DYCOPS-CAB, Florianopolis, Brazil
2. Backi, C. J.,Krishnamoorthy, D.and Skogestad, S., 2018. Slug handling with a virtual harp - based on nonlinear predictive control for a gravity separator, *IFAC-PapersOnLine*, 51(8), pp.120-125 (IFAC OOGP, Esbjerg, Denmark).
3. Backi, C. J., Krishnamoorthy, D., Verheyleweghen, A. and Skogestad, S., 2018. Combined nonlinear moving horizon estimation and model predictive control applied to a compressor for active surge control, *IEEE Conference on Control Technology and Applications*, Copenhagen, Denmark.

1.5.4 Relevant conference papers published before starting PhD studies, not included in the thesis

1. Krishnamoorthy, D., Pavlov, A. and Li, Q., 2016. Robust Extremum Seeking Control with application to Gas Lifted Oil Wells. IFAC-PapersOnLine, 49(13), pp.205-210
2. Krishnamoorthy, D., Bergheim, E.M., Pavlov, A., Fredriksen, M. and Fjalestad, K., 2016. modeling and Robustness Analysis of Model Predictive Control for

Electrical Submersible Pump Lifted Heavy Oil Wells. IFAC-PapersOnLine, 49(7), pp.544-549 (IFAC DYCOPS, Trondheim, Norway)

3. Pavlov, A., Krishnamoorthy, D., Fjalestad, K., Aske, E. and Fredriksen, M., 2014, October. Modeling and model predictive control of oil wells with electric submersible pumps. IEEE Conference on Control Applications (pp. 586-592). (Antibes, France)

1.5.5 Additional presentations (invited, or with abstract-only)

1. Krishnamoorthy, D., Biegler, L. T. and Jäschke, J., Fast Advanced-Step Economic MPC Based on Turnpike Properties, *AIChE Annual Meeting*, Orlando, FL, USA, Nov 2019.
2. Krishnamoorthy, D. and Skogestad, S. Novel Approaches to Online Process Optimization Under Uncertainty, *AIChE Annual Meeting*, Orlando, FL, USA, Nov 2019.
3. Krishnamoorthy, D. and Skogestad, S. Online Process Optimization with Active Constraint Set Changes using Simple Control Structures, *22nd Nordic Process Control workshop*, Lyngby, Denmark, Aug 2019.
4. Krishnamoorthy, D., Jäschke, J. and Skogestad, S. Overview and classification of online process optimization approaches, *IFAC DYCOPS Pre-symposium workshop*, Florianopolis, Brazil, Apr 2019.
5. Dos Santos, A., Secchi, A., de Souza, M., Skogestad, S. and Krishnamoorthy, D. Gray-box Identification using Polynominal NARMAX Models, *Foundations on Process Analytics and Machine Learning (FOPAM)*, Raleigh, NC, USA, Aug 2019 (Poster).
6. Krishnamoorthy, D., and Skogestad, S., 2019. A Fast Robust Extremum Seeking Control using Transient Measurements, *Foundation on Process Analytics and Machine Learning (FOPAM)*, Raleigh, NC, USA, Aug 2019 (Poster).
7. Krishnamoorthy, D. Production optimization using simple control loops, *Subsea Valley Conference*, Oslo, Norway, Apr 2019. (*Invited*)
8. Krishnamoorthy, D. Efficient Real-time optimization using transient measurements, *CAPE Forum*, Bucharest, Romania, Nov 2018.
9. Krishnamoorthy, D. and Skogestad, S. An Overview and Evaluation of Approaches for Online Process Optimization, *AIChE Annual Meeting*, Pittsburgh, PA, USA, Nov 2018.
10. Krishnamoorthy, D., Jäschke, J. and Skogestad, S. An Efficient Distributed Algorithm for Multistage Scenario Model Predictive Control Using Primal Decomposition, *AIChE Annual Meeting*, Pittsburgh, PA, USA, Nov 2018.
11. Krishnamoorthy, D., Jäschke, J. and Skogestad, S. Perspectives on Nonlinear Model Predictive Control under Uncertainty. *12th UKACC International Conference on Control*, Sheffield, United Kingdom, Sept 2018. *(Accepted, but withdrawn)*

12. Krishnamoorthy, D., Overview of Real-time optimization approaches under uncertainty, Federal university of Rio de Jenerio (UFRJ-COPPE), Brazil, May 2018. (*Invited*)
13. Krishnamoorthy, D., Efficient production optimization strategies using transient measurements, *VII Brazil-Norway Production Optimization workshop*, Rio De Jenerio, Brazil, May 2018. *(Invited)*
14. Krishnamoorthy, D., Jahanshahi, E. and Skogestad, S., Feedback Real time optimization strategy using transient measurements, *21st Nordic Process Control workshop*, Åbo, Finland, Jan 2018.
15. Krishnamoorthy, D., Foss, B. and Skogestad, S. Model Predictive Control Under Model Structural Uncertainty, *AIChE Annual Meeting*, Minneapolis, MN, USA, Nov 2017.
16. Krishnamoorthy, D., Straus, J. and Skogestad, S. Combining Self-optimizing control and extremum seeking control-Applied to Ammonia Reactor Case Study, *AIChE Annual Meeting*, Minneapolis, MN, USA, Nov 2017.
17. Krishnamoorthy, D. Production optimization under uncertainty, *NFA Subsea conference*, Kristiansand, Norway, Sept 2017. *(Invited)*
18. Krishnamoorthy, D. Multistage scenario-based model predictive control, *Federal University of Santa Catarina*, Florianopolis, Brazil, May 2017. (*Invited*)
19. Krishnamoorthy, D. Real-Time Optimization under Uncertainty Applied to a Gas Lifted Well Network, *VI Brazil-Norway Production Optimization workshop*, Rio De Jenerio, Brazil, Apr 2017. *(Invited)*
20. Krishnamoorthy, D., Pavlov, A. and Li, Q., Robust Extremum Seeking Control -applied to gas lift optimization, *20th Nordic Process Control workshop*, Stockholm, Sweden, Aug 2016.

Part I

Optimal Steady-state Economic Operation using Transient Measurements

Chapter 2

Steady-State Real-Time Optimization using Transient Measurements

In order to address the steady-state wait time, this chapter proposes a "hybrid" approach where the model adaptation is done using dynamic models and transient measurements and the optimization is performed using steady-state models. The proposed Hybrid RTO provides similar performance to dynamic RTO, at computation times similar to steady-state RTO.

Based on the article published in Computers and Chemical Engineering [95].

2.1 Introduction

Although there have been recent developments in different approaches to RTO [28], the most common approach to commercial RTO implementation is the so-called two-step model-adaptation approach [29], [123] as shown in Fig. 2.1. The steady-state model used in this approach is parameterized by a set of unknown or uncertain parameters, which are updated using measurement data in the first step. In the second step, the updated model is used to compute the optimal set of decision variables by solving a numerical optimization problem. The repeated identification and optimization scheme using steady-state models is used in many commercial RTO software packages [22].

However, traditional steady-state RTO faces some challenges, which limits its industrial use as described in Chapter 1. In a recent review paper on current practices of RTO [34], the authors conclude that a fundamental limiting factor of RTO implementation is the steady-state wait-time associated with the online update of the model. Since only steady-state models are used, the model adaptation step must be carried out using measurements that correspond to steady-state operation. If the process is frequently subject to disturbances or, if the settling times are rather long, this can lead to the plant being operated in transients for significant periods of time. With the inadequacy of steady-state measurements, the model is

not updated frequently. Consequently the plant is operated suboptimally for long periods of time.

Darby et al. [34] briefly suggest the approach of using dynamic terms that would impact only the model adaptation step as a potential research direction to address this issue. In this chapter, we further analyze the approach of using dynamic model adaptation and transient measurements. We show that by using a hybrid approach with dynamic models for estimation together with steady-state models for optimization, the problems with steady-state detection (SSD) and model adaptation can be handled more efficiently, hence leading to an efficient RTO implementation.

The main contribution of this chapter is the hybrid RTO approach (i.e. steady-state optimization with dynamic model adaptation) that directly addresses the issue of steady-state wait-time.

The rest of the chapter is structured as follows: A brief review of traditional RTO structures and the implementation issues are provided in section 2.2. Modifications to the traditional RTO approach are proposed in section 2.3 which are illustrated using an application example in section 2.4. Discussions on the use of steady-state versus dynamic problem formulation is provided in section 2.5 before concluding the chapter.

2.2 Traditional RTO

2.2.1 Steady-state RTO (SRTO)

The traditional steady-state RTO implementation in Fig. 2.1, based on the two-step approach, is briefly summarized below:

- Steady-state detection and data pre-processing - This initial step detects if the plant is operating close enough to steady-state, to start the RTO sequence.
- Steady-state parameter estimation (Step 1) - The model parameters are adjusted to match the current data, using regression techniques. The parameter estimation step usually consists of data reconciliation and model adaptation. The measurement data is screened for unreasonable data such as gross errors, for example based on material and energy balances. Suitable actions are taken to rectify or eliminate any erroneous data before it is used to update the model. Considerable process knowledge may be required to decide which model parameters needs to be updated as noted by Seborg et al. [158, Ch.19] and Quelhas et al. [140].
- Steady-state Optimization (Step 2) - Given an objective function, process constraints and an updated model, the optimum setpoints are computed using mathematical optimization methods.

The setpoints computed by the RTO are then provided to the lower layer supervisory control, where a dynamic optimization problem may be solved online, typically using simplified linear models with constraints (in the framework of NMPC) to minimize the deviation of the measurements from the setpoint over a period of time. Fig. 2.1 shows a typical RTO structure, where the steady-state process data is used for adapting the steady-state model which is used in the steady-state RTO.

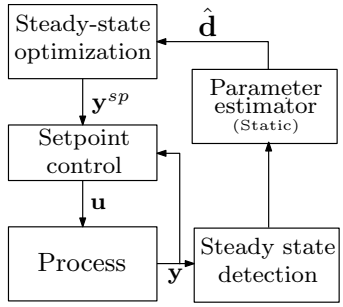


Figure 2.1: Traditional RTO with steady-state model adaptation and steady-state optimization.

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

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k) \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \end{aligned} \tag{2.1}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$, $\mathbf{u}_k \in \mathbb{R}^{n_u}$ and $\mathbf{y}_k \in \mathbb{R}^{n_y}$ are the states, process inputs and process measurements at time step k respectively. The model is parameterized by a set of time-varying parameters and disturbances jointly represented by $\mathbf{d}_k \in \mathbb{R}^{n_d}$. The model equations are represented by $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$ and $\mathbf{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$.

Let the steady-state counterpart for this model be described by,

$$\mathbf{y} = \mathbf{f}_{ss}(\mathbf{u}, \mathbf{d}) \tag{2.2}$$

where $\mathbf{f}_{ss} : \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_y}$ describes the steady-state input-output equilibrium map.

Once the plant is operating at steady-state, the model parameters are updated using the steady-state measurements (step 1 of 2). The model parameter adaptation scheme is based on minimizing the error between the model predicted value and the measurement data.

The updated parameter vector $\hat{\mathbf{d}}_k$ is then used in the optimization problem (step 2 of 2). The optimization problem then computes the optimal setpoints $\mathbf{y}^{sp}$ that optimizes the process performance, while satisfying process and operating constraints. The steady-state optimization problem using the two-step approach for this system can thus be stated mathematically as follows,

Step 1: Steady-state Estimation

$$\hat{\mathbf{d}}_k = \arg\min_{\mathbf{d}_k} \|\mathbf{y}_{meas} - \mathbf{f}_{ss}(\mathbf{u}_k, \mathbf{d}_k)\|_2^2 \tag{2.3}$$

Step 2: Steady-state Optimization

$$\begin{aligned} &\mathbf{u}_{k+1}^* = \arg\min_{\mathbf{u}} J(\mathbf{y}, \mathbf{u}) \\ &\text{s.t. } \mathbf{y} = \mathbf{f}_{ss}(\mathbf{u}, \hat{\mathbf{d}}_k) \\ &\quad \mathbf{g}(\mathbf{y}, \mathbf{u}) \leq 0 \end{aligned} \tag{2.4}$$

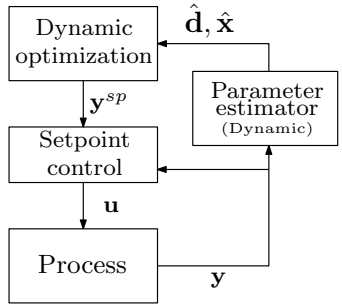


Figure 2.2: Dynamic RTO with dynamic model adaptation and dynamic optimization.

where $\mathbf{y}_{meas} \in \mathbb{R}^{n_y}$ denotes the measurements from the plant, $J : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}$ describes the objective function, $\mathbf{g} : \mathbb{R}^{n_u} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_c}$ describes vector of nonlinear constraints that may be imposed such as process and operating constraints including bounds on the process inputs and outputs. Note that J and $\mathbf{g}$ are not directly dependent on the disturbances $\mathbf{d}$, but implicitly via the process outputs $\mathbf{y}$ governed by the model (2.2).

Challenges with steady-state detection Many commercial RTO software uses either statistical methods or heuristic methods or a combination of both to verify the stationarity of the data for a fixed window length in the past. A detailed description of the different steady-state detection routines used in commercial RTO systems can be found in [22]. Tolerances are specified by the user to determine if the process is "close enough" to steady-state and, the process is said to have reached steady-state when all the measurements are within the specified tolerances [26]. If the tolerances are specified without proper evaluation of the data window length, then the steady-state detection might erroneously accept transient data as stationary data. Using transient data to update steady-state models results in estimation errors, which are then propagated to the optimization routine. Câmara et al. [22] demonstrated this issue using data from a real industrial application.

Another challenge in many processes is that, it may be frequently subject to disturbances. This results in the process being operated mostly at transients, thus hindering model adaptation. This is further worsened if the process has long settling times. In such processes, the model parameters are not updated frequently due to inadequate availability of steady-state measurements. Consequently, the process may be operated suboptimally for long periods of time, until the model parameters are updated again.

2.2.2 Dynamic RTO (DRTO)

In the recent past, there has also been many developments in the use of *Dynamic RTO* (DRTO) and the closely related economic nonlinear model predictive control (economic NMPC), that provides an optimal input trajectory using a dynamic model instead of a steady-state model. Consider the dynamic system (2.1), the

two-step approach to dynamic RTO at each time step k can be given by,

Step 1: Dynamic Estimation

$$\hat{\mathbf{d}}_k = \arg\min_{\mathbf{d}_k} \|\mathbf{y}_{meas,k} - \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)\| \tag{2.5}$$
$$s.t.\ \ \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{d}_{k-1})$$

Step 2: Dynamic Optimization

$$\mathbf{u}_k^* = \arg\min_{\mathbf{u}_k} \sum_{k=0}^{N-1} J(\mathbf{y}_k, \mathbf{u}_k) \tag{2.6}$$
$$\begin{aligned} s.t.\ \ & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \hat{\mathbf{d}}_k) \\ & \mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathbf{g}(\mathbf{y}_k, \mathbf{u}_k) \leq 0 \\ & \mathbf{x}_k = \hat{\boldsymbol{x}}_t \qquad\qquad \forall k \in \{0, \ldots, N-1\} \end{aligned}$$

where the subscript $(\cdot)_k$ represents each sample in the optimization horizon of length N and, $\hat{\boldsymbol{x}}_t$ represents the state estimate at the current time step.

Although the use of dynamic models for model adaptation and optimization may eliminate the requirements of steady-state detection, solving dynamic nonlinear optimization problem for large-scale systems may be challenging, even with today's computing power. Campos et al. [25] points out that, many numerical issues associated with DRTO must be addressed before it can be widely implemented in industrial applications. In addition, the dynamic model requires additional parameters, including a model of the lower-layer control system. Steady-state RTO is therefore still more prevalent in many industrial applications.

Steady-state RTO uses the same steady-state model in both the steps of the two-step approach. Similarly, dynamic RTO uses the same dynamic model in both the steps of the two-step approach. To address the computational challenges in dynamic optimization and to address the steady-state wait-time issue in steady-state RTO, a "hybrid RTO" structure can be considered, where dynamic models are used in the model adaptation step and steady-state models are used in the optimization step.

2.3 Hybrid RTO (HRTO)

If the primary objective is to optimize the steady-state performance of the process, then dynamic terms in the model needs to be introduced only in the model adaptation step. When dynamic models are used in the model adaptation, transient data can be used to update the model, without the need to discard big chunks of data. The updated model parameters can then be used in the steady-state model used in the optimizer, as shown in Fig. 2.3. To illustrate this, consider the discrete dynamic model (2.1) and the corresponding steady-state model (2.2). The model

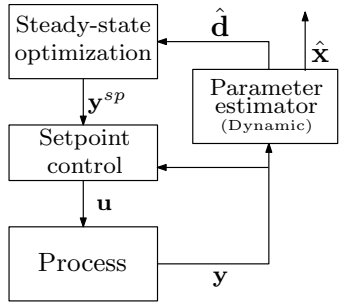


Figure 2.3: Hybrid RTO with dynamic model adaptation and steady-state optimization.

adaptation via the two-step approach would then be given by,

Step 1: Dynamic Estimation

$$\hat{\mathbf{d}}_k = \arg\min_{\mathbf{d}_k} \|\mathbf{y}_{meas,k} - \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)\| \tag{2.7}$$
$$\text{s.t. } \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{d}_{k-1})$$

Step 2: Steady-state Optimization

$$\mathbf{u}^*_{k+1} = \arg\min_{\mathbf{u}} J(\mathbf{y}, \mathbf{u}) \tag{2.8}$$
$$\text{s.t. } \mathbf{y} = \mathbf{f}_{ss}(\mathbf{u}, \hat{\mathbf{d}}_k)$$
$$\mathbf{g}(\mathbf{y}, \mathbf{u}) \leq 0$$

At every time step k, the dynamic model estimator provides the estimate of the uncertain variables $\hat{\mathbf{d}}_k$ and, the steady-state optimization problem is solved with the updated model to find the new optimal steady-state operating point. Therefore, as opposed to the traditional steady-state RTO, the steady-state optimization problem in the hybrid RTO approach is solved at each time step k and the resulting optimal setpoints are provided to the setpoint tracking control.

Development of model-based control design around 1960s, together with seminal works such as the Kalman filter [78], led to the development of identification theory in control literature. Different methods exists today that can be used to estimate the unknown variables in a dynamic model. Some of the methods that are commonly used include, but are not restricted to, recursive least squares estimation, nonlinear Kalman filter variants such as extended Kalman filter (EKF) and unscented Kalman Filter (UKF), optimization-based methods such as the moving horizon estimator (MHE) etc. In the reminder of this chapter, we consider the Hybrid RTO approach using an extended Kalman filter for online parameter estimation. The use of EKF where parameter estimation is the focus can be found in several examples in literature [164]. The framework of using an extended Kalman filter for combined state and parameter estimation can be found in Appendix B.

Modeling effort - In most chemical processes, the so-called first principle models used in RTO applications are based on the conservation of mass, energy and

momentum, which naturally gives rise to ordinary differential equations (ODE). Additional algebraic equations may be specified to model unknown variables, such as flow through an orifice, hydrostatic and frictional pressure drop, reaction stoichiometry, equation of state etc. Once the mathematical models have been defined for a single control volume, multi-staged models such as distillation columns or models with many control volumes can be easily modeled by joining up the same number of mathematical models. A good overview of mathematical modeling for many typical chemical processes can be found in [52].

Often, a chemical engineer starts out the modeling task using dynamic equations. The dynamic models developed using physical principles are converted to steady-state models by setting all the derivative terms to zero, [52]. In many cases, the development of models for dynamic estimation may require little extra modeling effort. However, dynamic models require additional parameters including mass and energy holdups. In addition, the dynamic model used for dynamic optimization (DRTO) needs a model of the lower-layer control system. In the hybrid RTO case (HRTO), where the dynamic model is only used for parameter estimation, a simpler representation of the lower level control system may be sufficient, for example, by not including the control system or assuming perfect control.

Developing and maintaining a dynamic model for the model adaptation step can be justified, if the performance improvement is significant when using the hybrid RTO approach, since this enables better RTO implementation.

2.4 Illustrative example - Oil production optimization

We now demonstrate the use of the hybrid RTO (HRTO) approach and compare it to traditional steady-state RTO (SRTO) and dynamic RTO (DRTO). We consider a gas lifted well network consisting of $n_w = 2$ gas lifted wells connected to a common riser manifold, as shown in Fig. A.1. In oil production wells, when the reservoir pressure is not sufficient to lift the fluids to the surface economically, artificial lift methods are used to boost the production from the wells. Gas lift is one such widely used artificial lift method where compressed gas is injected at the bottom of the well to reduce the density of the fluid column inside the well, thus reducing the pressure drop in the well tubing. The fluids from the reservoir enters the well tubing and mixes with the lift gas. The mixture then flows through the common riser manifold and finally enters the topside processing facility such as a separator where the oil and gas phases are separated. The production network may be constrained by total gas processing capacity or the total gas that is available for gas lift injection. The objective of the production optimization problem is to compute the gas lift injection rate for each well such that the total oil production is maximized while satisfying operational constraints.

Production from a cluster of $\mathcal{N} = \{1, \cdots, n_w\}$ gas lifted wells is modeled (see Appendix A). The process is described as semi-explicit index-1 DAE system of the form.

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) \tag{2.9}$$

$$\mathbf{G}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) = 0 \tag{2.10}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ and $\mathbf{z} \in \mathbb{R}^{n_z}$ represent the differential and algebraic states respectively, $\mathbf{u} \in \mathbb{R}^{n_u}$ represents the degrees of freedom, which are the gas lift injection rates for each well w_{gl_i}. $\mathbf{d} \in \mathbb{R}^{n_d}$ represents the vector of uncertain variables. In this work, we consider the gas-oil ratio (GOR) from the reservoir for each well to be the disturbance.

The steady-state optimization problem for such a system can be written as

$$\max_{w_{gl_i}} J = \left(\$_o \sum_{i \in \mathcal{N}} w_{po_i} - \$_{gl} \sum_{i \in \mathcal{N}} w_{gl_i} \right) \tag{2.11a}$$

s.t.

$$\mathbf{F}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) = 0 \tag{2.11b}$$

$$\mathbf{G}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) = 0 \tag{2.11c}$$

$$\sum_{i \in \mathcal{N}} w_{pg_i} \leq w_{g_{max}} \tag{2.11d}$$

where $\$_o$ and $\$_{gl}$ prices are the value of oil and cost of gas compression respectively. w_{po_i} and w_{pg_i} are the produced oil and gas rates from each well i. The steady-state process model is enforced as equality constraint in (2.11b)-(2.11c). $w_{g_{max}}$ is the total capacity constraint which is enforced in (2.11d). Hence from a process control point of view, this is equivalent to real-time optimization. For each of the RTO case shown in this chapter (SRTO, DRTO and HRTO), a setpoint tracking NMPC layer was used below, to track the setpoints of the gas lift injection rates provided by the RTO layer above. In this chapter, we use a nonlinear MPC, but similar results would be achievable with a more traditional linear MPC. The sampling time of the setpoint tracking controller was set to 5 min and a prediction horizon of 24 samples. A sufficiently long prediction horizon of 2 hours was chosen to ensure stability [119].

Any produced gas rate that exceeds the maximum capacity of $w_{g_{max}}$ in (2.11d) is flared to avoid pressure build-up in the topside processes. Gas flaring is often very expensive due to environmental costs in the form of carbon tax. To reflect this, the gas capacity constraint (2.11d) was implemented as soft constraints using exact penalty functions and slack variables, where the slack variables are penalized in the cost function [82] as shown below,

$$\begin{aligned} \max_{w_{gl_i}} J' = & \left(\$_o \sum_{i \in \mathcal{N}} w_{po_i} - \$_{gl} \sum_{i \in \mathcal{N}} w_{gl_i} \right) - \$_{fl} \|w_{fl}\| \\ \text{s.t.} & \sum_{i \in \mathcal{N}} w_{pg_i} \leq w_{g_{max}} + w_{fl} \\ & (2.11b) - (2.11c) \end{aligned} \tag{2.12}$$

where the flared gas rate $w_{fl} \geq 0$ is the slack variable and $\$_{fl}$ is the cost associated with gas flaring that is penalized in the cost function. Note that, the exact penalty function for soft constraint would typically not have any physical meaning. However, in this example, the slack variable is the flared gas and the corresponding

Table 2.1: The optimal gas lift injection rates for different GOR combinations used in the simulations.

GOR well 1	GOR well2	$w^*_{gl_1}$	$w^*_{gl_2}$
0.1	0.12	1.606	0.781
0.1033	0.115	1.497	0.956
0.0817	0.1127	2.217	1.042
0.1198	0.1278	0.9437	0.5013
0.0757	0.1197	2.414	0.799
0.0864	0.1176	2.06	1.26
0.1104	0.1176	1.26	0.8626
0.0854	0.1295	2.091	0.4516

penalty function would be the cost of gas flaring. An alternate equivalent formulation would be to compute the flared gas as a part of the system model and minimize the gas flaring in the cost function.

The NLP problem (2.12) was developed in `CasADi` v3.0.1 [4] using the `MATLAB` R2017a programming environment and solved using `IPOPT` version 3.12.2 [184] running with mumps linear solver on a 2.6GHz workstation with 16GB memory. The plant (simulator) was implemented using `IDAS` integrator [68].

In case of Dynamic RTO and for setpoint tracking NMPC, the system (2.9) is discretized using a third order direct collocation scheme. The dynamic optimization problem is similar to (2.11) with the steady-state model (2.11b)-(2.11c) replaced with the discretized dynamic process model. The resulting NLP was solved using `IPOPT` as described above.

In all the simulations shown in this chapter, the parameter GOR varies in the simulator as shown in Fig. 2.4 (solid lines). We simulate the system for a total simulation time of 12 hours. The gas processing capacity is assumed to be constrained at $w_{g_{max}} = 10kg/s$. The optimal steady-state gas lift injection rates for the different GOR combinations simulated in Fig. 2.4 are summarized in Table. 2.1. It is evident that the optimal gas lift injection rates are sensitive to changes in GOR. If the GOR used in the optimizer is not updated, then the plant will be operated suboptimally.

2.4.1 Steady-state RTO (SRTO)

We consider the traditional steady-state RTO approach in Fig. 2.1, where the steady-state RTO provides the optimal setpoints to the lower setpoint tracking layer. In this case, the steady-state detection was based on the comparison of total variance of a signal in the recent data window of fixed length as described in [22]. When steady-state operation is detected, GOR is estimated from the measurement data. Fig. 2.4 compares the true value of the GOR used in the simulator (solid lines) and the estimated GOR value used by the RTO (dashed lines). There is a significant delay before the models are updated and re-optimized due to the steady-state wait-time. This results in suboptimal operation for significant periods

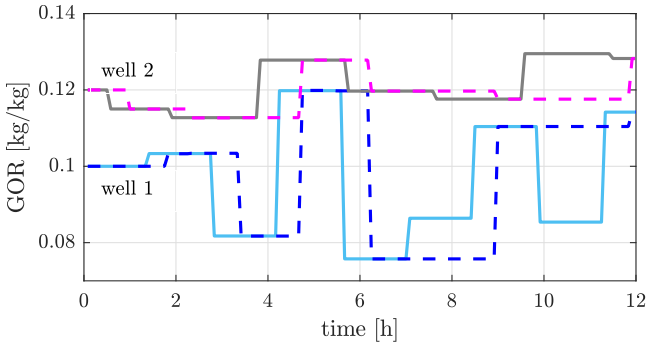


Figure 2.4: True GOR parameters used in the simulator (solid lines) and the estimated GOR using steady-state measurements used in SRTO (dashed lines). The steady-state wait time varies from about 30 minutes to several hours.

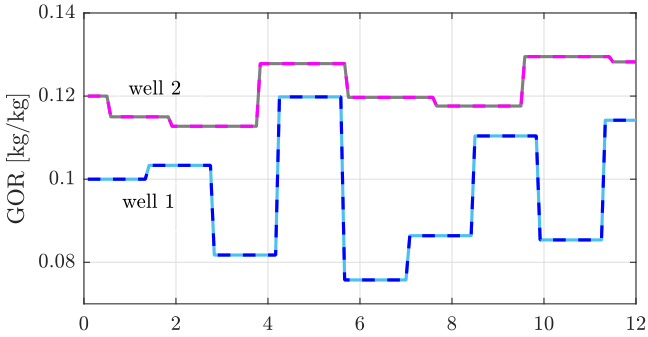


Figure 2.5: True GOR parameters used in the simulator (solid lines) and the almost identical estimated GOR using EKF used in HRTO and DRTO (dashed lines).

of time. In this simulation, the steady-state RTO updated the setpoints 10 times. The performance of the steady-state RTO approach is compared with dynamic RTO and the proposed hybrid RTO approach in Section 2.4.4

2.4.2 Hybrid RTO (HRTO)

With the proposed hybrid RTO as shown in Fig. 2.3, we estimate the uncertain parameters and disturbances using the dynamic model and optimize using the steady-state model. For the dynamic model adaptation, we use a discrete time extended Kalman filter to estimate the uncertain parameters (GOR of each well) as shown in Appendix B. The annulus pressure, wellhead pressure and bottomhole pressure for each well, manifold pressure, riser-head pressure, total oil and gas flow rates at the separator are commonly available measurements in an oil production network, and are here used for state and parameter estimation in the EKF.

The EKF updates the GOR estimate with the same sampling time as the op-

timizer, which is every 5 minutes. The estimated GOR using the EKF is shown in Fig. 2.5. The optimizer uses the updated GOR to compute the steady-state optimal gas lift rates, which are given as setpoints to the lower level setpoint-tracking NMPC layer. Some measurements such as bottom hole pressure and wellhead pressure measurements are plotted in the Fig. 2.6. It can be clearly seen that the system is in a transient phase for significant amount of time due to the frequent changes in GOR. Nevertheless, with dynamic estimation, the GOR is constantly adapted, as opposed to the steady-state wait time in the SRTO case. As a result, the hybrid RTO is able to compute the new optimal steady-state gas lift rates as the GOR changes, without having to wait for the system to settle to steady-state.

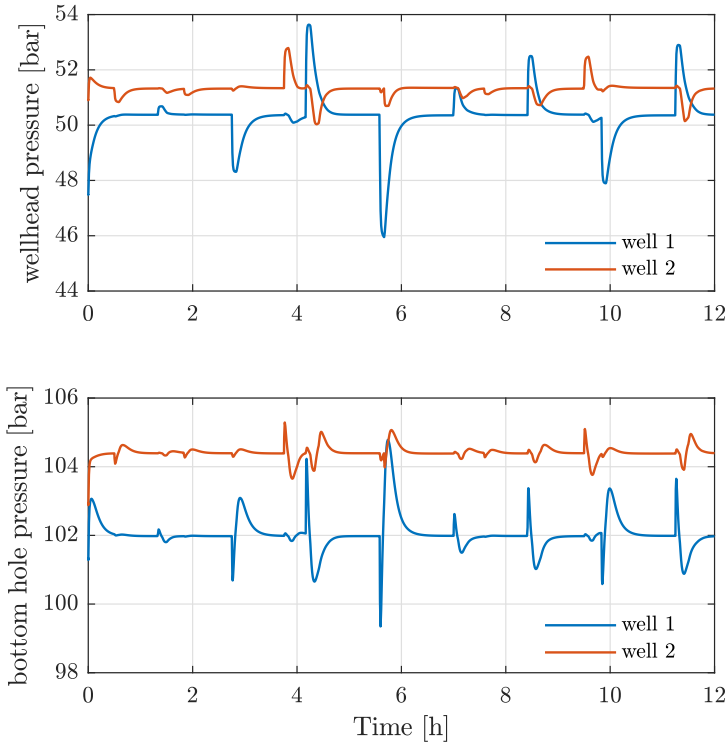


Figure 2.6: Plot showing some of the measurements used in EKF to estimate GOR.

2.4.3 Dynamic RTO (DRTO)

Here, we use a dynamic RTO approach as shown in Fig. 2.2, instead of a steady-state optimizer. But otherwise, the setup was the same as for the hybrid RTO described in Section 2.4.2. The simulation results are compared with SRTO and HRTO in the next subsection.

2.4.4 Comparison of SRTO, HRTO and DRTO

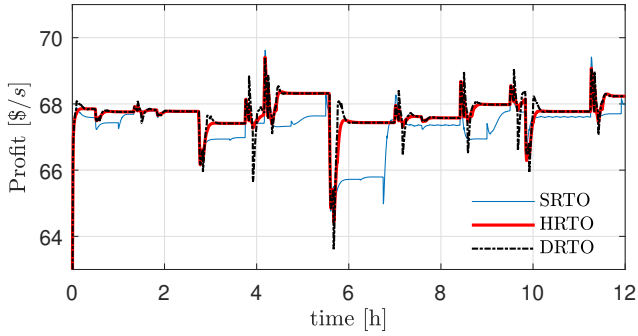
Table 2.2: Comparison of average computation time, maximum computation time and the total integrated oil production over a simulation time of 12 hours for the different RTO approaches.

	avg. time [s]	max time [s]	Integrated Profit [$\times 10^6$\$]	Total oil [ton]	Flared gas [ton]
SRTO	0.0184	0.0223	1.8256	2969.5	10.93
HRTO	0.0199	0.0282	2.7019	2980.2	2.25
DRTO	0.9025	3.3631	2.7509	2980.9	1.77

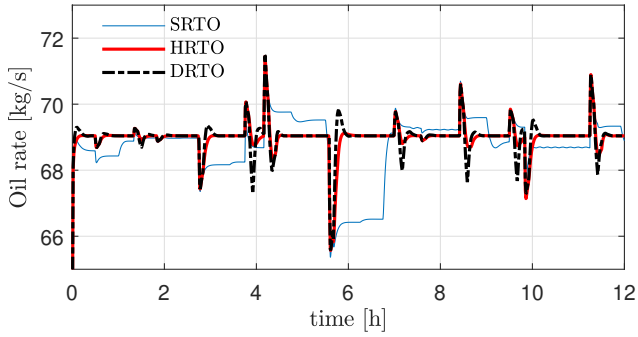
The objective function (2.12) along with the oil and gas production rates obtained with the SRTO, HRTO and DRTO are compared in Fig. 2.7a, Fig. 2.7b and Fig. 2.7c respectively. As expected, SRTO (shown in thin blue lines) leads to suboptimal operation and the total produced gas also violates the capacity constraint of $10kg/s$ in (2.11d) for significant periods of time. The hybrid RTO (red solid lines) and dynamic RTO (black dashed lines) have similar performance in terms of optimality. We see that the process is maintained at optimal operation and, disturbances are swiftly counteracted. The integrated profit (2.12), oil production, and the total gas flared over a period of 12 hours obtained with the three methods are summarized in Table.2.2 and Fig. 2.8.

The average computation times for the steady-state RTO, hybrid RTO and dynamic RTO are also shown in Table 2.2 and Fig. 2.8. From the simulation results and the computation times, it can be seen that the Hybrid RTO provides a similar performance to the dynamic RTO in terms of convergence to the optimal point, but the computation time of the hybrid approach is about two orders of magnitude less than DRTO, and about the same as SRTO. Additionally, for the HRTO and DRTO case, the average computation time for the EKF is 0.0026s, which is small compared to the computation time for the optimization problems.

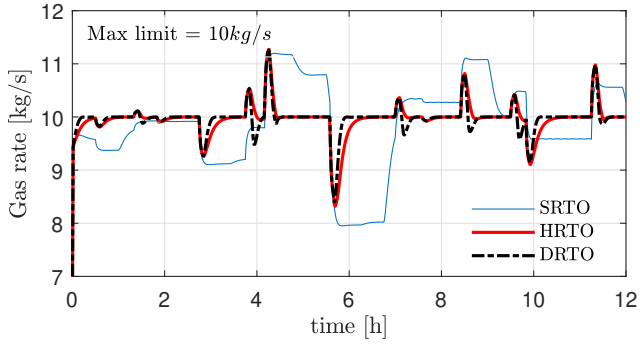
The decision variables (setpoints) provided by the HRTO and DRTO are shown in Fig. 2.9, whereas the gas lift rates actually implemented by the respective setpoint tracking controller are shown in Fig. 2.10. It can be seen that, when the disturbance causes the total gas rate to exceed its limit, the dynamic RTO manipulates the setpoints to quickly come out of constraint violations, whereas the hybrid RTO simply provides the steady-state optimal setpoints. However, since the gas rate constraint of 10kg/s is included in the control layer below as a dynamic constraint, the actual gas lift rates provided by the setpoint tracking controllers are more similar. For example, consider the time between 4 and 5 hours in Fig. 2.9 and Fig. 2.10. This shows that dynamic limitations (cf. Challenge 5 in Section 1.1) in many cases can be handled by the control layer below, which partly explains why the hybrid RTO scheme works well.



(a)



(b)



(c)

Figure 2.7: Comparison of (a) objective function (2.11), (b) oil production rate and (c) gas production rate. SRTO is shown in thin blue lines, HRTO is shown in solid red lines and DRTO is shown in black dashed lines.

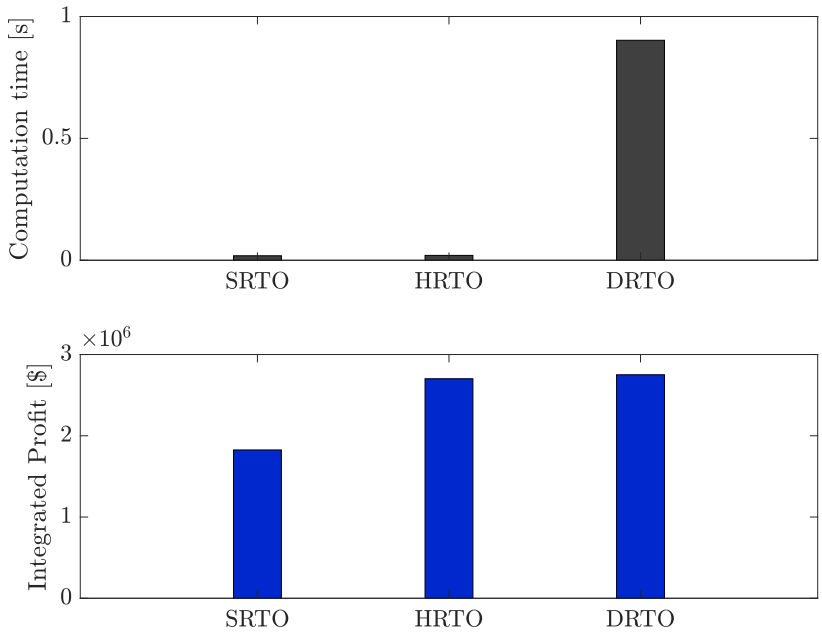


Figure 2.8: Comparison of average computation time and the total integrated oil production the different RTO approaches shows that HRTO provides similar performance as DRTO as computation times similar to SRTO.

2.5 Discussion

2.5.1 On steady-state versus dynamic optimization

In the previous section, we discussed the Hybrid RTO structure, where both dynamic and steady-state models are used. One question that naturally arises is that when dynamic models are used for model adaptation, why not use the dynamic models also in the optimizer. Indeed, there is a clear trend and extensive research towards dynamic RTO and the closely related economic NMPC, see for example [76], [42] and [49]. In the face of this current trend, there is a lack of clear understanding on when steady-state optimization is sufficient or under what conditions the use of dynamic optimization may be justified. Recently, some good discussions using case examples on appropriate problem formulations were provided by Foss et al. [50] for the petroleum production optimization problem, where the authors conclude that most production optimization problems can be solved using steady-state optimization. On the other hand, batch processes, cyclic operations, operations that involves frequent grade changes, start-up and shut-down etc. that involves transient operation would benefit from the use of dynamic optimization.

One of the main challenges with dynamic RTO, however, is computing power

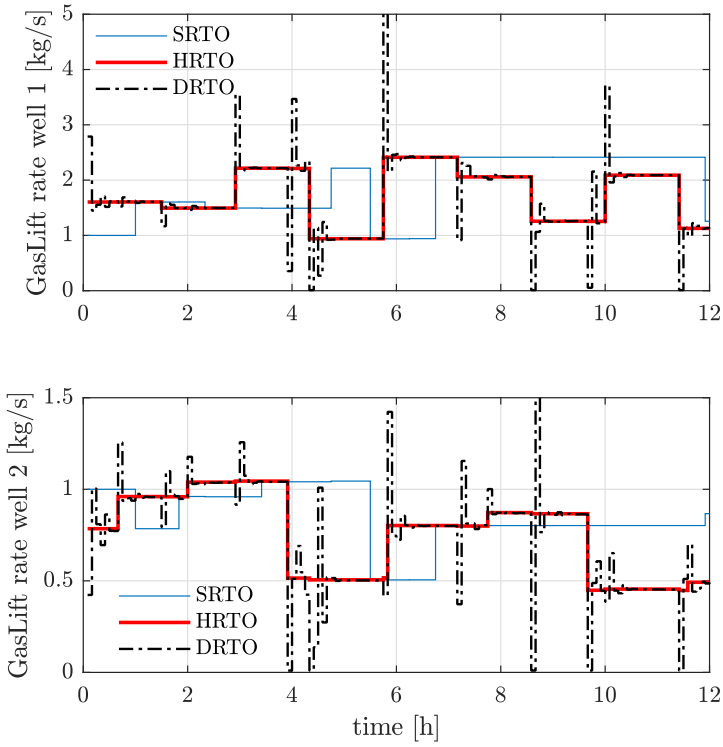


Figure 2.9: Optimal gas lift rate setpoint computed by SRTO (blue lines), HRTO (solid red lines) and DRTO (black dashed lines). These are the manipulated variables (decision variables) computed by the optimizers.

[25]. RTO often involves optimization of large-scale systems with large number of variables. This results in large nonlinear programming (NLP) problems. Additionally in dynamic optimization problems, the size of the problem increases significantly due to the additional dimension of time. As a result, dynamic optimization problems may be significantly more computationally demanding to solve than their steady-state counterpart. For example, in our case study, the dynamic RTO had 3056 optimization variables as opposed to 22 optimization variables in SRTO and HRTO. The computational delay may impose limitations on how often the optimal setpoints can be updated. In some cases, the computational delay may even lead to performance degradation or closed-loop instabilities [45]. This challenge will be considered later in Part II of this thesis, in the context of economic NMPC.

The challenge with computational power is even more pronounced in the case where RTO has discrete integer decision variables. Mixed-integer problems may be required if 1)the problem has discrete integer variables such as on-off switching, binary logic etc. or 2) if the nonlinear process model is modeled using piece-wise-affine (PWA) models or surrogate models or 3) if the problem has non-convex cost

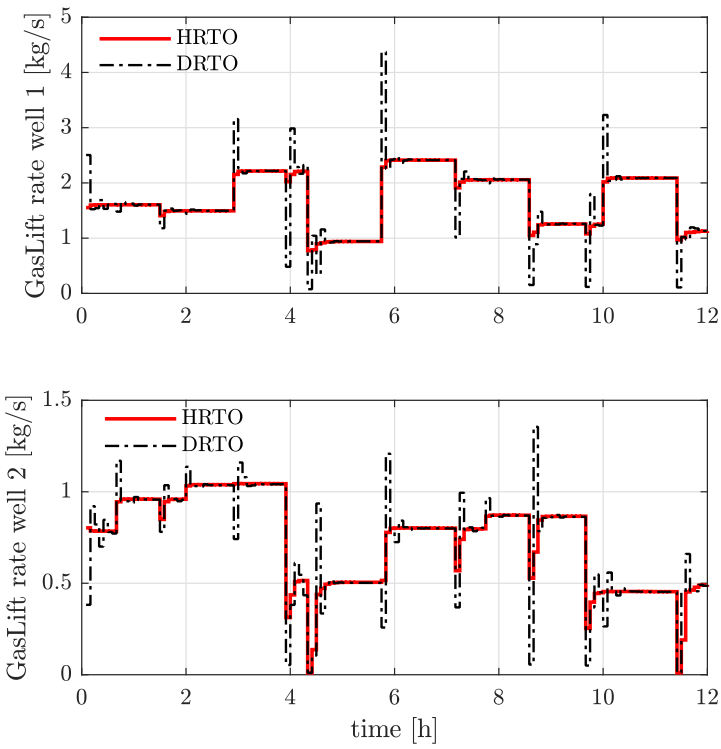


Figure 2.10: Optimal gas lift rates provided by respective setpoint tracking controllers for HRTO (solid red lines) and DRTO (black dashed lines). These are the implemented manipulated variables.

or constraints. Such problem are often reformulated and solved efficiently using the mixed-integer framework. See for example [10], [61], [69], [77] and the references therein [1]. Mixed-integer solvers employ methods such as branch-and-bound and cutting-plane methods combined with some heuristics and significant research in developing mixed-integer solvers directed towards steady-state optimization problems [59, 181]. Hence steady-state RTO remains the preferred formulation in many industrial applications. In such cases, the proposed hybrid RTO approach can help tackle the steady-state wait-time issue, which is one of the fundamental limiting factor in traditional steady-state RTO and at the same time circumvent the computational issues of dynamic RTO.

[1] Mixed-integer formulations are out of scope of this thesis.

2.5.2 Advantages of steady-state optimization (SRTO and HRTO)

However, computation cost is not the only reason steady-state optimization is prevalent in industrial applications. A fundamental advantage of steady-state optimization (in SRTO and HRTO) is that it does not have time as a variable. This avoids the causality issue, and allows for optimizing on decision variables other than the manipulated variables (MVs). For example, these decision variables may be pressure, level, composition and temperature. Consequently, this can 1) simplify the optimization and 2) allow the optimization to run on a much slower time scale because we can choose slow-varying variables as decision variables. This is also the principle behind self-optimizing control [165], where the goal is to choose the right decision variables $\mathbf{y}^{sp}$ in Fig. 2.1.

This advantage is not seen in our example because in all simulation cases, the decision variables are the gas lift injection rates (MVs). However, as a simple example, consider a small tank with one inflow (disturbance) and one outflow (MV). The setpoint for the level is assumed fixed. A dynamic RTO or economic NMPC would have the outflow as the decision variable, and it would need to be updated with the same frequency as the inflow disturbance. Essentially, the DRTO (or economic NMPC) is doing the job of the base layer PID controller, and needs to run at the speed of the base layer control system. However, with a steady-state RTO, the decision variable could be the level setpoint, which would remain constant, irrespective of the disturbance. Of course, optimizing on other decision variables assumes that we have a lower layer level controller, which takes care of disturbances on a fast time scale. In steady-state RTO, the decision variables are setpoints to CVs of the lower-layer control system and the only assumption one needs to make about the control system is that it has integral action.

2.5.3 Dynamic estimation methods

As mentioned earlier, in the hybrid RTO approach, the model adaptation step is performed using dynamic models. Although we used an extended Kalman filter in this chapter, we now provide a very brief discussion on the different dynamic estimation methods that can be used. Very simple methods such as filtered bias update or Implicit Dynamic Feedback (IDF) method may be used for simple parameter estimation problems. Implicit dynamic feedback is analogous to a PI controller as explained by Hedengren and Eaton [66] and may be used when a one-to-one pairing of measurements to the parameter is known.

For more complex multivariable systems, weighted least squares estimation or family of Kalman filters such as the extended Kalman filter (EKF) or unscented Kalman filter (UKF) may be used. Other recently developed Kalman filter based methods include sequential Monte Carlo methods and expectation maximization methods for parameter estimation [177]. It was noted by Leibman et al. [112] that extended Kalman filter is the most widely used tool for nonlinear weighted least square estimation in chemical engineering. In the presence of uncorrelated Gaussian white noise, this also corresponds to the maximum likelihood estimator (MLE) [112]. In terms of computational requirements, EKF for parameter estimation is

known to be simple to implement and computationally fast compared to other methods [177]. The solution provided by EKF is accomplished through matrix multiplications and does not need to solve nonlinear optimization problems online. Additionally, there has also been some research in faster implementation of EKF for some applications, see for example [142], where the authors analyze the EKF matrices and reduce the number of computations by exploiting the sparsity and structure of the EKF matrices. The confidence interval provided by the covariance estimates may also be useful. One of the challenges with practical implementation of Kalman filter is the tuning which include the measurement and process noise covariance matrix elements and a forgetting factor, if included in the formulation [112]. Often these tuning parameters are chosen arbitrarily. Recent works introduced a computationally efficient approach to identify the noise covariance for nonlinear systems [55].

Optimization-based estimation methods such as the moving horizon estimation (MHE) has been receiving more attention recently, where a numerical optimization problem is solved to reduce the error between the estimates and the measurement. MHE is especially favorable in the case of constrained estimation. However, this method is more computationally expensive due to the dynamic optimization problem that has to be solved at each sampling instant. This may not be favorable if one of the motivations of using steady-state optimization is to avoid solving dynamic optimization problems. MHE also often require additional observers in parallel such as EKF for estimating the arrival cost. For more detailed review of dynamic estimation methods, the reader is referred to [66], [112] and the references therein.

2.6 Chapter Summary

In this chapter, we presented a Hybrid RTO framework to address Challenge 2. By using a hybrid RTO with dynamic models for model adaptation, we are able to efficiently use transient data for updating the models. Hence the optimization does not need to wait for the process to settle, before the model parameters are updated. Additionally, by adopting a Hybrid RTO structure, with steady-state models for optimization, numerical and computational issues associated with dynamic optimization can be avoided. This leads to better utilization of the potential of RTO in many industrial applications. The use of hybrid RTO was demonstrated using an oil production network as case study, where similar performance to dynamic RTO was achieved at computation times similar to the traditional steady-state RTO. This method was also applied to a larger case example with $n_w = 6$ wells connected to a common riser manifold (20 differential states, 78 algebraic states and 6 inputs), see [105].

Additionally, we also showed in Fig. 2.10, that by using a setpoint tracking NMPC layer below, one can address the dynamic limitations (Challenge 5).

Chapter 3

A Feedback Real-time Optimization Strategy using a Novel Steady-state Gradient Estimate and Transient Measurements

This chapter presents a new feedback real-time optimization (RTO) strategy for steady-state optimization that directly uses transient measurements. The proposed RTO scheme is based on controlling the estimated steady-state gradient of the cost function using feedback. The steady-state gradient is estimated using a novel method, based on linearizing a nonlinear dynamic model around the current operating point. The gradient is controlled to zero using standard feedback controllers, for example, a PI-controller.

Based on the article published in Industrial and Engineering Chemistry Research [102].

3.1 Introduction

As we have seen in the previous chapters, real-time optimization is traditionally based on rigorous steady-state process models that are used by a numerical optimization solver to compute the optimal inputs and setpoints. The optimization problem needs to be re-solved every time a disturbance occurs. Since steady-state process models are used, it is necessary to wait, so the plant has settled to a new steady-state before updating the model parameters and estimating the disturbances. This steady-state wait-time is one of the fundamental limitations of the traditional RTO approach, and was also the main motivation in Chapter 2.

To address the problem of the steady-state wait-time associated with the traditional steady-state RTO, a hybrid RTO approach was proposed in Chapter 2, where the model adaptation is done using a dynamic model and transient measurements, whereas the optimization is performed using a static model. The hybrid RTO approach, thus requires solving a numerical optimization problem in order to

compute the optimal setpoints. As mentioned in the introduction (cf. Section 1.1), one of the challenges with traditional RTO is the numerical robustness and computational issues associated with solving numerical optimization problems. It also requires regular maintenance of both the dynamic model and its static counterpart.

In order to address the steady-state wait time and the numerical issues, in this chapter, we propose to convert the hybrid RTO strategy from Chapter 2 into a feedback steady-state RTO strategy. This is based on the principle that optimal operation can be achieved by controlling the estimated steady-state gradient from the inputs to the cost at a constant setpoint of zero, thereby achieving the necessary conditions of optimality. The proposed method involves a novel non-obvious method for estimating the steady-state gradient by linearizing a nonlinear dynamic model which is updated using transient measurements. To be more specific, the nonlinear dynamic model is used to estimate the states and parameters by means of a dynamic estimation scheme in the same fashion as in the hybrid and dynamic RTO approaches. However, instead of using the updated model in an optimization problem, the state and the parameter estimates are used to linearize the updated dynamic model from the inputs to the cost. This linearized dynamic model is then used to obtain the mentioned non-obvious estimate of the steady-state gradient at the current operating point (cf. Theorem 3.1). Optimal operation is achieved by controlling the estimated steady-state gradient to constant setpoint of zero by any feedback controller.

The concept of achieving optimal operation by keeping a particular variable at a constant setpoint is also the idea behind self-optimizing control, which is another direct input adaptation method, see [165] and [75]. It was also noted by Skogestad [165] that the ideal self-optimizing variable is the steady-state gradient from the input to the cost, which when kept constant at a setpoint of zero, leads to optimal operation (thereby satisfying the necessary condition for optimality), which complements the idea behind our proposed method.

The concept of estimating and driving the steady-state cost gradients to zero is also the same used in methods such as extremum seeking control [5, 107], necessary-conditions of optimality (NCO) tracking controllers [51], and "hill-climbing" controllers [108]. However, these methods are model-free and hence requires additional perturbations for accurate gradient estimation. The main disadvantages of such methods are that they require the cost to be measured directly, and generally give prohibitively slow convergence to the optimum[1] [180, 182].

The main contribution of this chapter is a novel gradient estimation method (Theorem 3.1), which is used in a feedback-based RTO strategy using transient measurements. The proposed method is demonstrated using a CSTR case study. The proposed method is compared with the traditional static RTO, dynamic RTO and the newer hybrid RTO approach. It is also compared with two direct input adaptation methods, namely self-optimizing control and extremum seeking control.

[1] This issue will be addressed later in Chapter 5.

3.2 Proposed method

In this section, we present the feedback steady-state RTO strategy. Consider a continuous-time nonlinear process,

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{F}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) \\ \mathbf{y}(t) &= \mathbf{H}(\mathbf{x}(t), \mathbf{u}(t)) \end{aligned} \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$, $\mathbf{u} \in \mathbb{R}^{n_u}$ and $\mathbf{y} \in \mathbb{R}^{n_y}$ are the states, process inputs and process measurements respectively. $\mathbf{d} \in \mathbb{R}^{n_d}$ is the set of process disturbances. $\mathbf{F} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \rightarrow \mathbb{R}^{n_x}$ describes the differential equations, and the measurement model is given by $\mathbf{H} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_y}$.
Let the cost that has to be optimized $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}$ be given by,

$$J(t) = \mathbf{G}(\mathbf{x}(t), \mathbf{u}(t)) \tag{3.2}$$

Note that the measurement model and the cost function are not directly affected by the disturbances, but are affected via the states. According to the plantwide control procedure [165], we also assume that any active constraints are tightly regulated, and the n_u degrees of freedom considered here are the remaining unconstrained degrees of freedom available for optimization. A more general treatment with change in active constraint regions will be provided in Chapter 4.

Assumption 3.1: (3.2) is sufficiently continuous and twice differentiable such that for any $\mathbf{d}$, (3.2) has a minimum at $\mathbf{u} = \mathbf{u}^*$. According to the Karush-Kuhn- Tucker (KKT) conditions, the following then holds:

$$\frac{\partial J}{\partial \mathbf{u}}(\mathbf{u}^*, \mathbf{d}) = \mathbf{J_u}(\mathbf{u}^*, \mathbf{d}) = 0 \tag{3.3}$$

$$\frac{\partial^2 J}{\partial \mathbf{u}^2}(\mathbf{u}^*, \mathbf{d}) = \mathbf{J_{uu}}(\mathbf{u}^*, \mathbf{d}) > 0 \tag{3.4}$$

Without loss of generality, we can assume that the optimization problem is a minimization problem.

The optimization problem can be converted to a feedback control problem by controlling the steady-state gradient $\mathbf{J_u}$ to a constant setpoint of $\mathbf{J_u}^{sp} = 0$. The main challenge is then to estimate the steady-state gradient efficiently. There are many different data-based gradient estimation algorithms that estimate the steady-state gradient using steady-state measurements, see [173]. In this chapter, we propose to estimate the steady-state gradient using a nonlinear dynamic model and the process measurements y_{meas} by means of a combined state and parameter estimation framework. In this way, we can estimate the exact gradients around the current operating point.

Any state estimation scheme may be used to estimate the states $\hat{\mathbf{x}}$ and the unmeasured disturbances $\hat{\mathbf{d}}$ using the dynamic model of the plant and the measurements $\mathbf{y}_{meas}$. In this chapter, for the sake of demonstration, we use an augmented extended Kalman filter (EKF) for combined state and parameter estimation, see [164] for detailed description of the extended Kalman filter.

Once the states and unmeasured disturbances are estimated, (3.2) is linearized to obtain a *local linear dynamic model* from the inputs $\mathbf{u}$ to the objective function J. Let $\tilde{\mathbf{x}}(t)$, $\tilde{\mathbf{u}}(t)$ and $\tilde{\mathbf{d}}(t)$ denote the original dynamic trajectory that would result if we keep $\mathbf{u}$ unchanged (i.e. no control). Let $\Delta\mathbf{u}(t)$ represent the additional control input and $\Delta\mathbf{x}(t)$ the resulting change in the states,

$$\begin{aligned}\mathbf{x}(t) &= \tilde{\mathbf{x}}(t) + \Delta\mathbf{x}(t)\\ \mathbf{u}(t) &= \tilde{\mathbf{u}}(t) + \Delta\mathbf{u}(t)\\ \mathbf{d}(t) &= \tilde{\mathbf{d}}(t) + \Delta\mathbf{d}(t)\end{aligned} \tag{3.5}$$

where we assume $\tilde{\mathbf{u}}(t) = \hat{\mathbf{u}}(t_0)$ (constant) and $\tilde{\mathbf{d}}(t) = \hat{\mathbf{d}}(t_0)$ (constant) and t_0 denotes the current time. Note that $\Delta\mathbf{d}(t) = 0$ because the control input does not affect the disturbances. For control purposes, the local linear dynamic model from the inputs to the cost in terms of deviation variables is then be given by,

$$\begin{aligned}\Delta\dot{\mathbf{x}} &= A\Delta\mathbf{x}(t) + B\Delta\mathbf{u}(t)\\ \Delta J(t) &= C\Delta\mathbf{x}(t) + D\Delta\mathbf{u}(t)\end{aligned} \tag{3.6}$$

where $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $C \in \mathbb{R}^{1 \times n_x}$ and $D \in \mathbb{R}^{1 \times n_u}$. The system matrices are evaluated around the current estimates $\hat{\mathbf{x}}$ and $\hat{\mathbf{d}}$,

$$\begin{aligned}A &= \left.\frac{\partial \mathbf{F}(\mathbf{x},\mathbf{u},\mathbf{d})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(t_0),\mathbf{d}=\hat{\mathbf{d}}(t_0)}\\ B &= \left.\frac{\partial \mathbf{F}(\mathbf{x},\mathbf{u},\mathbf{d})}{\partial \mathbf{u}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(t_0),\mathbf{d}=\hat{\mathbf{d}}(t_0)}\\ C &= \left.\frac{\partial \mathbf{G}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(t_0),\mathbf{d}=\hat{\mathbf{d}}(t_0)}\\ D &= \left.\frac{\partial \mathbf{G}(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(t_0),\mathbf{d}=\hat{\mathbf{d}}(t_0)}\end{aligned}$$

Note that, since we do not assume full state feedback, we need some nonlinear observer to estimate the states $\hat{\mathbf{x}}$ in order to evaluate the aforementioned Jacobians. Nonlinear observers may not be required if we have full state feedback information to compute the Jacobians, but this is seldom the case.

Theorem 3.1. *Given a nonlinear dynamic system (3.1) and (3.2) and assumption 3.1 holds, the model from the decision variables $\mathbf{u}$ to the cost J can be linearized around the current operating point using any nonlinear observer to get (3.6) and assuming that A is invertible, the corresponding steady-state gradient is then*

$$\hat{\mathbf{J}}_{\mathbf{u}} = -CA^{-1}B + D \tag{3.7}$$

The process can be driven to its optimum by controlling the estimated steady-state gradient to constant setpoint of zero using any feedback control law $\mathbf{u} = \mathcal{K}(\hat{\mathbf{J}}_{\mathbf{u}})$.

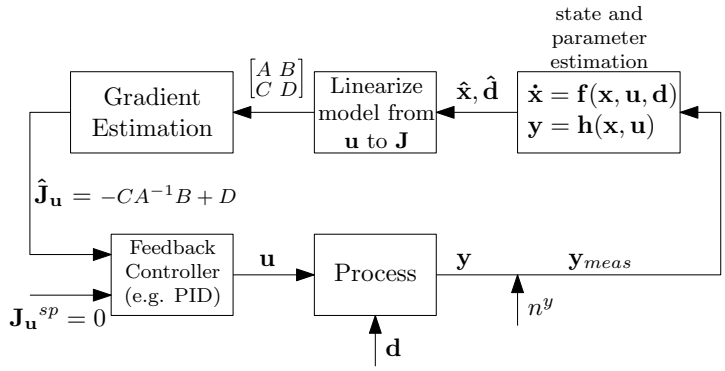


Figure 3.1: Block diagram of the proposed method

Proof. In (3.6), $\Delta\mathbf{x}(t)$, $\Delta\mathbf{u}(t)$ and $\Delta J(t)$ are deviation variables. Let $\Delta\mathbf{u}(t) = \delta\mathbf{u}$ be a small step change in the input occurring at $t = 0$, which will result in a steady-state change for the system as $t \to \infty$. This will occur when $\Delta\dot{\mathbf{x}} = 0$ and by eliminating $\Delta\mathbf{x}(t)$ it follows from (3.6) that the steady-state change in the cost is

$$\delta J = \lim_{t\to\infty} \Delta J(t) = \left(-CA^{-1}B + D\right)\delta\mathbf{u} \tag{3.8}$$

Here, the steady-state gradient is defined as $\mathbf{J_u} = \frac{\delta J}{\delta \mathbf{u}}$ and (3.7) follows. Driving the estimated steady-state gradients to a constant setpoint of zero ensures satisfying the necessary condition of optimality. □

The proposed method is schematically represented in Fig. 3.1. It can be seen that steady-state gradient is obtained from the dynamic model and not from the steady-state model as would be the conventional approach. With a dynamic model we are able to use the transient measurements to estimate the steady-state gradient.

The combined state and parameter estimation framework using extended Kalman filter is discussed in detail in [164]. Note that, although we use an extended Kalman filter to demonstrate the proposed method in the example, any observer may be used to estimate the states and the parameters. Using the estimated states, the dynamic model may be linearized and the steady-state gradient estimated using equations (3.6) - (3.7), which is the key point in the proposed method.

Note that we have used the dynamic model from $\mathbf{u}$ to J. To tune the controller, another dynamic model from the inputs $\mathbf{u}$ to the gradient $\hat{\mathbf{J}}_\mathbf{u}$ is required. The steady-state gain of this model is the Hessian $\mathbf{J_{uu}}$, which is constant if the optimal surface is quadratic.

Remark 3.1. For a multi-input system, it is natural to pair u_i with J_{u_i} (diagonal pairing). In this case, for the PID controllers to be stable, the determinant of the Hessian $\det(\mathbf{J_{uu}})$ must not change sign [168].

More discussions on controller tuning are provided in Section 3.4.4.

3.3 Illustrative example

In this section, we test the proposed method using a continuous stirred tank reactor (CSTR) process from [41] (Fig. 3.2). This case study has been widely used in academic research [1, 72, 187]. The proposed method is compared to traditional steady-state RTO, hybrid RTO and dynamic RTO. It also benchmark against two existing direct-adaptation-based method, namely self-optimizing control and extremum seeking control.

3.3.1 Exothermic Reactor

The CSTR case study consists of a reversible exothermic reaction where component A is converted to component B ($A \rightleftharpoons B$). The reaction rate is given as $r = k_1 C_A - k_2 C_B$, where the reaction constants follow the Arrhenius law: $k_1 = C_1 e^{\frac{-E_1}{RT}}$ and $k_2 = C_2 e^{\frac{-E_2}{RT}}$. The dynamic model consists of mass balances for components A and B and an energy balance:

$$\frac{dC_A}{dt} = \frac{1}{\tau}(C_{A,i} - C_A) - r \tag{3.9a}$$

$$\frac{dC_B}{dt} = \frac{1}{\tau}(C_{B,i} - C_B) + r \tag{3.9b}$$

$$\frac{dT}{dt} = \frac{1}{\tau}(T_i - T) + \frac{-\Delta H_{rx}}{\rho C_p} r \tag{3.9c}$$

Here, C_A and C_B are concentrations of the two components in the reactor whereas $C_{A,i}$ and $C_{B,i}$ are the concentration in the inflow. T_i is the inlet temperature and T is the reaction temperature. Other model parameters for the process are given in Table 3.1.

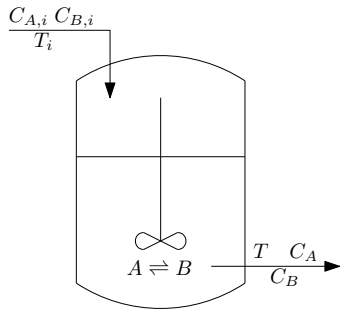


Figure 3.2: Process flowsheet of the CSTR.

The cost function to be minimize is defined as [1]

$$J = -[2.009 C_B - (1.657 \times 10^{-3} T_i)^2]. \tag{3.10}$$

The manipulated variable is $u = T_i$, the temperature in the inlet stream. The state variables are the concentrations and reactor temperature $x^\mathsf{T} = [C_A \ C_B \ T]$, the

Table 3.1: Nominal values for CSTR process

	Description	Value	Unit
F^*	Feed rate	1	$mol\,min^{-1}$
C_1	Constant	5000	s^{-1}
C_2	Constant	10^6	s^{-1}
C_p	Heat capacity	1000	$cal\,kg^{-1}\,K^{-1}$
E_1	Activation energy	10^4	$cal\,mol^{-1}$
E_2	Activation energy	15000	$cal\,mol^{-1}$
$C^*_{A,i}$	Inlet A concentration	1	$mol\,L^{-1}$
$C^*_{B,i}$	Inlet B concentration	0	$mol\,L^{-1}$
R	Universal Gas Constant	1.987	$cal\,mol^{-1}\,K^{-1}$
ΔH_{rx}	Heat of reaction	-5000	$cal\,mol^{-1}$
ρ	Density	1	$kg\,L^{-1}$
τ	Time constant	1	min

disturbances are assumed to be the feed concentrations $d^\mathsf{T} = [C_{A,i}\ C_{B,i}]$, and the available measurements are $y^\mathsf{T} = [C_A\ C_B\ T\ T_i]$.

Feedback steady-state RTO

The proposed feedback RTO strategy described in Section 3.2 is now implemented for the CSTR case study. For the state estimation, we use an extended Kalman filter as described by Simon [164]. The disturbances $\mathbf{d}^\mathsf{T} = [C_{A,i}\ C_{B,i}]$ are assumed to be unmeasured, and are estimated together with the states in the extended Kalman filter. A simple PI controller is used to control the estimated steady-state gradient to a constant setpoint of $\mathbf{J_u}^{sp} = 0$. The PI controller gains are tuned using SIMC tuning rules [166] with the proportional gain $K_p = 4317.6$ and integral time $\tau_I = 60$s. The process is simulated with a total simulation time of 2400s with disturbances in $C_{A,i}$ from 1 $molL^{-1}$ to 2 $molL^{-1}$ at time t = 400s and $C_{B,i}$ from 0 $molL^{-1}$ to 2 $molL^{-1}$ at time t = 1409s. The measurements are assumed to be available with a sampling rate of 1s.

Optimization-based approaches

In this subsection, the simulation results of the proposed method are compared with other optimization-based approaches, namely traditional static RTO (SRTO), dynamic RTO (DRTO) and hybrid RTO (HRTO) for the same disturbances as mentioned above. The traditional static RTO, dynamic RTO and the hybrid RTO structures were used to compute the optimal input temperature. Note that, in practice, this could correspond to a setpoint under the assumption of tight control at the lower regulatory control level.

Traditional static RTO (SRTO) In this approach, before we can estimate the disturbances and update the model, we need to ensure that the system is operating in steady-state. This is done using a steady-state detection (SSD) algorithm that is commonly used in industrial RTO system [22]. The resulting steady-state wait time is a fundamental limitation in many processes, where the plant may be operated

suboptimally for significant periods of time before the model can be updated and the new optimal operation re-computed.

Hybrid RTO (HRTO) As mentioned earlier, in order to address the steady-state wait time issue of traditional RTO approach, a hybrid RTO approach was proposed in Chapter 2, where a dynamic nonlinear model is used online to estimate the parameters and disturbances. The updated static model is then used by a static optimizer to compute the optimal inlet temperature as shown in Fig. 2.3. In this case study, we use the same extended Kalman filter as the one used in the proposed feedback RTO method for the dynamic model adaptation. We then compare the performance of the proposed feedback RTO to the hybrid RTO approach. These two approaches only differ in the fact that in hybrid RTO, a static optimization problem is solved to compute the optimal inlet temperature, whereas in the proposed method optimization is done via feedback.

Dynamic RTO (DRTO) Recently there has been a surge of research activity towards dynamic optimization and centralized integrated optimization and control such as economic nonlinear model predictive control (ENMPC), which is also closely related to dynamic RTO. Since the proposed method uses a nonlinear dynamic model online, a natural question that may arise is *why not use the same dynamic models also for optimization*. For the sake of completeness, we therefore compare the performance of the proposed method with dynamic RTO.

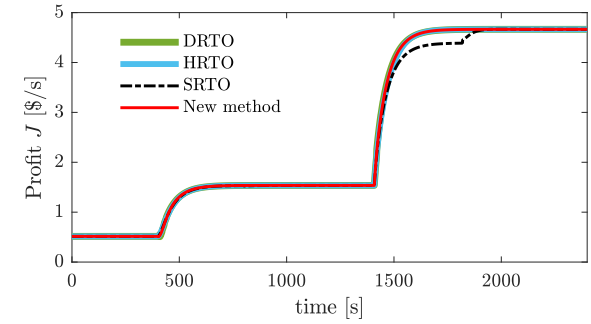
For the dynamic RTO, the same extended Kalman filter as in the proposed feedback RTO method and hybrid RTO is used to update the dynamic model online. The updated nonlinear dynamic model is then used in the dynamic optimization problem with a prediction horizon of 20min and a sampling time of 10s.

Comparison of RTO methods

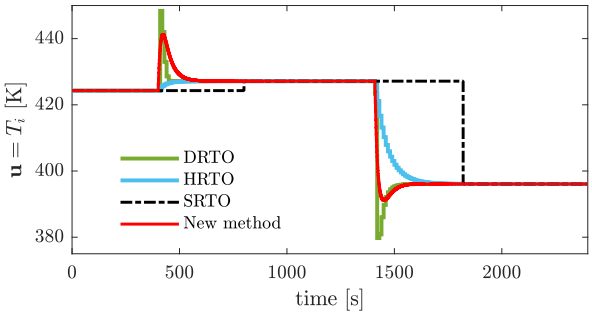
The cost J and the optimal control input $\mathbf{u}$ provided by the proposed feedback RTO method, static RTO, hybrid RTO and dynamic RTO are shown in Fig. 3.3a and Fig. 3.3b, respectively.

It can be clearly seen that for the static RTO (black dash-dotted lines), the steady-state wait time delays the model adaptation. Hence, the system operates suboptimally for significant time periods. Once the process reaches steady-state and the model is updated, we see that the steady-state RTO brings the system to the ideal optimal operation. For example, in this simulation case, it takes around 400s after each disturbances for the SRTO to update the optimal operating point. The change in the cost observed during the transients, before the new optimum point is re-computed, is due to the natural drift in the system. This is more clearly seen after the second disturbance at time $t = 1409$s.

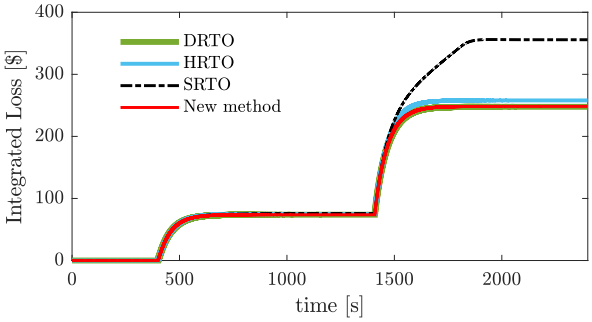
Hybrid RTO (cyan solid lines) and dynamic RTO (green solid lines) provide similar performance as the new proposed feedback RTO strategy (red solid lines), due to the fact that all these three approaches use transient measurements and a nonlinear dynamic model online. These three methods however differ in the way the optimization is performed. As mentioned earlier, dynamic RTO solves a dynamic optimization problem using the updated nonlinear dynamic model, hybrid RTO



(a)



(b)



(c)

Figure 3.3: Simulation results of the proposed feedback RTO method (red solid lines) compared to traditional static RTO (black dash-dotted lines), hybrid RTO (cyan solid lines) and dynamic RTO (green solid lines) for disturbance in $C_{A,i}$ at time t = 400s and $C_{B,i}$ at time t = 1409s. (a) Plot comparing the cost function. (b) Plot comparing the input usage. (c) Plot comparing the integrated loss (3.11).

solves a static optimization problem using the updated static counterpart of the model, whereas feedback RTO estimates the steady-state gradient by linearizing the nonlinear dynamic model and controls the estimated steady-state gradients to a constant setpoint of zero.

The integrated loss is given by

$$L_{int}(t) = \int_0^t (J_{opt,SS}(t) - J(t))\,\mathrm{d}t. \tag{3.11}$$

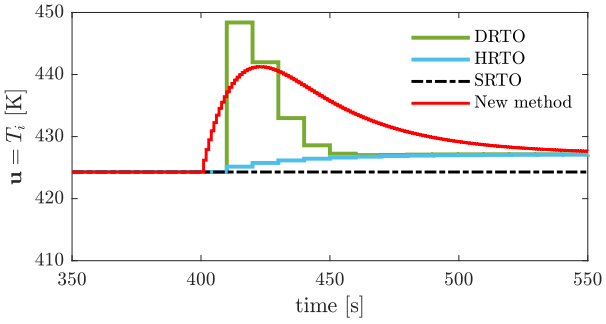
To compare the different approaches further, the integrated loss for the different RTO approaches are shown in Fig. 3.3c and also noted in Table. 3.2 for $t = 2400$s.

We note here that until time $t = 1409$s, the new feedback RTO method has the lowest loss of 73.73\$, closely followed by DRTO and hybrid RTO with a loss of 73.81\$ and 74.77\$, respectively. Following the second disturbance, the integrated loss for the interval $t = 1409$s to $t = 2400$s is the lowest for the DRTO with 247.69\$. The new feedback RTO has a very similar loss of 248.07\$ followed by hybrid RTO with an integrated loss of 257.97\$. The static RTO is much worse with a loss of 355.78\$. This is mainly because of the fact that in the new feedback RTO approach, optimization is done via feedback, and hence can be implemented at higher sampling rate. The static, dynamic and hybrid RTO approaches requires additional computation time to solve the optimization problem online, and hence may be implemented at slower sampling rates. As mentioned earlier, in our case study, the feedback RTO approach is implemented every 1s as opposed to static, dynamic and hybrid RTO approaches which are solved every 10s. This is clearly shown in Fig. 3.4a, where the control input from Fig. 3.3b is magnified between time 350s to 550s. Following the disturbance at $t = 400$s, the static, dynamic and hybrid RTO updates the setpoint at time step $t = 410$s unlike the feedback RTO, which updates the new control input already at time $t = 401$s giving it an advantage over other RTO methods. As a result, the new feedback RTO method has the lowest integrated loss up to time $t = 1409$s. The control input between time 1300s to 1500s is shown in Fig. 3.4b, where following the disturbance at $t = 1409$s, new proposed feedback RTO, the static, dynamic and hybrid RTO, all update the optimal control input at time step $t = 1410$s simultaneously. Therefore, the DRTO has the best integrated loss as expected. This example clearly shows the importance of being able to implement a controller at higher sampling rates.

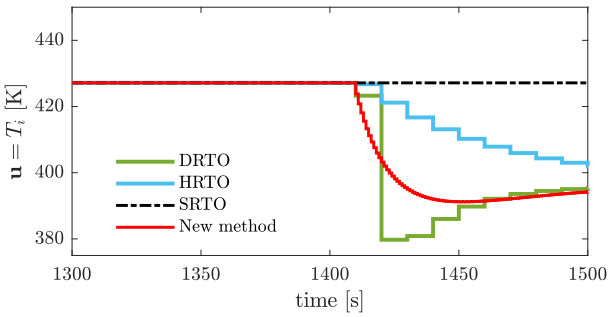
The simulations are performed on a workstation with the Intel Core i7-6600U CPU (dual-core with the base clock of 2.60GHz and turbo-boost of 3.40GHz), and 16GB memory. The average computation times for the different RTO approaches are also compared in Table 3.2. The proposed feedback RTO method is the least computationally expensive method due to the fact the optimization is done via feedback, and as expected, dynamic RTO is the most computationally intensive.

Comparison with direct-input adaptation methods

For the sake of completeness, we also compare the proposed method with two feedback-based direct-input adaptation methods, namely, self-optimizing control and extremum seeking control, where the optimization problem is converted into a feedback control problem,.



(a)

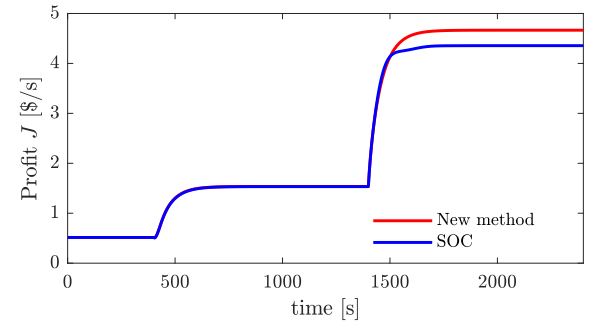


(b)

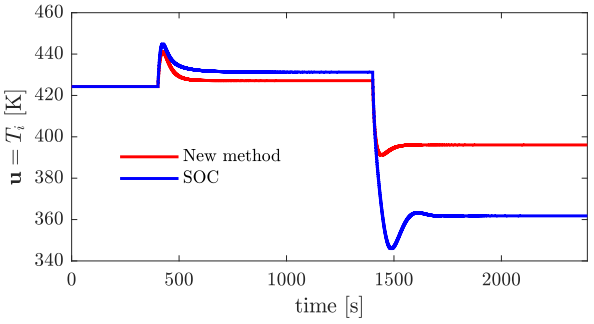
Figure 3.4: Magnified plot of the control input from Fig. 3.3b between time (a) 350s to 550s and (b) 1300s to 1500s.

Table 3.2: Average computation time and integrated loss at the end of simulation time for the proposed method compared with traditional static RTO, hybrid RTO and DRTO.

	Computation time [s]	Integrated loss $t = 1409$s [$]	Integrated loss $t = 2400$s [$]
New method	0.004	73.73	248.07
SRTO	0.007	75.75	355.78
HRTO	0.01	74.77	257.97
DRTO	0.14	73.81	247.69



(a)



(b)

Figure 3.5: Simulation results of the proposed feedback RTO method (red solid lines) compared to self-optimizing control (blue solid lines) for disturbance in $C_{A,i}$ at time t = 400s and $C_{B,i}$ at time t = 1409s. (a) Plot comparing the cost function. (b) Plot comparing the input usage.

Self-optimizing control (SOC) Self-optimizing control, is about finding the right controlled variable, which when controlled to a constant setpoint, leads to acceptable loss [165]. In general, the idea is to find a linear measurement combination that can be used as a self-optimizing variable [2]. Self-optimizing control will be formally introduced later in Chapter 6. At this point, the reader is simply referred to [2, 75, 165].

Since we have 3 measurements, 2 disturbances and 1 control input, the nullspace method can be used to identify the self-optimizing variable. For the case study considered here, the optimal selection matrix computed using nullspace method (around the nominal optimal point when $d^{\mathsf{T}} = [C_{A,i}\ C_{B,i}] = [1.0\ 0.0]$) is given by $\mathbf{H} = \begin{bmatrix} -0.7688 & 0.6394 & 0.0046 \end{bmatrix}$, see [1]. The resulting self-optimizing variable $\mathbf{c} = -0.7688C_A + 0.6394C_B + 0.0046T$ is controlled to a constant setpoint of $\mathbf{c}_s = 1.9012$. The PI controller used in the self-optimizing control structure were tuned using SIMC rules with proportional gain $K_p = 188.65$ and integral time $\tau_I = 75$s.

The simulations were performed with the same disturbances as in the previous case. The objective function for the two methods are shown in Fig. 3.5a and the

corresponding control input usage is shown in Fig. 3.5b. When compared to self-optimizing control, we can see that there is an optimality gap when the disturbances occur. This is because self-optimizing control is based on linearization around the nominal optimal point, as opposed to linearization around the current operating point in the proposed feedback RTO approach. Because of the nonlinear nature of the processes, the economic performance degrades for operating points far from the nominal optimal point hence leading to steady-state losses.

Extremum seeking control (ESC) The concept of estimating and driving the steady-state gradient to zero in the proposed feedback RTO strategy is also used in data-driven methods such as extremum seeking control [107] and NCO-tracking [51]. However, the gradient estimation scheme is fundamentally different, i.e. the steady-state gradient in the proposed approach is estimated using a dynamic nonlinear model, whereas the steady-state gradient in extremum seeking control and NCO tracking are estimated directly from the cost measurement. For the sake of brevity, we restrict our comparison to extremum seeking control. Extremum seeking control will be studied in more detail in Chapters 5 and 6. At this point, the reader is simply referred to [70, 107, 180].

In this subsection, we consider the least-square based extremum seeking control proposed by Hunnekens et al. [70] because it has been shown to provide better performance than classical extremum seeking control [32, 70]. The least square based extremum seeking controller also estimates the gradient rather than just the sign of the gradient [31]. The least-square based extremum seeking controller estimates the steady-state gradient using the measured cost and input data with the moving window of fixed length in the past. The gradient estimated by the least square method is then driven to zero using a simple integral action. The integral gain was chosen to be $K_{ESC} = 2$.

Due to the slow convergence of the extremum seeking controller, the process with simulated with a total simulation time of 36000s (600min) with disturbances in $C_{A,i}$ from 1 $molL^{-1}$ to 2 $molL^{-1}$ at time t = 10800s and $C_{B,i}$ from 0 $molL^{-1}$ to 2 $molL^{-1}$ at time t = 21600s. The results using extremum seeking control are compared with those of the proposed method in Fig. 3.6a. It can be seen that the extremum seeking controller reaches the optimal point, however, the convergence to the optimum point is very slow compared to the proposed method. The proposed method has a fast action to the disturbances, and hence reaches the optimum significantly faster than the extremum seeking controller. The integrated loss compared to the ideal steady-state optimum (3.11) shown in Fig. 3.6c reflects this.

It should be added this is a simulation example, because strictly speaking it may not be possible to directly measure an economic cost J with several terms. The simple cost in (3.10) may be computed by measuring individually the composition C_B and the inlet temperature T_i, but more generally for process systems, direct measurement of all the terms and adding them together is not accurate. This is discussed further in the discussions section.

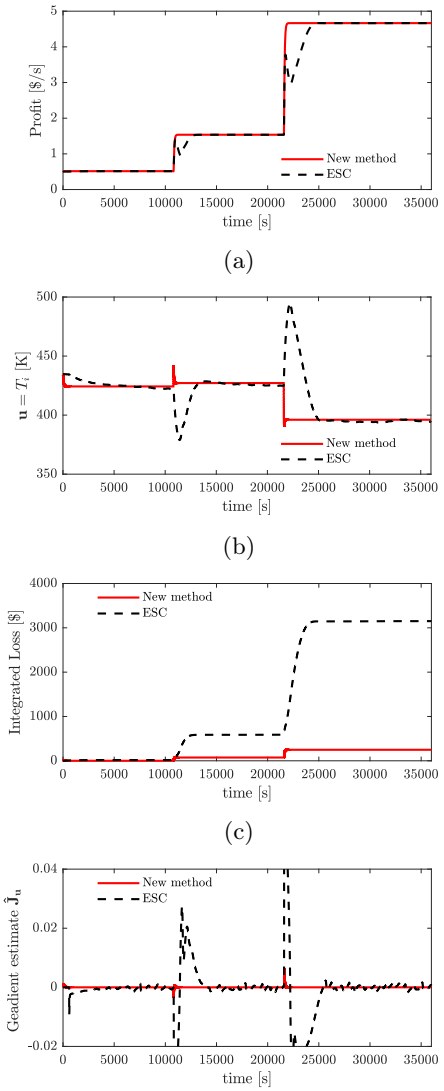


Figure 3.6: Comparison of the proposed feedback RTO method (red solid lines) with extremum seeking control (black dashed lines) for disturbance in $C_{A,i}$ at time t = 10800s and $C_{B,i}$ at time t = 21600s. (a) Plot comparing the cost function. (b) Plot comparing the input usage. (c) Plot comparing the integrated loss. (d) Plot comparing the estimated gradients.

3.4 Discussion

3.4.1 Comparison with optimization-based approaches

With the traditional steady-state RTO approach, it was seen clearly that the steady-state wait time resulted in suboptimal operation, clearly motivating the need for alternative RTO strategies that can use transient measurements. Dynamic RTO framework that use transient measurements, provide the most optimal solution, but it comes at the cost of solving computationally expensive optimization problems as noted in Table 3.2. This is even worse for large-scale systems where the sample time will be restricted by the computational time. Indeed, the computational delay has also been shown to result in performance degradation or even instability [45].

The feedback RTO strategy proposed in this chapter is closely related to the hybrid RTO scheme proposed in Chapter 2 and has similar performance (see Fig. 3.3a). The main difference is that in the new strategy, the steady-state gradient is obtained as part of the solution to the estimation problem, and the optimization is then solved by feedback control rather than numerically solving a steady-state optimization problem. Thus, we avoid maintaining the steady-state models in addition to the dynamic model.(i.e. avoids duplication of models in addition to the numerical optimization).

3.4.2 Comparison with self-optimizing control

Self-optimizing control is based on linearization around the nominal optimal point [2]. The economic performance degrades for operating points far from the nominal optimal point due to the nonlinear nature of the process. This is the reason for sustained steady-state loss of self-optimizing control seen in the simulation results. The proposed method however, is based on linearization around the current operating point, and hence does not lead to steady-state losses. The price for this performance improvement is the use of the model online instead of offline. In other words, the proposed method requires computational power for the nonlinear observers, which are not required in the standard self-optimizing control. However, nonlinear observers such as extended Kalman filters can be used, as demonstrated in the simulations, which are known to be simple to implement and computationally fast [177]. The EKF used in the simulations had an average computation time of 0.0036s.

3.4.3 Comparison with extremum seeking control

Extremum seeking control estimates the steady-state gradient by fitting a local linear static model using the cost measurements [107]. Therefore, transient measurements cannot be used for the gradient estimation. On the other hand, since our proposed method linearizes the nonlinear dynamic system to get a local linear *dynamic* model, it does not require a timescale separation for the gradient estimation. Hence the convergence to the optimum is significantly faster compared to the extremum seeking control, as demonstrated in our simulation results.

In practice, extremum seeking methods may also not be completely model-free, and may then suffer from structural errors, although it will be different from when using model-based optimization. The reason is that a direct measurement of the cost J is often not possible, especially if J is an economic cost with many terms, and it may then be necessary to use model-based methods to estimate one or more terms in the cost J. Typically the cost function for a process plant is of the form,

$$J = c_q Q + c_f F - c_{p1} P_1 - c_{p2} P_2 \tag{3.12}$$

where Q, F, P_1 and $P2$ are flows in [kg/s] of utility, feed and products 1 and 2 respectively, and c_q, c_f, c_{p1}, c_{p2} are the corresponding prices in [$/kg]. Extremum seeking control requires all the flow terms in (3.12) to be accurately measured. In practice, even if one of the flow terms is not accurately measured (e.g. to due to sensor failure), then the best way to get accurate flows is to estimate them using a nonlinear process model, e.g., using data reconciliation. This means that for optimization of larger process systems, a extremum seeking or NCO tracking controller will not be truly model-free, because a model is needed to get an acceptable measurement (estimate) of the cost.

However, it is also important to note that extremum seeking (and NCO-tracking) approaches can handle structural uncertainty. The method proposed in this chapter, as any other model-based method, works well only when the model is structurally correct. We do note that, in the presence of plant-model mismatch, the proposed method may lead to an optimality gap, leading to some steady-state loss, unlike the model-free approaches, which would perform better. Therefore, extremum seeking or NCO-tracking methods (including modifier adaptation) methods should be considered to handle structural mismatch.

3.4.4 Tuning

As mentioned earlier, the steady-state gradient is controlled to a constant setpoint of zero using feedback controllers. The controller tuning is briefly discussed in this section. For the CSTR case study, PI controllers were used. The PI controllers were tuned using the SIMC PID tuning rules [166]. For each input, let the process model from the corresponding gradient $\mathbf{y} = \mathbf{J_u}$ to the input $\mathbf{u}$ be approximated by a first order process. For a scalar case,

$$\mathbf{J_u} = \frac{\mathbf{k}}{(\tau_1 s + 1)} e^{-\theta s} \mathbf{u} \tag{3.13}$$

where, τ_1 is the dominant time constant, θ is the effective time delay and $\mathbf{k}$ is the is the steady-state gain. These three parameters can be found experimentally or from the dynamic model [166]. Note that the process dynamics include both the effect of the process itself and the estimator, see Figure 1. In our case we found experimentally by simulations that $\mathbf{k} = 2.25 \times 10^{-4}$, $\tau_1 = 60$s, $\theta = 1$s. The time delay for the process is very small, and mainly caused by the sampling time of 1 s.

In general, the steady-state gain is equal to the Hessian, $\mathbf{K} = \mathbf{J_{uu}}$, which according to Assumption 3.1 should not change sign. The Hessian $\mathbf{J_{uu}}$ was computed for the CSTR case study and it was verified that this assumptions holds. In particular, the value of $\mathbf{K} = \mathbf{J_{uu}}$ for the three steady-states shown in Fig. 3.3 were

$\mathbf{J_{uu}} = 2.25 \times 10^{-4}$ (nominal), $\mathbf{J_{uu}} = 3.89 \times 10^{-4}$ and $\mathbf{J_{uu}} = 6.33 \times 10^{-4}$, respectively. The gain increases by a factor of 3, which may lead to instability, but if robust PID tuning is used, tuning the controller at the nominal point should be sufficient. For a PI-controller,

$$c(s) = K_C + \frac{K_I}{s} \tag{3.14}$$

the SIMC-rules give the proportional and integral gain [166]

$$K_C = \frac{1}{\mathbf{k}} \frac{\tau_1}{\tau_c + \theta} \quad K_I = \frac{K_C}{\tau_I} \tag{3.15}$$

where the integral time is $\tau_I = \min(\tau_1, 4(\tau_c + \theta))$ and τ_c is the desired closed-loop time response, which is the sole tuning parameter [166].

It is generally recommended to select $\tau_c \geq \theta$ [166] to avoid instability and a larger τ_c gives less aggressive control and more robustness. In our case, the controlled variable is $\mathbf{J_u}$ (the gradient), but there is little need to control $\mathbf{J_u}$ tightly because it is not an important variable in itself. Therefore, to get better robustness we recommend selecting a larger value for τ_c (assuming that $\tau_1 > \theta$, which is usually the case):

$$\tau_c \geq \tau_1 \tag{3.16}$$

Selecting $\tau_c > \tau_1$ means that the closed-loop response is slower than the open-loop response. This avoids excessive use of the input $\mathbf{u}$ and the system is more robust with respect to gain variations. This is confirmed by the simulations in Fig. 3.7b for three different choices of τ_c. With $\tau_c = 10\text{s} \ll \tau_1 = 60\text{s}$, we get aggressive input changes with large overshoots in $\mathbf{u} = T_{in}$ for both disturbances. The control of gradient is good (Fig. 3.7c), but this in itself is not important, and the improvement in profit J is fairly small compared to the choice $\tau_c = \tau_1 = 60\text{s}$, which is the nominal value used previously. The integrated loss when $\tau_c = 10\text{s}$ was 245.99$ as opposed to 248.07$ when $\tau_c = \tau_1 = 60\text{s}$. With $\tau_c = 4\tau_1$ the input change is even smoother, but the performance in terms of the profit (J) is almost the same (with an integrated loss of 259.07$).

3.4.5 Stability Robustness

The stability of the system is determined by the closed-loop stability involving the plant input $\mathbf{u}$ (manipulated variable) and the estimated gradient $\mathbf{J_u}$ as the plant output $\mathbf{y}$. Thus, the overall plant for this analysis includes the real process, the estimator (extended Kalman Filter in our case) and the "measurement" $\mathbf{y} = \mathbf{J_u} = CA^{-1}B + D$. In this case we use a simple PI-controller and the conventional linear measures for analyzing robustness are to compute the gain margin, phase margin or delay margin, or $\mathcal{H}_\infty$ measures such as the peak of the sensitivity function, M_S.

The gain and phase margins at the three different steady-states are shown for the three alternative PI-controllers in Table 3.3. Note that the gain margin varies from 17.3-6.14 for the least robust controller with $\tau_c = 10\text{s}$, which is still much larger than the expected variations in the process gain (which is about a factor 3 from 2.25×10^{-4} to 6.33×10^{-4}). This is of course a nonlinear plant and the

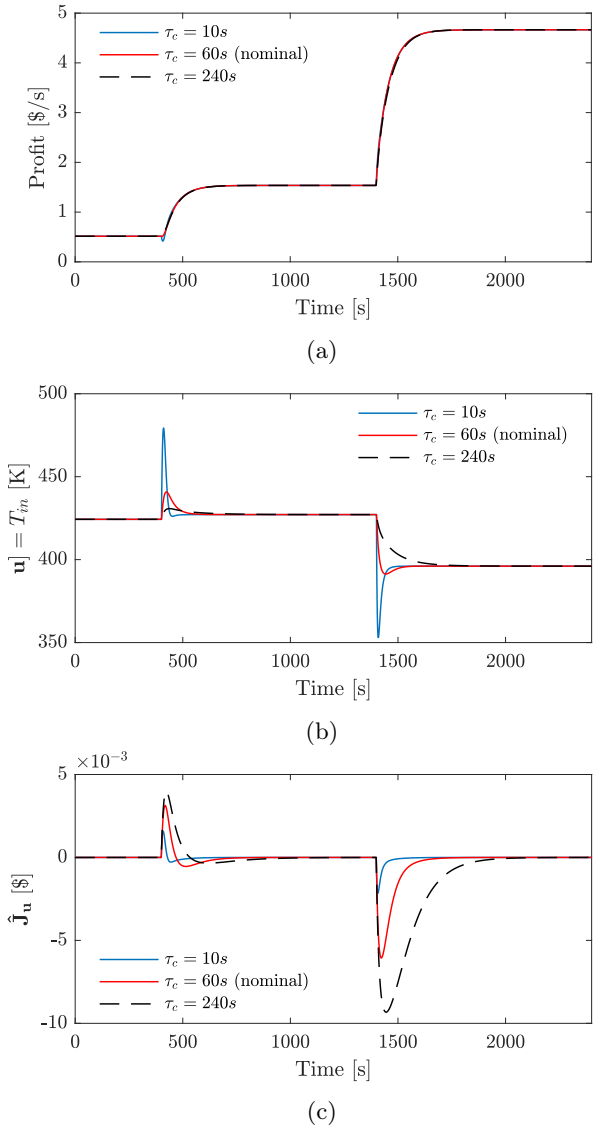


Figure 3.7: Effect of controller tuning (τ_c) for the new method. (a) Objective function J. (b) Control input $\mathbf{u}$. (c) Estimated gradient $\hat{\mathbf{J}}_\mathbf{u}$.

Table 3.3: Gain and Phase Margins at the different steady-states for the three PI controllers with varying τ_c.

		Gain Margin	Phase Margin
Steady-state 1	τ_c=10s	17.3	84.8°
	τ_c=60s	95.8	89.1°
	τ_c=240s	379	89.8°
Steady-state 2	τ_c=10s	9.99	81.0°
	τ_c=60s	55.4	88.4°
	τ_c=240s	219	89.6°
Steady-state 3	τ_c=10s	6.14	75.3°
	τ_c=60s	34.1	87.4°
	τ_c=240s	135	89.3°

conventional linear analysis of closed-loop stability will have the same limitations as it has for any nonlinear plant. It is also possible to consider nonlinear controllers for the plant, but it does not seem necessary because of the large robustness margins for the linear controllers.

In summary, stability is not an important concern for the tuning of the controllers in our case. The important issue is the trade-off between input usage and the speed of response. This was also observed in the other case studies [18, 96].

3.4.6 Other multivariable case studies

In addition to the CSTR case study presented here, the new proposed feedback RTO method has been successfully applied to the following processes:

1. a 3-bed ammonia reactor with 3 inputs [18]
2. an oil and gas production optimization problem with n_w inputs - [96] with $n_w = 2$ wells and in [100] with $n_w = 6$ wells
3. an evaporator process with 1 input in [103]
4. an isothermal CSTR process with 4 components and 2 inputs in [89]
5. a benchmark William-Otto reactor with 6 components and 2 inputs, shown in Appendix D

These application examples are appended to this thesis and can be found in Appendix C. In all these case studies, the new feedback RTO method was compared with other optimization methods and was shown to provide consistent results as in the CSTR case study shown in this chapter. It is worth noting that, the ammonia reactor process studied in [18] and the gas-lift optimization problem studied in [96, 100] are examples of coupled multivariable process.

3.5 Chapter Summary

To conclude, in this chapter, we proposed a novel model-based direct input adaptation approach, where the steady-state gradient $\mathbf{J_u}$ is estimated as $\hat{\mathbf{J}}_\mathbf{u} = -CA^{-1}B+$

D by linearizing the nonlinear model around the current operating point. The steady-state gradient is thus estimated using transient measurements as shown in Fig. 3.1.

Chapter 4

Online Process Optimization with Active Constraint Set Changes using Simple Control Structures

The objective of this chapter is to show that online process optimization can be achieved using simple control structures. In particular, we show that changes in active constraint regions can be handled using simple logics such as selectors, without needing to identify the exact location of the active constraint regions a priori, nor using a detailed model online.

Based on the articles published in Industrial and Engineering Chemistry Research [89] and Control Engineering practice [100].

4.1 Introduction

In Chapter 2 and 3, hybrid RTO and feedback RTO approaches were presented, where we used a nonlinear dynamic model and a dynamic state and parameter estimator to optimize the process. The model adaptation step mostly works under the assumption that any plant-model mismatch observed in the measurements arises from the set of uncertain parameters $\mathbf{d}$. Although this may reduce the error between the model predictions and the plant observations, this may not result in optimal operation if the model has the wrong structure. Updating the model parameters may not be sufficient to alleviate the effects of model structural mismatch. For example, Roberts and Williams [147], Marchetti et al. [120] and several others show that it is difficult to achieve the model-adequacy conditions[48] using the two-step model adaptation approach.

The RTO layer generally has as degrees of freedom, the setpoints for the controlled variables (CV^{sp}), which is given to the control layer as illustrated in Fig. 4.1. The control layer has degrees of freedom $\mathbf{u}$, which are the physical manipulated variables (MV), and in addition to achieving feasible operation, its main objective is to keep the outputs $\mathbf{y}$ or controlled variables (CV) at the optimal values computed by the RTO layer. The main purpose of this chapter is to study how we can eliminate the RTO layer, even for the case when the set of constraints that are

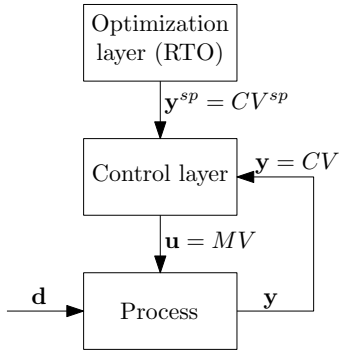


Figure 4.1: Typical hierarchical decomposition into optimization and control layer.

active changes with changing operating conditions. In other words, the objective is to indirectly move the optimization into the control layer.

The idea of achieving optimal operation using feedback control structure dates back to the 80s, where Morari et al. [130] attempted to synthesize a *feedback optimizing control* structure by translating the economic objectives into process control objectives. However, this idea of "feedback optimizing control" from Morari et al. [130] received very little attention, until Skogestad [165] presented the self-optimizing control structure, where the objective is to find a simple feedback control strategy with near-optimal cost, subject to constraints using feedback controllers. Here, Skogestad [165] advocates that, it is important to tightly control the constraints that are at its limit at the optimum. In this case, there is no loss by keeping the constraint at its limit (active constraint control). Next, if there are remaining unconstrained degrees of freedom, one should identify suitable controlled variables which translates to the economic objectives. In other words, one should identify the so-called "self-optimizing" control variables. The idea is that, controlling the self-optimizing variables at constant setpoints give an acceptable loss compared to the true optimum, when disturbances occur. Offline optimization is typically needed to find good self-optimizing variables. The ideal self-optimizing variable is the gradient of the cost function, $\mathbf{J_u}$, which should then be controlled to a constant setpoint of zero [62], thereby satisfying the *necessary conditions of optimality*.

To summarize, if the optimization problem at hand is an unconstrained problem, then to drive the process to its optimum, we need good models. However, if the problem is constrained such that the optimal operation happens when the constraints are active, then the simplest and most effective approach is to control the active constraints tightly using feedback controllers (active constraint control), without the need for a model.

In many cases, the active constraint set changes as a function of disturbances. For example, when a disturbance changes, some of the active constraints may no longer be active and some other constraints may become active. When using simple feedback controllers, a different active constraint set requires reconfiguration of the control loops and also possible identify different self-optimizing variables for the new operating conditions. This is schematically shown in Fig. 4.2, where the

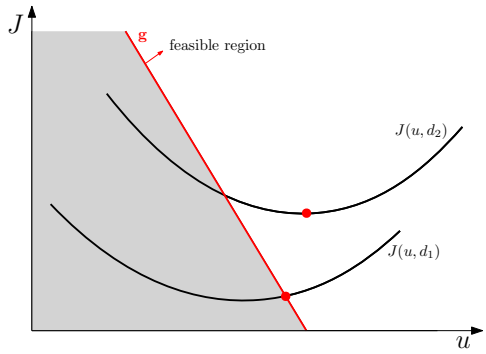


Figure 4.2: Schematic representation showing the change in the active constraint set for two different disturbances, where for disturbance d_1 the optimum occurs at the constraint, whereas, for disturbance d_2, the optimum is unconstrained.

cost function is shown for two different disturbance values d_1 and d_2. The infeasible region is shown in gray shaded area. For disturbance d_1, the optimum occurs when the constraint (shown in red line) is active, whereas, when the disturbance changes to d_2, the constraint is no longer active and a new self-optimizing variable is required, since the problem is an unconstrained optimization problem. This requires the need to change the controlled variables. This chapter focuses on the use of simple control structures to achieve optimal operation, even in the case where the active constraint set changes. Using different example cases, we will show that, often simple control logics are sufficient to handle changes in the active constraint sets without needing to use a model online.

When we have one manipulated variable (MV) controlling two controlled variable (CV), i.e. CV-CV switching, then minimum/maximum selectors can be used. Alternatively, when we have more than one candidate MV to control one controlled variable (CV), then other advanced control structures such as split range, or input resetting can be used. Split range control may also be used when MV-CV pairings need to be changed, e.g. when a MV saturates. However, this chapter focuses on CV-CV switching and the reader is referred to Appendix F and [146] for more detailed description on MV-MV switching.

The main contribution of this chapter is to show that, for many simple processes and unit operations, online *steady-state* process optimization with changes in active CV constraint regions can indeed be achieved by using simple feedback control structures, without having a separate online optimization layer. Some well known case studies are presented that demonstrate the effectiveness of the proposed control structures and how changes in the active constraint regions can be handled using selector logic.

Remark 4.1. It is important to note that the objective of this chapter is not to convince that simple PID control structures are superior to advanced control and model-based optimization routines such as model predictive control, but rather to demonstrate that systematic control structure design using classical feedback controllers may be useful in some processes to achieve optimal operation.

One main advantage of using simple feedback controllers is that it addresses the challenges related to human aspects, since this is by far, the most trusted and used technology in practice [152, 153]. However, for many large-scale multivariable processes with complex interconnections and constraints, this approach can quickly become messy with several SISO loops and logic blocks.

4.2 Control structure design for online process optimization

Consider a steady-state optimization problem,

$$\begin{aligned} \min_{\mathbf{u}} \quad & J(\mathbf{u}, \mathbf{d}) \\ \text{s.t.} \quad & \\ & \mathbf{g}(\mathbf{u}, \mathbf{d}) \leq 0 \end{aligned} \tag{4.1}$$

where $\mathbf{u} \in \mathbb{R}^{n_u}$ denotes the vector of manipulated variables (MV) and $\mathbf{d} \in \mathbb{R}^{n_d}$ denotes the vector of disturbances, $J : \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}$ is the scalar cost function and $\mathbf{g} : \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_g}$ denotes the vector of constraints and let $n_a \leq n_g$ denote the number of active constraints $\mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})$ at the optimum for a given disturbance $\mathbf{d}$. The Lagrangian of (4.1) is written as,

$$\mathcal{L}(\mathbf{u}, \mathbf{d}) = J(\mathbf{u}, \mathbf{d}) + \lambda^{\mathsf{T}} \mathbf{g}(\mathbf{u}, \mathbf{d}) \tag{4.2}$$

where $\lambda \in \mathbb{R}^{n_g}$ is the vector of Lagrangian multipliers for the constraints. The Karush-Kuhn-Tucker (KKT) optimality conditions for (4.1) state that the necessary condition of optimality is when,

$$\nabla_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mathbf{d}) = \nabla_{\mathbf{u}} J(\mathbf{u}, \mathbf{d}) + \lambda^{\mathsf{T}} \nabla_{\mathbf{u}} \mathbf{g}(\mathbf{u}, \mathbf{d}) = 0 \tag{4.3a}$$

$$\mathbf{g}(\mathbf{u}, \mathbf{d}) \leq 0 \tag{4.3b}$$

$$\lambda^{\mathsf{T}} \mathbf{g}(\mathbf{u}, \mathbf{d}) = 0 \tag{4.3c}$$

$$\lambda \geq 0 \tag{4.3d}$$

where (4.3b) is the primal feasibility, (4.3d) is the dual feasibility and (4.3c) is the complementary slackness condition.

Typically, the optimal solution for (4.1) is computed by solving the KKT conditions (4.3), either analytically or using numerical methods. However, the objective in this section is to translate the KKT conditions into control objectives, and thereby move the optimization layer into the control layer.

In order to achieve this, the first question one needs to answer is what are the potential combination of active constraints that may be encountered during operation. For a process with n_g constraints, we have a maximum of 2^{n_g} possible combinations of operating regions [165]. To be systematic, we can start our by writing all the possible constraint combinations. But in practice there are often only a few of these combinations that are applicable for a given set of disturbances. Often with good process understanding and "engineering intuition", one can tell a-priori which active constraint combinations that may be encountered. Hence we eliminate the constraint combinations that are not feasible or not likely.

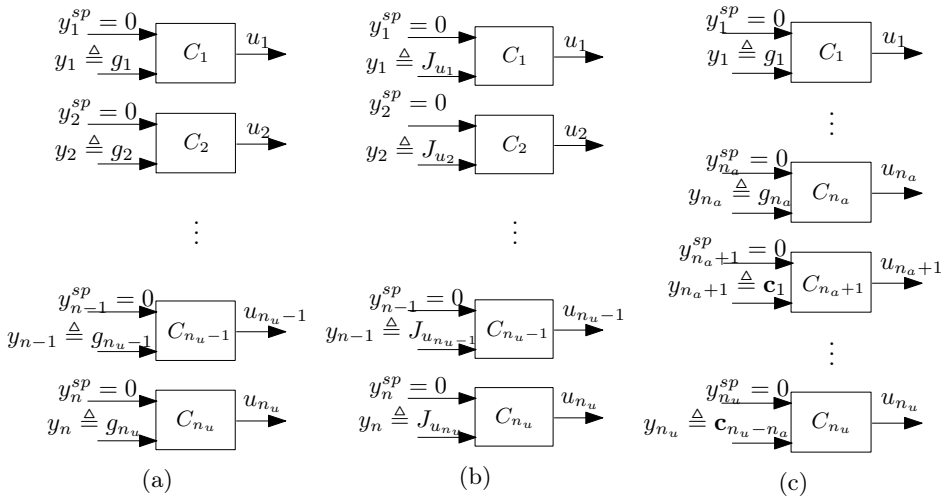


Figure 4.3: Control structure design for (a) Fully constrained case ($n_a = n_u$), (b) Fully unconstrained case ($n_a = 0$) and (c) Partially constrained case ($0 < n_a < n_u$)

By "active constraints", we mean the set of constraints $\mathbf{g}_{\mathbb{A}} \subset \mathbf{g}$, which optimally should be at the limiting value, i.e., $\mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d}) = 0$. For each active constraint, we need to find an associated controlled variable, usually simply the constraint itself, i.e. CV $= \mathbf{g}_{\mathbb{A}}$. In some cases, the constraint is on a manipulated variable (MV); in this case one can simply select CV = MV, so actually no controller is needed to control an MV-constraint. Disturbances may cause the active constraints to change, meaning that we get different operating regions, as motivated in Fig. 4.2.

A more rigourous alternative is to identify using offline optimization (requires good models), the possible active constraint regions for known disturbances. Once the relevant combination of active constraints are identified, control structures can be designed to handle the different combinations of active constraints.

4.2.1 Selection of controlled variables

Selecting the right controlled variable is important to achieve optimal operation using simple feedback control structures. Depending on the number of active constraints n_a, relative to the number of manipulated variables (MV) n_u, we have four different cases.

Case 1: Fully constrained case ($n_a = n_u$) When the number of active constraints at the optimum is equal to the number of MVs (i.e. $n_a = n_u$), then the simplest and easiest approach to achieve optimal operation is to select CV $= \mathbf{g}_{\mathbb{A}}$, and simply maintain the active constraints at their limits using measurement feedback (active constraint control) [118, 165]. Therefore, in this case, we have n_u feedback controllers that controls the $n_u = n_a$ active constraints $\mathbf{y} = \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d}) = 0$ as shown in Fig. 4.3a.

If a constraint is on the MV, then we do not need any controller at all for this constraint, since the MV can simply be held constant at its limit. More generally, for other constraints, we need to control an associated CV, e.g. CV = $\mathbf{g}_{\mathbb{A}}$ (active constraint control). Active constraint control approach is an old idea and has been used in many examples [6, 47, 71, 118, 130, 145]. In fact, the example used by Morari et al. [130] in one of the earliest works on feedback optimizing control happened to result in an implementation with controlling the constraints at its limit. Note that in the case of fully constrained optimum, we can achieve optimal operation without using a model for optimization, since the process constraints are usually measured and can be maintained at its limit using feedback.

Case 2: Fully unconstrained case ($n_a = 0$) When there are no active constraints at the optimum (i.e. $n_a = 0$), we have n_u unconstrained MVs which can be used to drive the process to its optimum. The optimization problem (4.1) reduces to

$$\min_{\mathbf{u}} \quad J(\mathbf{u}, \mathbf{d}) \tag{4.4}$$

since $\mathbf{g}_{\mathbb{A}} = \emptyset$ and the necessary condition of optimality (4.3) is given by,

$$\nabla_{\mathbf{u}} J(\mathbf{u}, \mathbf{d}) = 0 \tag{4.5}$$

Optimal operation can then be achieved by driving the steady-state gradient $\nabla_{\mathbf{u}} J$ of the cost J from the inputs $\mathbf{u}$ to a constant setpoint of zero by n_u feedback controllers, thereby fulfilling the necessary conditions of optimality. Therefore. we have n_u feedback controllers, each controlling the steady-state gradient to a constant setpoint of 0, i.e. the controlled variables (CV) $y = \nabla_{\mathbf{u}} J$ are controlled to a constant setpoint of $y_{sp} = 0$ as shown in Fig. 4.3b.

There are different ways that can be used to estimate the steady-state gradient $\nabla_{\mathbf{u}} J$. One approach is to use a dynamic model of the process along with the measurements as described in Chapter 3. Alternatively, the plant gradients can be estimated directly from the cost measurements (if available) as done in NCO-tracking control[51] and several variants of extremum seeking control[36, 70, 107]. The reader is referred to [173] for a comprehensive review of several model-based and model-free gradient estimation techniques that can be employed.

Case 3: Partially constrained case ($0 < n_a < n_u$) When the number of active constraints at the optimum is less than the number of MVs, we have a partially constrained case. In this case, we first use the n_a available MVs to control the active constraints. For the remaining $(n_u - n_a)$ uncontrolled MVs, we ideally want to control the steady-state cost gradient $\mathbf{J_u}$ to a constant setpoint such that the KKT conditions in (4.3) are satisfied.

As mentioned in case 2, in the fully unconstrained case, the steady-state gradients are controlled to a constant setpoint of zero in order to fulfill the necessary condition of optimality as shown in (4.1). However, in the partially constrained case, the steady-state gradients $\nabla_{\mathbf{u}} J(\mathbf{u}, \mathbf{d})$ may have to be controlled to a non-zero value in order to satisfy the KKT conditions. To further explain this, the

Lagrangian function (4.2) of the optimization problem for this particular case reduces down to

$$\mathcal{L}(\mathbf{u}, \mathbf{d}) = J(\mathbf{u}, \mathbf{d}) + \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})^{\mathsf{T}} \lambda \tag{4.6}$$

where, $\lambda \in \mathbb{R}^{n_a}$ represents the Lagrange multipliers of the n_a active constraints $\mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})$. Note that the Lagrange multipliers corresponding to the active constraints $\mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})$ are non-zero due to the complementary slackness condition (4.3c), i.e. $\lambda > 0$. The necessary condition of optimality is then given by,

$$\nabla_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mathbf{d}) = \nabla_{\mathbf{u}} J(\mathbf{u}, \mathbf{d}) + \nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})^{\mathsf{T}} \lambda = 0 \tag{4.7}$$

$$\nabla_{\mathbf{u}} J(\mathbf{u}, \mathbf{d}) = -\nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})^{\mathsf{T}} \lambda \tag{4.8}$$

Therefore, the steady-state gradients must be controlled to a constant value equivalent to $-\nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})^{\mathsf{T}} \lambda$ in order to fulfill the necessary conditions of optimality. However, the expression in (4.8) cannot be used directly for feedback control, because it still contains the Lagrange multiplier, which is an unknown variable. In order to achieve necessary condition of optimality using feedback controllers, we eliminate the Lagrange multiplier by looking into the nullspace $\mathbf{N}$ of the active constraint gradients $\nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})$.

A shown by Jäschke and Skogestad [73], $\mathbf{N}$ is defined as the nullspace of $\nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})$ if $\mathbf{N}^{\mathsf{T}} \nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d}) = 0$. Since the number of active constraints $n_a < n_u$, Linear inequality constraint qualification (LICQ) is satisfied and therefore, $\nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})$ has full row rank. Consequently, the nullspace $\mathbf{N}$ is well defined. We then have the following theorem:

Theorem 4.1 (Linear combination of gradients as self-optimizing variables)**.** *Given a steady-state optimization problem (4.1) with n_u manipulated variables $\mathbf{u}$, n_d independent disturbances $\mathbf{d}$ and n_a active constraints $\mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})$. Let $\mathbf{N} \in \mathbb{R}^{n_u \times (n_u - n_a)}$ be the nullspace of the active constraint gradients $\nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})$, such that $\mathbf{N}^{\mathsf{T}} \nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d}) = 0$. Then the necessary conditions of optimality can be achieved by controlling the linear combination of the gradients*

$$\mathbf{c} = \mathbf{N}^{\mathsf{T}} \nabla_{\mathbf{u}} J(\mathbf{u}, \mathbf{d}) \tag{4.9}$$

to a constant setpoint of zero.

Proof. To eliminate the Lagrange multiplier, we pre-multiply (4.8) by $\mathbf{N}^{\mathsf{T}}$ to get

$$\mathbf{N}^{\mathsf{T}} \nabla_{\mathbf{u}} J(\mathbf{u}, \mathbf{d}) = -\mathbf{N}^{\mathsf{T}} \nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})^{\mathsf{T}} \lambda \tag{4.10}$$

Since $\mathbf{N}$ is in the nullspace of $\nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})$, $\mathbf{N}^{\mathsf{T}} \nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}}(\mathbf{u}, \mathbf{d})^{\mathsf{T}} = 0$.

$$\mathbf{N}^{\mathsf{T}} \nabla_{\mathbf{u}} J(\mathbf{u}, \mathbf{d}) = -0\lambda \tag{4.11}$$

By construction, $\mathbf{c}$ is a vector with $(n_u - n_a)$ elements. Hence, to achieve optimal operation, we can then control the $(n_u - n_a)$ elements of the gradient combinations $\mathbf{c}$ to a constant setpoint of zero using $(n_u - n_a)$ feedback controllers. □

To summarize, in the case with $0 < n_a < n_u$, we use n_a MVs to control the active constraints, and for the remaining $(n_u - n_a)$ MVs, we control the linear gradient combination $\mathbf{c} = \mathbf{N}^\mathsf{T}\nabla_\mathbf{u} J(\mathbf{u}, \mathbf{d})$ to a constant setpoint of zero as shown in Fig. 4.3c.

This is similar to null-space method proposed by Alstad and Skogestad [2], but instead of choosing a linear combination of measurements $\mathbf{c} = \mathbf{H}\mathbf{y}$, we propose to choose a linear combination of the cost gradients as the self-optimizing variables. Using a linear combination of gradients as proposed here, leads to zero steady-state loss, since Theorem 4.1 is based on the KKT conditions. Whereas, the linear measurement combination proposed by Alstad and Skogestad [2], only leads to acceptable loss (non-zero steady-state loss). Moreover, the linear combination of the gradients is always controlled to a constant setpoint of zero, hence eliminating the need to adjust the setpoint online, unlike the linear measurement combination.

It is important to note that, since this self-optimizing variable is computed based on $\mathbf{g}_\mathbb{A}(\mathbf{u}, \mathbf{d}) = 0$, it is only valid in this particular active constraint region. Therefore, the control loops tracking the gradient combinations $\mathbf{c} = \mathbf{N}^\mathsf{T}\nabla_\mathbf{u} J(\mathbf{u}, \mathbf{d})$ using the unconstrained MVs must not be used in the regions other than where it is designed for and the neighboring regions (A neighboring region is an operating region where only one controlled variable (constraint) has changed). Therefore, it is not necessary (and in fact it will be incorrect) to track the unconstrained CVs in all other regions. It is only necessary (and correct) to track the variable $\mathbf{c}$ in the regions where they are active and its neighboring regions.

Case 4: Over-constrained case ($n_a > n_u$) If the number of active constraints becomes larger than the number of MVs, then the problem becomes infeasible, since we do not have sufficient MVs to control all the active constraints. In this case, the only possible solution would be to give up controlling less important constraints. This is analogous to using soft constraints on less important constraints to avoid infeasibility issues in numerical optimization problems. In this case, a priority list can be used to determine the less important constraints and only the first n_u important constraints are controlled at their limit using feedback controllers. The remaining $(n_a - n_u)$ constraints are given up. The control structure in this case would remain the same as shown in Fig. 4.3a. The reader is referred to [146] for a typical priority list that is commonly used. Note that MV hard constraints, for example due to physical limitations of the actuator cannot be given up and are therefore typically high up in the priority list.

4.2.2 Use of selector logic to switch between active constraint regions

So far, we considered the choice of controlled variables for four different cases. In practice, depending on the operating point (disturbances), the active constraint set may change within one of the cases, or from one case to another. One way to handle this is to use simple controller logics.

Selectors are commonly used as logic elements when one MV u is used to control several controlled variables (CV) y, i.e to handle CV-CV switching. In this

approach, there is a controller for each CV and the MV-value is selected among the controller outputs using a minimum or maximum selector, depending on the process gains. The following theorem summarizes the systematic design of selectors for CV-CV switching.

Theorem 4.2. *(CV-CV switching using selectors) Consider a process with one MV and*

- *at most 1 CV equality constraint that can be given up (setpoint control), denoted by* y_0
- *any number of CV inequality constraints that may be optimally active, denoted by* y_i, $i = \{1, \ldots, n\}$

For each output y_i *design a SISO controller which computes* u_i*, and let the actual input* u *used to control the system be determined by a max- or min-selector*

$$u = \max_{i\in[0,n]}(u_i) \text{ or } u = \min_{i\in[0,n]}(u_i)$$

Further, let a logic variable y_i^{lim} *be defined for each CV inequality constraint* $(i = 1, \ldots, n)$

$$y_i^{lim} = 1 \text{ for a max-constraint}$$

$$y_i^{lim} = -1 \text{ for a min-constraint}$$

Then the CV-CV switching is feasible only if

$$sgn(G_i)sgn(y_i^{lim}) = sgn(G_j)sgn(y_j^{lim}) \quad \forall i, j \in \{1, \ldots, n\}$$

where G_i *is the steady state process gain for the* i^{th} *CV* [1]*.*

Furthermore, if $sgn(G_i)z_i = 1$*, use a minimum selector block, and if* $sgn(G_i)z_i = -1$ *use a maximum selector block.*

Proof. When a CV inequality constraint y_i is active for any $i \in \{1, \ldots, n\}$, all the other CV constraints must be satisfied for the system to be feasible. Assume now that y_i is controlled but there is a disturbance that brings another output y_j to exceed its constraint. Then to remain feasible, we need to change the input u to avoid that y_j exceeds its constraint value. However, this input change will also effect the other outputs, so to guarantee feasibility, we must require that all the constrained outputs y_i $\forall i = \{1, \ldots, n\}$ move in the same direction relative to their constraints. This proves the CV-CV feasibility condition.

Next, if $sgn(G_i)sgn(y_i^{lim}) = 1$, then increasing u moves output y_i closer to its constraint, so to maintain feasibility we must use a min-selector. Similarly, if $sgn(G_i)sgn(y_i^{lim}) = -1$, then decreasing u moves output y_i closer to its constraint, so to maintain feasibility we must use a max-selector. This proves the last part of the theorem.

□

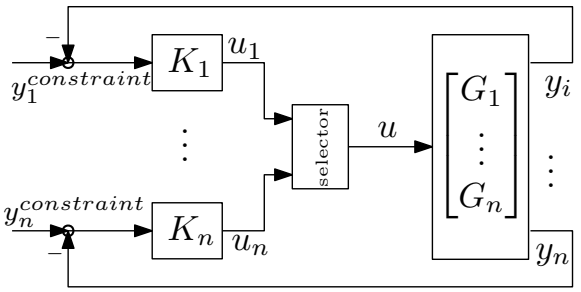


Figure 4.4: Schematic representation of a process with n CVs and 1 MV using a selector logic block with scaled variables.

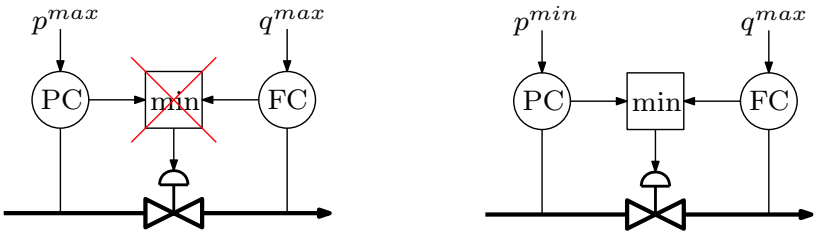


Figure 4.5: Flow line with a valve as MV and flow and inlet pressure as CVs. (a) Example of bad pairing. Theorem 4.2 is not satisfied and the selector logic does not work. (b) Theorem 4.2 is satisfied and the selector logic works.

For example, consider a flow line with a valve as the MV. Suppose we have two potential CV constraints, namely, flow q with its maximum limit scaled to q^{max} ($y_q^{lim} = 1$) and the inlet pressure p with its maximum limit scaled to p^{max} ($y_p^{lim} = 1$). The process gain from the valve to the flow is positive ($sgn(G_q) = 1$), whereas the process gain from the valve to the inlet pressure is negative ($sgn(G_p) = -1$). One can clearly, see that, with both the CVs being controlled to the max limit, $sgn(G_q)sgn(y_q^{lim}) \neq sgn(G_p)sgn(y_p^{lim})$. In this case, the flow controller would want to open the valve, whereas the pressure controller would want to close the valve (opposing MV change direction), as shown in Fig. 4.5a. the constraint feasibility is not guaranteed in this case.

However, if we instead want to control the pressure at its minimum limit p^{min} ($y_p^{lim} = -1$), and the flow at its maximum limit as before, $sgn(G_q)sgn(y_q^{lim}) = sgn(G_p)sgn(y_p^{lim}) = 1$ (see Fig. 4.5b). Therefore, using a minimum selector, the valve will be opened until either the minimum pressure limit is reached or the maximum flow limit is reached, thereby providing the desired response and ensuring feasibility of all the constraints .

When using selectors, it is important to note that there will be integral "windup" in any deselected/inactive controller [162]. A simple remedy to this problem is to

[1]Note that the possible CV equality constraint is not included in the evaluation

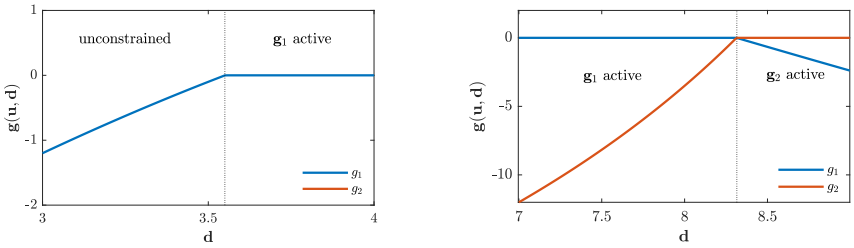
use anti-windup logic or bumpless transfer logic [58, 65] to avoid any undesired transients when switching between different controllers and controller modes.

4.2.3 MV-CV Pairing

When designing simple feedback controllers to achieve optimal operation, another important question that arises is what MVs must be used to control the active constraints and/or the gradient combinations. In other words how to pair the MV and CVs to achieve optimal operation and enable switching between the active constraint regions. As for any decentralized control structure design, useful tools such as the relative gain array (RGA) can be used to decide on the CV-MV pairing. Once the different active constraint regions are identified, the corresponding MV-CV pairing must be chosen to design the control loops such that the necessary CVs in each operating regions is controlled by an MV. Hence the active constraint regions play a very important role in choosing the MV-CV pairings. As a rule of thumb, in each active constraint region, the MV-CV pairings must be chosen based on the following:

1. Pair-close rule [168] - In order to avoid large time delays and sluggish control, it would also be wise to control a CV using an MV that is physically close to one another.
2. Non-negative RGA - CV-MV pairing must be chosen such that the steady-state RGA of the resulting transfer matrix is non-negative and close to identity matrix at crossover frequencies [168].
3. One must also try to avoid pairing important CVs with MVs that quickly saturate [167], and instead pair such MVs with less important CVs that may be given up. Moreover, if we want to switch between two active constraint regions, where the switching is between a CV and MV constraint, then by following this rule, we do not need any additional logics to switch between the CV and MV constraint. In this case, the MV would saturate automatically by giving up on the CV that is no longer active in the new operating region. If this rule is not followed, more logic blocks such as split-range control would be required to re-pair the MV and CV to achieve the same objective, making the control structure unnecessarily complicated.
4. Same MV change sign when using selectors (Theorem 4.2) - Another important consideration, when choosing the CV-MV pairing is the different combination of the active constraints that must be considered. Once the different possible CV combinations that may be encountered, are identified, selector blocks can be used to switch between two or more CVs using the same MV. When grouping the CVs together that needs to be controlled by the same MV, one must not switch between different CVs using the same MV, with opposing MV change according to Theorem 4.2.

Note that there may be several different possible MV-CV pairings to achieve the same objectives, and the pairing rules listed above can guide in selecting a good control structure design that would help reduce the number of logic blocks required to reconfigure the control loops.



(a) (b)

Figure 4.6: Example 1 - Optimal constraint values as a function of disturbance. (a) $\mathbf{d} \in [3, 4]$ (b) $\mathbf{d} \in [7, 9]$

4.3 Example 1 - Toy example

To illustrate this, consider a simple toy example with $n_u = 1$ degree of freedom. The optimization problem is given by,

$$\min_{\mathbf{u}} \quad J = (\mathbf{u} - \mathbf{d})^2 \tag{4.12}$$

s.t.

$$\mathbf{g}_1 : (4.8 - 0.4\mathbf{d})\mathbf{u} - 12 \leq 0 \tag{4.13}$$

$$\mathbf{g}_2 : 5\mathbf{u} + \mathbf{d} - 49 \leq 0 \tag{4.14}$$

For this problem, we have $n_g = 2$ potential constraints, and therefore, we can have a maximum of $2^{n_g} = 4$ active constraint regions. These correspond to;

1. Fully unconstrained (R-I)
2. Only $\mathbf{g}_1$ active (R-II)
3. Only $\mathbf{g}_2$ active(R-III)
4. Both $\mathbf{g}_1$ and $\mathbf{g}_2$ active (infeasible)

Since we only have one MV, we can control at most one CV at any given time. Therefore, the last combination with both constraints active is infeasible, and we only need to design controllers for regions R-I, R-II and R-III.

To understand better the optimal solution (and not because it is necessary for the subsequent controller design), we show the optimal active constraint values for $\mathbf{d} \in [3, 4]$ and $\mathbf{d} \in [7, 9]$ in Fig. 4.6, which was computed by solving the numerical optimization problem offline. It can be seen that, for $\mathbf{d} \leq 3.55$ the problem is unconstrained (R-I) and for $3.55 < \mathbf{d} < 8.3$, constraint $\mathbf{g}_1$ is active (R-II). At $\mathbf{d} = 8.3$ the optimal active constraint switches from g_1 to g_2 (R-III).

The unconstrained optimum for $d \leq 3.55$ is achieved when the cost gradient $\mathbf{J_u} = 0$, which from (4.12) corresponds to $\mathbf{u} = \mathbf{d}$. We therefore propose a control structure with three controllers; one to control the unconstrained optimum ($\mathbf{u} = \mathbf{d}$), and two other to control the constraints g_1 and g_2, respectively. Since in this example, $sgn(G_1)sgn(y_1^{lim}) = sgn(G_2)sgn(y_2^{lim}) = 1$ for $\mathbf{g}_1$ and $\mathbf{g}_2$ respectively, to switch between the three controllers, we use a minimum selector logic based on

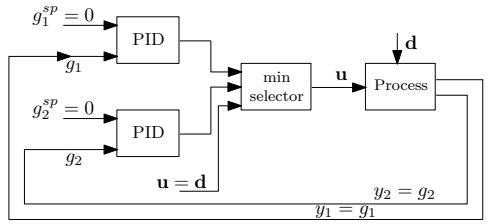


Figure 4.7: Example 1 - Block diagram of the control structure to handle changes in the active constraint set.

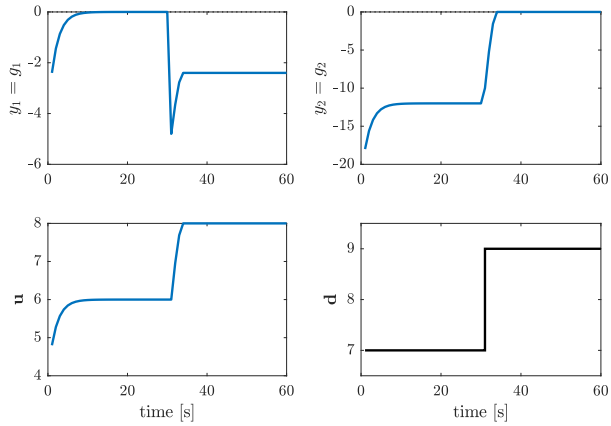


Figure 4.8: Example 1 - Simulation results showing the switching between the active constraint regions using a minimum selector block.

Theorem 4.2. The resulting control structure is shown in Fig. 4.7. The simulation results using this control structure are shown in Fig. 4.8.

4.4 Example 2 - Exothermic reactor

In this section, we consider optimal operation of a continuously stirred tank reactor (CSTR) from [41] which has been widely studied in academic research [1, 72, 187]. The CSTR process has a heater to adjust the feed temperature $u_1 = T_i$ and also the feed rate can be manipulated, $u_2 = F$. The dynamic model of the CSTR is given by (3.9) and the nominal values for the CSTR process are given in Table. 3.1.

For this CSTR process, the two degrees of freedom are $\mathbf{u} = [T_i \quad F]^\mathsf{T}$ and the objective is to maximize a weighted sum of the throughput rate F and the product concentration C_B, while penalizing utility costs associated with a high feed temperature T_i. This is subject to maximum feed rate F, maximum temperature T, and maximum impurity C_A constraint in the outflow. The optimization problem

is formulated as,

$$\begin{aligned} \min_{T_i,F} \quad & -F - 2.009C_B + (1.657 \times 10^{-3}T_i)^2 \\ \text{s.t.} \quad & \\ & \mathbf{g}_1 : F/F^{max} - 1 \leq 0 \\ & \mathbf{g}_2 : T/T^{max} - 1 \leq 0 \\ & \mathbf{g}_3 : C_A/C_A^{max} - 1 \leq 0 \end{aligned} \tag{4.15}$$

The concentration of component A in the feed stream $C_{A,i}$ is a disturbance and varies in the range from 0.7 to $1.1mol/l$.

For this system, we have $n_g = 3$ constraints and $2^3 = 8$ potential active constraint regions, namely,

1. Fully unconstrained (never)
2. only $\mathbf{g}_1$ active (R-I)
3. only $\mathbf{g}_2$ active (never)
4. only $\mathbf{g}_3$ active (never)
5. $\mathbf{g}_1$ and $\mathbf{g}_2$ active (R-II)
6. $\mathbf{g}_2$ and $\mathbf{g}_3$ active (R-III)
7. $\mathbf{g}_1$ and $\mathbf{g}_3$ active (unlikely)
8. $\mathbf{g}_1$, $\mathbf{g}_2$ and $\mathbf{g}_3$ active (infeasible)

Since we only have two MVs, we can only control at most 2 constraints active at any time. Therefore the potential constraint region 8 listed above is infeasible and can be eliminated. Also, from the cost function in (4.15), the feed rate F will be maximized (i.e. $\mathbf{g}_1$ is active) as long as there are any unconstrained degrees of freedom. This eliminates the fully unconstrained region, the region where only $\mathbf{g}_2$ is active and region where only $\mathbf{g}_3$ are active. Furthermore, based on engineering insight and for the given range of disturbance and activation energy, the reactor temperature constraint ($\mathbf{g}_2$) will become active before the impurity constraint ($\mathbf{g}_3$) becomes active. This eliminates region 7 as unlikely and leaves us with active constraint regions R-I, R-II and R-III. Solving the numerical optimization problem offline for the expected disturbances confirms that we need to consider the three regions. This is illustrated in Fig. 4.9 which shows the three constraint variables as a function of $C_{A,i}$.

In R-I, when $C_{A,i} < 0.85mol/l$, only the feed rate constraint ($\mathbf{g}_1$) is active, leaving one MV unconstrained. This belongs to the partially constrained case (case 3). Here, we can use the feed rate $u_1 = F$ to maintain the throughput at its constraint value of F^{max}, and use the inlet temperature $u_2 = T_i$ to control the gradient $\mathbf{J_u} = \nabla_{T_i} J$ (the steady-state gradient from T_i to J) at a constant setpoint of zero.

This follows from (4.7) which gives,

$$\begin{aligned} \nabla_{T_i}\mathcal{L} =& \nabla_{T_i} J + \lambda \nabla_{T_i}(F - F^{max}) = 0 \\ \Rightarrow& \nabla_{T_i} J + \lambda \cdot 0 = 0 \\ \Rightarrow& \nabla_{T_i} J = 0 \end{aligned} \tag{4.16}$$

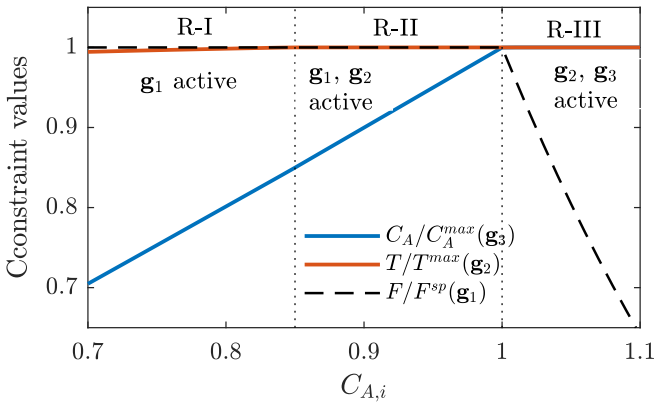


Figure 4.9: Example 2 - Optimal constraint values as a function of the disturbance. The three active constraint regions R-I, R-II and R-III are clearly marked.

In this case, we estimate the steady-state gradient $\mathbf{J_u} = \nabla_{T_i} J$ by linearizing a nonlinear dynamic model around the current operating point, as proposed by Krishnamoorthy et al. [102]. Note that any gradient estimation method may be used here to estimate the steady-state cost gradient[173].

In R-II, when $0.85 < C_{A,i} < 1mol/l$, there are two active constraints, namely, the feed rate ($\mathbf{g}_1$) and the reactor temperature ($\mathbf{g}_2$). This is a fully constrained case (case 2) and we use the feed rate F to maintain the throughput at F^{max} as in region R-I, and use the inlet temperature T_i to maintain the reactor temperature at its maximum limit.

In R-III, when $C_{A,i} > 1mol/l$, we also have a fully constrained case with the reactor temperature constraint ($\mathbf{g}_2$) and the product concentration constraint ($\mathbf{g}_3$) being active. The feed rate is no longer at its maximum and is instead used to control the concentration of component A in the outlet less than the maximum limit of $0.5mol/l$ in order to meet the product requirement. As in region R-II, the inlet temperature is used to control the reactor temperature at its maximum limit.

To switch between the different active constraint regions, minimum selector blocks are used. The control structure including the selectors to handle changes in the active constraint is shown in Fig. 4.10. The proposed control structure was tested in simulations with varying values of the disturbance $C_{A,i}$ and the results are shown in Fig. 4.11. All the controllers are PI controllers and are tuned using the SIMC tuning rules [166]. The proportional gain K_C and integral gain K_I are shown in Table. 4.1.

Table 4.1: Controller Tunings used for example 2 (Fig. 4.10)

	TC	GC	CC	FC
K_C	0.0167	4167	3.3661	0
K_I	0.0167	64.1149	0.2805	1

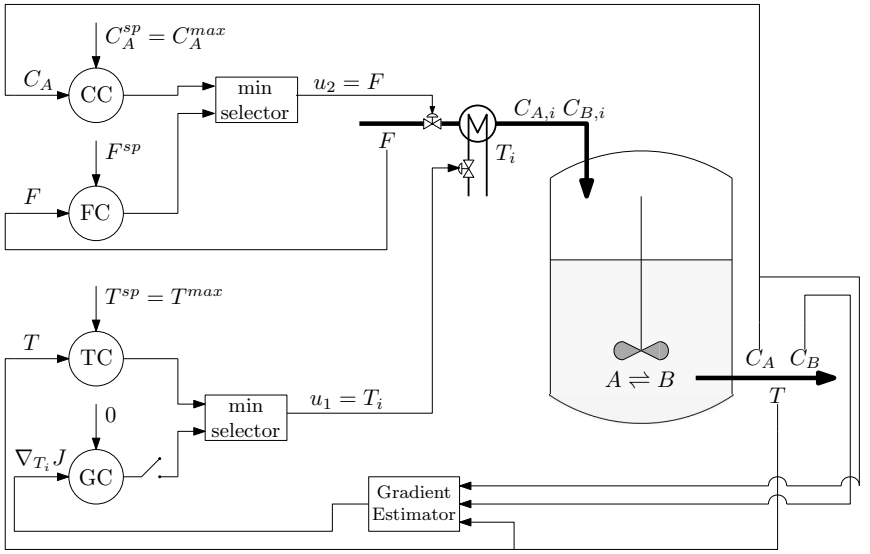


Figure 4.10: Example 2 - Proposed control structure design for optimal operation over Regions R-I, R-II and R-III

The simulation starts with $C_{A,i} = 0.75mol/l$ and we can see that the flow controller (denoted FC in Fig. 4.10) maintains the feed rate its setpoint of $F^{max} = 1mol/min$ and the gradient controller (denoted by GC in Fig. 4.10) drives the system to its optimum. At time $t = 20$min, the feed concentration changes to $C_{A,i} = 0.9mol/l$. In this case the temperature controller (denoted by TC in Fig. 4.10) takes over from the gradient controller and maintains the reactor temperature at its setpoint of $T^{max} = 425K$. Finally, at time $t = 40$min, the feed concentration further increases to $C_{A,i} = 1.1mol/l$. In order to meet the purity requirement on the product, the concentration controller (denoted by CC in Fig. 4.10) takes over the feed rate control to maintain the concentration of component A at its maximum limit of $C_A^{max} = 0.5mol/l$ by reducing the feed rate.

The true optimal steady-state solution computed by solving a steady-state numerical optimization problem is used as a base case and is compared with the converged solution in Table. 4.2. By comparing these simulation results with the true optimal solution, we find as expected that the simple feedback control structure provides optimal operation without needing to solve online numerical optimization problems. The simulation results also demonstrate that simple logics are sufficient to handle changes in active constraint sets without using the model online to detect change in active constraint regions.

As mentioned earlier, the gradient controller (GC) that controls the steady-state gradient must be inactive in region R-III, since it is not a neighboring region to region R-I. Therefore, one should only to track the gradient in R-I and its neighboring region R-II. One way to do this is by turning off the GC controller when the concentration controller CC that controls C_A becomes active, indicating operation in region 3. Fig. 4.11 also shows as dashed lines, the simulation results when the

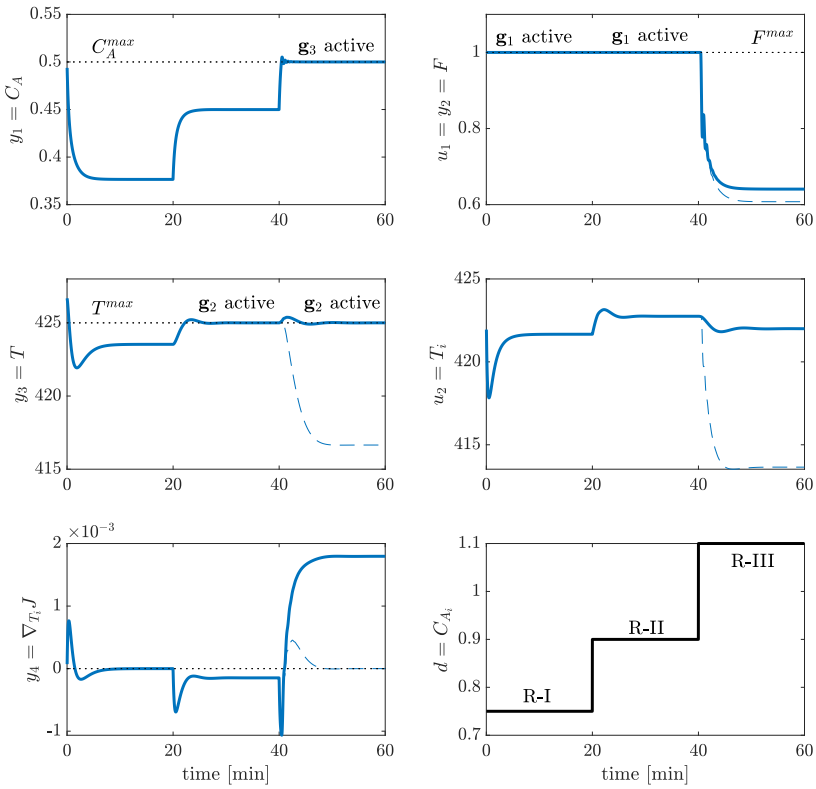


Figure 4.11: Example 2 - Simulation results using the proposed control structure (solid thick lines). The thin dashed lines show the simulation results if the GC controller is incorrectly activated in R-III as a motivating example.

gradient controller GC incorrectly controls the gradient from R-I also in R-III. This motivates the need for the switch in Fig. 4.10 which automatically deselects the gradient controller in region R-III, when the concentration C_A becomes active. As shown in the simulation results, if the gradient controlled is not turned off in region R-III, we get non-optimal operation with $T < T^{max}$ ($\mathbf{g}_2$ not active as it should be).

Comment: Although the case with $\mathbf{g}_1$ and $\mathbf{g}_3$ active was not relevant for the considered disturbance (and listed as unlikely), it may occur if, for example the activation energy also changes. However, switching to a case where the constraints $\mathbf{g}_1$ and $\mathbf{g}_3$ are active will require a re-pairing of MVs and CVs, since controllers CC and FC are the two controllers that need to be used in this region and they now both use $u_2 = F$. Additional logics such as split range control logic would be needed to handle this.

Table 4.2: Comparison of the true optimal steady-state solution and the converged solution.

		$C_{A,i} = 0.75$	$C_{A,i} = 0.9$	$C_{A,i} = 1.1$
True optimum	$u_1 = F$	1.000	1.000	0.6408
	$u_2 = T_i$	421.7	422.7	422.0
Converged solution	$u_1 = F$	1.000	1.000	0.6408
	$u_2 = T_i$	421.7	422.7	422.0

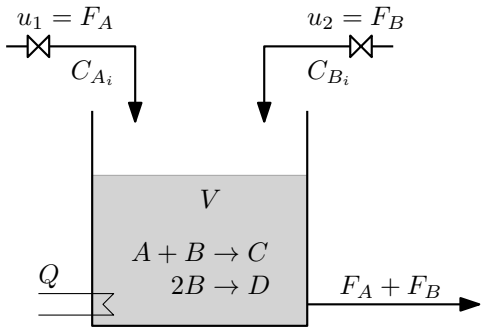


Figure 4.12: Example 3 - Isothermal CSTR [27, 171]

4.5 Example 3 - Isothermal CSTR

In this section, we consider the optimal operation of another CSTR, as studied by Srinivasan et al. [171] and Chachuat et al. [27]. It consists of an isothermal CSTR with two exothermic reactions, namely,

$$A + B \rightarrow C \tag{4.17a}$$

$$2B \rightarrow D \tag{4.17b}$$

The desired product is C, while D is an undesired by-product. The CSTR has two feed streams $u_1 = F_A$ and $u_2 = F_B$ with corresponding known inlet concentrations C_{A_i} and C_{B_i} respectively (see Fig. 4.12).

We modify the steady-state model from Chachuat et al. [27] to get a dynamic model. Assuming perfect temperature control (isothermal) and level control (constant V), the dynamic model of the process is given by the following component mass balances:

$$\frac{\mathrm{d}n_A}{\mathrm{d}t} = F_A C_{A_i} - (F_A + F_B)C_A - k_1 C_A C_B V \tag{4.18a}$$

$$\frac{\mathrm{d}n_B}{\mathrm{d}t} = F_B C_{B_i} - (F_A + F_B)C_B - k_1 C_A C_B V - 2k_2 C_B^2 V \tag{4.18b}$$

$$\frac{\mathrm{d}n_C}{\mathrm{d}t} = -(F_A + F_B)C_C + k_1 C_A C_B V \tag{4.18c}$$

where n_A, n_B and n_B are the number of moles of components A, B and C respectively. $C_A = n_A/V$, $C_B = n_B/V$ and $C_C = n_C/V$ are the concentration of

components A, B and C in the product stream.

The heat produced by the chemical reaction is given by,

$$Q = (-\Delta H_1)k_1 C_A C_B V + (-\Delta H_2)k_2 C_B^2 V \tag{4.19}$$

where ΔH_1 and ΔH_2 denotes the enthalpy of the reactions 1 and 2, respectively. The nominal model parameters are the same as the one used in [27], and are shown in Table. 4.3.

Table 4.3: Nominal values for CSTR process from Case study 2[27].

	Description	Value	Unit
$C^*_{A,i}$	Inlet A concentration	2	mol
$C^*_{B,i}$	Inlet B concentration	1.5	mol
ΔH_1	Enthalpy of reaction 1	-7×10^4	J/mol
ΔH_2	Enthalpy of reaction 2	-1×10^5	J/mol
V	Tank Volume	500	l
k_1	Reaction rate 1	1.5	$l/mol/h$
k_2	Reaction rate 2	0.014	$l/mol/h$
F^{max}	Maximum flow rate	22	l/h
Q^{max}	Maximum heat	1000	kJ/h

The objective is to use feed streams $\mathbf{u} = [F_A \quad F_B]$ to maximize the production of component C, which is expressed as the amount of product C, given by the expression, $(F_A + F_B)C_C$ multiplied by the yield factor $(F_A + F_B)C_C/F_A C_{A_i}$ [27]. In addition there are constraints on the cooling Q and the total outflow $(F_A + F_B)$. The optimization problem is then expressed as,

$$\min_{F_A, F_B} \quad J = -\frac{(F_A + F_B)^2 C_C^2}{F_A C_{A_i}}$$

$$\text{s.t} \tag{4.20}$$

$$\mathbf{g}_1 : Q/Q^{max} - 1 \leq 0 \tag{4.21}$$

$$\mathbf{g}_2 : (F_A + F_B)/F^{max} - 1 \leq 0$$

Since we have $n_g = 2$ constraints, we have a maximum of $2^2 = 4$ potential active constraint regions, namely,

1. Fully unconstrained (unlikely)
2. Only $\mathbf{g}_1$ active (R-I)
3. Only $\mathbf{g}_2$ active (R-III)
4. Both $\mathbf{g}_1$ and $\mathbf{g}_2$ active (R-II)

The reaction rate $d = k_1$ is a disturbance and varies in the range from 0.3 to $1.5 l/mol\, h$. In this range, the active constraints at the optimum changes with k_1.

For the purpose of verification of the potential active constraints listed above, and not for the subsequent control structure design, we solve the numerical optimization problem offline for the expected disturbances, which verifies that we only need to consider three different combinations of active constraints; only $\mathbf{g}_1$

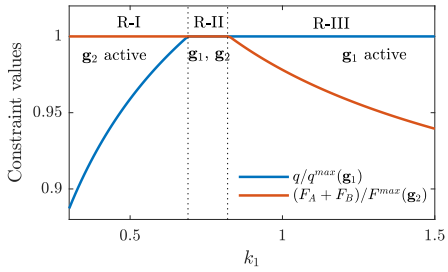


Figure 4.13: Example 3 - Optimal values of the normalized constraint variables Q/Q^{max} (solid blue lines) and $(F_A + F_B)/F^{max}$ (solid red lines) as a function of the reaction rate k_1. The three active constraint regions are clearly marked.

active (R-I), both $\mathbf{g}_1$ and $\mathbf{g}_2$ active (R-II) and only $\mathbf{g}_2$ active (R-III). The active constraint regions as a function of the k_1 is shown in Fig. 4.13. It can be seen that there are three active constraint regions, which are marked with R-I, R-II and R-III respectively.

In region R-I, when $k_1 < 0.69$, only the outlet flow constraint is active ($\mathbf{g}_2$). This belongs to the partially constrained case (case 3), where we use feed stream $u_1 = F_A$ to control the feed flow (F_A+F_B) at its maximum limit. We use $u_2 = F_B$ to drive the system to its optimum using a gradient controller. Following Theorem 4.1, expression (4.8) in this region looks like,

$$\begin{aligned} \nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u},\mathbf{d}) = \nabla_{\mathbf{u}}J(\mathbf{u},\mathbf{d}) + \nabla_{\mathbf{u}}(F_A + F_B - F^{max})^{\mathsf{T}}\lambda = 0 \\ \Rightarrow \begin{bmatrix} \frac{\partial J}{\partial F_A} \\ \frac{\partial J}{\partial F_B} \end{bmatrix} = -\lambda \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned} \tag{4.22}$$

To eliminate the Lagrange multiplier, we pre-multiply (4.22) by the nullspace of the active constraint gradients, which in this case is given by $[1 \quad 1]^{\mathsf{T}}$ and the corresponding nullspace $\mathbf{N} = [-0.7071 \quad 0.7071]$. This gives the gradient combination

$$\mathbf{c}_1 = \frac{\partial J}{\partial F_A} - \frac{\partial J}{\partial F_B} = 0 \tag{4.23}$$

Therefore, we use the second degree of freedom to maintain gradient combination $\mathbf{c}_1$ to a constant setpoint of zero. In this simulation study, we estimate the steady-state gradients $\nabla_{\mathbf{u}}J$ by linearizing a nonlinear dynamic model around the current operating point as described in Chapter 3.

In region R-II, when $0.69 < k < 0.82$, both constraints $\mathbf{g}_1$ and $\mathbf{g}_2$ are active. This belongs to the fully constrained case (case 2), where optimal operation can be achieved by simply controlling the heat produced to its maximum limit using F_B and the outlet flowrate at its maximum limit using F_A.

In region R-III, when $k > 0.82$, the outlet flowrate constraint is no longer active and only the maximum cooling constraint is active ($\mathbf{g}_1$). This again corresponds the partially constrained case. Following Theorem 4.1, expression (4.8) in this region

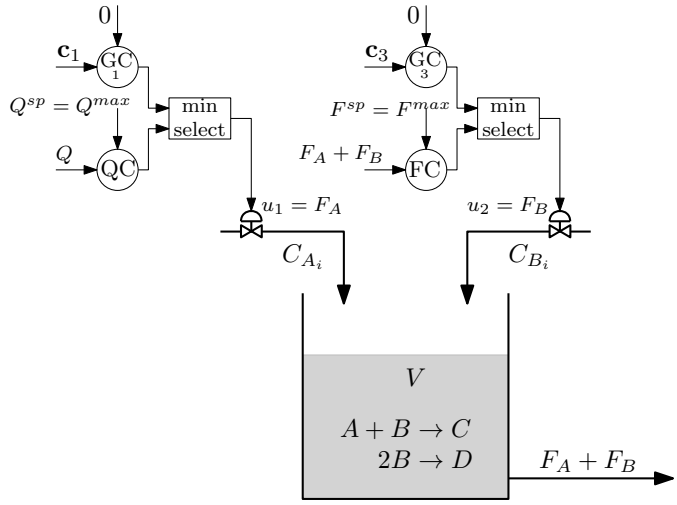


Figure 4.14: Example 3 - Proposed control structure design for optimal operation over Regions R-I, R-II and R-III

looks like,

$$\begin{aligned} \nabla_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mathbf{d}) = \nabla_{\mathbf{u}} J(\mathbf{u}, \mathbf{d}) + \nabla_{\mathbf{u}} (Q - Q^{max}) \lambda = 0 \\ \Rightarrow \begin{bmatrix} \frac{\partial J}{\partial F_A} \\ \frac{\partial J}{\partial F_B} \end{bmatrix} = -\lambda \begin{bmatrix} \frac{\partial Q}{\partial F_A} \\ \frac{\partial Q}{\partial F_B} \end{bmatrix} \end{aligned} \tag{4.24}$$

To eliminate the Lagrange multiplier, we pre-multiply (4.24) by the nullspace of the active constraint gradients, which in this case is denoted by $\mathbf{N} = [n_1 \quad n_2]$. This gives the gradient combination of

$$\mathbf{c}_3 = n_1 \frac{\partial J}{\partial F_A} + n_2 \frac{\partial J}{\partial F_B} = 0 \tag{4.25}$$

Therefore, we use the second degree of freedom $u_2 = F_B$ to maintain gradient combination $\mathbf{c}_3$ to a constant setpoint of zero.

To switch between the different active constraint regions, minimum selector blocks, as shown in Fig. 4.14. The proposed control structure was tested in simulations with varying values of k_1 and shown in Fig. 4.15. All the controllers used in this simulation are PI controllers and are tuned using SIMC tuning rules[166]. The proportional gain K_C and the integral gain K_I for the PI controllers are shown in Table. 4.4. As mentioned earlier, the GC1 controller need not be tracked in R-III and similarly GC3 controller need not be tracked in R-I. In other words, GC1 controller can be turned off when GC3 controller is active and vice versa.

The simulation starts in R-III with $k_1 = 1.5 \quad l/mol\,h$. In this case, the cooling is maintained at its maximum limit using $u_1 = F_A$ (QC in Fig. 4.14) and the gradient combination $\mathbf{c}_3$ is controlled using $u_2 = F_B$ (GC3 in Fig. 4.14). At time $t = 20$min, k_1 is ramping down to $k_1 = 0.75 \quad l/mol\,h$ at time $t = 40$min, possibly

Table 4.4: Controller Tunings used in the controllers shown in Fig. 4.14

	QC	GC1	GC3	FC
K_C	3.1807	2.1191	2.6647	0
K_I	1.2723	0.0706	0.1665	1

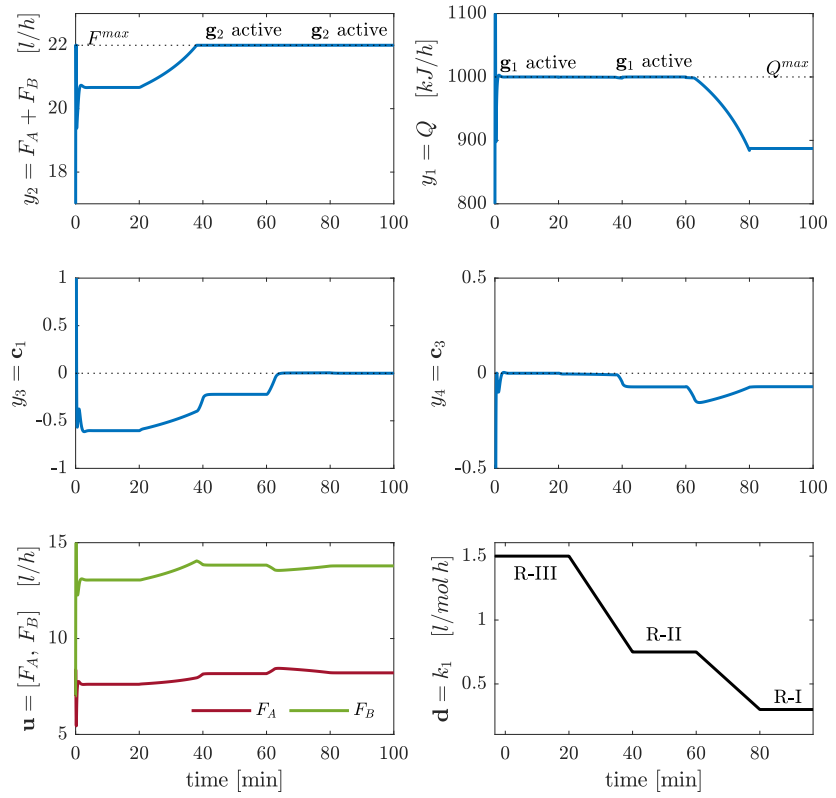


Figure 4.15: Example 3 - Simulation results using the proposed control structure.

caused by deactivation of the catalyst. When this happens, the maximum limit on $F_A + F_B$ is reached and the flow control FC takes over from the GC3 controller. The cooling is still maintained at its maximum limit by the QC controller. At time $t = 60$min, k_1 further ramps down to $k_1 = 0.3 \quad l/mol\,h$ at time $t = 80$min. When this happens, the constraint on the cooling is no longer active and the gradient controller GC1 takes over to maintain the gradient combination $\mathbf{c}_1$ at a constant setpoint of 0.

The true optimal steady-state solution computed by solving a steady-state numerical optimization problem is used as a base case and is compared with the con-

Table 4.5: Comparison of the true optimal steady-state solution and the converged solution.

		$k_1 = 1.5$	$k_1 = 0.75$	$k_1 = 0.3$
True optimum	$u_1 = F_A$	7.615	8.171	8.211
	$u_2 = F_B$	13.05	13.83	13.79
Converged solution	$u_1 = F_A$	7.615	8.171	8.211
	$u_2 = F_B$	13.05	13.83	13.79

verged solution in Table. 4.5. This simulation example shows that simple feedback control structures provide optimal operation without needing to solve online numerical optimization problem. The simulation results also demonstrate that simple logics are sufficient to handle changes in active CV constraint regions. As mentioned earlier, the GC1 controller that controls the unconstrained optimum in region R-I is turned off in region R-III and similarly, the GC3 controller that controls the unconstrained optimum in region R-III is turned off in region R-I.

Although the fully unconstrained case was not relevant active constraint region for the considered disturbance, this can be easily included in the proposed control structure by adding one more control loop for each MV that controls the steady-state gradients $\nabla_{F_A} J$ and $\nabla_{F_A} J$ using $u_1 = F_A$ and $u_2 = F_B$ to a constant setpoint of zero. The minimum selector box implemented for each MV will then select the minimum of the three controller outputs to be implemented on the process.

4.6 Example 4- Oil production optimization

In this section, we show how we can achieve optimal operation of a gas-lifted well network using simple feedback control structures. We consider a production network with n_w gas-lifted wells, producing to a common riser manifold (See Appendix A). The optimization problem can be stated as:

$$\begin{aligned} \min_{w_{gl_i}} \quad & J = -\$_o \sum_{i=1}^{n_w} w_{po_i} + \$_{gl} \sum_{i=1}^{n_w} w_{gl_i} \\ \text{s.t.} \quad & \sum_{i=1}^{n_w} w_{pg_i} \leq w_{pg}^{max} \\ & \sum_{i=1}^{n_w} w_{gl_i} \leq w_{gl}^{max} \end{aligned} \tag{4.26}$$

where $\$_o$ and $\$_{gl}$ are the oil price and the cost of gas compression respectively. w_{po_i} and w_{pg_i} are the produced oil and gas from from well i respectively. The gas-lift rate is denoted by w_{gl_i}, which are the manipulated variables. w_{pg}^{max} and w_{gl}^{max} are the maximum gas processing capacity, and the maximum gas available for gas-lift respectively.

A gas-lifted production network with n_w wells connected to a riser has $2n_w + 1$ degrees of freedom, namely, n_w gas-lift injection rates, n_w production valves on

the wells and 1 valve on the riser. Typically, the gas-lifted wells are predominantly controlled by the gas-lift injection rate only. The production valves on the wells and the riser are kept at its maximum limit (which is either constrained by the physical opening or other effects such as casing-heading etc.), since decreasing the valve position reduces the flow from the reservoir. Thus, the production valves are all kept at a constant opening and only n_w gas-lift injection rates are the degrees of freedom in this work [96].

4.6.1 Control structure design - Unconstrained case

In the unconstrained case, the ideal self-optimizing variable would be the cost gradient with respect to the inputs, which must be equal to zero at the optimum. From (4.26), this would be given by the expression,

$$\frac{\partial J}{\partial w_{gl_i}} = -\$_o \frac{\partial w_{po_i}}{\partial w_{gl_i}} + \$_{gl} = 0 \quad \forall i \in 1, \ldots, n_w \tag{4.27}$$

In order to achieve the necessary condition of optimality, rearranging (4.27) gives,

$$\frac{\partial w_{po_i}}{\partial w_{gl_i}} = \frac{\$_{gl}}{\$_o} \quad \forall i \in 1, \ldots, n_w \tag{4.28}$$

The term $\frac{\partial w_{po_i}}{\partial w_{gl_i}}$ is commonly known as *marginal gaslift-oil-ratio* (often abbreviated as marginal GOR), which is defined as the change in oil rate per unit change in the gas-lift injection rate. In the rest of the chapter, we denote marginal GOR by

$$\nu_i := \frac{\partial w_{po_i}}{\partial w_{gl_i}}$$

Therefore, in the unconstrained case, the marginal GOR for all the wells ν_i must be controlled to constant setpoint of $\frac{\$_{gl}}{\$_o}$ [96]. That is, the controlled variables are

$$CV_1 = \nu_1 \rightarrow \frac{\$_{gl}}{\$_o} \tag{4.29}$$

$$CV_i = \nu_{i-1} - \nu_i \rightarrow 0, \quad \forall i = 2, \ldots, n_w \tag{4.30}$$

4.6.2 Control structure design - Produced gas capacity constraint active

If the total gas processing capacity is low, then the optimum occurs when all the available gas processing capacity is fully utilized. Hence the total gas processing capacity constraint becomes active. We use one well to control this active constraint tightly. For the remaining $(n_w - 1)$ well's gas-lift, the optimum happens when the marginal GOR is equal for all the wells. This is because, for any parallel unit, the optimum happens when the marginal cost is equal as shown by [37]. One can also easily prove this again using Theorem 4.1. This concept has been used in gas-lift optimization in several works such as [79, 96, 161, 183] to name a few.

When it comes to selecting which well to use to control the active constraint tightly, the well with the largest MV (flow) should be used to control the active constraint. This gives better control of the active constraint (high gain and higher controllability range). Controlling the active constraint with a small MV may quickly saturate, leading to constraint violation or suboptimal operation. In this case, we assume that the wells are numbered in decreasing order in terms of the flow, with well 1 having the largest flow. Therefore well 1 is used to control the active constraint.

Therefore, in this case, the $(n_w - 1)$ self-optimizing CVs are given by,

$$CV_i = \nu_{i-1} - \nu_i, \quad \forall i = 2, \ldots, n_w \tag{4.31}$$

which are controlled to a constant setpoint of zero, thereby achieving equal marginal GOR. We note that (4.31) is the same as (4.30). For well 1, the controlled variable is given by

$$CV_1 = \sum_{i=1}^{n_w} w_{pg_i} \tag{4.32}$$

which is controlled at its maximum limit of w_{pg}^{max}, thereby achieving active constraint control.

4.6.3 Control structure design - Gas-lift constraint active

When the total available gas-lift is limited such that each of the wells cannot be operated at its local optimum, then the optimum occurs when all the available gas is used for gas-lift. Therefore the total gas-lift constraint is active, and one of the well's gas-lift (or more precisely, one degree of freedom related to gas-lift) must be used to control the total gas-lift rate at its maximum limit of w_{gl}^{max}. Again we use the well with the largest MV (flow) for active constraint control as mentioned in Section 4.6.2. For the remaining $(n_w - 1)$ well's gas-lift rate, the optimum happens when the marginal GOR is equal. Therefore, in this case, the $(n_w - 1)$ self-optimizing CVs are given by (4.31). For active constraint control, the total gas-lift rate must be at its constraint. So,

$$CV_1 = \sum_{i=1}^{n_w} w_{gl_i} \tag{4.33}$$

which should be controlled to a constant setpoint of w_{gl}^{max}. This may be viewed as an MV constraint, because we can simply keep MV_1 at a constant value of $w_{gl_1} = w_{gl}^{max} - \sum_{i=2}^{n_w} w_{gl_i}$.

4.6.4 Control structure design - Summary of cases

A comparison of the three cases shows that, in all the cases, we must control $(n_w - 1)$ self-optimizing CVs given by (4.31), whereas CV_1 changes:

Case 4.6.1: $CV_1 = \nu_1$ with $CV_1^{sp} = \frac{\$_{gl}}{\$_o}$

Case 4.6.2: $CV_1 = \sum_{i=1}^{n_w} w_{pg_i}$ with $CV_1^{sp} = w_{pg}^{max}$

Case 4.6.3: $CV_1 = \sum_{i=1}^{n_w} w_{gl_i}$ with $CV_1^{sp} = w_{gl}^{max}$

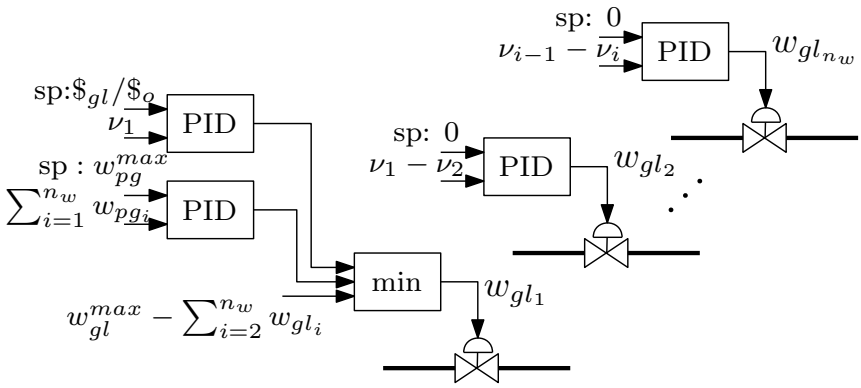


Figure 4.16: Example 4: Schematic representation of the proposed control structure design for optimal operation of a gas-lifted well network.

For all the three cases, CV_i is given by $\nu_{i-1} - \nu_i, \quad \forall i = 2, \ldots, n_w$ Thus, with the chosen pairings, the CV only changes for well 1 depending on the three operating regions. Since this is a CV-CV switching, we use a minimum selector block to select between (4.29), (4.32) and (4.33). The proposed control structure is schematically represented in Fig. 4.16.

4.6.5 Simulation results

In this simulation study, we consider a network of $n_w = 6$, as such we have 6 MVs. For the last 5 MVs, we control the CVs (4.31) to a constant setpoint of zero, thereby ensuring equal marginal GOR for all the wells. For the first well, we design three controllers to control (4.29), (4.33) and (4.32), and use a minimum selector block to select between the three controllers as shown in Fig. 4.16. Simple PI controllers are used for all the controllers, and the controller gains were tuned using SIMC rules [166]. The controller tuning parameters are shown in Table 4.6. For this simulation case example, the model is the same as the one shown in AppendixA with $n_w = 6$. The model parameters are shown in Table 4.7.

We use the following data: The ratio of the oil price to cost of compression $\frac{\$_{gl}}{\$_o} = 0.25$. The total available gas-lift is limited to $w_{gl}^{max} = 10kg/s$, and the nominal gas processing capacity is constrained at $w_{pg}^{max} = 30kg/s$. The production network is simulated for a total of 10 hours. There are different methods to estimate the steady-state gradient, which is the marginal GOR in this case. In this chapter, the marginal GOR is estimated using a nonlinear process model and measurements as described in Chapter 3 (see also [96]). Initially, we simulate a case where, the GOR of the different wells are such that the optimum occurs when all the available gas is used for gas-lift. We see that the proposed control structure design achieves this by automatically selecting (4.33) as CV_1. The difference in the marginal GOR for the wells is also controlled to zero, implying that the wells are operated at equal marginal GOR, as shown in Fig. 4.17. At time $t = 3$hrs, a disturbance causes the GOR of well 3 to increase from 0.09 kg/kg to 0.11 kg/kg. The total gas-lift

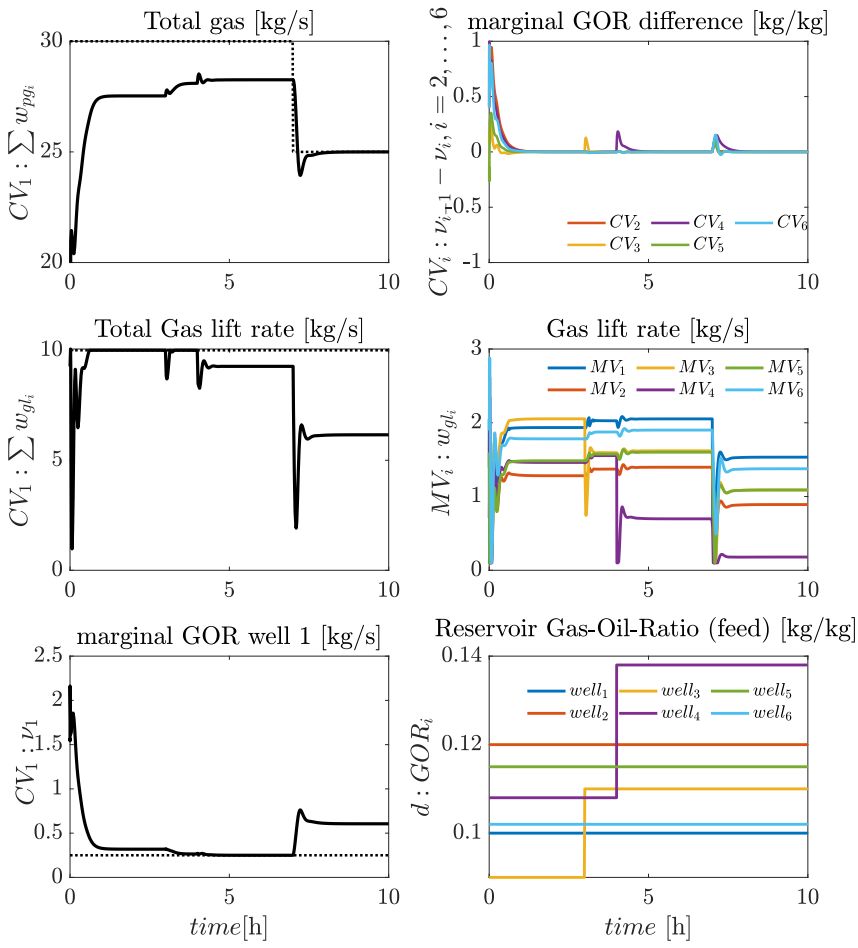


Figure 4.17: Simulation results showing the optimal operation of a production network with 6 gas-lifted wells for the three different cases.

Table 4.6: Controller parameters used in the gas-lift case study

well	CV	CV^{sp}	K_P	K_I
well 2	$\nu_1 - \nu_2$	0	13.79	0.0173
well 3	$\nu_2 - \nu_3$	0	9.12	0.0114
well 4	$\nu_3 - \nu_4$	0	13	0.016
well 5	$\nu_4 - \nu_5$	0	12.5	0.015
well 6	$\nu_5 - \nu_6$	0	7.07	0.0089
	$\sum w_{pg_i}$	w_{tg}^{max}	0.267	3.3e-4
well 1	ν_1	$\$_{gl}/\$_o$	9.302	0.0116
	$\sum w_{gl_i}$	w_{gl}^{max}	-	-[1]

[1] No controller is needed for the maximum gas-lift rate $\sum w_{gl_i}$. This is achieved by having a constant MV $w_{gl_1} = w_{gl}^{max} - \sum_{i=2}^{n_w} w_{gl_i}$

constraint is still active, and the gas-lift injection rates for the different wells are adjusted automatically to optimally allocate the total available gas-lift to reflect the new operating conditions.

At time $t = 4$hrs, another disturbance causes the GOR of well 4 to increase from 0.108 kg/kg to 0.138 kg/kg. In this case, neither the total gas-lift constraint, nor the gas capacity constraint is active (unconstrained case). When this disturbance happens, the min selector block automatically chooses (4.29) as CV_1, and we see that the marginal GOR of all the wells are now controlled to a constant value of $\frac{\$_{gl}}{\$_o} = 0.25$.

At time $t = 7$hrs, the total topside gas processing capacity reduces from $w_{pg}^{max} = 30kg/s$ to $w_{pg}^{max} = 25kg/s$. The optimum is then when all the available gas processing capacity is fully utilized. In this case, the min selector automatically switches CV_1 to (4.32), and we see that the total gas produced is controlled at its maximum limit of $w_{pg}^{max} = 25kg/s$. At the same time, the marginal GOR for all wells are kept equal by the proposed controller structure, thus leading to optimal operation for the new operating condition.

From the simulation results, it can be clearly seen that optimal operation under varying operating conditions can be achieved using simple feedback controllers, without the need for advanced optimization tools. Experimental study of optimal control of electrical submersible pump (ESP) lifted oil wells using such simple PID controllers with logics were also tested on a large-scale experimental test facility with a full scale ESP and live viscous crude oil, which shows that classical feedback controllers with simple logic blocks are sufficient to achieve optimal operation of small-scale processes [100] [2].

4.7 Chapter Summary

In this chapter, we formalized a framework for optimal control structure design for four different operating cases, namely

[2] The experimental test results are published in [100] which is appended to this thesis in Appendix E

- Case 1 - fully constrained (active constraint control)
- Case 2 - fully unconstrained (control cost gradient to zero)
- Case 3 - partially constrained (active constraint control and control linear gradient combination to zero for unconstrained DOF)
- Case 4 - over constrained (give up less important constraints [3])

In the partially constrained case, we also proposed to control a linear combination of cost gradients as the self-optimizing variable for the unconstrained degrees of freedom. Further, we showed that simple controller logics such as selectors can be used to handle changes in the active constraint regions. Using four different examples, we showed that optimal operation of processes can be achieved using simple feedback control structures, without needing to solve computationally expensive optimization problems online. In all the examples, it was shown that true optimal solution can be obtained using simple feedback controllers despite changes in the active constraint regions, without the need to use models online to detect changes in active constraint regions.

[3] analogous to soft constraints

Table 4.7: Model parameters used in the gas lift well model equations from Appendix A

parameter	description	unit	well 1	well 2	well 3	well 4	well 5	well 6	riser
L_w	length of well tubing	m	1500	1500	1500	1500	1500	1500	-
H_w	height of well tubing	m	1000	1000	1000	1000	1000	1000	-
D_w	diameter of well tubing	m	0.121	0.121	0.121	0.121	0.121	0.121	-
L_{bh}	length of well below injection	m	500	500	500	500	500	500	-
H_{bh}	height of well below injection	m	500	500	500	500	500	500	-
D_{bh}	diameter of well below injection	m	0.121	0.121	0.121	0.121	0.121	0.121	-
L_a	length of well annulus	m	1500	1500	1500	1500	1500	1500	-
H_a	height of well annulus	m	1000	1000	1000	1000	1000	1000	-
D_a	diameter of well annulus	m	0.189	0.189	0.189	0.189	0.189	0.189	-
L_r	length of riser	m	-	-	-	-	-	-	500
H_r	height of riser	m	-	-	-	-	-	-	500
D_r	diameter of riser	m	-	-	-	-	-	-	0.121
ρ_o	oil density	kg/m^3	800	800	790	800	820	805	-
GOR_0	nominal gas-oil-ratio	kg/kg	0.1	0.12	0.09	0.108	0.115	0.102	-
p_{res}	reservoir pressure	bar	150	155	155	160	155	155	-
PI	Productivity index	$kg/s/bar$	7	7	7	7	7	7	-
C_iv	injection valve characteristics	m^2	0.1E-3	0.1E-3	0.1E-3	0.1E-3	0.1E-3	0.1E-3	-
C_pc	production valve characteristics	m^2	2E-3	2E-3	2E-3	2E-3	2E-3	2E-3	-
C_pr	riser valve characteristics	m^2v	10E-3	10E-3	10E-3	10E-3	10E-3	10E-3	-
T_a	Annulus temperature	$^\circ C$	28	28	28	28	28	28	-
T_w	well tubing temperature	$^\circ C$	32	32	32	32	32	32	-
T_r	riser temperature	$^\circ C$	-	-	-	-	-	-	30
M_w	molecular weight of gas	g	20	20	20	20	20	20	20

Chapter 5

A Fast Robust Extremum Seeking Scheme using Transient Measurements

In this chapter, we propose a novel robust extremum seeking scheme, where existing domain knowledge in the form of plant dynamics is fixed in the extremum seeking scheme without the need for rigorous nonlinear models. By fixing the linear dynamics, we can use real-time transient measurements to robustly estimate the local steady-state gradient from the input to the cost, even in the presence of neglected plant dynamics and non-minimum phase elements.

Based on the article submitted to Automatica [87].

5.1 Introduction

This chapter primarily focuses on online optimization methods where the steady-state gradient is directly estimated from the cost measurement without the need for rigorous nonlinear models. Here, the idea is to constantly perturb the system in order to estimate the steady-state gradient. The estimated steady-state cost gradient is then driven to a constant setpoint of zero, thereby satisfying the necessary condition of optimality. One of the attractive properties of extremum seeking methods is that they do not require the development of rigorous nonlinear models (Challenge 1).

The use of extremum seeking control dates back to 1951, with the work of Draper and Li [39]. However it started gaining popularity in the academic community only recently, following the work of Krstić and Wang [107]. Since then, there has been several advancements in extremum seeking (ES) methods including, least square-based ES [70], sliding-mode ES [53, 133], greedy ES [182], discrete-time ES[33], newton-based ES [56], Lie-bracket approximation based ES [40] to name a few. In addition there are other similar methods such as NCO-tracking [51], hill-climing control [108] etc. A common trait among all these approaches is that it involves estimating the cost gradient directly from the measurements and driving

the estimated gradient to zero using a control law (most commonly simple integral action).

However, implementation of these methods are typically based on the assumption that the dynamic plant acts likes a static map between the input and the cost. In order to estimate the steady-state cost gradient, the perturbation signal must be much slower than the plant dynamics (typically a factor of 10), such that the dynamic plant can be approximated as a static map. Furthermore, the integral gain to drive the steady-state gradient to zero must be small enough such that the convergence to the optimum is much slower than the perturbation signal (typically, another factor of 10). In summary, this means that the overall convergence rate is about two orders of magnitude slower than the original plant dynamics [106, 180]. For many dynamic processes, this may lead to prohibitively slow convergence.

Therefore, the main disadvantage of extremum seeking control, is often the slow convergence to the optimum. Despite the very appealing characteristic of not requiring a detailed model, this makes extremum seeking control impractical for real-time optimization of most processes. The concept of extremum seeking control was briefly introduced in Chapter 3 to compare the performance with the feedback RTO approach, where we showed that slow convergence was one of the main issue of the classical extremum seeking approach.

The main reason for the slow convergence of the extremum seeking algorithm is the steady-state wait time, because of the local linear static map assumption used in almost all variants of extremum seeking control. Naively using transient measurements, leads to erroneous steady-state gradient estimation. In order to address this issue, one potential solution is to explicitly include the plant dynamics in the ES scheme. In this chapter, we will focus on using transient measurements with extremum seeking control.

The use of measurements to repeatedly identify a local linear *dynamic* model around the current operating point for online optimization of slow chemical processes was first proposed by Bamberger and Isermann [7] in 1978. We will refer to this as the BI-approach. Here, ARX models were repeatedly identified online for Hammerstein plants, and the input was updated using an adaptation law in the same fashion as in most extremum seeking approaches. This approach was further analyzed by Garcia and Morari [54], where local linear dynamic models were recursively identified to estimate the steady-state gradient, and drive the process to its optimum using a gradient descent algorithm. Later in 1989, McFarlane and Bacon [127] presented an empirical strategy for open-loop online optimization using black-box ARX models similar to the one proposed by Bamberger and Isermann [7] and Garcia and Morari [54].

The BI-approach proposed by Bamberger and Isermann [7] and further studied by Garcia and Morari [54], Golden and Ydstie [57] and McFarlane and Bacon [127] can in fact be seen as a *dynamic* variant of extremum seeking control for Hammerstein plants, where the cost measurements are directly used to estimate the steady-state gradients, which is then driven to zero using integral action. In simulations, local linear dynamic models such as ARX models were shown to be an effective way of taking into account the plant dynamics, enabling fast convergence to the optimum for slow dynamic processes. Incorporating the plant dynamics in the gradient estimation problem, effectively removes the assumption that the plant

behaves like a static map. Therefore, one does not need a time scale separation between the plant dynamics and the excitation frequency, unlike the classical ES control [180]. This leads to significantly faster convergence to the optimum. Local linear dynamic model in the context of extremum seeking control was recently used for Hammerstein plants (see Appendix O), where we showed that local linear dynamic models lead to more than one order of magnitude faster convergence, compared to classical extremum seeking control.

Using transient measurements to experimentally optimize slow dynamic process was proposed even earlier by Sawaragi et al. [155], however this differs from the concept of extremum seeking in the sense that it does not estimate the steady-state gradient from the measurements. Sawaragi et al. [155] evaluates the profit value directly based on online experiments, and use an *if-else* logic to decide on how to update the inputs, as opposed to using the gradient. Besides, the method presented by Sawaragi et al. [155] is very particular to a distillation column, and does not present a generic method, unlike the method proposed by Bamberger and Isermann [7] and Garcia and Morari [54], where the authors present a generic method to estimate the steady-state gradient using an ARX model, and use the gradient to update the control input.

Following the introduction of ARX-based local linear dynamic model to estimate the gradient in the 1980s [7], this approach has remained dormant, despite the recent surge of interest in extremum seeking control[1]. This is probably because of the robustness problems of these methods that impede practical applicability, which will be motivated using a simple example in this chapter.

However, identifying an ARX model online, as opposed to estimating only the steady-state gradient as in classical extremum seeking control, involves fitting additional parameters. Consequently,there is a need for sufficient excitation to accurately estimate all the parameters of the ARX model. As the system approaches its optimum, the steady-state relation between the input and the cost is flat, and there is not sufficient excitation in the measured cost to accurately estimate all the parameters of the ARX model. This may lead to numerical robustness problems in the gradient estimation, causing the control action to respond erratically close to the optimum. Therefore, when identifying ARX models online in [7] and [127], the optimizer was turned off once the plant reached its optimum. Consequently, robustness problems were not reported in these early works. However, in practice, with disturbances and noise, it is very difficult to know when to turn off the identification, and when to activate it again. Indeed, Garcia and Morari [54] acknowledged this problem and suggested to stop estimating the auto-regressive part of the ARX model as the system converges. However, this was not studied in any detail. Also, this method still requires the ARX model orders to be chosen offline.

Estimating all the parameters of the ARX model also leads to robustness problems due to the unmodeled/neglected plant dynamics, that are not captured by the chosen ARX model. In other words, the method proposed by Bamberger and Isermann [7],Garcia and Morari [54] and McFarlane and Bacon [127], may only work when the ARX model structure is able to capture the plant dynamics with suffi-

[1] This may also be partially because control researchers are largely uninformed of developments outside of their own interest areas [153].

cient accuracy. If the plant dynamics are quite different from the chosen ARX model structure, then the steady-state gradient estimated using the ARX model may not be correct, and can lead to stability robustness issues. For example, this may be the case when there are time delays or when there are unstable zeros (inverse response). This makes the problem sensitive to the chosen ARX model structure, which has not been previously studied. Using illustrative counter-examples, we demonstrate the robustness problems of the approach proposed by Bamberger and Isermann [7] (BI-approach).

Hammerstein systems are also popular when studying extremum seeking control, see for e.g. [7, 104, 113, 127, 136, 137, 160] to name a few. However, all these works assume perfect knowledge of the linear dynamics, and none of these works explicitly considers the impact of neglected dynamics on the closed-loop system, such as time delays and inverse responses, which could potentially lead to instabilities and performance degradation. There are also some papers on Hammerstein models, e.g. [160], but from a process control point of view, these are closer to parameter estimation based real-time optimization (RTO) than extremum seeking control.

Hammerstein plants consist of a nonlinear steady-state mapping between the inputs and the cost $h(u)$, followed by linear dynamics $G(s)$, as shown in Fig. 5.1. As opposed to earlier ARX approaches, we propose to fix the linear dynamics based on prior knowledge. We can then use transient measurements to estimate only the unknown nonlinear steady-state component of the process. Note that the proposed method does not require any knowledge of the nonlinear steady-state map, but knowledge of the plant dynamics is required. Since the linear dynamics is assumed to be known and fixed, we use classical robust control theory to provide stability robustness bounds for any neglected or varying plant dynamics.

To this end, the main contribution of this chapter is a novel dynamic extremum seeking scheme, where the plant dynamics are fixed using a linear model in order to efficiently use the transient measurements for the gradient estimation. For practical implementation, this leads to an improved and robust formulation compared to the method proposed by Bamberger and Isermann [7]. In addition, we also provide robust stability bounds on any neglected or varying plant dynamics, which has not been previously studied in the context of extremum seeking control for Hammerstein plants.

The remainder of the chapter is organized as follows. We introduce some preliminaries in Section 5.2. In Section 5.3, we revisit the BI-approach proposed by Bamberger and Isermann [7] and use an illustrative example to motivate the robustness problems of the BI-approach. Our robust dynamic extremum seeking scheme is then introduced in Section 5.5 followed by a stability robustness analysis in Section 5.6. Simulation examples in section 5.7 show the effectiveness of the proposed approach, before concluding the chapter.

5.2 Preliminaries

Consider a nonlinear single-input-single output (SISO) plant, where the objective is to drive the cost J to its minimum by using the input u.

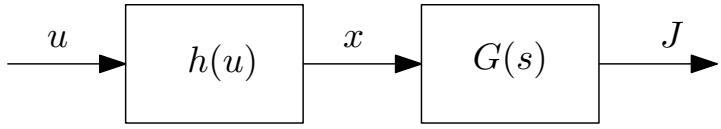


Figure 5.1: Hammerstein process

Assumption 5.1: The plant cost J can be measured.

Assumption 5.2: The effect of the input u on the plant cost J, can be represented as a Hammerstein model, with a combination of a nonlinear time invariant map $h(u) : \mathbb{R} \to \mathbb{R}$, and a proper, stable, finite-dimensional, linear, time-invariant (FDLTI) dynamic system $G(s)$ at its output, see Fig.5.1.

This assumption is often justified, since Hammerstein models are shown to be good model representation for many engineering systems, particularly those where the dominating dynamics are due to slow sensor response [43, 84] or slow input dynamics.

Assumption 5.3: The nonlinear map $h(u)$ is sufficiently smooth and continuously differentiable such that it has a unique minimizer at $u = u^*$

$$\frac{\partial h}{\partial u}(u^*) = 0 \tag{5.1}$$

$$\frac{\partial^2 h}{\partial u^2}(u^*) > 0 \tag{5.2}$$

Assumption 5.3 ensures that $h(u)$ has a unique minimizer at $u = u^*$ and the goal is to drive u to the neighborhood of u^*.

Assumption 5.4: A nominal linear model for the plant dynamics $G_0(s)$ is assumed to be known.

It is reasonable to assume that the user often has information about the nominal plant dynamics $G_0(s)$, for example, obtained using step response tests around some nominal operating conditions.[2] We do not need any *a priori* knowledge of the nonlinear steady-state map $h(u)$. Note that we only need to know the nominal plant dynamics $G_0(s)$ and not the exact plant dynamics $G(s)$. We will later show that, as long as the unmodeled dynamics are bounded, robust stability of the closed-loop system can be ensured.

5.3 ARX-based dynamic extremum seeking control: BI-approach

5.3.1 Controller design used in the BI-approach

In order to use the transient measurements, we use the method proposed by Bamberger and Isermann [7] (BI-approach) in the context of dynamic extremum seek-

[2] Often, a good operator may also be able to guess the dominant time constants of the process

ing scheme, which is based on identifying a local linear *dynamic* model around the current operating point, using transient measurements. The core idea of the BI-approach is that, by repeatedly identifying a locally linear ARX model online, one can essentially use the transient measurement data to estimate the cost gradient. The cost and input measurements from a fixed moving window containing the last N data samples are used to continuously fit an ARX model of the form,

$$\begin{aligned} J(t) = &- a_1 J(t-1) - \cdots - a_{n_a} J(t-n_a) \\ &+ b' u(t-1) + \cdots + b'_{n_b} u(t-n_b) + e(t) \end{aligned} \tag{5.3}$$

Remark 5.1. The input and cost measurements in (5.3) are assumed to be pre-processed, such that they are mean-centered [114].

Remark 5.2. If J_u is the local linear approximation of the steady-state map $h(u)$, then the local linear dynamic system (5.3) along with the nonlinear steady-state effect can be written with $b'_i := J_u b_i$ for all $i = 1, \ldots, n_b$.

Given N data samples of the cost $[J(N), J(N-1), \ldots, J(1)]$ and the input $[u(N), u(N-1), \ldots, u(1)]$, such that $J(N)$ is the latest sample and $J(1)$ is the oldest sample, we estimate the ARX polynomials,

$$\theta = [a_1 \quad \cdots \quad a_{n_a} \quad b'_1 \quad \cdots \quad b'_{n_b}]^\mathsf{T} \tag{5.4}$$

using linear least squares estimation

$$\hat{\theta} = arg \min_{\theta} \|\psi - \Phi\theta\|_2^2 \tag{5.5}$$

where ψ is given by the expression[3]

$$\psi = \begin{bmatrix} J(N) & J(N-1) & \ldots & J(n+1) \end{bmatrix}^\mathsf{T} \tag{5.6}$$

and Φ is given by the expression

$$\Phi = \left[\begin{array}{ccc|ccc} -J(N-1) & \ldots & -J(N-1-n_a) & u(N-1) & \ldots & u(N-1-n_b) \\ -J(N-2) & \ldots & -J(N-2-n_a) & u(N-2) & \ldots & u(N-2-n_b) \\ \vdots & & \vdots & \vdots & & \vdots \\ -J(n) & \ldots & -J(n+1-n_a) & u(n) & \ldots & u(n+1-n_b) \end{array}\right] \tag{5.7}$$

Introducing the notation

$$A_{poly}(q) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}$$

and

$$B_{poly}(q) = b' q^{-1} + \cdots + b'_{n_b} q^{-n_b}$$

with q^{-1} being the unit delay operator, we get a local linear dynamic model of the form,

$$J(t) = \frac{B_{poly}(q)}{A_{poly}(q)} u(t) \tag{5.8}$$

[3] $n = \max(n_a, n_b)$

The steady-state gradient around the current operating point can then be estimated by,

$$J_u = A_{poly}(q)^{-1} B_{poly}(q) \tag{5.9}$$

Alternatively, the identified ARX polynomials $A_{poly}(q)$ and $B_{poly}(q)$ can be converted to continuous time state-space system[4] as shown below,

$$\begin{aligned} \dot{x} &= Ax + Bu \\ J &= Cx + Du \end{aligned} \tag{5.10}$$

The steady-state gain is then given by setting $\dot{x} = 0$ and eliminating the states x in (5.10),

$$J = \underbrace{\left(-CA^{-1}B + D\right)}_{\hat{\mathbf{J}}_\mathbf{u}} u \tag{5.11}$$

which yields the same expression as in the feedback RTO approach in Chapter 3 (cf. equation (3.7)).

Once the steady-state gradient $\hat{\mathbf{J}}_\mathbf{u}$ is estimated, a simple integral action can be used to drive the system to its extremum

$$u = \frac{K_I}{s} J_u \tag{5.12}$$

where K_I is the integral gain, and T_s is the sample time. Additional perturbation ω, such as a pseudo random binary sequence (PRBS) signal is added to the input to provide sufficient excitation.

$$u(t+1) = \hat{u}(t+1) + \omega$$

5.4 Motivating example

Consider the Hammerstein system with $G(s) = \frac{1}{(\tau_1 s+1)}$, $h(u) = -0.1u^2+4u+5$ with $u \in \mathbb{R}$ and $\tau_1 = 174$s. This has an optimum at $u^* = 20$. We consider a simulation case with no uncertainty in the ARX model structure, where we repeatedly identify a first-order ARX model of the form

$$J(t) = -a_1 J(t-1) + b_1 u(t-1)$$

with sampling time T_s. Note that, in this case, we need to estimate two ARX model parameters, corresponding to the time constant and the steady-state gain, given by,

$$J_u = \frac{b_1}{1+a_1}, \qquad \tau = \frac{-T_s}{ln(-a_1)}$$

which is just one more than for the classical extremum seeking control.

[4] for example using `idss` and `d2c` command in `MATLAB`. Although one does not need the state-space formulation, we simply show this here to correlate with the gradient estimation scheme from Chapter 3.

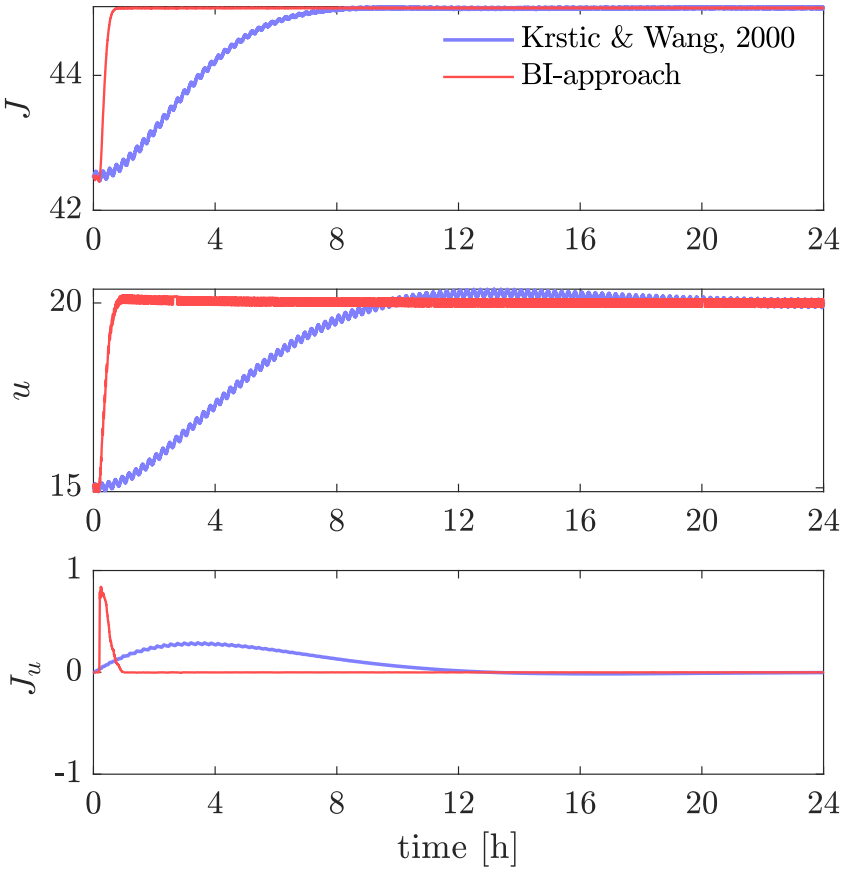


Figure 5.2: Example 1: Simulation results comparing the performance of the BI-approach [7] with classical extremum seeking control from [107].

Fig. 5.2 shows the BI-approach compared to classical extremum seeking control, which shows that by identifying a local linear dynamic model, the problem converges significantly faster than the classical extremum seeking scheme from [107]. Using the classical extremum seeking approach, a process with a settling time of approximately 3 min, takes about 12 to 16 hours to converge, whereas by using the BI-approach, the process converges within 1 hour. This significant performance improvement is possible due to the inclusion of the process dynamics.

However, the BI-approach has robustness problems, and may work only under specific conditions. Before presenting our proposed robust approach, we first motivate the robustness problems when using the BI-approach. Identifying an ARX model online involves estimating all the parameters of A_{poly} and B_{poly}. If the excitation of the process is not sufficient, then all the $(n_a + n_b)$ ARX model parameters

may not be estimated accurately. In the context of extremum seeking control, this is especially a problem as the system approaches its optimum, where the steady-state relation between the input and the cost $h(u)$ is typically flat.

Fig. 5.3 shows the same simulation results as in Fig. 5.2, for a period of two hours. The first subplot shows the cost function, the second subplot shows the control input, the third subplot shows the estimated steady-state cost gradient and the last subplot shows the estimated time constant of the identified ARX model. From the first three subplots, it seems like the BI-approach is working well and the system converges to the optimum, similar to the results by Bamberger and Isermann [7], McFarlane and Bacon [127]. However, from the last subplot we see that the estimated time constant is converging to zero, whereas the true value is 174s. This is due to the steady-state gain from input to the cost approaching zero, which leads to rank deficiency of the information matrix close to the optimum, as pointed out by Garcia and Morari [54]. This may also lead to numerical bursting and instability.

As mentioned earlier, when identifying ARX models online in [7, 127], the optimizer was simply turned off once the plant reached its optimum. However, in practice, it is desirable to keep estimating the gradient even after reaching the optimum. This is to ensure that the system is driven to its new optimum if there are disturbances. As suggested in [54], the problem may be partially solved by not estimating the autoregressive part (as in our case) as the system converges to the optimum. However, it still would not resolve the issue of unmodeled plant dynamics not considered in the chosen ARX model.

This is demonstrated using the same example as above, where we repeatedly identify the same first order ARX model of the form $J(t) = -a_1 J(t-1) + b_1 u(t-1)$ based on the nominal dynamics of $G_0(s) = 1/(174s + 1)$. However, we now add additional inverse response dynamics, such that the plant is given by

$$G(s) = G_0(s)\frac{(-5s+1)}{(10s+1)} \tag{5.13}$$

The ARX model structure is first order, which cannot include the inverse response dynamics in (5.13). Thus, the ARX identification leads to erroneous gradient estimation. This is shown in Fig. 5.4, where we see also from the first three subplots that the closed-loop system behaves erratically due to the neglected inverse response, leading to poor performance.

To summarize, the BI-approach proposed by Bamberger and Isermann [7] may fail when,

- the process approaches its optimum and the excitation is not sufficient to estimate all the ARX model parameters accurately,
- the process includes neglected dynamics not captured by the chosen ARX model.

5.5 Proposed approach

In order to address the robustness problems motivated above, we propose to fix the parameters of plant dynamics $G_0(s)$ in the estimation problem. This is illustrated

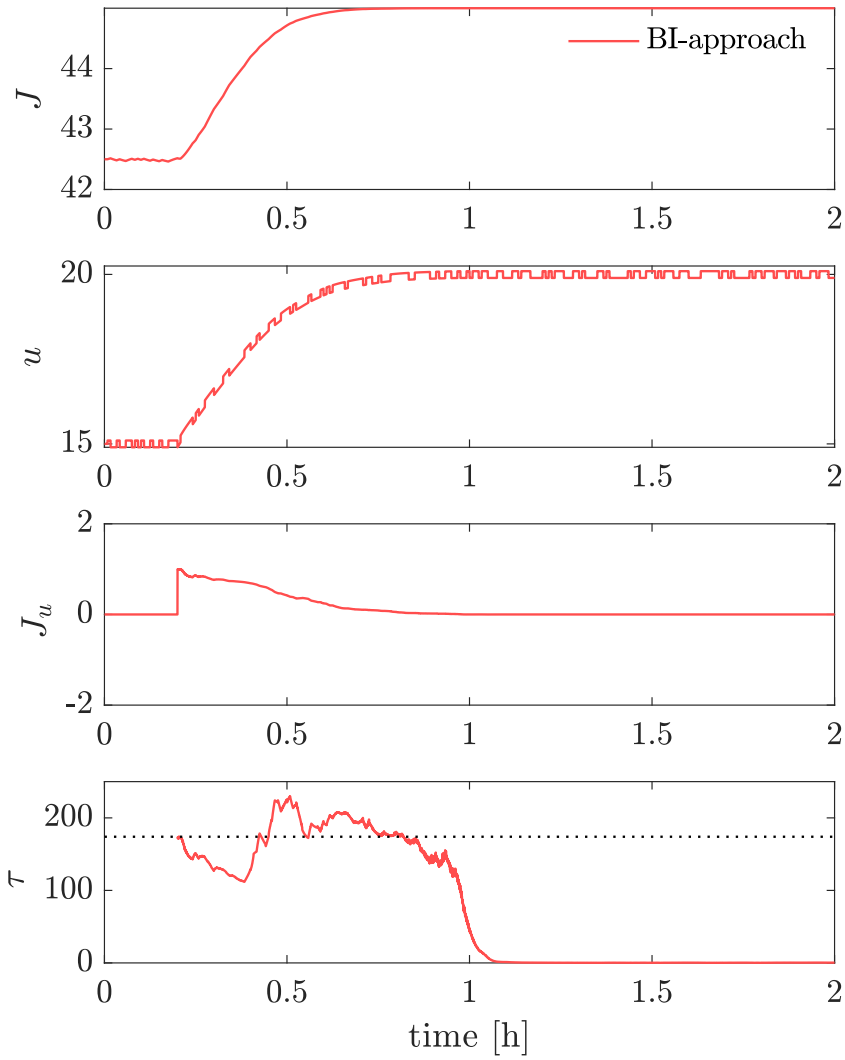


Figure 5.3: Example 1: First 2 hours from Fig. 5.2 showing the robustness issues with the BI-approach when identifying a first order ARX model from [7] even with no uncertainty in the plant dynamics $G_0(s)$.

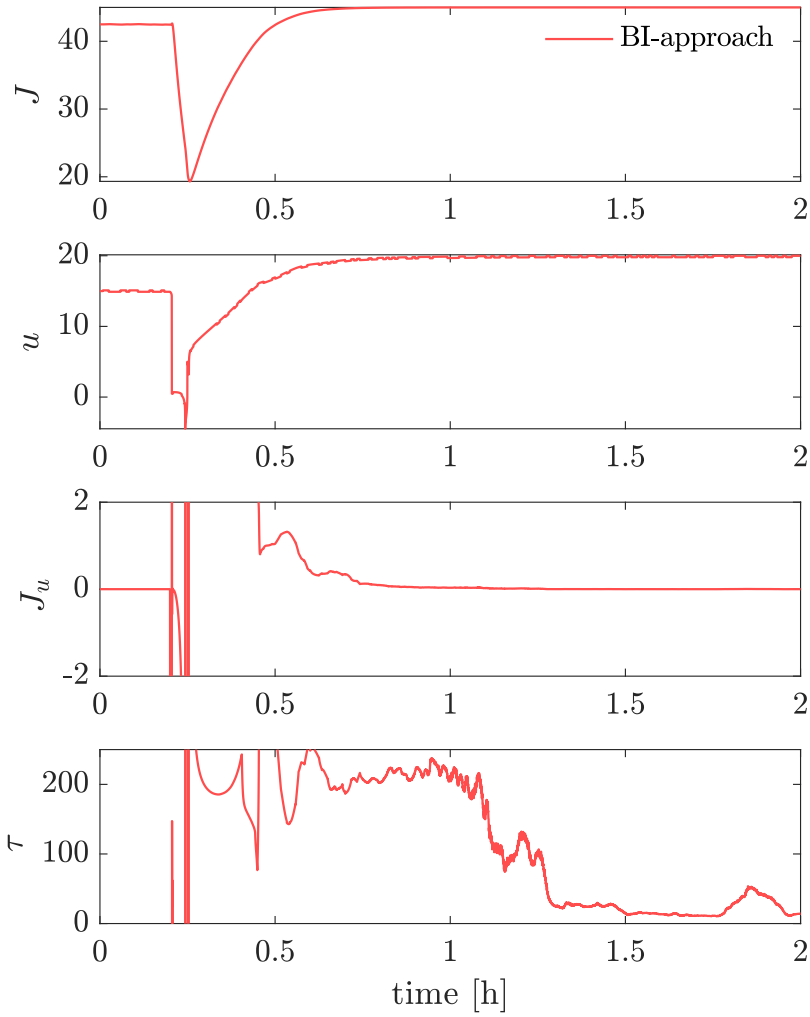


Figure 5.4: Example 1: BI-approach with neglected RHP-zero. Simulation results showing the robustness issues when identifying a first order ARX model from [7] with neglected inverse response dynamics in the process.

below in Fig. 5.6 and Fig. 5.7, which show a large improvement compared to the simulations in Fig. 5.3 and Fig. 5.4, respectively.

In our proposed approach, the online measured data is used to estimate only the local steady-state gain. In other words, if domain knowledge in the form of nominal process dynamics $G_0(s)$ is known a-priori (e.g. from step response tests), we can incorporate this knowledge in the online identification problem to estimate the steady-state effect of the process $h(\mathbf{u})$, around the current operating point. This way, we can incorporate the "known" system dynamics in the online identification problem, which enables us to use the transient measurements to estimate the steady-state gradient more accurately. Since the nominal linear dynamics $G_0(s)$ is known and fixed, we can then provide robustness margins for the neglected dynamics in $G(s)$ using classical robust control theory.

We first transform the nominal linear dynamics $G_0(s)$ (with unit gain) along with the local steady-state gain J_u into discrete-time of the form,

$$J = G_0(q)h(u) = J_u \underbrace{\left(\frac{b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b}}{1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}} \right)}_{\text{fixed}} u \tag{5.14}$$

Note that, here the ARX model is written with $b := b'/J_u$ (cf. Remark 5.2). For the known nominal ARX model of the form (5.14), we can then fix the parameters $[a_1, \ldots, a_{n_a}, b_1, \ldots, b_{n_b}]$ and estimate only the steady-state gain J_u of the system around the current operating point. Thus, in our proposed method, we solve the least square estimation problem

$$\hat{\theta} = arg \min_{\theta} \|\psi - \Phi\theta\|_2^2 \tag{5.15}$$

with

$$\theta = J_u \tag{5.16}$$

$$\psi = \begin{bmatrix} J(N) + a_1 J(N-1) + \cdots + a_{n_a} J(N-n_a) \\ J(N-1) + a_1 J(N-2) + \cdots + a_{n_a} J(N-1-n_a) \\ \vdots \\ J(n+1) + a_1 J(n+2) + \cdots + a_{n_a} J(n+1-n_a) \end{bmatrix} \tag{5.17}$$

and

$$\Phi = \begin{bmatrix} b_1 u(N-1) + \cdots + b_{n_b} u(N-n_b) \\ b_1 u(N-2) + \cdots + b_{n_b} u(N-1-n_b) \\ \vdots \\ b_1 u(n) + \cdots + b_{n_b} u(n-1-n_b) \end{bmatrix} \tag{5.18}$$

over a fixed moving window of the past N samples[5], the solution to which is given by $\hat{\theta} = (\Phi^\mathsf{T}\Phi)^{-1}\Phi^\mathsf{T}\psi$. Notice that θ is now a scalar as opposed to (5.4).

The control input is then updated using a simple integral action with gain K_I to drive the estimated cost gradient J_u to zero.

$$u = \frac{K_I}{s} J_u \tag{5.19}$$

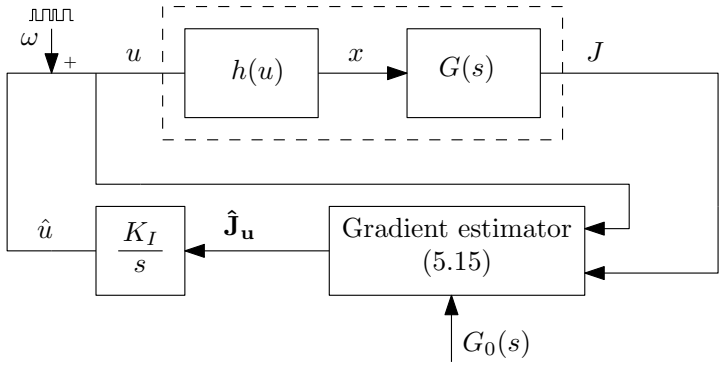


Figure 5.5: Schematic representation of the proposed robust dynamic extremum seeking approach.

For the persistence of excitation, a pseudo random binary sequence (PRBS), ω is added as in Fig. 5.5, which shows the schematic representation of the proposed approach, where we fix the plant dynamics $G_0(s)$ and only estimate the steady-state gradient J_u using the transient measurements. By incorporating the domain knowledge in terms of the process dynamics, we can reduce the number of parameters to be estimated to one (i.e $J_u \in \mathbb{R}$), which makes the estimation problem more numerically robust. Note that, here we have used a fixed moving window of the past N samples in the same fashion as a in [70]. Alternate formulations using recursive least square may also be used.

The proposed method is an improvement of the approach suggested by Garcia and Morari [54], where the autoregressive part of the ARX model $[a_1, \ldots, a_{n_a}]$ was fixed, as the system converges, and only the parameters $[J_u b_1, \ldots, J_u b_{n_b}]$ were estimated online. In our proposed approach, however, we fix also $[b_1, \ldots, b_{n_b}]$ at all times and estimate only J_u online.

Consider, the same motivating example as in Section 5.4, where we fix $G_0(s) = 1/(174s + 1)$, which is equivalent to $a_1 = -0.9943$ and $b_1 = 0.005731$. Fig. 5.6 shows the simulation results obtained with no model error, i.e. with $G_0(s) = G(s)$. The simulation results show that by fixing the plant dynamics and estimating only the local steady-state gain, the process converges to the optimum with the same speed of convergence as the BI-approach.

In the case of additional inverse response dynamics, Fig. 5.7 shows simulation results using the proposed approach, compared to the BI-approach. It can be seen clearly that the proposed approach is robust to neglected plant dynamics.

5.6 Robust stability

We saw in the simulation example in Fig. 5.7 that the proposed approach is robust to neglected plant dynamics. In this section, we explain this using robust

[5] $n = \max(n_a, n_b)$. $J(N)$ is the latest sample and $J(1)$ is the oldest sample.

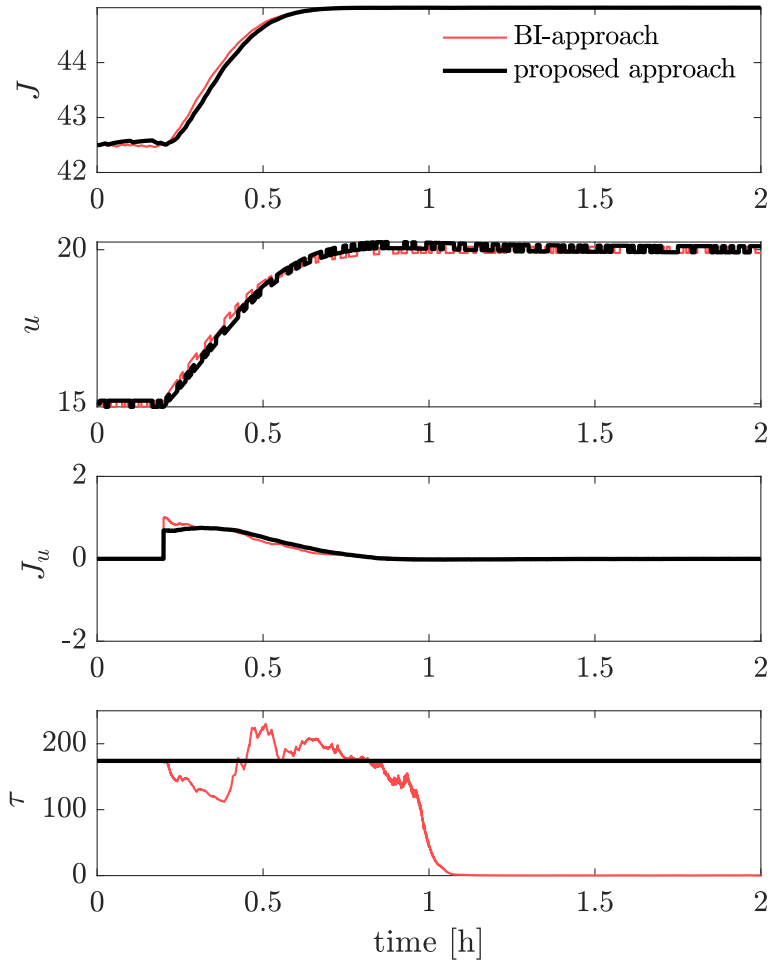


Figure 5.6: Example 1: Proposed method with no uncertainty. Simulation results for the dynamic extremum seeking scheme (black) compared with the BI-approach[7] (red).

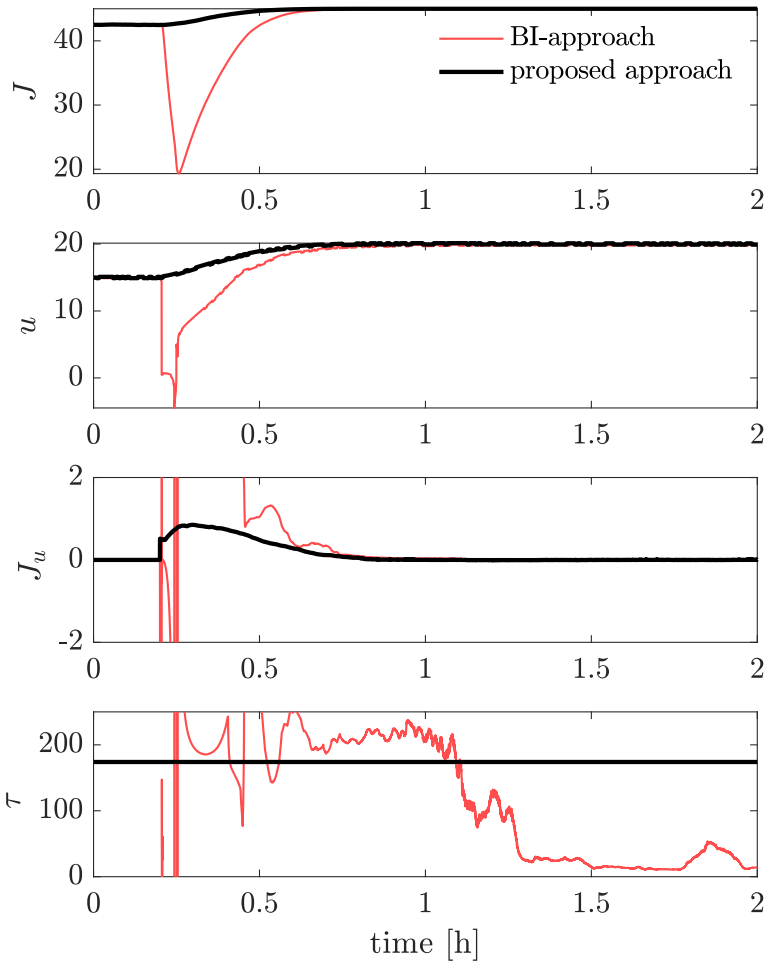


Figure 5.7: Example 1: Proposed method with neglected RHP-zero: Simulation results using a nominal dynamics of $G_0(s) = \frac{1}{(174s+1)}$ for a process with dynamics $G_p(s) = G_0(s)\frac{(-5s+1)}{(10s+1)}$ using the proposed method (black) compared with the BI-approach [7] from Fig. 5.4 (red).

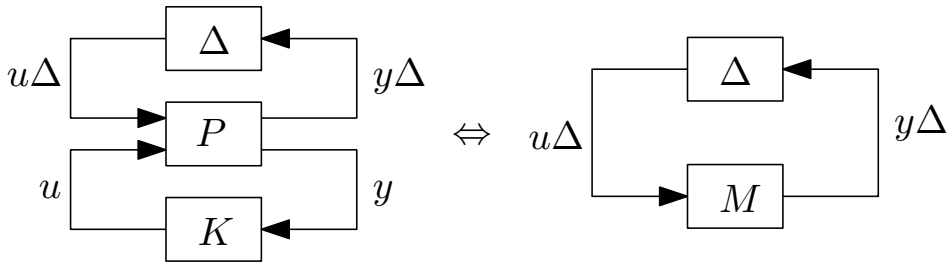


Figure 5.8: Generalized $M\Delta$ structure.

control theory, and provide robust stability margins for neglected dynamics for a generalized case.

In practice, the actual plant dynamics $G(s)$ are not the same as $G_0(s)$ fixed in the steady-state gain estimator (5.14) and (5.15). There could be many reasons for this, including neglected dynamics and changes due to nonlinearities. In this section, we will show that robust stability of the proposed gray-box approach can be ensured (at least locally), as long as the neglected dynamics are bounded.

To analyze the robust stability, let us consider the general method for formulating control problems [38] as shown in Fig. 5.8, where

$$P := \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}$$

denotes the generalized plant and $K(s)$ is the controller, which in our case is selected as an integral controller $K(s) := \frac{K_I}{s}$. Δ represents the uncertainty, which should be less than 1 in magnitude at all frequencies. The plant dynamics $G(s)$ is given by upper linear fractional transformation (LFT)

$$G := P_{22} + P_{21}\Delta(I - P_{11}\Delta)^{-1}P_{12} \tag{5.20}$$

It can be seen that when there is no uncertainty, $\Delta = 0$ and hence $P_{22} = G_0$. Using the proposed dynamic extremum seeking scheme, the nominal loop transfer function around the current operating point can be represented as,

$$L_0 := J_u G_0 K \tag{5.21}$$

where J_u represents the local steady-state gain of the system.

To analyze the effect of the uncertainty on the stability, we consider a generic $M\Delta$ structure as shown in Fig. 5.8 [168], where M is given by the lower LFT,

$$M := P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21} \tag{5.22}$$

The robust stability in the generalized case is given by the following theorem.

Theorem 5.1 (Bounds on neglected dynamics)**.** *[168] Consider the dynamic extremum seeking scheme using the fixed nominal linear dynamics $G_0(s)$ and an integral control action $K(s)$. Assuming the nominal closed-loop system and the*

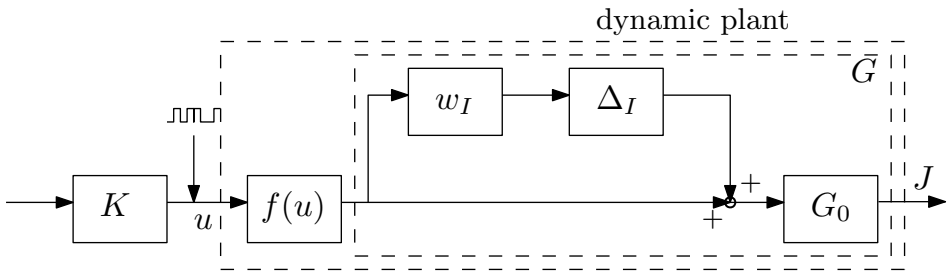


Figure 5.9: Unmodeled dynamics represented as multiplicative uncertainty.

perturbation Δ are stable, the closed-loop system including the unmodeled plant dynamics is then robustly stable in the neighborhood of the current operating point for any plant dynamics with $\|\Delta\|_\infty \leq 1$, as long as the unmodeled dynamics are bounded by

$$|M| < 1, \quad \forall\omega \tag{5.23}$$

Proof. The proof is obtained by applying the Nyquist stability condition, which says that $M\Delta$ must not encircle -1 for all Δ. Thus,

$$\text{RS} \Leftrightarrow |1 + M\Delta| > 0, \quad \forall|\Delta| \leq 1, \ \forall\omega \tag{5.24}$$

The worst-case scenario for violating this condition is when $|\Delta| = 1$, and the terms $|M\Delta|$ and 1, point in the opposite direction. Thus,

$$\text{RS} \Leftrightarrow 1 - |M| > 0 \tag{5.25}$$

$$\Leftrightarrow |M| < 1, \quad \forall\omega \tag{5.26}$$

□

The exact structure of M depends on how the uncertainty is modeled [168]. For example, in the case of additive uncertainty,

$$P := \begin{bmatrix} 0 & 1 \\ w_A & G_0 \end{bmatrix}, \qquad M := w_A K(1 + G_0 K)^{-1} \tag{5.27}$$

for multiplicative uncertainty

$$P := \begin{bmatrix} 0 & 1 \\ w_I G_0 & G_0 \end{bmatrix}, \qquad M := w_I G_0 K(1 + G_0 K)^{-1} \tag{5.28}$$

and for inverse multiplicative uncertainty

$$P := \begin{bmatrix} w_{iI} & 1 \\ w_{iI} G_0 & G_0 \end{bmatrix}, \qquad M := w_{iI}(1 + G_0 K)^{-1} \tag{5.29}$$

In general, various sources of dynamic uncertainty and neglected dynamics can often be represented as multiplicative uncertainty. Therefore, we choose to use this form with $G(s) = G_0(s)\tilde{G}(s)$, and the neglected dynamics can be written as

$$\tilde{G}(s) = (1 + w_I(s)\Delta_I(s)) \tag{5.30}$$

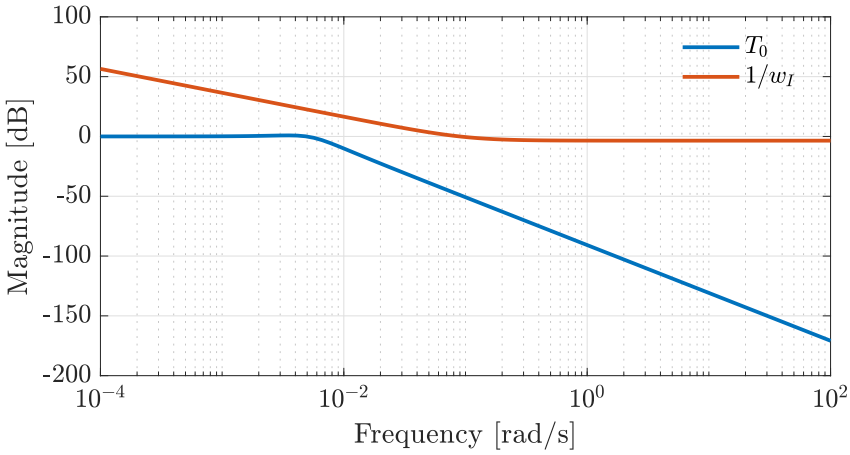


Figure 5.10: Example 1: Checking robust stability condition (5.31) with neglected RHP zero. T_0 is based on G_0 and w_I represents the multiplicative (relative) difference between G_0 and G in (5.13).

where $|\Delta_I(j\omega)| \leq 1 \forall \omega$, represents complex perturbations, and w_I represents the multiplicative weights. The local linear system with multiplicative uncertainty is shown in Fig. 5.9.

For multiplicative uncertainty, we get $M = w_I T_0$, which leads to the following corollary of Theorem 5.1.

Corollary 5.2 (Bounds on multiplicative uncertainty)**.** *Consider the dynamic extremum seeking scheme using the nominal linear dynamics $G_0(s)$ and an integral control action $K(s) = \frac{K_I}{s}$. The closed-loop system is then robustly stable in the neighborhood of the current operating point for any plant dynamics, as long as the unmodeled dynamics are bounded by*

$$|T_0| < \frac{1}{|w_I|}, \quad \forall \omega \tag{5.31}$$

where $T_0 = (1 + L_0)^{-1} L_0$ is the complementary sensitivity function of the locally linear nominal system and $|w_I| := |1 - \tilde{G}|$.

Proof. Substituting M in Theorem 5.1 with (5.28) yields $|T_0| < \frac{1}{|w_I|}$. □

5.6.1 Simulation 1 revisited

This explains why our proposed approach is able to converge despite the neglected unstable zero in Fig. 5.7. The neglected dynamics can be represented by multiplicative uncertainty with weight

$$w_I(j\omega) = \left| \frac{G(j\omega) - G_0(j\omega)}{G_0(j\omega)} \right| = \left| 1 - \frac{(-5j\omega + 1)}{(10j\omega + 1)} \right|$$

With the same integral controller as before, the magnitude of the nominal complementary sensitivity function $|T_0|$ is shown in Fig. 5.10, together with the magnitude of $\frac{1}{|w_I|}$. The figure confirms that the bound $|T_0| < \frac{1}{|w_I|}$ according to Theorem 5.1 holds for all frequencies with good margins. Hence the closed-loop response of the proposed extremum seeking scheme by fixing the dynamics with $G_0(s) = 1/(174s + 1)$ is locally robustly stable. The simulation results in Fig. 5.7 confirms this.

5.7 Simulation example 2

In this section, we further demonstrate the proposed ES scheme using another simulation example from [160], which is based on the application of extremum seeking to dampen large pressure oscillations in a lean burn combustor [8]. In this example, we see clearly the robustness problems like numerical bursting with the BI-approach, which coincidentally, was not seen in example 1. In addition, we will also see the effect of pole uncertainty, neglected time delay uncertainty and the case where neglected inverse response leads to divergence from the optimum when using the BI-approach and how these can be addressed using the proposed approach.

The plant is described as a Hammerstein system with the nonlinear steady-state effect described by,

$$h(u) = d + \sum_{i=1}^{3} p_{2i-1} \sin(iu) + p_{2i} \cos(iu) \tag{5.32}$$

and the nominal dynamics given by,

$$G_0(s) = \frac{1}{(s/b + 1)} \tag{5.33}$$

The parameter vector $p = [p_1, \ldots, p_6]$ is equal to

$$[14.16, -20.29, 8.214, -3.017, 2.992, 0.787] \times 10^{-3}$$

and $b = 4$. At steady-state, minimum cost occurs when $u^* = -0.61$rad [160]. Note that in [160], the authors assumed that the model structure $h(u)$ and the nominal dynamics (5.33) were known, and only the parameters p were unknown. However, in this chapter, we assume that we have no knowledge about $h(u)$ and only the nominal dynamics (5.33) is known. The system was perturbed with a PRBS signal every 0.04s with amplitude $a = 0.2$ (as opposed to 0.4 in [160]).

5.7.1 No uncertainty

In the first simulation, we assume the plant dynamics and the nominal dynamics are the same (i.e. no neglected plant dynamics). Fig. 5.11 shows the simulation results using the BI-approach and the proposed robust dynamic extremum seeking approach. It can be clearly seen that the proposed scheme is able to quickly drive the process to its optimum, and stay there without causing numerical bursting. The BI-approach identifies the full ARX as proposed in [7, 127]. We see that both

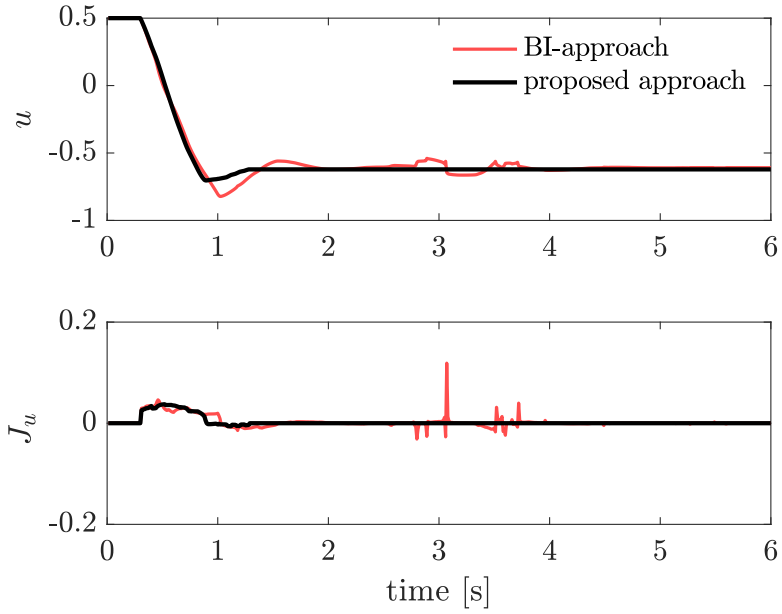


Figure 5.11: Example 2 with no uncertainty: Optimizer output using the proposed scheme (black), compared to the BI-approach [7] (red)

our proposed approach and the BI-approach converges to the optimum. However, unlike in our proposed approach, the estimated gradient in the BI approach has numerical bursting after the process has converges, which leads to unnecessary changes in the control input.

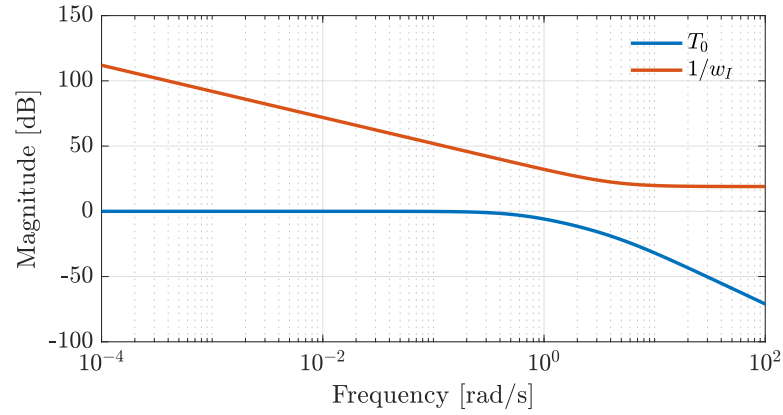
5.7.2 Pole uncertainty

In [160], the authors assumed that the decay rate b was constant. However, to demonstrate the robustness of the proposed method, we now simulate the case where the decay rate changes from $b = 4$ to $b = 4.45$. The robust stability condition in Theorem 5.1 is verified as shown in Fig. 5.12a for the pole uncertainty. Fig. 5.12b shows the simulation results of the proposed method which are compared to the BI-approach. Again, it can be clearly seen that, despite the pole uncertainty, the proposed robust dynamics extremum scheme behaves much better than the BI-approach.

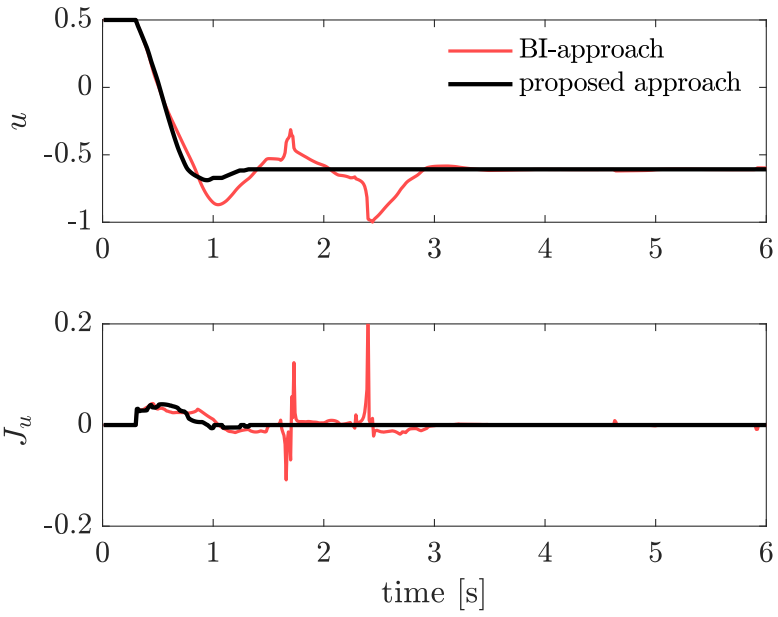
5.7.3 Time delay uncertainty

We then simulate a case where we add a time delay, that is not considered in the nominal dynamics (in our approach and in the BI-approach). The plant dynamics in this case is given by

$$G(s) = G_0(s)e^{-0.01s}$$

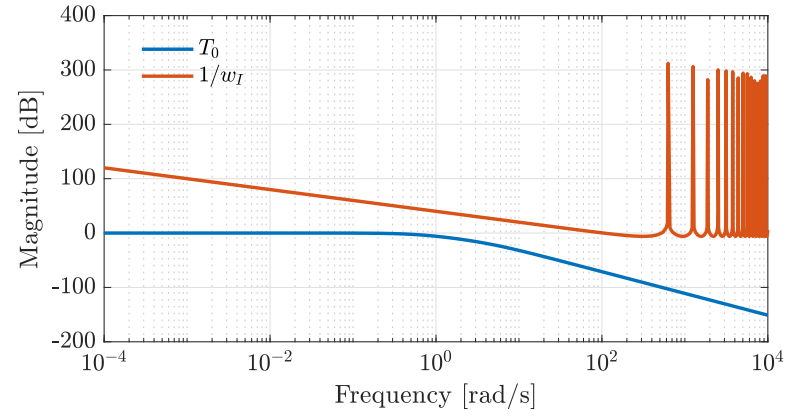


(a)

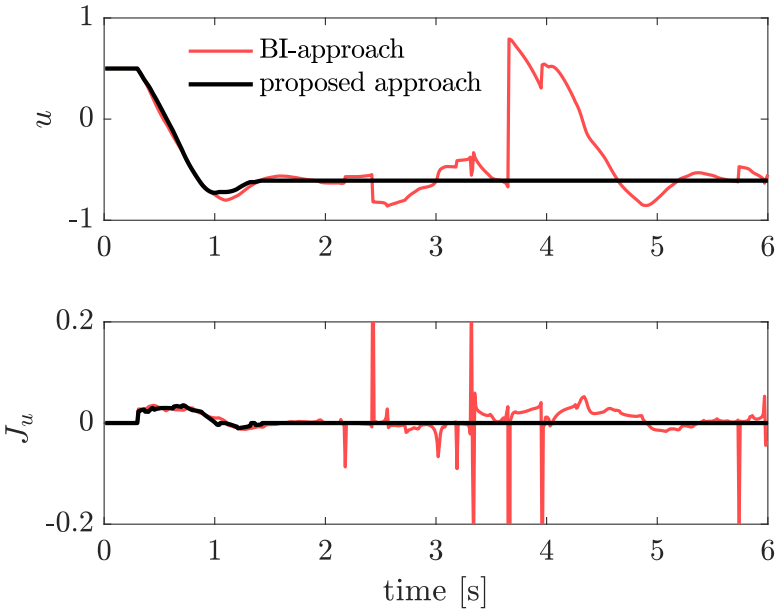


(b)

Figure 5.12: Example 2 with pole uncertainty: (a) Checking robust stability with pole uncertainty. (b) Optimizer output using the proposed scheme (black), compared to the BI-approach [7] (red) for pole uncertainty.

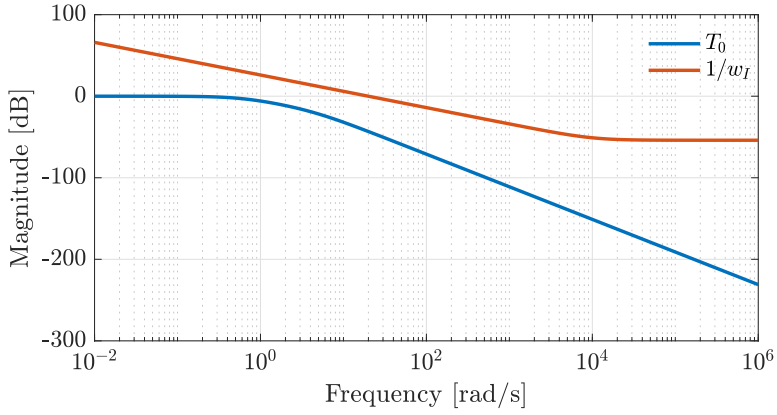


(a)

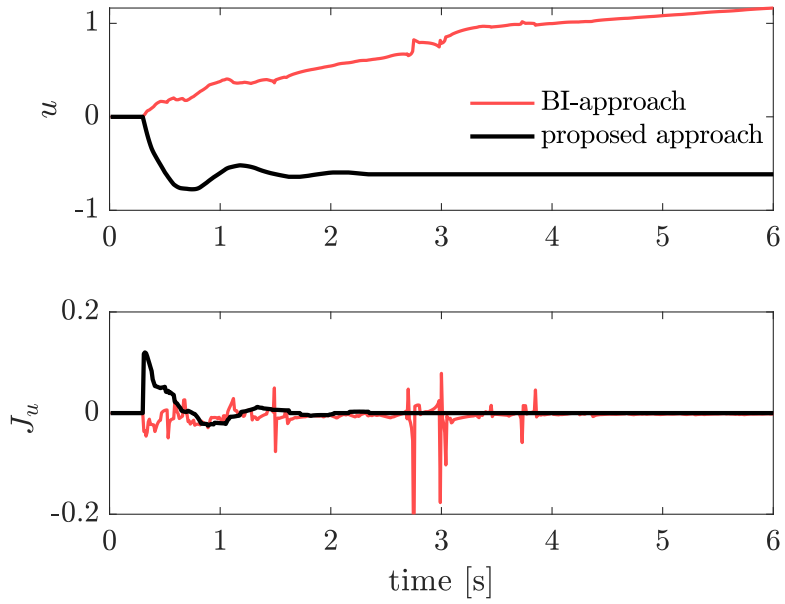


(b)

Figure 5.13: Example 2 with time delay uncertainty: (a) Checking robust stability with time delay uncertainty. (b)Optimizer output using the proposed scheme (black), compared to the BI-approach [7] (red) for neglected time delay.



(a)



(b)

Figure 5.14: Example 2 with neglected RHP-zero and lag: (a) Checking robust stability with neglected RHP zero and lag. (b)Optimizer output using the proposed scheme (black), compared to the BI-approach [7] (red) in the presence of neglected inverse response due to RHP-zero.

The robust stability condition in Theorem 5.1 is verified again, as shown in Fig. 5.13a. The simulation results using our proposed method and the BI-approach are shown in Fig. 5.13b. It can be clearly seen that our proposed method is able to drive the process to its optimum in a stable manner depsite the neglected time delay. Whereas in the BI-approach, the performance degrades drastically around the optimum due to the neglected time delay.

5.7.4 RHP-zero uncertainty

We then simulate a case, where we add a neglected RHP-zero, that is not considered in the nominal dynamics (in our approach and in the BI-approach). The plant dynamics in this case is given by,

$$G_p(s) = G_0(s)\frac{-0.05s+1}{0.0001s+1}$$

The robust stability condition in Theorem 5.1 is verified again, as shown in Fig. 5.14a. The simulation results using our proposed method and the BI-approach are shown in Fig.5.14b. It can be clearly seen that our proposed extremum scheme is able to drive the process to its optimum robustly despite the neglected inverse response. On the other hand, the BI-approach further deviates away from the optimum, clearly failing to reach the optimum.

5.8 Chapter summary

In this chapter, we proposed a novel robust dynamic extremum seeking scheme for Hammerstein plants, where we proposed to fix the plant dynamics in the gradient estimation problem. This removes the need for the assumption, that the plant behaves like a static map. This enables us to use the transient measurements for the gradient estimation.

Using a motivating example, we showed that identifying ARX models in order to use the transient measurements as proposed by Bamberger and Isermann [7], may lead to robustness problems as the process converges, or due to neglected plant dynamics.

To address the robustness problems, we presented an improved approach to incorporate the plant dynamics. Furthermore, we also provided bounds on any neglected dynamics to ensure robust stability of the closed-loop system. The proposed scheme was demonstrated using two examples. Compared to the classical extremum seeking [107], the proposed approach was shown to converge significantly faster to the optimum, and compared to the BI-approach, the proposed scheme was shown to be robust to neglected plant dynamics, such as time delay and inverse responses induced by unstable zeros.

Chapter 6

On Combining Self-optimizing Control and Extremum Seeking Control in the context of Real-time Optimization

This chapter presents a hierarchical combination of self-optimizing control (SOC) and extremum seeking control (ESC) that utilize the advantages of ESC and SOC to handle disturbances in both slow and fast time scales. This chapter shows that extremum seeking control and self-optimizing control are complementary rather than competing.

Based on the article published in the Journal of Process Control [176].

6.1 Introduction

We have so far considered the use the steady-state gradient as the ideal self-optimizing variable in the presence of unconstrained degrees of freedom. In this chapter we will consider the linear measurement combination as a self-optimizing control variable [165] and show how this can be combined with model-free methods such as extremum seeking control. The different methods work in different timescales and handle different types of uncertainty. In this chapter we will show that the two approaches are in fact complimentary and not contradictory.

Self-optimizing control deals with using the model offline to find an appropriate set of controlled variables. The objective is to translate economic objectives into control objectives [165]. This may eliminate the need for an expensive online optimizer to re-optimize when disturbances occur or at least the re-optimization may be performed less frequently. One of the main advantages of self-optimizing control is that it moves the slower economic optimization into the faster control layer. It can also handle unmeasured disturbances as long as they have been modeled. This is achieved by selecting an optimal combination of measurements, which when held at a constant setpoint, gives acceptable loss in the presence of varying disturbances. However, unmodeled disturbances will generally result in a loss.

The self-optimizing control and extremum seeking control methods have been developed relatively independently since 2000. Jäschke and Skogestad [72] successfully combined self-optimizing control with NCO-tracking in a hierarchical structure and demonstrated that the measurement based optimization techniques and the model based self-optimizing concepts are complementary. However, the gradient was estimated using finite differences which gave relatively poor NCO-tracking. The authors suggested that more advanced gradient estimation and input adaptation methods may give a better overall performance as a future research direction. The use of extremum seeking control on top of self-optimizing control was briefly discussed in [81] using the classical extremum seeking method. However, the authors only considered measured disturbances for a single-input single output system. Additionally, based on the simulation results presented, Keating and Alleyne [81] did not consider a clear time scale separation between the extremum seeking and self-optimizing controllers.

In this chapter, we extend the work from [72] and [81] and provide a detailed description on the combination of the model-free extremum seeking controller with model-based self-optimizing control. We propose to use an improved gradient estimation method using least squares estimation and provide a framework for a multivariable system. We then apply the proposed control structure to a multivariable ammonia reactor case study with 3 control inputs and consider both unmeasured and unmodeled disturbances. We compare the performance of the proposed method with just the self-optimizing control and just the extremum seeking control implemented independently and demonstrate clear performance improvements due to the time-scale separation between the extremum seeking and self-optimizing controllers.

6.2 Background

Consider the process where the steady-state optimal operation of the process can be formulated as (4.1), along with Assumptions 5.1 and 5.2.

6.2.1 Self-optimizing control

Self-optimizing control is a strategy for selecting optimal measurement combination $\mathbf{c}$ as controlled variables, such that the impact of known but unmeasured disturbances $\mathbf{d}$ on the optimal operation is minimized. This is achieved by using the system model offline to compute an optimal measurement combination. The ideal self-optimizing variable would be the gradient $\mathbf{J_u}$ which should be controlled to a constant setpoint of 0. However, in most applications, the gradient cannot be measured. An alternative is to identify a controlled variable $\mathbf{c} \in \mathbb{R}^{n_c}$ (with $n_c = n_u$) as a function of the available measurements $\mathbf{y} \in \mathbb{R}^{n_y}$. The simplest approach to select a linear combination of measurements is given by,

$$\mathbf{c} = \mathbf{H}\mathbf{y}_m \tag{6.1}$$

where, $\mathbf{y}_m = \mathbf{y} + \mathbf{n}^y$ is the vector of available measurements, which generally is corrupted by measurement noise $\mathbf{n}^y$, and $\mathbf{H} \in \mathbb{R}^{n_c \times n_y}$ is the measurement combination or selection matrix. In addition to finding $\mathbf{H}$, we must also decide on the

setpoint $\mathbf{c}_s$ which is typically chosen as the nominal optimal value, $\mathbf{c}_s = \mathbf{H}\mathbf{y}_0^{opt}$, where the subscript 0 denotes the nominal operation point with $\mathbf{d} = \mathbf{d}_0$.

Several approaches can be used to calculate the optimal measurement combination $\mathbf{c} = \mathbf{H}\mathbf{y}$. The reader is referred to the recent review paper by Jäschke et al. [75] for a comprehensive review on this subject. Most approaches are based on local linearization around the nominal optimal point. In this chapter, we consider the exact local method as introduced by Halvorsen et al. [62] and further developed by Alstad et al. [3] and Yelchuru and Skogestad [188]. In this method, the optimization problem (4.1) is approximated by a quadratic approximation and a linearized measurement model. Let the linearized measurement model be represented by,

$$\mathbf{y} = \mathbf{G}^{\mathbf{y}}\mathbf{u} + \mathbf{G}_{\mathbf{d}}^{\mathbf{y}}\mathbf{d} \tag{6.2}$$

where $\mathbf{G}^{\mathbf{y}} \in \mathbb{R}^{n_y \times n_u}$ and $\mathbf{G}_{\mathbf{d}}^{\mathbf{y}} \in \mathbb{R}^{n_y \times n_d}$ are the gain matrices from $\mathbf{u}$ to $\mathbf{y}$ and $\mathbf{d}$ to $\mathbf{y}$ respectively. The optimal selection matrix $\mathbf{H}$, in terms of minimizing the loss for J_{ss} with respect to the expected disturbances and measurement noise, is then given by, [3]

$$\mathbf{H}^{\mathsf{T}} = (\mathbf{Y}\mathbf{Y}^{\mathsf{T}})^{-1}\mathbf{G}^{\mathbf{y}} \tag{6.3}$$

where,

$$\mathbf{Y} = \begin{bmatrix} \mathbf{F}\mathbf{W}_{\mathbf{d}} & \mathbf{W}_{\mathbf{n}^{\mathbf{y}}} \end{bmatrix} \tag{6.4}$$

and $\mathbf{W}_{\mathbf{d}}$ and $\mathbf{W}_{\mathbf{n}^{\mathbf{y}}}$ are diagonal scaling matrices for the expected magnitudes of the disturbance and the measurement noise, respectively. $\mathbf{F} = \frac{\partial y^{opt}}{\partial d}$ is the optimal sensitivity matrix which describes how the measurement vector that correspond to the optimal operation $\mathbf{y}^{opt}$, change with unit change in the disturbance. The optimal sensitivity matrix may be determined analytically using

$$\mathbf{F} = -(\mathbf{G}^{\mathbf{y}}{\mathbf{J}_{\mathbf{uu}}}^{-1}\mathbf{J}_{\mathbf{ud}} - \mathbf{G}_{\mathbf{d}}^{\mathbf{y}}) \tag{6.5}$$

or it may also be determined numerically by perturbing the disturbances and resolving the optimization problem as described by Alstad and Skogestad [2]. Note that the optimal $\mathbf{H}$ in (6.3) is not unique, but the non-uniqueness may be absorbed into the controller.

As seen from the equations above, the optimal selection matrix $\mathbf{H}$ in (6.3) is based on the plant model $\mathbf{G}^y$ and the optimal sensitivity matrix $\mathbf{F}$ for the expected disturbances. Due to the linearization around the nominal optimal point, the controlled variables combination is only locally valid around this nominal optimal point. If a disturbance moves the process far from the nominal optimal point, the local model approximation may be poor, resulting in higher steady-state loss. Over time, as the plant-model mismatch increases, the increase in the loss may no longer be acceptable. This requires re-optimization and computation of new optimal setpoints $\mathbf{c}_s$. Additionally, any unmodeled disturbances that are not accounted for in the optimal sensitivity matrix cannot be handled efficiently.

6.2.2 Extremum seeking control

Unlike self-optimizing control, which is based on local linearization of the *model* around the nominal operating point, extremum seeking control is based on local

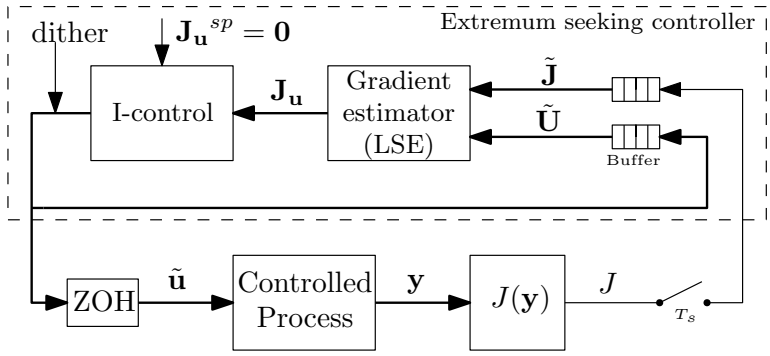


Figure 6.1: Block diagram of the least-square based extremum seeking controller.

linearization of the *measured cost* around the current operating point as explained in Chapter 5. We only considered a single-input-single-output (SISO) system in Chapter 5, however, in this chapter, we will consider a multivariable system.

There are different ways of estimating the gradient based on the input and cost measurements. In this chapter we will consider the linear least square estimation method to estimate the steady-state gradient [70]. The least squares approach also provides a natural platform extending to multivariable systems. Improved performance using a recursive least squares approach was also reported in [32].

The goal is to estimate the gradient from the inputs $\tilde{\mathbf{u}}$ to the measured cost J. In the least squares based extremum seeking control , the last N samples of data are used to fit a local linear cost model of the form,

$$J = \mathbf{J}_{\tilde{\mathbf{u}}}{}^{\mathsf{T}}\tilde{\mathbf{u}} + m \tag{6.6}$$

where $\mathbf{J}_{\tilde{\mathbf{u}}} \in \mathbb{R}^{n_u}$ is the vector of gradients from $\tilde{\mathbf{u}}$ to J and $m \in \mathbb{R}$ is the bias.

At the current sample time k, let $\tilde{\mathbf{J}} = [J_k \ \cdots \ J_{k-N+1}]^{\mathsf{T}} \in \mathbb{R}^N$ be the vector of the last N samples of the measured cost and $\tilde{\mathbf{U}} = [\tilde{\mathbf{u}}_k \ \cdots \ \tilde{\mathbf{u}}_{k-N+1}]^{\mathsf{T}} \in \mathbb{R}^{N\times n_u}$ be the vector of the last N samples of the input. A moving window of fixed length N is then used to estimate the gradient using the linear least squares method [114]

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\| \tilde{\mathbf{J}} - \Phi^{\mathsf{T}}\boldsymbol{\theta} \right\|_2^2 \tag{6.7}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{(n_u+1)}$ is the vector of parameters to be estimated and is given by

$$\boldsymbol{\theta} = \begin{bmatrix} \mathbf{J}_{\tilde{\mathbf{u}}}{}^{\mathsf{T}} & m \end{bmatrix}^{\mathsf{T}} \tag{6.8}$$

and $\Phi \in \mathbb{R}^{N(n_u+1)}$ is the regressor vector given by

$$\Phi = \begin{bmatrix} \tilde{\mathbf{u}}_k^{\mathsf{T}} & 1 \\ \tilde{\mathbf{u}}_{k-1}^{\mathsf{T}} & 1 \\ \vdots & \vdots \\ \tilde{\mathbf{u}}_{k-N+1}^{\mathsf{T}} & 1 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{U}} & \mathbf{1} \end{bmatrix}^{\mathsf{T}} \tag{6.9}$$

The analytical solution to the least squares problem is given by, [114]

$$\hat{\boldsymbol{\theta}} = \left[\Phi^{\mathsf{T}}\Phi\right]^{-1}\Phi^{\mathsf{T}}\tilde{\mathbf{J}} \tag{6.10}$$

The application of ordinary least squares requires that $N > n_u$. Note that in theory, it is not necessary to use a dither signal when this approach is used, but for practical purposes it is recommended, and in our case study we use a sinusoidal dither signal with an appropriately chosen amplitude.

Once the gradient vector $\mathbf{J}_{\tilde{\mathbf{u}}}$ is estimated, n_u integral controllers can be used to drive the gradients to zero using as degrees of freedom $\tilde{\mathbf{u}}$ (setpoints to the lower level controllers). The integral controller in general can be written as,

$$\tilde{\mathbf{u}}_{k+1} = \tilde{\mathbf{u}}_k + T_s\mathbf{K}_I\hat{\mathbf{J}}_{\tilde{\mathbf{u}}} \tag{6.11}$$

where $\mathbf{K}_I \in \mathbb{R}^{n_u \times n_u}$ is the gain matrix and T_s is the sample time. In this chapter, we use decentralized control where $\mathbf{K}_I$ is diagonal. A schematic representation of the least squares-based extremum seeking control is shown in Fig. 6.1

An additional dither signal is added to the control input to provide sufficient excitation in the input and cost measurements for accurate gradient estimation. For a multi-input system, each input should have a unique perturbation frequency in order to estimate the gradient of the cost measurement with respect to each input. Since the linear model assumption is valid only locally around the current operating point, the gradient is estimated using only recent samples of data (i.e. a moving window of fixed length N). It was shown in [70] that the least squares based extremum seeking control is stable and that the error is small for a sufficiently small product of the adaptation gain and sample size $\mathbf{K}_I N$.

6.3 Proposed method

In this chapter, we propose a hierarchical implementation with separate optimization and control layers as proposed by Halvorsen et al. [62] and shown in Fig. 6.2.

Due to the timescale separation required between the optimization and control layers [165], the extremum seeking controller is in the slow optimization layer, and thus replaces the conventional RTO. Self-optimizing control is in the faster setpoint control layer below and tracks the updated setpoint given by the extremum seeking controller. In other words, the extremum seeking controller uses the measured cost J to compute the setpoint $\mathbf{c}_s$ which is provided to the self-optimizing control. The controller output from the extremum seeking controller is $\tilde{\mathbf{u}} = \mathbf{c}_s$ in (6.6)-(6.10)[1].

It may be argued that the self-optimizing control layer is redundant since an extremum seeking scheme can directly manipulate the process to optimize the objective function. However, by using a purely data-driven approach, we ignore any *a-priori* knowledge about the system and the effect of disturbances. In addition, the extremum seeking controller does not make use of measurements besides the cost measurements. Finally, the convergence to the optimum is slow. The proposed

[1] In the case where extremum seeking controller controls the plant directly, $\tilde{\mathbf{u}} = \mathbf{u}$

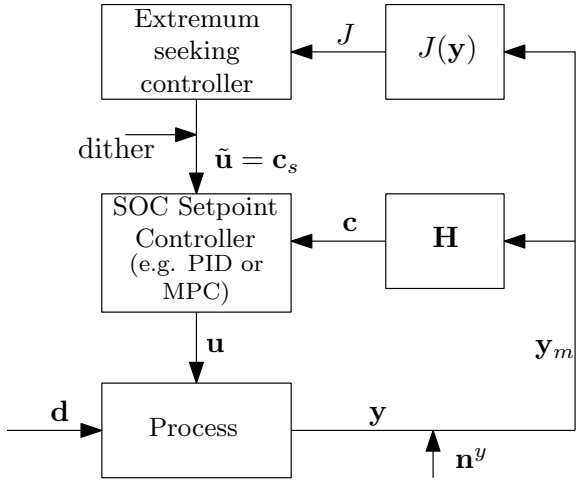


Figure 6.2: Hierarchical implementation of combined self-optimizing control and extremum seeking control . The extremum seeking controller used in this chapter is shown in Fig. 6.1.

Table 6.1: Properties of self-optimizing control and extremum seeking control .

Self-optimizing control	extremum seeking control
offline model required	model-free
fast rejection of disturbances	slow rejection of disturbances
local linearization around nominal optimal point	local linearization around current operating point
handles unmeasured but expected disturbances	handles unmeasured and unexpected disturbances
needs no cost measurement	requires measurement of cost

hierarchical combination of extremum seeking control and self-optimizing control avoids the shortcomings of the extremum seeking scheme and improves the convergence to the optimum. This is primarily due to a faster initial reaction of the self-optimizing layer to known (modeled) disturbances. Following a disturbance, the self-optimizing control quickly brings the operation point close to the optimal region, and on a slower timescale, the extremum seeking control fine-tunes the setpoint and removes any loss associated with the self-optimizing control.

The extremum seeking layer handles the plant-model mismatch and unmodeled disturbances and removes any steady-state loss by adjusting the setpoint $\mathbf{c}_s$. This also avoids re-optimization e.g. using real-time optimization.

In summary, we use the knowledge about the system to stay in the near-optimal region using self-optimizing control in the presence of disturbances. The extremum seeking control helps to remove, or at least reduce the losses due to plant-model

mismatch, it handles any unexpected disturbances, and it fine-tunes the optimal operating point. The key properties of the two methods are summarized and compared in Table 6.1, which shows that the self-optimizing control and extremum seeking control are complementary rather than competing.

Note that, the proposed method is not just restricted to the least squares-based extremum seeking control , and other extremum seeking control methods, such as classical extremum seeking control , or recursive least squares based extremum seeking control etc. [32, 107, 180] may be used instead on top of the self-optimizing control layer in Fig. 6.2.

6.3.1 Stability issues

In this section, we provide some discussions on the stability of the combined self-optimizing control and extremum seeking control layout presented in Fig. 6.2. As mentioned earlier, extremum seeking controller has three timescales, namely,

- fast - controlled plant dynamics
- medium - dither frequency
- slow - convergence to the optimum

A good extremum seeking controller is tuned such that there is a clear timescale separation between these three timescales. The self-optimizing controller is included in the controlled plant and belongs to the fast timescale [107]. By introducing the self-optimizing control below the extremum seeking control , the perturbation frequency and the adaptation (integral) gain $\mathbf{K}_I$ must be chosen such that the timescale separation enables the static map assumption. Therefore, when seen from the slow timescale of the extremum seeking controller, the closed-loop system comprising of the self-optimizing controller and plant is a static map $J = \mathbf{h}(\mathbf{c}_s)$.

The stability results for the classical extremum seeking controller provided by Krstić and Wang [107] and the least squares based extremum seeking controller provided by Hunnekens et al. [70], both assume a smooth stabilizing control law parameterized by a "performance parameter". This performance parameter is used as the handle by the extremum seeking controller. In our paper, the control law is given by the self-optimizing control layer and the "performance parameter" is equivalent to the setpoint for the self-optimizing variable $\mathbf{c}_s$.

To summarize, existing stability results from [107] for the classical extremum seeking control and [70] for the least squares based extremum seeking control also hold for the combined hierarchical structure in Fig. 6.2, if the following two conditions are met:

1. The self-optimizing setpoint control layer is closed-loop stable.
2. The perturbation frequency of the ESC is sufficiently small compared to the timescale of the controlled plant which includes the stabilizing self-optimizing controller.

Choice of tuning parameters According to the stability analysis results by Hunnekens et al. [70], the product $K_I N$ is the only important quantity that must be chosen small enough. This is a reasonable measure, since a small product of the

adaptation gain K_I and the time window N intuitively means that the measurements used for gradient estimation are in the small neighborhood of the current operating point $\mathbf{u}$. As for the choice of the window length for the past measurements, N must be sufficiently small such that the error in the gradient estimate is bounded as described in detail in [70]. Alternatively, instead of using a fixed moving window of the past N samples, a forgetting factor may be used in a recursive least squares estimation framework, as adopted in [32]. As for any extremum seeking control framework, the adaptation gain K_I must be chosen sufficiently small, such that there is a clear time scale separation between the dither and the convergence to the optimum. This is required in order to validate the assumption of the plant being a static map, which is required for the extremum seeking control . For more detailed guidelines on tuning the adaptation gain, the reader is referred to [180].

6.3.2 Effect of disturbances

Very little of the literature on extremum seeking control consider explicitly the effect of disturbances. Disturbances typically trigger fast dynamics, which may invalidate the assumption of the plant operating close to a static map. Consequently, the introduction of the fast dynamics leads to erroneous gradient estimation, especially with the least squares gradient estimation method used in this chapter. In other words, if the cost measurement in the gradient estimation time window is in transients due to the disturbances, then the least squares method fits a wrong gradient $\hat{\mathbf{J}}_{\mathbf{u}}$. The effect of abrupt disturbances on the extremum seeking scheme has been well motivated by Krishnamoorthy et al. [92], Marinkov et al. [122], and Marinkov et al. [121], along with some modifications to improve disturbance rejection. However, all these modifications require the disturbances to be measured. Measured disturbances may also be handled by the least squares-based extremum seeking scheme described in Section 6.2.2, by explicitly including the measured disturbances as a part of the regressor Φ in (6.10) and replacing (6.6) with

$$J = \mathbf{J}_{\tilde{\mathbf{u}}}{}^{\mathsf{T}}\tilde{\mathbf{u}} + \mathbf{J}_{\mathbf{d}'}{}^{\mathsf{T}}\mathbf{d}' + m \tag{6.12}$$

where $\mathbf{J}_{\mathbf{d}'} \in \mathbb{R}^{n_{d'}}$ is the vector of gradients from the measured disturbances $\mathbf{d}'$ to the cost J.

Unfortunately, unmeasured disturbances may still result in erroneous gradient estimation. Given that the extremum seeking control problem at hand is essentially a static optimization problem, the only way to avoid this problem is to use a steady-state detection, as used in the traditional steady-state RTO. The extremum seeking scheme can be triggered only if the cost measurement in the gradient estimation window is close to steady-state operation. Dynamic changes in the cost measurement resulting from a disturbance will be flagged by the steady-state detection routine and the extremum seeking scheme is temporarily halted until the cost measurement in the gradient estimation time window comes close to steady-state operation. By doing so, the static map assumption used by the extremum seeking scheme is always valid. This halt is typically known as the *steady-state wait time* and is commonly used in the traditional steady-state RTO paradigm. In fact, the method proposed by Marinkov et al. [121] is precisely a steady-state wait time routine implemented using a supervisory state machine.

In many processes with long settling times, the steady-state wait time can lead to a very slow convergence to the optimum. Alternatively, in some process systems, one can make use of some heuristics to avoid the steady-state wait time, such as to bound the magnitude of the individual gradients ($\hat{J}_{\tilde{u}_i}$) in (6.11) to a value $J_{u_i,max}$

$$\left|\hat{J}_{\tilde{u}_i}\right| \leq J_{u_i,max} \tag{6.13}$$

The bounding limits aggressive input usage. Based on (6.11), we propose to introduce a maximum change in the input between samples, $\Delta\tilde{\mathbf{u}}_{i,max}$, and choose,

$$J_{u_i,max} = \frac{\Delta\tilde{\mathbf{u}}_{i,max}}{T_s\mathbf{K}_I} \tag{6.14}$$

Note that although this approach is used in our case study, this additional heuristic of bounding the gradient is not part of the core methodology presented in this chapter, but should be viewed as an alternative approach to the steady-state wait time.

6.4 Case study - Ammonia synthesis reactor

In this chapter, we apply the proposed hierarchical combination of extremum seeking control and self-optimizing control to a three-bed ammonia reactor with heat integration. A flowsheet, including the control structure for the proposed method, can be found in Fig. 6.3. The model was first described in [131]. The model consists of three sequential reactor beds and one heat exchanger. The inlet stream to the reactor system (denoted by subscript in) is split into four streams; one quench flow to each bed and a preheated flow to the first reactor bed. The quench split ratios correspond to the three manipulated variables $\mathbf{u}_0 = \begin{bmatrix} u_{0,1} & u_{0,2} & u_{0,3} \end{bmatrix}^\mathsf{T}$. The three reactor beds are discretized into a cascade of continously stirred tank reactors (CSTR). A detailed model description can be found in [174].

The objective is to maximize the extent of reaction ξ for a given feed, that is

$$J = \xi \tag{6.15}$$
$$= \dot{m}_{in}\left(w_{NH_3,30} - w_{NH_3,in}\right) \tag{6.16}$$

The disturbances are the inlet conditions;

$$\mathbf{d} = \begin{bmatrix} \dot{m}_{in} & p_{in} & T_{in} & w_{NH_3,in} \end{bmatrix}^\mathsf{T} \tag{6.17}$$

6.4.1 Controller design

The potential instability in case of disturbances as described by Morud and Skogestad [131] requires a stabilizing "slave" control layer below the self-optimizing control layer. Straus and Skogestad [174] showed that if the reactor is operated close to the nominal optimum and without control, reactor extinction may result even from small disturbances compared to the large disturbances investigated by Morud and Skogestad [131]. Hence, also for the case when extremum seeking control is utilized

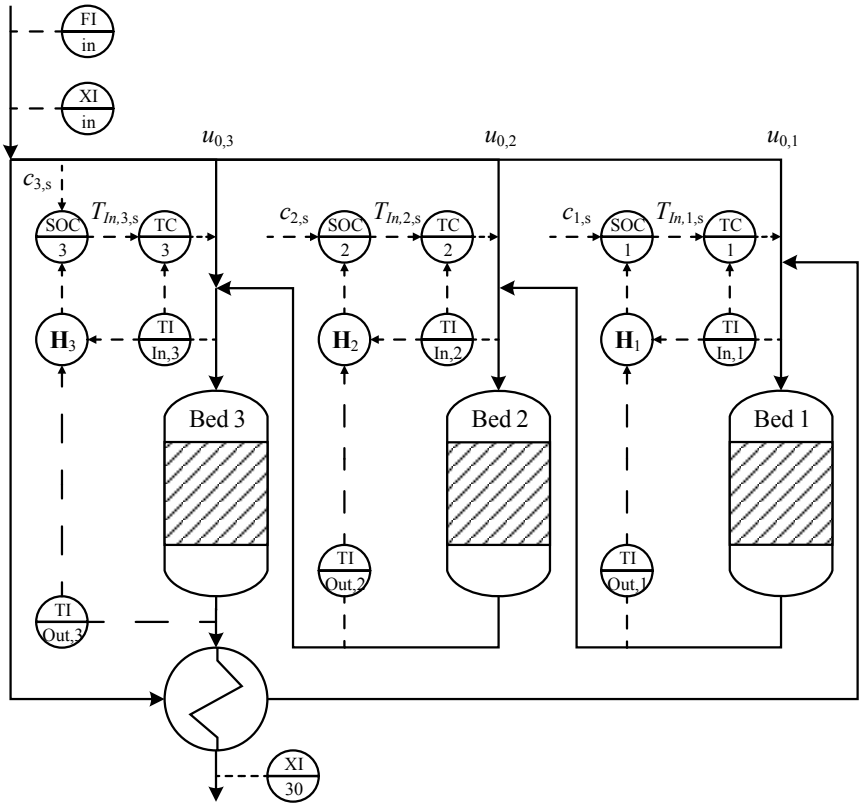


Figure 6.3: Flowsheet of the reactor, modified from [131] to include the proposed control structure.

without self-optimizing control, a stabilizing "slave" control layer is required resulting in a cascade control structure in all investigated control structures. The chosen "slave" controllers are temperature controllers and more detailed explanation can be found in [176]. In our case, the SOC controllers give the setpoints to the respective slave temperature controllers. The measurements $\mathbf{y}$ for self-optimizing control are selected to be the inlet and outlet temperature of each reactor bed; *i.e.*

$$\mathbf{y}_i = \begin{bmatrix} T_{In,i} \\ T_{i0} \end{bmatrix} \qquad \forall i = 1, 2, 3 \tag{6.18}$$

Hence, only two measurements were used for the calculation of $\mathbf{H}_i$ in (6.3). The slave temperature loops as well as the master self-optimizing control loops were tuned using the SIMC rules [166], which can be found in [176]. The extremum seeking controllers were implemented in discrete time resulting in a discrete-continuous representation. The tuning parameters for the extremum seeking control were chosen based on trial and error to achieve satisfactory performance and can be found in [176]. These slow integral controllers give the setpoints to either the base layer

temperature ($\mathbf{T}_s$, denoted T+ESC) or the self-optimizing controllers ($\mathbf{c}_s$, denoted T+SOC+ESC). The estimation of the gradient $\hat{\boldsymbol{\theta}}$ according to (6.10) is performed using $\tilde{\mathbf{u}}$ as the setpoint of the respective slave controller (T or SOC). It is assumed that the disturbances are unmeasured.

6.4.2 Results

In order to compare the proposed methods, two disturbances were investigated; a disturbance in the inlet mass flow rate $\dot{m}_{in}$, corresponding to a modeled disturbance in self-optimizing control, and an unmodeled disturbance in the reaction rate r. These disturbances were chosen as they correspond to the largest losses for the self-optimizing control structure (not shown). Hence, the improvement using extremum seeking control is most pronounced. In addition, both disturbances would result in reactor extinction, if the stabilizing temperature controllers would not be present. The integrated loss (cost difference),

$$J_{int}(t) = \int_0^{\mathsf{T}} \left[\xi_{opt,SS}(t') - \xi(t')\right] \mathrm{d}t' \tag{6.19}$$

is used to compare the proposed methods.

The first considered disturbance is a +20 % step change in the inlet mass flow rate as this disturbance results in the highest steady-state loss for self-optimizing variables [175]. This disturbance is considered in the calculation of the SOC variables. The cost $J = \xi$ and the integrated loss (6.19) are shown in Fig. 6.4. The cases with extremum seeking control (solid lines) settle to the new optimum in contrast to pure self-optimizing control (dashed red line). The combination of self-optimizing control and extremum seeking control gives a large reduced loss in produced tons of ammonia. As seen in Fig. 6.4, this reduction corresponds to 4.95 t ammonia in the investigated time-frame of 18 hours. One could argue that this is caused by suboptimal tuning parameters in the pure extremum seeking control . By taking a look at the time the disturbance is occurring, we claim that this is not the case. Fig. 6.5 shows a close-up of the response in the cost function for the first 1.2 hours after the disturbance occurs. From this figure, it can be clearly seen that both ESCs (solid lines) initially follow their respective slave controllers, before deviating when the ESCs start changing the setpoints to the slave controllers. Both ESC control structures are in fact moving initially in the wrong direction, that is, to a reduced extent of reaction. This can be explained by the past measurements, before the disturbance, which are still used at this point. One approach to circumvent this behavior is to use a smaller time horizon (smaller N). This results on the other hand in a drift away from the optimal setpoint on a long time scale. Hence, it is preferable to have a slightly suboptimal initial performance.

A disturbance in the reaction rate r is an unmodeled disturbance which is not considered in the calculation of the optimal selection matrices according to (6.3). It can be considered a plant-model mismatch. The simulation results for a −20 % step change in the reaction rate r are shown in Fig. 6.6. Similarly to a disturbance in the inlet mass flow $\dot{m}_{in}$, the control structure based on the proposed method with both self-optimizing and extremum seeking control settles to the new optimum

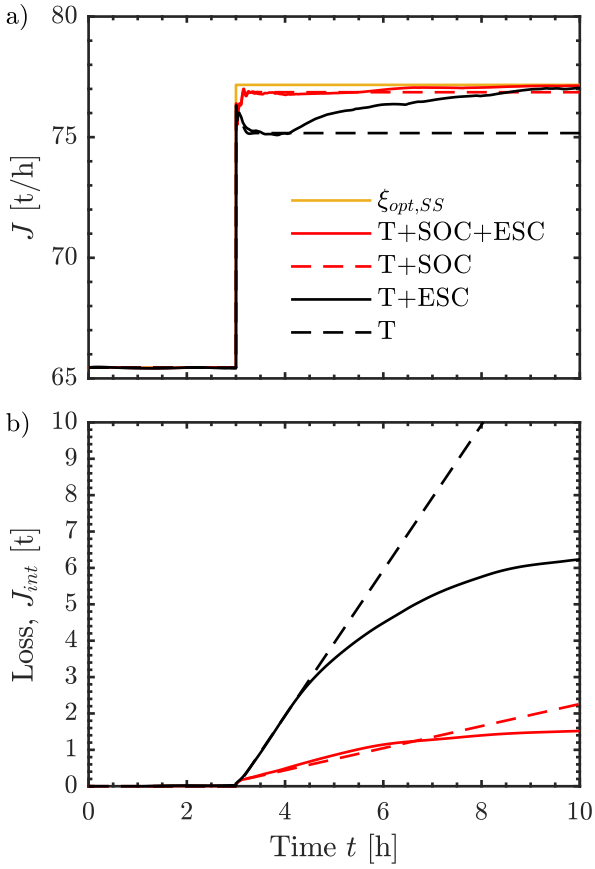


Figure 6.4: Response of a) extent of reaction ξ and b) integrated loss to a +20 % disturbance in inlet mass flow rate, $\Delta\dot{m}_{in} = +54$ t/h, at $t = 3$ h.

after 7 hours whereas extremum seeking control alone requires around 13 hours. During the time the controllers require to settle to the new optimum, the loss is reduced in the proposed control structure with SOC. Over 18 hours, the proposed control structure has a reduced loss of 6.71 tons of produced ammonia. Here it has to be noted, that despite this disturbance was not included in the design phase, the self-optimizing control structure has a reduced loss. This can be explained by general favorable properties of self-optimizing feedback with regard to disturbances and plant-model mismatch.

6.5 Chapter Summary

In this chapter, we have shown that extremum-seeking control and self-optimizing control are complementary rather than competing. This is caused by the differ-

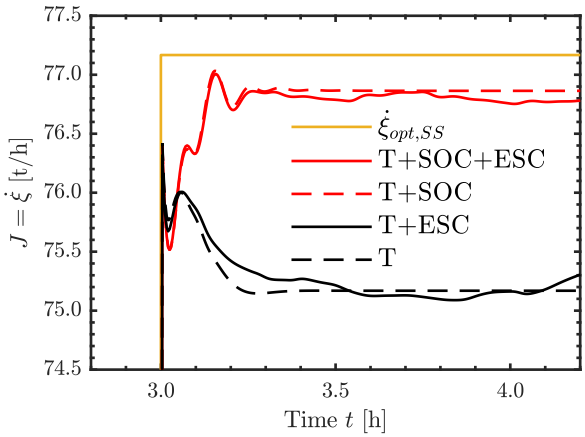


Figure 6.5: Closeup of Fig. 6.4 a) at the time when the disturbance occurs.

ent timescale at which the control strategies are operating. By combining self-optimizing control and extremum-seeking control, we are able to utilize the advantages of each method and improve the convergence to the optimum. Using a three-bed ammonia reactor case study, we demonstrate that the combined system can handle unmeasured disturbances and at the same time correct for plant-model mismatch.

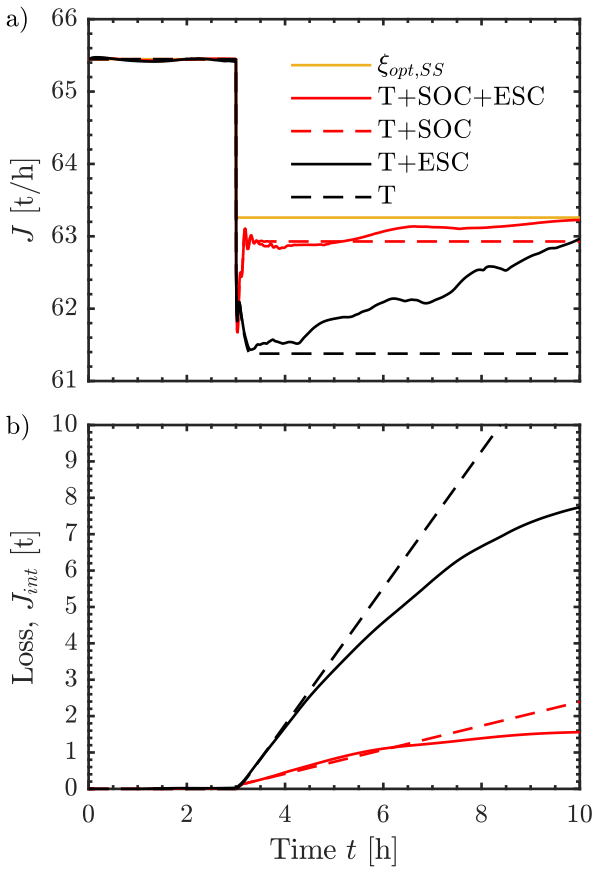


Figure 6.6: Response of a) extent of reaction ξ and b) integrated loss to a -20 % disturbance in pre-exponential factors of the Arrhenius equations at $t = 3$ h.

Part II

Addressing Computation Time for Dynamic Process Optimization

Chapter 7

Dynamic Online Process Optimization under Uncertainty

This chapter serves as an introduction to part II, and provides useful discussions on problem formulation under uncertainty. More importantly this chapter aims to provide a clear understanding of open-loop and closed-loop optimization.

7.1 Introduction

In part I of the thesis, we considered the steady-state optimal operation of a process. The main focus of the different approaches studied in Part I of this thesis, was to drive the process to its optimal steady-state operating point. Optimizing during the transients were not considered in Part I. But for some processes, optimizing the transients becomes important, such as processes with frequent changes in feed, product specifications, market disturbances, grade transitions, cyclic operations, batch processes etc., which makes steady-state optimization less relevant. Optimal operation of such processes requires solving dynamic optimization problems. This is done under the context of dynamic real-time optimization (DRTO) [158], where dynamic process models are used to solve a dynamic optimization problem. The optimal setpoint trajectories are then given to a setpoint tracking layer below(commonly, model predictive control). To this end, dynamic real-time optimization (DRTO) and model predictive control (NMPC) have emerged as crucial technologies in achieving this. More recently, the DRTO and NMPC layers have been tightly integrated into a single layer approach, known as economic NMPC [143], where there is no time scale separation such as in the conventional two-layer approach, as shown in Fig. 7.1.

Among different application areas, process industries are probably the most suited for centralized optimizing control structures like economic NMPC, since information from different parts of the plant are typically gathered into one platform (traditionally the control room) for processing and decision-making. Throughout this thesis, we solve dynamic optimization problems in the context of economic NMPC as shown in Fig. 7.1b.

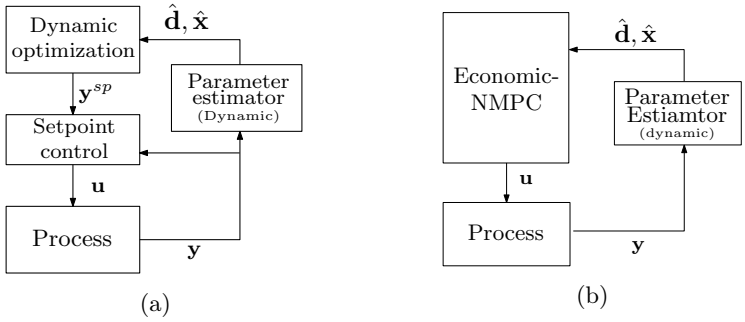


Figure 7.1: (a) Dynamic RTO (b) Economic NMPC

7.2 Problem formulation

Suppose we want to solve a dynamic optimal control problem (OCP),

$$\min_{\mathbf{x}(t),\mathbf{u}(t)} \quad \int_{t_0}^{t_f} \ell(\mathbf{x}(t),\mathbf{u}(t))\mathrm{d}t \tag{7.1a}$$

$$\text{s.t.} \tag{7.1b}$$

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{p}(t)) \tag{7.1c}$$

$$\mathbf{g}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{p}(t)) \leq 0 \tag{7.1d}$$

$$\mathbf{x}(t_0) = \hat{\boldsymbol{x}}_t \tag{7.1e}$$

$$\mathbf{x} \in \mathcal{X}, \quad \mathbf{u} \in \mathcal{U} \tag{7.1f}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ denotes the states, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ denotes the control inputs and $\boldsymbol{p}(t) \in \mathbb{R}^{n_p}$ denote the model parameters and disturbances, which are considered to be uncertain and the continuous time ODE process model is described by the function $\mathbf{F} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$. . The states and the inputs are constrained to lie in the compact sets $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{n_x}$ and $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^{n_u}$ respectively, $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ denotes the stage cost that is integrated over the decision horizon $[t_0, t_f]$, $\mathbf{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_c}$ denotes the nonlinear inequality constraints and $\hat{\boldsymbol{x}}_t$ is the state measured/estimated at the current time.

Before this can be posed as a standard optimization problem, the infinite dimensional optimal control problem (7.1) is first discretized into a finite dimensional nonlinear programming problem (NLP), divided into N equally spaced sampling intervals in $\mathcal{K} = \{0, \cdots, N-1\}$. Discretization can be performed using different approaches such as single shooting, multiple shooting or direct collocation as described in [15]. Once the system has been discretized, the OCP can be posed as a

standard NLP problem,

$$\min_{\mathbf{x}_k, \mathbf{u}_k} \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) \tag{7.2a}$$

s.t.

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{p}_k) \qquad \forall k \in \mathcal{K} \tag{7.2b}$$

$$\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{p}_k) \leq 0 \qquad \forall k \in \mathcal{K} \tag{7.2c}$$

$$\mathbf{x}_0 = \hat{\boldsymbol{x}}_t. \tag{7.2d}$$

$$\mathbf{x}_k \in \mathcal{X}, \quad \mathbf{u}_k \in \mathcal{U} \qquad \forall k \in \mathcal{K} \tag{7.2e}$$

where, the discretized process model is described by the function $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$.

Since dynamic RTO and economic NMPC are based on using a dynamic model of the process, solving (7.2) in real-time requires fast optimization algorithms. As many industrial applications call for increasingly complex, detailed and large-scale process models [9], a major concern is the computational resources needed to solve the resulting large-scale optimization problem. While advances in numerical optimization strategies have enabled us to solve increasingly larger optimal control problems (OCPs), real-time implementation is still challenging, even with today's computing power [14, 25]. The non-negligible amount of time taken to solve the numerical optimization problem online leads to computational delays, that are known to degrade the control performance [154], and can also destabilize the system [30, 45].

One of the main bottlenecks in solving dynamic optimization problems is therefore, the online computation time. To address this issue, several sensitivity-based methods were proposed, see [35], [186], [189], [74] and [178] to name a few. All these sensitivity-based methods exploit the fact that the OCP is parametric in the initial condition $\hat{\boldsymbol{x}}_t$. Here, the computationally expensive NLP problem is solved offline, and the NLP sensitivities are used to obtain fast approximate solutions when new measurements of the state become available. The application of one such sensitivity-based economic NMPC for a gas-lift optimization problem is provided in Appendix N. The reader is referred to [13] for an overview of the recent developments in fast sensitivity-based NMPC. In this thesis, we will focus on solving dynamic optimization problems under uncertainty.

In the presence of uncertainty, the online computation time may increase tremendously, depending on the problem formulation and how the uncertainty is handled. The problem formulation not only affects the complexity and computation time, but also the performance (e.g. robustness vs. conservativeness). For this reason, choosing an appropriate problem formulation is an important aspect to enable successful implementation of dynamic RTO and economic NMPC (Challenge 6 from Section 1.1).

7.3 Robust vs. adaptive problem formulation

Optimization problem formulation under uncertainty can be broadly divided into two categories, based on how and "where" the uncertainty is handled, namely,

- **Adaptive formulation:** measurement from the process is used to update the process model using a parameter estimator.
- **Robust and stochastic formulation:** A-priori information about the uncertainty is used to explicitly account for the uncertainty *in* the optimization problem.

The aim of this section is *not* to provide a comprehensive review of the two solution strategies, but rather to state the key differences in the two strategies, and provide a guideline on when to use which formulation. The discussion is also restricted to dynamic optimization problems, in the context of economic NMPC.

Adaptive formulation: In the adaptive approach, the idea is to estimate the uncertain variables from the process measurements, and use the updated value of the parameters in the optimization problem as shown in Fig. 7.1. In the adaptive approach, the uncertainty is handled "outside" the optimization problem, i.e. the optimization problem is supplemented with an external parameter estimation block, and the optimization problem itself, is deterministic. For example, the dynamic RTO formulation used in Chapters 2 and 3 use an adaptive approach, where a parameter estimator iteratively updates the model that is used to solve the dynamic optimization problem in a deterministic fashion.

Ideally, the adaptive mechanism used should asymptotically converge to the true system, thereby improving the performance over time. However, since the model adaptation is based on observed measurements, this is a *reactive* strategy, and the transient effects of the parameter estimation error have proven problematic in the context of predictive control. This was also observed in the simulation example used in Chapter 2, where it was seen, that the process constraints are violated dynamically, due to the *lag* in the parameter estimation step (cf. Fig. 2.7c). The parameters that are sensitive to the cost and the constraints, must also be observable. In addition, it is also difficult to guarantee closed-loop stability when the dynamic optimization problem is combined with an adaptation algorithm. Therefore, the development of adaptive problem formulation in the context of model predictive control, has received very little attention, and still remains an open problem [85, 126].

The goal of the parameter estimation is to minimize the error between the model predictions and the measurements. However, this does not always coincide with the optimization objective. A good prediction model may still be a bad optimization model, if the gradient of the process model used in the optimization problem does not accurately predict the plant gradients.

In general, an adaptive formulation may be preferred if,

- there are no hard constraints on the states or measurements, that may need to be satisfied in the future
- significant performance improvement can be achieved by updating the model using a suitable parameter estimator.

Robust and stochastic formulation: If the uncertainty characteristics of the uncertain parameters (i.e. $\boldsymbol{p} \in \mathcal{P}$) are known a-priori, for example in the form of a

compact set-membership or a probability density function, then the uncertainty can be explicitly handled in the optimization problem. Unlike the adaptive formulation, this is an *anticipative* approach, where the uncertainty is explicitly handled inside the optimization problem, i.e. the optimization problem formulation is no longer deterministic, but depends on the uncertainty characteristics.

To ensure *robust* constraint feasibility for any realization of the parameter from the compact set $\boldsymbol{p} \in \mathcal{P}$, the uncertain parameters may be assumed to take their worst-case realization in the optimization problem. This was first introduced in 1973 by Soyster, where every uncertain parameter in convex programming was taken equal to its worst-case value within a set [169]. Since then, optimization for the worst-case value of the parameters within a set has become effectively known as *Robust Optimization*. In the context of dynamic optimization and model predictive control, the min-max NMPC formulation was proposed by Campo and Morari [24], which computes an optimal input trajectory that minimizes the cost of the worst-case uncertainty realization. Although this ensures robust constraint satisfaction, this often leads to overly conservative and hence suboptimal solutions. This is because the optimization is performed for the worst-case scenario in an open-loop fashion.

Alternatively, in the stochastic approach, the cost and the constraints are defined as expectations using a probability density function. In its simplest form, if the system is linear, and the uncertainty is represented by a Gaussian distribution, then the expectations of the cost and the constraints can be computed [128]. However, when dealing with economic objectives, linear models are seldom used and the use of a Gaussian distribution may be too restrictive. In such cases, computationally expensive Monte Carlo simulations will have to be performed to solve the stochastic optimization problem [125].

In general, robust/stochastic formulations may be preferred if,

- there are not sufficient measurements available to update the model in a meaningful way
- a-priori information about the uncertainty in the form of probability density function or compact set membership is available,
- there are hard constraints that needs to be satisfied (requires robust formulation)

In the robust and stochastic formulations, the resulting optimal control framework is significantly more complex than deterministic optimization. This is especially true for nonlinear systems, where the problem can be too complex to be solved exactly [125].

Another major concern with the conventional robust and stochastic framework is that, the optimization problem formulation is highly dependent on the accurate knowledge of the uncertainty characteristics. For example, if the uncertainty characteristics change, then this requires the optimal control problem to be rebuilt again (which requires skilled engineers).

Moreover, in many real applications, the probability density function (PDF) or the uncertainty set for the uncertain parameters is not readily available, but only a finite number of data samples may be available. Classical stochastic NMPC frameworks make use of such data indirectly to infer the probability distribution of

the uncertain problem parameters by means of statistical estimation methods. The estimated probability density function is then subsequently used in the optimization problem [134]. The classical stochastic NMPC problem is based on a two-step approach:

1. estimate the PDF or the compact set from the finite data samples
2. use the estimated PDF or the compact set in the optimization problem.

The main issue with this two-step approach is that the estimation step often aims to achieve maximum prediction accuracy without tailoring it to the optimization problem. Hence, the estimated probability density function or the compact set, itself may be uncertain as noted by Parys et al. [134] (leading to recent developments in so-called distributionally robust optimization). Given finite data samples, the uncertainty representations may be chosen directly from this data set, thus releasing the assumption of the uncertainty having any particular distribution. Thus, in such cases, scenario-based optimization techniques may be used to get a sampled-average approximation of the optimization problem, as suggested by Calafiore and Fagiano [21], Campi et al. [23], Fagiano et al. [44] to name a few. It is also worth noting that handling model structural uncertainty in this framework is still an open challenge.

Recently, there has been some interest in combining the two approaches to formulate adaptive-robust optimization problems. This approach differs from typical *certainty-equivalence* adaptive formulations, in that the adaptation mechanism does not aim to directly estimate the parameter itself. Instead, the set-valued description of the uncertain parameter is adapted online by eliminating values from the compact set that do not explain the observed trajectories with sufficient likelihood [60]. We explore a similar idea in [97] (see Appendix L). In this part of the thesis, we will consider the robust formulation in more detail.

Notational remark: At this point, we make a clarifying remark on the notation of the uncertain variable. In part I of this thesis, we had used $\mathbf{d}$ to denote the uncertain parameter/disturbance in the model, that are updated using a suitable parameter estimator. The estimated value $\hat{\mathbf{d}}$ is then used in the optimization problem. In this part of the thesis, we will use $\boldsymbol{p}$ to denote the uncertain parameters/disturbances in the model that are not updated, but are instead known to belong to a compact set denoted by $\mathcal{P}$.

7.4 Dynamic optimization under uncertainty

Research on dynamic RTO and model predictive control (NMPC) under uncertainty has been steadily gaining more interest, and several different optimization problem formulations have been studied in the literature. However, robust and stochastic problem formulations seems to be the most popular in the academic community, although the same level of interest and enthusiasm is not reflected in industrial practice [25, 49].

Motivated by the requirements in process industries, Mayne [125] questions if research on robust and stochastic NMPC is going in the right direction, particularly pointing out the complexity, ease of maintenance, and the potential objections to

using robust and stochastic NMPC formulations. Forbes et al. [49] also noted that robust and stochastic approaches are currently not commercially exploited in the process industry owing to its complexity. This chapter deals with the research question:

"What is the most appropriate problem formulation to handle uncertainty in dynamic RTO and economic NMPC? "

A good problem formulation must ideally have the following properties[1].

- **P1. Easy to understand** - Being able to understand the control actions is important to gain operator confidence. As mentioned in Section 1.1, if the operators do not understand the control actions, then it will most likely not be used in practice [49].
- **P2. Low complexity** - The problem formulation must be easy to maintain and make changes by process engineers without the need for skilled experts. Without regular maintenance and service, the benefits of using MPC will diminish [49, 125].
- **P3. Not too conservative** - Since the goal of the optimization problem is to minimize some economic objective, an overly conservative problem formulation is not desirable.
- **P4. Low computational effort** - Since the optimization problem is solved online, the computational effort must be low (cf. Challenge 3 in section 1.1). The computation delay can lead to performance degradation as motivated in [45].
- **P5. Feedback** - It is a well known fact that feedback is required when uncertainty is present. As Mayne [126] pointed out, in order to introduce the notion of feedback, one must optimize over control policies as done in Dynamic programming, instead of optimizing over control actions.

Before continuing further, we first broadly categorize the different formulation strategies, based on whether the optimization and implementation are done in open-loop or closed-loop. This is also schematically represented in Fig. 7.2.

- *Open-loop optimization with open-loop implementation* - In the first category, both the optimization and the implementation are performed in open-loop. Using the model, the optimizer computes an optimal input trajectory, which is then implemented on the system without any notion of feedback. This is an open-loop dynamic optimization problem.
- *Open-loop optimization with closed-loop implementation*- In this category, the optimization is performed in open-loop using a model, but the optimal solution is implemented in a closed-loop fashion. Using the model, the optimizer computes an optimal input trajectory over the prediction horizon $[0, \ldots, N-1]$. Only the optimal control input at the first sample is implemented on the system. When new measurements are available, the optimization problem is solved again to re-compute a new optimal input trajectory

[1]Note that this is not an exhaustive list, but only some of the key properties inspired from [125] and [49], and my personal industrial experience.

	Open loop implementation	Closed loop implementation
Open loop optimization	Dynamic optimization	Standard MPC / Receding horizon control
Closed loop optimization	–	Multistage MPC /Feedback min-max MPC

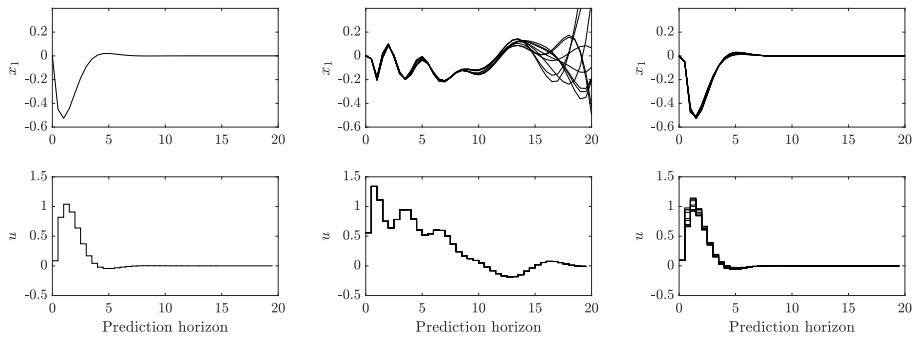
Figure 7.2: Classification of optimization problem formulations

at each sample time to account for the feedback information. This is carried out in a receding horizon fashion. Dynamic matrix control (DMC)/Model predictive control (MPC) falls under this category.

- *Closed-loop optimization with closed-loop implementation* - In this category, both the optimization and the implementation are performed in a closed-loop fashion. By "Closed-loop optimization", we mean, the optimization problem explicitly takes into account the fact that new measurements will be available in the future and that a new optimal input trajectory will be recomputed in the future. By doing so, we optimize over different control policies, rather than a single control trajectory [126]. This introduces the notion of *recourse*, which is an attractive property in handling uncertainty in the optimization problem.

To explain this better, first let us consider that the model in (7.2) is *perfect*, and $\boldsymbol{p}$ is known accurately. Then for an optimal input trajectory $\mathbf{u}^p_{[t,t+N]}$, the predicted state trajectory is given by $\mathbf{x}^p_{[t,t+N]}$ (see Fig. 7.3a). However, in the presence of uncertainty, described by $\boldsymbol{p} \in \mathcal{P}$, an optimal input trajectory $\mathbf{u}^p_{[t,t+N]}$ would give rise to a set of state trajectories $\{\mathbf{x}^p_{[t,t+N]}\}_{\mathcal{P}}$ depending on the value of the uncertain parameter $\mathbf{p} \in \mathcal{P}$ (see Fig. 7.3b). Optimizing over a single control trajectory $\mathbf{u}^p_{[t,t+N]}$ ignores the fact, that new information will be made available at the next time step, and a new optimal input trajectory will be re-computed. In other words, the optimization is performed in an open-loop fashion (although the implementation may be in closed-loop as in predictive control, where if the optimal control problem is re-solved at each sampling time with only the first control input move implemented on the process). Closed-loop optimization, on the other hand, involves computing a set of possible control trajectories $\{\mathbf{u}^p_{[t,t+N]}\}_{\mathcal{P}}$ instead of a single control trajectory $\mathbf{u}^p_{[t,t+N]}$, thereby introducing *recourse* action (see Fig. 7.3c). To provide a simple analogy, consider the uncertain optimal control problem as a decision-making process in an evolutionary strategic game (like chess). In order to make the best decision, even a moderately skilled player would prepare a strategy consisting of several back-up moves based on the expected evolution of the game (analogous to several control trajectories $\{\mathbf{u}^p_{[t,t+N]}\}_{\mathcal{P}}$), instead of only a single sequence of moves (analogous to single control trajectory $\mathbf{u}^p_{[t,t+N]}$).

To illustrate this, consider the example of driving the Van der Pol oscillator to



(a) (b) (c)

Figure 7.3: Van der Pol oscillator example: (a) Open-loop optimization with perfect model with single control trajectory $\mathbf{u}^p_{[t,t+N]}$ and corresponding state trajectory $\mathbf{x}^p_{[t,t+N]}$. (b) Open-loop optimization with single control trajectory $\mathbf{u}^p_{[t,t+N]}$ and corresponding set of state trajectories $\{\mathbf{x}^p_{[t,t+N]}\}_{\mathcal{P}}$. (c) Closed-loop optimization with set of possible control trajectories $\{\mathbf{u}^p_{[t,t+N]}\}_{\mathcal{P}}$ and corresponding set of state trajectories $\{\mathbf{x}^p_{[t,t+N]}\}_{\mathcal{P}}$.

its origin with a quadratic integral cost.

$$
\begin{aligned}
\min \quad & \int_0^{10} (x_1^2 + x_2^2 + u^2)\mathrm{d}t \\
\text{s.t.} \quad & \\
& \dot{x}_1 = (p - x_2^2)x_1 - x_2 + u \\
& \dot{x}_2 = x_1 \\
& p \in [0.8, 1.2]
\end{aligned}
$$

Fig. 7.3a shows the case with ideal case with perfect information, as a benchmark. Fig. 7.3b shows the open-loop optimization case, where a single control trajectory $\mathbf{u}^p_{[t,t+N]}$ is computed to optimize over different realizations of uncertainty. Fig. 7.3c shows the closed-loop optimization case, where different control trajectories $\{\mathbf{u}^p_{[t,t+N]}\}_{\mathcal{P}}$ are computed to optimize over different scenarios.

In the context of the classification in Fig. 7.2, traditional deterministic NMPC, min-max NMPC, stochastic NMPC etc. belongs the category of *Open-loop optimization with closed-loop implementation*. In a recent review paper, Mayne [126] argues that in the presence of uncertainty, a better strategy would be to optimize over control policies (closed-loop optimization), as done in dynamic programming, rather than over a sequence of control actions (open-loop optimization).

One such closed-loop optimization approach is the multistage scenario-based NMPC formulation, which will be considered in this thesis. In this approach, the uncertainty is discretized to get a finite number of realizations, and the future evolution of the uncertainty is propagated via a discrete scenario tree. The idea

of using a scenario tree for solving linear NMPC problems was first introduced by Scokaert and Mayne [157], with some initial results reported by Kerrigan and Maciejowski [83] and Bernardini and Bemporad [11]. This was later extended to nonlinear systems by Lucia et al. [116] in the context of multistage robust economic NMPC.

Remark 7.1. At this stage, it is important to note that the multistage scenario-based formulation considered in this work must not be confused with other scenario-based MPC approaches proposed by Calafiore and Fagiano [21], Fagiano et al. [44], Schildbach et al. [156] etc. One of the main difference between multistage scenario approach used in this work and the other scenario-based approaches is that, they compute a single control trajectory $\mathbf{u}_k$ to optimize the expected value over all the scenarios. Hence, there is no notion of feedback in the optimization problem (i.e. open-loop optimization with closed-loop implementation). Fig. 7.3b belongs to category. In contrast, the multistage scenario-based NMPC approach computes different control trajectories $\mathbf{u}_{k,j}$ for different scenarios, subject to the non-anticipativity constraints (i.e. closed-loop optimization with closed-loop implementation) [116, 157]. Fig. 7.3c belongs to this category.

Multistage scenario-based formulation as a promising alternative Based on the discussions, we now consider the multistage NMPC as one of the possible problem formulations and briefly analyze the method in light of the desirable properties listed above

- **P1. Easy to understand** - One of the most common reasons as to why NMPC under uncertainty is not used in practice is the fact that the operators often do not understand the rationale behind the control actions, due to its complexity and simply do not trust the controllers (human aspect described in Chapter 1). Forbes et al. [49] and Mayne [125] suggests that providing the predicted trajectories of key variables together with the constraints, can help operators understand the controller. The multistage scenario-based formulation makes it easier to implement this suggestion, since the use of a discrete scenario tree is easy and intuitive to understand.
- **P2. Easy to implement and maintain** - By representing the uncertainty as a scenario tree with finite number of discrete scenarios, the problem formulation does not assume any particular uncertainty distribution. One can design and implement the problem based on historical data [99](see Appendix K). Considering only a finite number of scenarios also leads to a tractable formulation. The scenario approach also provides flexibility in the problem formulation, making it easy to maintain and service. For example, scenarios that are no longer relevant (based on new information) can be easily discarded/updated, and similarly, new scenarios can be added/updated easily.
- **P3. Not too conservative** - The multistage scenario-based MPC was also shown to be less conservative than min-max MPC [116], which we will also demonstrate in the next chapter.
- **P5. Feedback** - The multistage scenario tree formulation explicitly introduces the notion of feedback in the optimization problem, which can be seen

as an *effective implementation of an optimization problem over different control policies.* This is also clearly demonstrated in Appendix H.

For the aforementioned reasons, the multistage scenario-based formulation can be considered a promising alternative for dynamic optimization under uncertainty, that particularly addresses some of the challenges discussed by Mayne [125]. Therefore, this thesis will focus on the multistage scenario-based economic NMPC formulation proposed by Scokaert and Mayne [157] and Lucia et al. [116].

The key issue that is yet to be addressed is the online computation time. A major drawback of the multistage scenario-based approach is that it leads to large optimization problems. The multistage optimization problem grows exponentially leading to very large optimization problems.

The next, and a very important research question that we answer in the subsequent chapters is:

"How can we reduce the online computation time of the multistage scenario-based formulation?"

In Chapter 9, we show that the multistage NMPC problem can be decomposed into smaller subproblems, to facilitate parallelization of the online computation. Additionally, by exploiting the NLP sensitivities, we show that the online computation time can be further reduced in Chapter 10.

7.5 Chapter summary

To summarize, in this thesis, we consider the multistage scenario-based optimization problem as an effective way of handling uncertainty in the dynamic optimization problem for the following reasons:

1. It is a closed-loop optimization approach that computes several different control trajectories instead of a single control trajectory, thereby introducing recourse action.
2. The resulting problem formulation remains tractable and does not assume any particular uncertainty characteristics, as opposed to conventional robust and stochastic NMPC, where the uncertainty characteristics heavily influence the problem formulation.
3. The problem formulation is easy to maintain (e.g. add/remove scenarios) and the use of discrete scenario tree is intuitive to understand, making it easier for practitioners to adopt this approach (human aspects).

It is important to note that there are several possible alternative formulations to economic NMPC problems under uncertainty and the multistage scenario-based formulation must be viewed as one of the several alternatives.

Chapter 8

Multistage Scenario-based Economic NMPC

The objective of this chapter is to formalize the multistage scenario-based economic NMPC problem, and show that it provides less conservative solution than the worst-case optimization approach using a gas-lift optimization problem.

Based on the articles published in Processes [91] and Computer Aided Chemical Engineering [93].

8.1 Problem formulation

In the multistage scenario-based problem formulation, the uncertainty space $\mathcal{P}$ is discretized to get M finite realizations of the uncertain parameter. The future evolution of the uncertainty in the prediction horizon is then represented by a scenario tree as shown in Fig.8.1, where a "scenario" is represented by the path from the root node to a leaf node [116]. In order to curb the exponential growth of the problem size by repeated branching at each time step, the branching is stopped after a certain number of time samples, known as robust horizon N_r, after which the uncertain parameters are treated as constants, as justified in [116]. Consequently, the number of scenarios in the NMPC[1] problem to be solved, is given by $S = M^{N_r}$. Let the set of scenarios be denoted as $\mathcal{S} := \{1, \ldots, S\}$. Allowing for the different cost weights ω_j to represent the likeliness of the different scenarios j, the resulting

[1]Remark: It should be understood that the problems considered in this thesis are economic optimization problems with nonlinear process models. However, for the sake of simplicity, with a slight abuse of notation, we use the term "NMPC" to broadly denote economic NMPC problems.

dynamic optimization problem can be formulated as,

$$\min_{\mathbf{x}_{k,j},\mathbf{u}_{k,j}} \quad \sum_{j=1}^{S} \omega_j \left[\sum_{k=0}^{N-1} \ell(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) \right] \tag{8.1a}$$

$$\text{s.t} \quad \mathbf{x}_{k+1,j} = \mathbf{f}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \boldsymbol{p}_{k,j}) \qquad \forall j \in \mathcal{S}, \forall k \in \mathcal{K} \tag{8.1b}$$

$$\mathbf{g}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \boldsymbol{p}_{k,j}) \leq 0 \qquad \forall j \in \mathcal{S}, \forall k \in \mathcal{K} \tag{8.1c}$$

$$\mathbf{x}_{0,j} = \hat{\boldsymbol{x}}_t \qquad \forall j \in \mathcal{S} \tag{8.1d}$$

$$\mathbf{x}_{k,j} \in \mathcal{X}, \quad \mathbf{u}_{k,j} \in \mathcal{U} \qquad \forall j \in \mathcal{S}, \forall k \in \mathcal{K} \tag{8.1e}$$

$$\sum_{j=1}^{S} \bar{\mathbf{E}}_j \mathbf{u}_j = \mathbf{0} \tag{8.1f}$$

where the subscript $(\cdot)_{k,j}$ represents the j^{th} scenario at time sample k. The cost function is given by $\ell(\mathbf{x}_{k,j}, \mathbf{u}_{k,j})$, and $\mathbf{g}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \boldsymbol{p}_{k,j})$ represents the nonlinear inequality constraints. Initial condition are enforced in (8.1d) for all the scenarios, where $\hat{\boldsymbol{x}}_t$ denotes the state measurements or estimates at the current time step t.

Note that we compute different control trajectories for different scenarios, which introduces the notion of feedback in the optimization problem. However, the control inputs are subject to certain restrictions which are required to capture the decision-making process accurately. These constraints are known as non-anticipativity constraints (also known as causality constraints). During the decision-making process, the control actions cannot anticipate the future evolution of the uncertainty. The control inputs predicted for time step k, must only depend only on the state trajectory for time step k. Therefore, every time a state branches, the corresponding control input is the same, i.e.

$$\mathbf{x}_{k,j_1} = \mathbf{x}_{k,j_2} \Leftrightarrow \mathbf{u}_{k,j_1} = \mathbf{u}_{k,j_2}$$

Eq.(8.1f) enforces the non-anticipativity constraints, which ensures that the states that branch at the same parent node, have the same control input. Note that $\mathbf{u}_j$ here represents the sequence of optimal control input for the j^{th} scenario, i.e. $\mathbf{u}_j = \left[\mathbf{u}_{0,j}^{\mathsf{T}} \cdots \mathbf{u}_{N-1,j}^{\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{n_u N}$ and $\bar{\mathbf{E}}_j$ is given by

$$\bar{\mathbf{E}} = \left[\begin{array}{c|c|c|c|c} E_{1,2} & -E_{1,2} & & & \\ & E_{2,3} & -E_{2,3} & & \\ & & \ddots & \ddots & \\ & & & E_{S-1,S} & -E_{S-1,S} \end{array}\right] \tag{8.2}$$

$$\Rightarrow \left[\begin{array}{c|c|c|c} \bar{\mathbf{E}}_1 & \bar{\mathbf{E}}_2 & \cdots & \bar{\mathbf{E}}_S \end{array}\right]$$

where,

$$E_{j,j+1} = \left[\begin{array}{ccc|ccc} I_{n_u} & & & 0 & \cdots & 0 \\ & \ddots & & \vdots & \ddots & \vdots \\ & & I_{n_u} & 0 & \cdots & 0 \end{array}\right] \tag{8.3}$$

If $n_{o,(j,j+1)}$ denotes the number of common nodes between two consecutive scenarios j and $j+1$, then $E_{j,j+1} \in \mathbb{R}^{r \times n_u N}$ and $\bar{\mathbf{E}}_j \in \mathbb{R}^{q \times n_u N}$ where,

$$q = n_u \sum_{j=1}^{S} n_{o,(j,j+1)} \quad \text{and} \quad r = n_u n_{o,(j,j+1)}$$

as described in [86]. Formulating the non-anticipativity constraints using this chain structure also results in sparse structures, which may be exploited by many solvers [86].

It is important to note that, the non-anticipativity constraint ensures that the control input computed at the first time step is identical for all the scenarios, i.e.

$$\mathbf{u}_{0,1} = \mathbf{u}_{0,2} = \cdots = \mathbf{u}_{0,S} \tag{8.4}$$

This is a very important property, since this enables closed-loop implementation of the multistage NMPC formulation in a receding horizon fashion.

The scenario tree with $M = 3$ discrete realization of parameters, and a robust horizon of $N_r = 2$ samples, resulting in a tree with $S = 9$ scenarios is schematically represented in Fig. 8.1, for both the state and control trajectories. This also clearly shows the concept of non-anticipativity constraints, where the control input corresponding to the states that branch at the same node are equal.

To this end, the first step to designing a multistage scenario-based NMPC is to select the discrete realizations of the uncertainty from the uncertainty set. Common practice is to select a combination of maximum, minimum and nominal values of the different uncertain parameters [116]. It is important to note that, in order to ensure robust constraint satisfaction, the worst-case scenario must be included as one of the scenarios. For now, we assume that the M discrete realizations of the uncertainty are given. The scenario selection problem will be considered in more detail in Chapter K, where we propose a novel approach to generate the scenario tree based on principal component analysis (PCA), that exploits the hidden data structures.

8.2 Methods and tools

In the rest of the thesis, the multistage scenario-based optimization problem is discretized using third order direct collocation unless otherwise stated explicitly. This gives a polynomial approximation of the system model. The set of three collocation points and the initial state in each interval $[k, k+1]$ is denoted by the index $c \in \mathcal{C} := \{0, 1, 2, 3\}$, and the location of these points are computed using the Radau scheme (see [15]). The discretized states are simply denoted by $\mathbf{x}_k \triangleq \mathbf{x}_k^0$ and the states at the other collocation points (known as helper states) are given by $\mathbf{x}_k^c \in \mathbb{R}^{n_x}$ for all $c \in \{1, 2, 3\}$. To ensure continuity of the states between two consecutive time intervals, the final state variables $\mathbf{x}_k^3$ and the initial conditions of the next time interval $\mathbf{x}_{k+1} \triangleq \mathbf{x}_{k+1}^c$ must be equal. This is commonly referred to as shooting gap constraints, and are additional constraints appended to the NLP problem (7.2). The control inputs $\mathbf{u}_k$ which are discretized at each sampling interval, are assumed to be piece-wise constant over each interval and hence are not

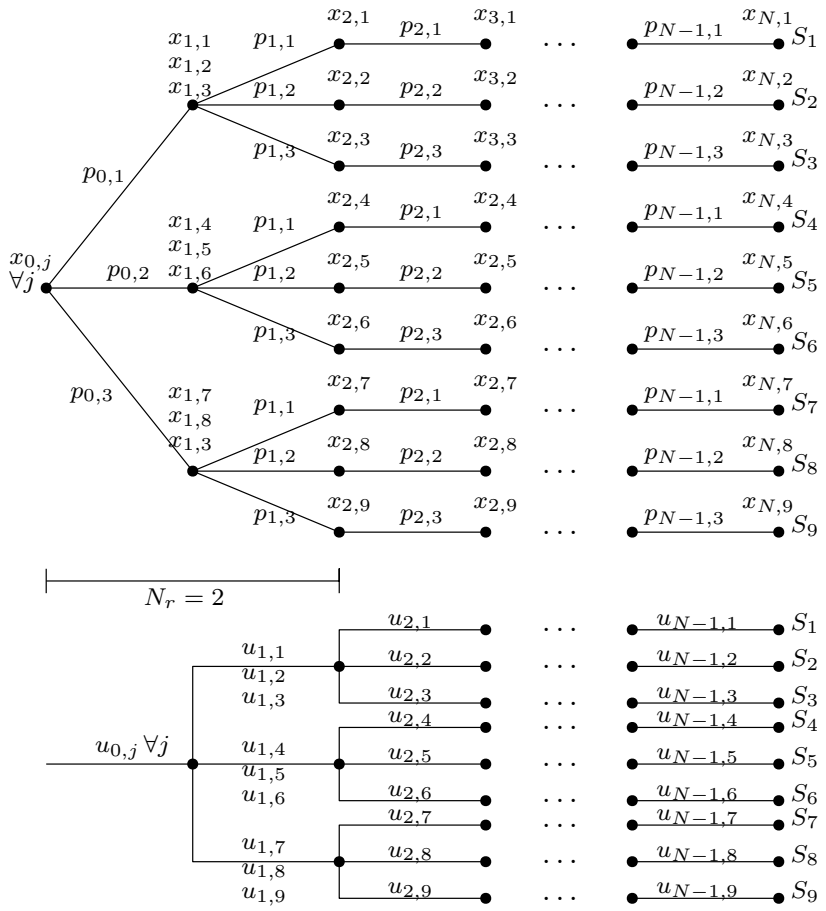


Figure 8.1: Schematic representation of a scenario tree for $M = 3$ discrete realization of parameters and a robust horizon of $N_r = 2$ samples.

discretized at the collocation points. The parameters $\boldsymbol{p}$ are also treated as piecewise constant over each interval $[k, k+1]$. For the sake of generality, the collocation points, and the shooting gap constraints will not be shown explicitly in any of the OCP formulations in this thesis.

The resulting NLP problem is implemented in `CasADi` version 3.4.5 [4] using the `MATLAB` programming environment. The resulting nonlinear programming problem was solved using `IPOPT` version 3.12.2 [184] running with a `MUMPS` linear solver. All the computation times reported in this thesis are based on simulations carried out on an Intel core i-7, 2.6GHz workstation with 16GB memory. The plant simulator was implemented using the `IDAS` integrator[68] for differential algebraic equation (DAE) models and the `CVODES` integrator [159] for ordinary differential equation (ODE) models.

8.3 Application example - Gas lift optimization

In this section, we demonstrate the multistage NMPC approach using a network of two gas lifted oil wells ($n_w = 2$) producing to a common riser manifold as shown in Fig. A.1. The process is assumed to be constrained by a maximum gas capacity of $w_{g_{Max}} = 8kg/s$. The uncertainty arises in the form of the gas-oil-ratio (GOR) from the reservoir (uncertainty in the feed). The optimization problem is expressed as,

$$\min_{w_{gl,i}} \; -c_o \sum_{i=1}^{n_w} w_{po,i} + c_{gl} \sum_{i=1}^{n_w} w_{gl,i} \tag{8.5a}$$

s.t.

$$\sum_{i=1}^{n_w} w_{pg,i} \leq w_{pg}^{max} \tag{8.5b}$$

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, GOR) \tag{8.5c}$$

$$GOR \in \mathcal{P} \tag{8.5d}$$

$$\forall i \in \{1, \ldots, n_w\}$$

where $w_{gl,i}$ is the gas lift injection rate for each well and is the manipulated variable ($n_u = 2$), $w_{po,i}$ and $w_{pg,i}$ are the oil and gas production rates from each well respectively, w_{pg}^{max} is the total gas processing capacity. c_o and c_{gl} are economic terms that represents the value of oil and cost of gas compression respectively. (8.5c) represents the nonlinear dynamic model of the gas lifted wells discretized using direct collocation. The reader is referred to Appendix A for more detailed description of the gas lifted well models. $\mathcal{P}$ denotes the uncertainty characteristics to which the uncertain parameter is known to belong. Since we have two wells, we have two uncertain parameters ($n_p = 2$), namely, the gas-oil ratio for each well. The resulting NLP problem was solved with a prediction horizon $N = 60$ and a sampling time of $T_s = 300$ s. The first control input is then applied to the plant.

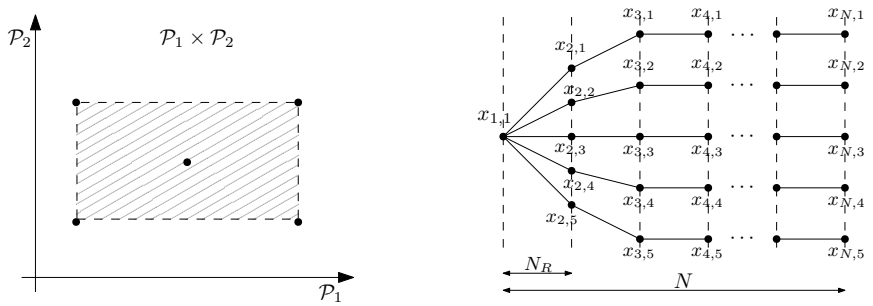
In this simulation example, we consider the uncertainty in $GOR_i \in \mathcal{P}_i$ to be equally distributed with $\mathcal{P}_1 \in [0.05, 0.15]$kg/kg and $\mathcal{P}_2 \in [0.11, 0.13]$kg/kg. Hence, the uncertainty set $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2$ is given as a box uncertainty set.

For the nominal optimization case, the nominal value of the uncertain parameter GOR was used in the optimization problem, and, for the worst-case optimization, the maximum values of the gas-oil-ratio of the two wells was used in the optimization problem. For the multistage scenario NMPC, $M = 5$ discrete realizations of the uncertainty are considered that corresponds to the combination of minimum, maximum and nominal values of the gas-oil-ratio of the two wells (see Table 8.1), and a robust horizon of $N_R = 1$ was chosen. This is schematically represented in Fig. 8.2.

The optimization problem considered here computes the optimal gas lift rate for each well. We assume that we have perfect low level controllers that adjust the gas lift choke z_{gl} to provide the desired gas lift rates. These assumptions are justified, since the main focus in this chapter is to compare the nominal, worst-case and multistage scenario-based optimization approaches. An Extended Kalman Filter (EKF) was implemented for state feedback. The annulus pressure, well head

Table 8.1: The GOR value used in the optimizer for nominal, worst-case and scenario based approach

	Nominal	worst-case	multistage scenario-based				
GOR well 1	0.1	0.15	0.05	0.05	0.1	0.15	0.15
GOR well 2	0.12	0.13	0.11	0.13	0.12	0.11	0.13



(a) (b)

Figure 8.2: (a) Uncertainty subspace and the possible models for the scenario tree denoted by •, (b) Scenario tree with $N_R = 1$ and $M = 5$ models $\Rightarrow S = 5^1 = 5$ scenarios.

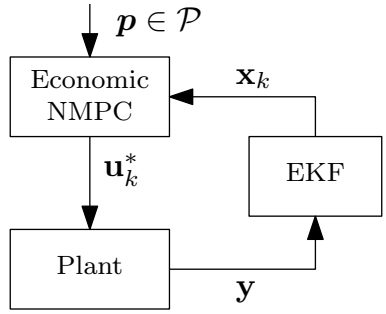


Figure 8.3: Block diagram of the implemented control structure including the EKF for state estimation.

pressure, bottom hole pressure, wellhead choke flow rate and gas inflow rate are all assumed to be measured and used as measurements in the EKF. An economic NMPC control structure was chosen together with an EKF for state estimation as shown in Fig.8.3. The simulation starts with the true GOR the same as the nominal GOR for both the wells. At sampling instant $N = 15$, true GOR gradually increases to 0.125 in well 1 and well 2, respectively, and remains constant at these values until $N = 45$. At sampling instant $N = 45$, the GOR suddenly increases to 0.15 and 0.13 (worst-case realization) in wells 1 and 2, respectively. The true GOR profile is shown is Figure 8.4 bottom subplot.

8.3.1 Simulation results

Nominal optimization

The system is first simulated for the nominal optimization case, where the optimization assumes the GOR to be at its nominal value as shown in Table 8.1. When the true GOR is at its nominal value, there is no plant-model mismatch and the total gas capacity constraint is active, as expected. However, when the true GOR in the system increases, this leads to constraint violations, as seen in Fig. 8.4 in green curves.

Worst-case optimization

Then, the system is simulated with the worst-case optimization, where the optimizer assumes GOR to take its worst-case value. When the true GOR is at its nominal value, we see that the optimal solution implemented is rather conservative. The gas capacity constraint is no longer active, and there is spare capacity that can be utilized. When the GOR increases, we see that the constraints are not violated, even when the GOR does take its worst-case value at $N \geq 45$. This is clearly shown in Fig. 8.4 (red curves). The solution is robust feasible at the cost of conservativeness.

Multistage scenario-based optimization

Finally, the system is simulated with the multistage scenario-based optimization with five different GOR values as shown in Table 8.1. All the scenarios are assumed equally probable and are therefore provided with equal weights for all the scenarios. When the GOR is at its nominal value, the optimizer solves for the optimal inputs that are feasible for all the possible scenarios, and we see that the gas capacity constraint is not active. However, the solution is less conservative than the worst-case optimization, as clearly seen in Fig. 8.4 (blue curves).

As the GOR increases, the implemented solution proves to be robust feasible, and the constraints are satisfied even when the GOR takes its worst-case value. However, when the true GOR assumes its worst-case value, the total oil produced is less than the worst-case optimization. This is due to the fact that there is no plant-model mismatch in the worst-case optimization case, whereas in the scenario tree optimization, the optimal solution is computed that maximizes the oil rate for the other scenarios in addition to the worst-case scenario.

The overall reduced conservativeness is due to the recourse action in the multistage formulation, which can be clearly seen in Fig. 8.5, which shows the closed-loop implemented solution along with the predicted trajectories. Note that Fig. 8.5 shows only the multistage formulation (blue) and the worst-case formulation (red). Non-anticipativity constraints are also clearly seen in Fig. 8.5 (subplots 2 and 3), where the first control sample is the same for all the scenarios.

Monte-Carlo simulations

To compare the performance of the nominal, worst-case and scenario optimization methods, we now present a Monte Carlo simulation with 35 simulations. The GOR

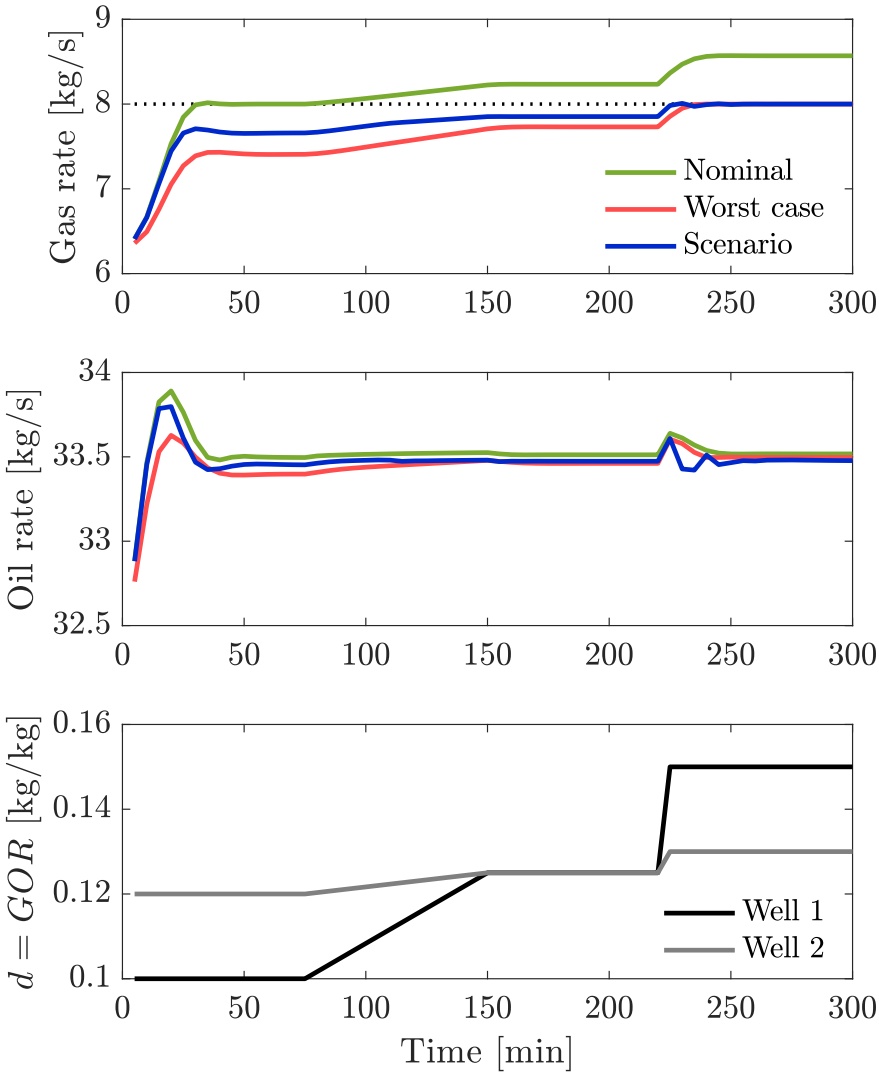


Figure 8.4: Closed-loop implemented solution for nominal optimization (green), worst-case optimization (red) and multistage scenario-based optimization (blue). Top subplot shows the total gas rate along with the max gas processing constraint, middle subplot shows the total oil production and bottom subplot shows the GOR disturbance for well 1 (black) and well 2 (gray).

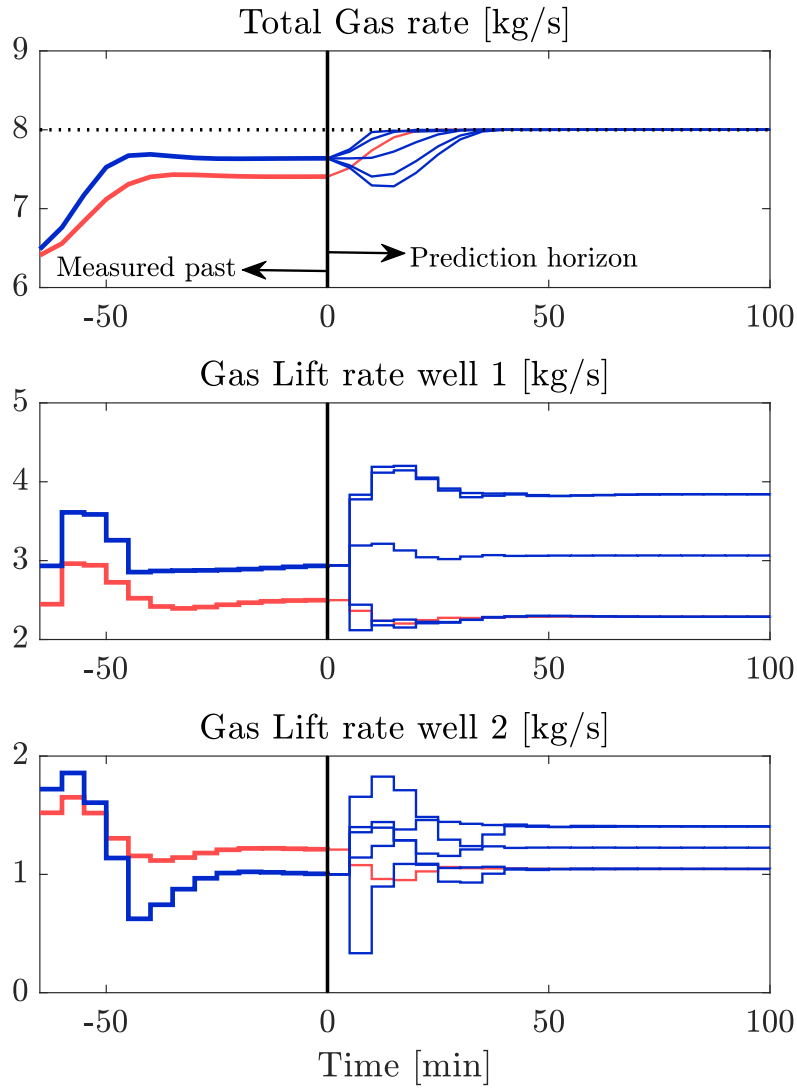


Figure 8.5: Closed-loop implemented solution, along with the first 100 min of predicted trajectories for worst-case optimization (red) and multistage scenario-based optimization (blue), demonstrating the recourse introduced by the multistage approach.

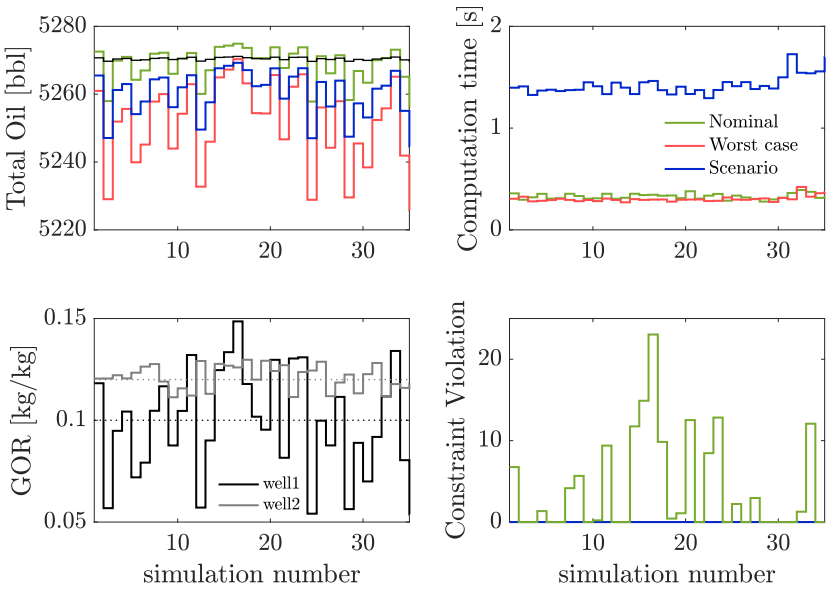


Figure 8.6: Monte Carlo Simulation results each with a simulation time of 75min

values used in the nominal, worst-case and scenario optimization are summarized in Table 8.1. For each Monte Carlo simulation, the true GOR used in the simulator was randomly picked from a uniform distribution $\mathcal{P}$ and is shown in Fig.8.6. Each Monte Carlo simulation was run for 75min (15 samples) and the total accumulated oil over 75min (integrated objective) is compared.

As a benchmark, the results from the ideal-case are also plotted in the top left subplot (shown in solid black lines), which corresponds to the case where we have perfect information about all the parameters. The nominal optimization seems to produce more oil than the ideal-case in some simulations, however the total gas capacity constraints are violated in these simulations which demonstrates that nominal optimization can lead to infeasibility. The constraint violations for all the cases plotted in Fig.8.6 shows that the constraints are violated only in the nominal optimization case.

However, it is important to note that this is not a general conclusion for the multistage scenario-based optimization. The multistage approach can be constraint feasible if at least one of the scenarios corresponds to the worst-case scenario, which was the case in this simulation study (see Table.8.1). Fig.8.6 shows that the worst-case and scenario optimization were robust feasible at the cost of conservativeness. However, the scenario optimization was significantly less conservative than the worst-case optimization. On average, the scenario optimization was less conservative than the worst-case optimization. The average computation time for nominal, worst-case and scenario optimization are also compared and we see that the improved performance of the scenario optimization comes at the cost of higher computation time.

8.4 Chapter summary

In this chapter, we formalized the multistage scenario NMPC problem in (8.1) and introduced the concept of non-anticipativity constraints, which enables closed-loop implementation. Using a simulation case study, we showed that the multistage scenario-based optimization provides a robust, yet less conservative solution compared to nominal optimization and worst-case optimization respectively. However, the computation time of the multistage scenario approach was significantly higher than the nominal and worst-case formulations. In the next two chapters, we will study how we can reduce the computation time using decomposition methods.

Chapter 9

A Primal Decomposition Algorithm for Distributed Multistage Scenario Model Predictive Control

This paper proposes a primal decomposition algorithm for efficient computation of multistage scenario model predictive control. Since the different scenarios are only coupled via the non-anticipativity constraints, the different scenarios can be decomposed into smaller subproblems using primal decomposition and solved iteratively using a master problem to coordinate the subproblems. We argue in favor of primal decomposition algorithms, since it ensures feasibility of the non-anticipativity constraints throughout the iterations, which is crucial for closed-loop implementation.

Based on the article published in Journal of Process Control [101] and presented as a Keynote talk at IFAC ADCHEM 2018 [94].[1]

9.1 Introduction

In Chapter 8, the multistage scenario-based NMPC framework was presented, where the uncertainty propagation was represented via a discrete scenario tree. This was shown to be less conservative than the worst-case optimization problem, due to the recourse action.

One of the main drawbacks of this approach is that the problem size grows exponentially with 1) the number of uncertain parameters, 2) the number of uncertainty realizations, and 3) the length of the prediction horizon. As justified by Lucia et al. [116], one way to curb the exponential growth of the problem size is by considering the branching of the tree only up to a certain number of samples in the prediction horizon, known as robust horizon. Another solution is to exploit the structure of the problem to decompose the problem into several smaller subproblems. The different scenarios are independent and additively separable, except for

[1] Finalist of the IFAC Young Author Award

the non-anticipativity constraints, which couples the different scenarios together. To this end, decomposition methods can be used to solve the different scenarios independently and later use a master problem to iteratively co-ordinate the different subproblems.

Scenario decomposition using dual decomposition methods were proposed by Lucia et al. [117] and Martí et al. [124], where the different subproblems are solved by relaxing the non-anticipativity constraints. A master problem then updates the Lagrangian multipliers corresponding to the non-anticipativity constraints iteratively. Therefore, the non-anticipativity constraints are feasible only upon convergence of the master and subproblem iterations. Dual decomposition methods may require a relatively large number of iterations between the master problem and the subproblems to converge. This leads to challenges with practical implementation as noted by Martí et al. [124]. If the iterations between the master problem and the subproblems do not converge within the required sample time of the NMPC, the non-anticipativity constraints remains infeasible. As a result, the different scenarios may give different optimal control inputs at the first sample, which is not acceptable for real-time closed-loop implementation.

To overcome the closed-loop implementation issue with dual decomposition, we propose a primal decomposition algorithm for scenario decomposition [94][101]. In contrast to dual decomposition, primal decomposition produces a primal feasible solution with monotonically decreasing objective value at each iteration [12]. Therefore, primal decomposition ensures that the non-anticipativity constraints are *always* feasible through out the iterations. This implies that even if the master problem and subproblem iterations are prematurely terminated, the non-anticipativity constraints are still feasible and the first control input provided by all the scenarios are the same (cf. (8.4)). As noted earlier, this is an important property for closed-loop implementation of the multistage scenario NMPC problem.

The main contribution of this chapter is a distributed framework for the multistage NMPC problem based on primal decomposition. In addition, we also introduce a novel back-tracking algorithm to select a suitable step-length in the master problem update in order to ensure the feasibility of the nonlinear constraints in the different subproblems. The proposed primal decomposition approach and the novel backtracking algorithm are demonstrated using a continuously stirred tank reactor (CSTR) case study.

The remainder of the chapter is organized as follows. Section 9.2 describes the different decomposition approaches for scenario decomposition. Section 9.3 introduces a novel backtracking algorithm for choosing a suitable step length in the master problem. The proposed method is then demonstrated using a CSTR case study in Section 9.4. Finally, discussions are provided in Section 9.5 before concluding the chapter in Section 9.6

9.2 Distributed multistage scenario MPC

The multistage scenario MPC problem (8.1) consists of S independent MPC problems, except for the non-anticipativity constraints (8.1f), which couple the different scenarios together. Different decomposition approaches can be used to split the

multistage scenario problem into smaller subproblems and use a master problem to co-ordinate the different subproblems.

9.2.1 Using Dual Decomposition [124]

Scenario decomposition using dual decomposition approaches is the most common strategy. Here, each scenario subproblem is solved by relaxing the non-anticipativity constraints, cf. [117], [124] and [86].

The scenario optimization problem (8.1) with the relaxed non-anticipativity constraints is expressed as,

$$\min_{\mathbf{x}_{k,j},\mathbf{u}_{k,j}} \quad \sum_{j=1}^{S}\left[\omega_j \sum_{k=0}^{N-1} \ell(\mathbf{x}_{k,j},\mathbf{u}_{k,j})\right] + \boldsymbol{\lambda}^{\mathsf{T}} \sum_{j=1}^{S} \bar{\mathbf{E}}_j \mathbf{u}_j \tag{9.1a}$$

s.t.

$$\mathbf{x}_{k+1,j} = \mathbf{f}(\mathbf{x}_{k,j},\mathbf{u}_{k,j},\boldsymbol{p}_{k,j}) \qquad \forall j \in \mathcal{S}, \forall k \in \mathcal{K} \tag{9.1b}$$

$$\mathbf{g}(\mathbf{x}_{k,j},\mathbf{u}_{k,j},\boldsymbol{p}_{k,j}) \leq 0 \qquad \forall j \in \mathcal{S}, \forall k \in \mathcal{K} \tag{9.1c}$$

$$\mathbf{x}_{0,j} = \hat{\boldsymbol{x}}_t \qquad \forall j \in \mathcal{S} \tag{9.1d}$$

$$\mathbf{x}_{k,j} \in \mathcal{X} \qquad \mathbf{u}_{k,j} \in \mathcal{U} \tag{9.1e}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^q$ is the Lagrange multiplier corresponding to the non-anticipativity constraint (8.1f). Eq. (9.1a) is now additively separable in $\mathbf{x}$ and $\mathbf{u}$ and each j^{th} scenario subproblem can be reformulated as a function of $\boldsymbol{\lambda}$ as shown below,

$$\Gamma(\boldsymbol{\lambda},\boldsymbol{p}_j) := \min_{\mathbf{x}_{k,j},\mathbf{u}_{k,j}} \quad \omega_j \sum_{k=0}^{N-1} \ell(\mathbf{x}_{k,j},\mathbf{u}_{k,j}) + \boldsymbol{\lambda}^{\mathsf{T}} \bar{\mathbf{E}}_j \mathbf{u}_j \tag{9.2a}$$

s.t.

$$\mathbf{x}_{k+1,j} = \mathbf{f}(\mathbf{x}_{k,j},\mathbf{u}_{k,j},\boldsymbol{p}_{k,j}) \qquad \forall j \in \mathcal{S}, \forall k \in \mathcal{K} \tag{9.2b}$$

$$\mathbf{g}(\mathbf{x}_{k,j},\mathbf{u}_{k,j},\boldsymbol{p}_{k,j}) \leq 0 \qquad \forall j \in \mathcal{S}, \forall k \in \mathcal{K} \tag{9.2c}$$

$$\mathbf{x}_{0,j} = \hat{\boldsymbol{x}}_t \qquad \forall j \in \mathcal{S} \tag{9.2d}$$

$$\mathbf{x}_{k,j} \in \mathcal{X} \qquad \mathbf{u}_{k,j} \in \mathcal{U} \tag{9.2e}$$

The subproblems are solved independently, and the Lagrange multiplier $\boldsymbol{\lambda}$ is iteratively updated in the master problem. Under very specific conditions, the non-anticipativity constraints only become feasible upon convergence of $\boldsymbol{\lambda}$, provided the duality gap vanishes.

The implication of this is that, relaxing the non-anticipativity constraints may impede real-time closed-loop implementation. In the receding horizon control framework, at each time step, the first control move is implemented in the plant. In multistage scenario MPC, the non-anticipativity constraints ensure that the first control move is equal for all the scenarios. However, if the master problem and subproblems fail to converge within the required sampling time, the non-anticipativity constraints are not satisfied. Consequently, the first control input computed by the different scenarios are different, thus impeding closed-loop implementation.

One way to address this problem is to take a weighted average of the manipulated inputs at the first sample, based on the probabilities of the different scenarios [149]. However, this may not be a good approach since the weighted average can lead to an infeasible solution. Martí et al. [124] proposed to compute an average of the control inputs at the first sample such that the worst-case constraint violation for the local subproblems is minimized, which is given by solving an additional linear programming (LP) problem.

In this thesis, we instead propose a primal decomposition approach to solve this problem, which always ensures the feasibility of the non-anticipativity constraints.

9.2.2 Using Primal Decomposition - Proposed Approach

To address the problem of non-anticipativity constraint feasibility, we propose a primal decomposition algorithm, which always produces a primal feasible point by iterating directly on the coupling variables [12]. Therefore, at any point in time, the non-anticipativity constraints are always feasible and the first control move provided by all the scenarios are the same, which is an important property for closed-loop implementation as described earlier.

The primal subproblem for the j^{th} subproblem can be written by introducing a new auxiliary variable $\mathbf{t}_l \in \mathbb{R}^{n_u}$, $\forall l \in \{1, \dots, \sum_{m=1}^{N_r} M^{m-1}\}$ for each non-anticipativity constraints. Note that the number of non-anticipativity constraints is given by $\sum_{m=1}^{N_r} M^{m-1}$. Each scenario subproblem is then expressed as a function of the auxiliary variables as shown below,

$$
\begin{aligned}
\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j) := \min_{\mathbf{x}_{k,j}, \mathbf{u}_{k,j}} \quad & \sum_{k=0}^{N-1} \ell(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) && && \text{(9.3a)}\\
\text{s.t} \quad & \mathbf{x}_{k+1,j} = \mathbf{f}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \boldsymbol{p}_{k,j}) && \forall j \in \mathcal{S}, \forall k \in \mathcal{K} && \text{(9.3b)}\\
& \mathbf{g}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \boldsymbol{p}_{k,j}) \leq 0 && \forall j \in \mathcal{S}, \forall k \in \mathcal{K} && \text{(9.3c)}\\
& \mathbf{x}_{0,j} = \hat{\boldsymbol{x}}_t && \forall j \in \mathcal{S} && \text{(9.3d)}\\
& \bar{\mathbf{E}}_j \mathbf{u}_j = \bar{\boldsymbol{\tau}}_j && \forall j \in \mathcal{S} && \text{(9.3e)}\\
& \mathbf{x}_{k,j} \in \mathcal{X} \qquad \mathbf{u}_{k,j} \in \mathcal{U} && && \text{(9.3f)}
\end{aligned}
$$

where $\bar{\boldsymbol{\tau}}$ is given by

$$
\bar{\boldsymbol{\tau}} = \left[\begin{array}{c|c|c|c|c} \tau_{1,2} & -\tau_{1,2} & & & \\ & \tau_{2,3} & -\tau_{2,3} & & \\ & & \ddots & \ddots & \\ & & & \tau_{S-1,S} & -\tau_{S-1,S} \end{array}\right] \tag{9.4}
$$

$$
= \left[\begin{array}{c|c|c|c} \bar{\boldsymbol{\tau}}_1 & \bar{\boldsymbol{\tau}}_2 & \cdots & \bar{\boldsymbol{\tau}}_S \end{array}\right] \tag{9.5}
$$

and $\tau_{j,j+1} \in \mathbb{R}^{n_u n_{o,(j,j+1)}}$ is a matrix that is composed of the auxiliary variables $\boldsymbol{t}_l \in \mathbb{R}^{n_u}$. Let $\boldsymbol{X}^*(\boldsymbol{t}_l, \boldsymbol{p}_j)$ and $\boldsymbol{\lambda}^*(\boldsymbol{t}_l, \boldsymbol{p}_j)$ respectively, denote the optimal primal and dual solution of the j^{th} subproblem (9.3).

The master problem to update the auxiliary variables $\boldsymbol{t}_l$ is then given by,

$$\min_{\boldsymbol{t}_l} \sum_{j=1}^{S} \boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j) \tag{9.6}$$

The simplest approach to update $\boldsymbol{t}_l$ is to use the subgradient method, where $\boldsymbol{t}_l$ is updated along the descent direction [20]. The search direction for the master problem (9.6) is given by the subgradient

$$\sigma = \sum_{j=1}^{S} \nabla_{\boldsymbol{t}_l} \boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j)$$

These are simply the Lagrange multipliers that corresponds to the non-anticipativity constraints (9.3e), which are computed by solving the different scenario subproblems [20]. Hence the subgradient for the master problem is essentially available for "free" without the need for any additional computations.

$\boldsymbol{t}_l$ is iteratively updated along the search direction with a suitable step length α given by

$$\boldsymbol{t}_l^{(v+1)} = \boldsymbol{t}_l^{(v)} + \alpha_l (\sum_{j=1}^{S} \nabla_{\boldsymbol{t}_l} \boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j))^{(v)}, \qquad \forall l \in \{1, \dots, \sum_{m=1}^{N_r} M^{m-1}\} \tag{9.7}$$

where (v) denotes the v^{th} iteration as explained in [12, 19, 20].

One commonly used stopping criteria for the master problem and subproblem iterations is that the change in $\boldsymbol{t}_l$ between two subsequent iterations, denoted by $\Delta \boldsymbol{t}_l = \|\boldsymbol{t}_l^{(v+1)} - \boldsymbol{t}_l^{(v)}\|$ must be less than a certain tolerance ϵ.

The distributed scenario MPC using primal decomposition is schematically represented in Fig. 9.2.

To summarize, the solution to the multistage scenario-MPC problem (8.1) can be reached by iteratively solving the smaller subproblems (9.3) with auxiliary primal variables $\boldsymbol{t}_l$, which are then updated using the gradient descent step (9.7). For convex and differentiable problems, it can be shown that solving the primal decomposition problem is equivalent to solving the original centralized problem (see Appendix I). In any case, by using auxiliary primal variables $\boldsymbol{t}_l$, the non-anticipativity constraints always remain feasible throughout the iterations, which is crucial for closed-loop implementation.

To illustrate this, consider a scenario tree with $M = 3$, $N_r = 2$ and $S = 9$ as shown in Fig.8.1. For such a tree $l = 4$ and $\boldsymbol{t}_l \in \{\boldsymbol{t}_1^\mathsf{T}, \boldsymbol{t}_2^\mathsf{T}, \boldsymbol{t}_3^\mathsf{T}, \boldsymbol{t}_4^\mathsf{T}\}$. The non-anticipativity constraints for the scenario tree is such that the control input for all the scenarios at the first control sample is the same.

$$\mathbf{u}_{1,j} = \mathbf{u}_{1,j+1} = \boldsymbol{t}_1, \qquad \forall j \in \{1, \dots, S-1\} \tag{9.8}$$

where $S - 1 = 8$ in this case.

The non-anticipativity constraints at the second time sample is then given by

$$\begin{aligned} \mathbf{u}_{2,1} &= \mathbf{u}_{2,2} = \mathbf{u}_{2,3} = \boldsymbol{t}_2 \\ \mathbf{u}_{2,4} &= \mathbf{u}_{2,5} = \mathbf{u}_{2,6} = \boldsymbol{t}_3 \\ \mathbf{u}_{2,7} &= \mathbf{u}_{2,8} = \mathbf{u}_{2,9} = \boldsymbol{t}_4 \end{aligned} \tag{9.9}$$

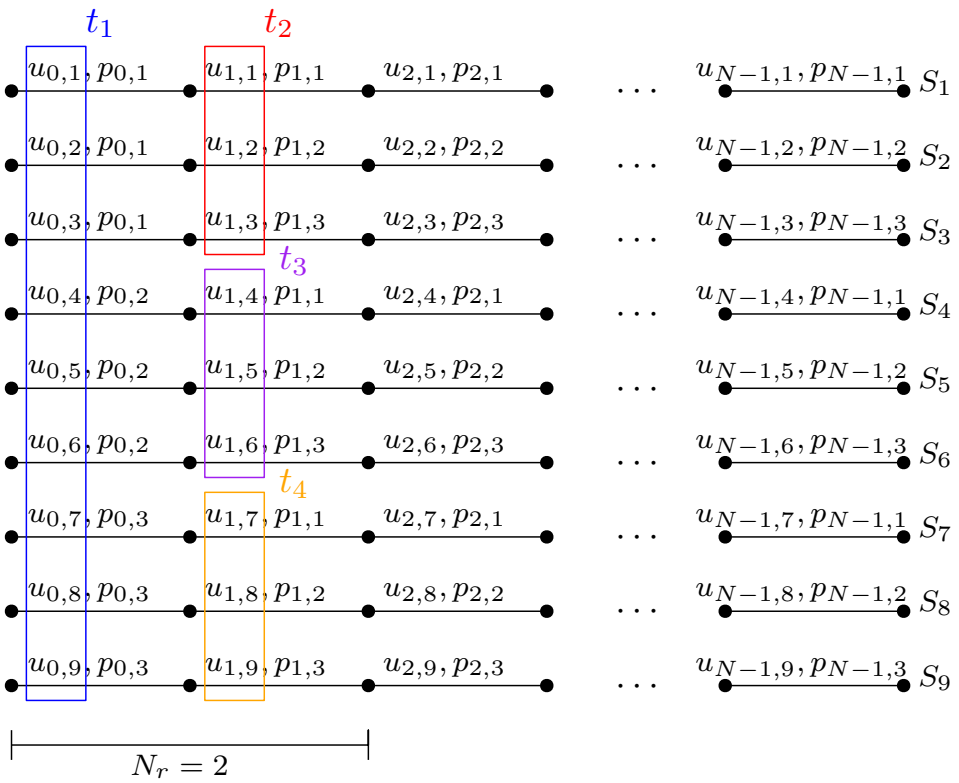


Figure 9.1: Schematic representation of the decomposed scenarios showing the non-anticipativity constraints enforced using the auxiliary variables $\boldsymbol{t}_l$.

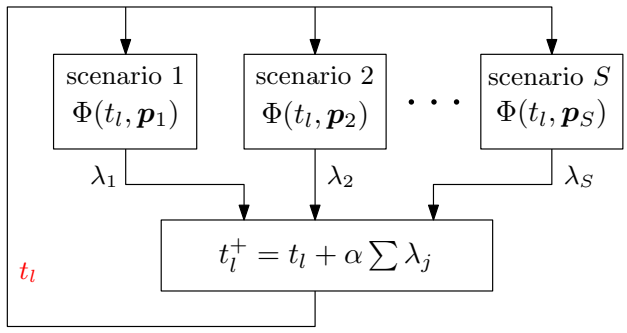


Figure 9.2: Schematic representation of the distributed scenario MPC using primal decomposition

This is schematically represented in Fig. 9.1. The auxiliary variables are then updated in the master problem using the gradient descent step (9.7),

$$\boldsymbol{t}_1^+ = \boldsymbol{t}_1 + \alpha_1 \sum_{j=1}^{9} \nabla_{\boldsymbol{t}_1} \Phi$$

where $\sum_{j=1}^{9} \nabla_{\boldsymbol{t}_1} \Phi = \sum_{j=1}^{9} \lambda_{1,j}$

$$\boldsymbol{t}_2^+ = \boldsymbol{t}_2 + \alpha_2 \sum_{j=1}^{9} \nabla_{\boldsymbol{t}_2} \Phi$$

where $\sum_{j=1}^{9} \nabla_{\boldsymbol{t}_2} \Phi = \lambda_{2,1} + \lambda_{2,2} + \lambda_{2,3}$

$$\boldsymbol{t}_3^+ = \boldsymbol{t}_3 + \alpha_3 \sum_{j=1}^{9} \nabla_{\boldsymbol{t}_3} \Phi$$

where $\sum_{j=1}^{9} \nabla_{\boldsymbol{t}_3} \Phi = \lambda_{2,4} + \lambda_{2,5} + \lambda_{2,6}$

$$\boldsymbol{t}_4^+ = \boldsymbol{t}_4 + \alpha_4 \sum_{j=1}^{9} \nabla_{\boldsymbol{t}_4} \Phi$$

where $\sum_{j=1}^{9} \nabla_{\boldsymbol{t}_4} \Phi = \lambda_{2,7} + \lambda_{2,8} + \lambda_{2,9}$

9.3 Back-tracking algorithm

As mentioned in the previous section, when using primal decomposition, we solve the subproblems by fixing the manipulated inputs for the non-anticipativity constraints to be equal to an auxiliary variable $\boldsymbol{t}_l$ for all the scenarios by means of the equality constraint (9.3e). The auxiliary variable $\boldsymbol{t}$ is then iteratively updated using a step length α as shown in (9.7). The equality constraint (9.3e) ensures that the non-anticipativity constraints are always feasible throughout the iterations. However, if the step length α is not suitably chosen, then the nonlinear constraints $\mathbf{g}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \boldsymbol{p}_{k,j}) \leq 0$ may become infeasible by fixing the control input at $\boldsymbol{t}_l^+$ using the equality constraint (9.3e). Choosing a step length too small on the other hand leads to a very slow convergence. Hence, careful selection of the step length α is important in the presence of nonlinear constraints.

Therefore, in this chpater, we propose a feasibility ensuring backtracking algorithm to suitably choose the step length α, such that we can choose a sufficiently large step length and backtrack when required to ensure that the nonlinear constraints $\mathbf{g}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \boldsymbol{p}_{k,j}) \leq 0$ remain feasible throughout iterations. The proposed backtracking algorithm is based on a forward integration of the system dynamics and the nonlinear constraints one time step ahead using the proposed step length. In other words, a one-step-ahead model prediction for each scenario using the prospective $\boldsymbol{t}_l^+$ is used to evaluate the nonlinear constraint feasibility before $\boldsymbol{t}_l^+$

is fixed in the subproblems using the equality constraint (9.3e) in the next iteration. The prospective $\boldsymbol{t}_l^+$ is given by $\boldsymbol{t}_l^+ = \boldsymbol{t}_l + \alpha\sigma$, where $\sigma = (\sum_{j=1}^{S} \nabla_{\boldsymbol{t}_l} \boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j))$ is the search direction or the subgradient. The step length α is backtracked until the nonlinear constraints in the one-step-ahead model prediction is feasible. This is illustrated in Algorithm 9.1.

Algorithm 9.1 Feasibility-ensuring backtracking algorithm

Define $c < 1$.

Input: at each iteration between master problem and subproblem: initial state $\hat{\boldsymbol{x}}$, initial α_0, $\boldsymbol{t}$ and subgradient $\sigma = \sum_{j=1}^{S} \nabla_{\boldsymbol{t}} \boldsymbol{\Phi}(\boldsymbol{t}, \boldsymbol{p}_j)$

for $j = 1, 2, \ldots, S$ **do**
  $\alpha \leftarrow \alpha_0$
  Evaluate $\mathbf{x}_{k+1,j} = \mathbf{f}(\hat{\boldsymbol{x}}, (\boldsymbol{t} + \alpha\sigma), \boldsymbol{p}_{k,j})$,
$\forall k \in \{1, \ldots, N_r\}$
  while $\mathbf{g}(\mathbf{x}_{k+1,j}, (\boldsymbol{t} + \alpha\sigma), \boldsymbol{p}_{k,j}) > 0$ **do**
    $\alpha \leftarrow c\alpha$
  end while
end for

Output: α

Theorem 9.1. *Suppose the multistage scenario-based MPC problem (8.1) has a feasible optimal solution and the gradient descent step has an initial feasible point $\boldsymbol{t} \in \mathcal{F}$, then $\exists\ \alpha \in [0, \overline{\alpha}]$ such that*

$$\boldsymbol{t} + \alpha\sigma \in \mathcal{F} \quad \forall 0 < \alpha \leq \overline{\alpha} \tag{9.10}$$

Proof. As shown in [111, Ch.9], the subgradient σ for the master problem is feasible by construction[2]. Hence there is a maximum value $\overline{\alpha}$ such that (9.10) holds. □

Theorem 9.1 shows that the proposed backtracking algorithm always converges. The task of the feasibility ensuring backtracking algorithm is then to find the upper bound $\overline{\alpha}$ using Algorithm 9.1 such that the updated value $\boldsymbol{t}^+ = \boldsymbol{t} + \alpha\sigma$ is feasible. The availability of an initial feasible guess for $\boldsymbol{t}_l$ is discussed in Section 9.5.

The sketch of the proposed primal decomposition algorithm for multistage scenario MPC problem using the feasibility ensuring backtracking algorithm is given in Algorithm 9.2.

9.4 Case study

In this section, we test the proposed primal decomposition-based scenario decomposition on a continuous stirred tank reactor (CSTR) process from [1, 41]. This

[2] See also Appendix I

Algorithm 9.2 Distributed multistage scenario MPC using Primal decomposition

Define tolerance $\epsilon > 0$.

Input: at each time step: initial state $\hat{\boldsymbol{x}}$ and $\Delta \boldsymbol{t}_l > \epsilon$, initial α

while $\Delta \boldsymbol{t}_l > \epsilon$ **do**
  Initialize iteration number $v = 1$
  for $j = 1, 2, \ldots, S$ **do**
    $\left[\boldsymbol{X}^*(\boldsymbol{t}_l, \boldsymbol{p}_j), \boldsymbol{\lambda}^*(\boldsymbol{t}_l, \boldsymbol{p}_j)\right] \leftarrow$ solution NLP $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j)$
  end for
  for $l \in \{1, \ldots, \sum_{m=1}^{N_r} M^{N_r - 1}\}$ **do**
    Update subgradients $\nabla_{\boldsymbol{t}_l} \Phi(\boldsymbol{t}_l, \boldsymbol{p}_j)$
    Backtrack α using Algorithm 9.1
    Update $\boldsymbol{t}_l^{(v+1)} = \boldsymbol{t}_l^{(v)} + \alpha_l (\sum_{j=1}^{S} \nabla_{\boldsymbol{t}_l} \boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j))^{(v)}$
    Update $\Delta \boldsymbol{t}_l = \|\boldsymbol{t}_l^+ - \boldsymbol{t}_l\|$
  end for
end while
Update iteration number $v = v + 1$
Set $\boldsymbol{t}_l^* \leftarrow \boldsymbol{t}_l$
Reset α to initial guess.

Output: $\boldsymbol{t}_l^*$, $\boldsymbol{X}^*(\boldsymbol{t}_l^*) = [\boldsymbol{X}^*(\boldsymbol{t}_l^*, \boldsymbol{p}_1), \ldots, \boldsymbol{X}^*(\boldsymbol{t}_l^*, \boldsymbol{p}_S)]^\mathsf{T}$

is the same case example that was used in Chapter 3 (cf. 3.3.1) and the reader is referred to (3.1) for the process model.

The objective is to maximize the product concentration C_B while penalizing the utility cost of heating the input stream using the inlet temperature $\mathbf{u} = T_i$ as the manipulated variable as described in Chapter 3. In addition, the reactor temperature now has a maximum limit of 425K.

$$\begin{aligned} \min_{T_i} J &= -[2.009 C_B - (1.657 \times 10^{-3} T_i)^2] \\ \text{s.t. } & (3.1) \\ & T \leq 425 \end{aligned} \tag{9.11}$$

We assume that the concentration of component B in the feed stream is uncertain and is known to vary in the range $C_{B,i} \in [0, 0.2] mol/l$. We design a multistage scenario NMPC with a prediction horizon of $T = 300$s equally divided into $N = 20$ samples. For the scenario tree, we consider $M = 3$ discrete realizations of the uncertainty, namely, $C_{B,i} \in \{0.0, 0.1, 0.2\} mol/l$.

9.4.1 Simulation case 1

In the first simulation case, we design the multistage NMPC with a robust horizon of $N_r = 1$, leading to $S = M^{N_r} = 3$ scenarios. The process was simulated for a total simulation time of 600s. The concentration of component B in the feed stream changes at time $t = 300$s from $0.15 mol/l$ to $0.0 mol/l$. The process was first

simulated with a fully centralized multistage scenario NMPC problem (8.1) to be used as a benchmark (shown in thick yellow curves in Fig. 9.3).

The process was then simulated using the proposed primal decomposition-based scenario decomposition (9.3), where the step length α_l was initialized with $\alpha = 2000$. At each iteration, the step length α was suitably adjusted using the proposed back-tracking algorithm introduced in Section 9.3. At each time step, warm-starting was implemented for the auxiliary variables $\boldsymbol{t_l}$ using the predicted control trajectories from the previous time step. The simulation results are compared with the fully centralized approach in Fig. 9.3. The cost function J, outlet temperature T, and the inlet temperature T_i are shown in the left hand side subplots (solid red curves). The corresponding absolute errors compared to the fully centralized approach is shown in the right hand side subplots (solid red curves). From the plots, it can be clearly seen that the primal decomposition approach of solving the multistage scenario NMPC problem results in the same solution as the centralized solution, thus indication proper formulation. The total number of master and subproblem iterations taken at each time step are shown in the bottom right subplot and the step-length α value obtained from the proposed backtracking algorithm is shown in the bottom left subplot.

As mentioned earlier, one of the main motivations to use primal decomposition is that it enables closed-loop implementation even if the master problem and subproblems have not fully converged. In order to test this, the iterations between the master problem and the subproblems were capped at 5 iterations to prematurely terminate the iterations. The simulation results are shown in Fig. 9.3 using black dashed curves. It can be seen that by prematurely terminating the iterations, the non-anticipativity constraints remain feasible, however, the closed-loop solution is suboptimal. By warm starting the auxiliary variables, the proposed primal decomposition approach eventually converges to the optimal solution provided by the centralized approach even when the master and subproblem iterations are prematurely terminated. If the problem is convex and differentiable, then this can be guaranteed [19]. This is also clearly seen in the error subplots, where the absolute error compared to the centralized approach (shown in black dashed lines) diminishes over time.

Effect of the step-length size The step-length α backtracked using the proposed backtracking algorithm is shown in the bottom left subplot. When the disturbance in the input feed stream changes at time $t = 300$s, the optimal solution drives the process to the constraint on reactor temperature. At time $t = 410$s, the step-length α is backtracked to a small value when operating close to the constraint. Keeping the step length constant at $\alpha = 2000$, resulted in infeasibility of the reactor temperature constraint. This is because, for the operating conditions after about 400s of simulation, the upper bound on the step length $\overline{\alpha}$ for which the master problem remains feasible as shown in (9.10) is lower than the initially used step length value $\alpha = 2000$. By using the proposed backtracking algorithm, the step length was backtracked to find the upper bound $\overline{\alpha}$ as shown in the bottom-left subplot in Fig. 9.3, such that it ensures the nonlinear process constraints also remain feasible throughout the iterations while updating the auxiliary variables in

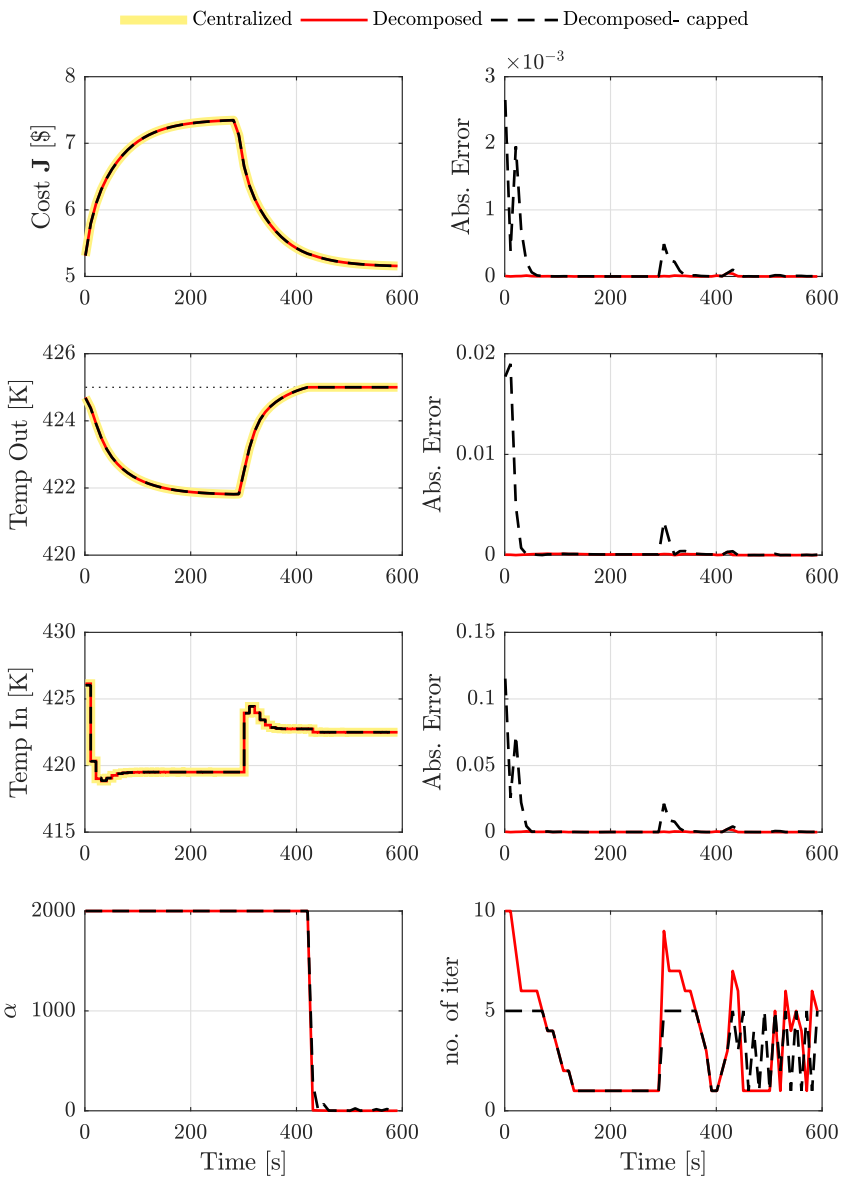


Figure 9.3: Simulation results with $N_r = 1$ showing the optimal solution provided by the centralized approach (thick yellow lines), primal decomposition approach (solid red lines) and the primal decomposition approach with the maximum number of iterations capped at 5 (black dashed lines).

Table 9.1: CPU times (in sec) for simulation 1 and 2

	$N_r = 1$			$N_r = 2$		
	max	avg	min	max	avg	min
Centralized	0.219	0.168	0.154	0.549	0.429	0.377
Decomposed	0.137	0.073	0.053	0.127	0.085	0.077

the master problem.

9.4.2 Simulation case 2

In this subsection, we simulate the system with a robust horizon of $N_r = 2$, leading to $S = M^{N_r} = 9$ scenarios. The process was simulated for a total simulation time of 600s. The concentration of component B in the feed stream changes at time $t = 300$s from $0.15 mol/l$ to $0 mol/l$ just as in simulation case 1.

The process was then simulated using the primal decomposition based scenario decomposition, where the step length α_l was initialized with 500 for all l. At each iteration, the step lengths were suitably adjusted using the proposed back-tracking algorithm in Section 9.3. The simulation results and the corresponding absolute errors for this simulation case is shown in Fig. 9.4.

The proposed method was also simulated with the total number of iterations capped at 15 iterations. The simulation results are shown in Fig. 9.4 using black dashed curves. It can be seen that the proposed primal decomposition approach eventually converges to the optimal solution provided by the centralized approach even when the master and subproblem iterations are prematurely terminated.

As can be seen from the simulation results from Fig. 9.3 and Fig. 9.4, the solution obtained by the proposed primal decomposition approach is almost identical to the one provided by solving the multistage problem in a centralized fashion. The Primal decomposition method was also shown to enable closed-loop implementation when the iterations between the master and scenario subproblems are prematurely terminated. The computation times for the multistage problem solved as a fully centralized problem and using the primal decomposition are also shown in Table. 9.1, which shows that, by using the proposed primal decomposition approach for multistage NMPC, the same solution can be obtained at less computation times.The proposed primal decomposition based multistage scenario decomposition approach was also demonstrated using an oil and gas production optimization case study in our recent work [94].

9.5 Discussions

9.5.1 Why Primal decomposition?

One of the key challenges today in real-time implementation of optimizing controllers such as model predictive control, is the computation time. The late arrival of a solution in many cases may simply not be acceptable. A solution to the optimization problem must ideally be available within the sampling time.

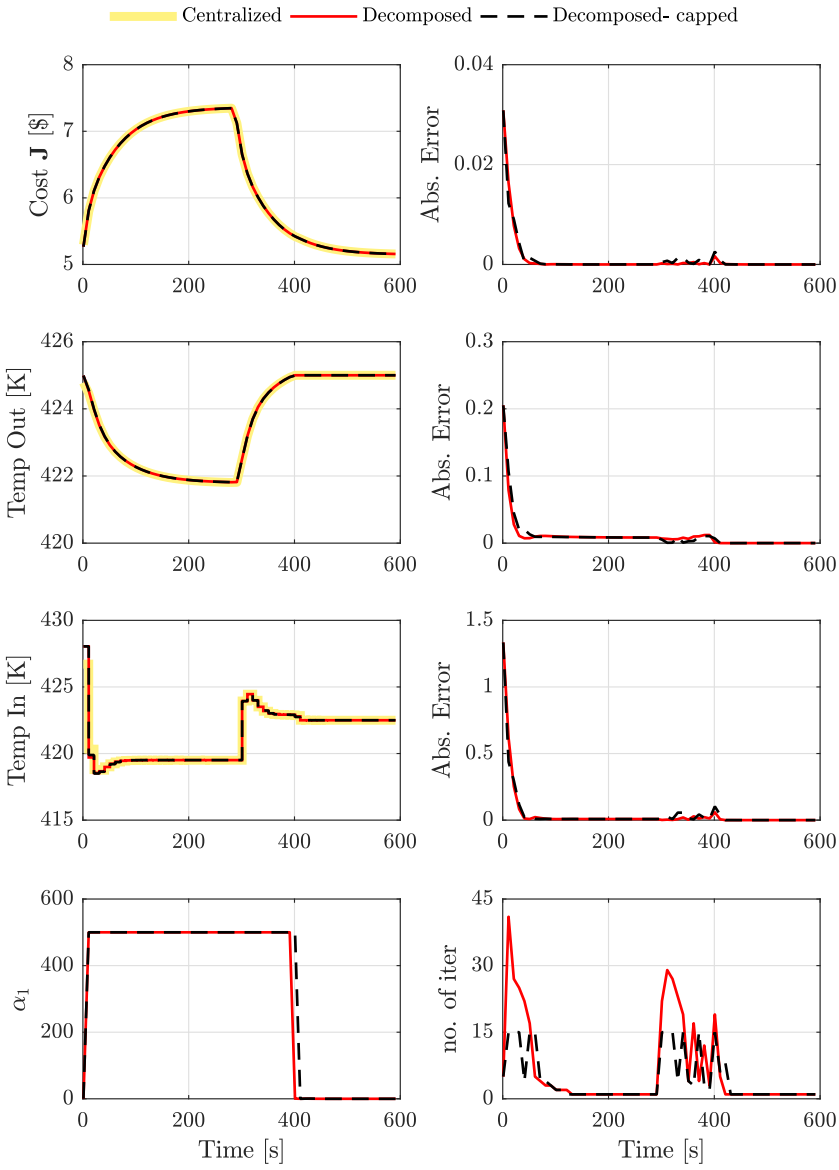


Figure 9.4: Simulation results with $N_r = 2$ showing the optimal solution provided by the centralized approach (thick yellow lines), primal decomposition approach (solid red lines) and the primal decomposition approach with the maximum number of iterations capped at 15 (black dashed lines).

Decomposition methods for scenario decomposition has its roots in multistage stochastic optimization problems studied in the operations research community, see [16, 67, 150, 151] to name a few. Problems studied in the operations research field do not focus on real-time closed-loop implementation in the same fashion as NMPC in the process control community. The nature of the problems studied in operations research community often call for offline optimization problems, and closed-loop implementation is not a focus, as opposed to NMPC applications in the process industries. For example, in many process control applications, a new control input must be computed at every sampling instant, which may be in the time scale of seconds to minutes. As a result, the dual decomposition methods developed for multistage stochastic optimization problems may not be directly applicable for the multistage NMPC problem. If the dual decomposition does not converge, then the non-anticipativity constraints are not feasible. The feasibility of the non-anticipativity constraints is crucial for closed-loop implementation, since it ensures that the first control input is the same for all the scenarios. Keeping in mind the closed-loop implementation, primal decomposition is a favorable approach than dual decomposition, since the non-anticipativity constraints are *always* feasible, even if the problem does not converge to the optimal solution.

Since primal decomposition provides a primal feasible solution with monotonically decreasing objective value with each iteration, premature termination of the iterations only results in suboptimal operation and does not violate the non-anticipativity constraints. By warm-starting the subsequent time steps, the solution may eventually converge to the true optimal solution, under certain conditions. This was also seen in the error plots in Fig. 9.3 and Fig. 9.4. Therefore, primal decomposition approach addresses the practical implementation issues of distributed multistage scenario NMPC problem.

9.5.2 Feasibility-ensuring backtracking algorithm

In this chapter, we also proposed a feasibility ensuring backtracking algorithm to suitably choose the step length size in the master problem update such that the nonlinear constraints in the subproblems remain feasible throughout the iterations. In algorithm 9.1, we check the forward simulation for all the discrete realizations of the uncertainty used in the scenario tree to backtrack the step length used in the master problem. However, if the worst-case realization of the uncertainty is known a-priori, we then need to check the feasibility of the local constraints only for the worst-case scenario w.r.t to the nonlinear constraints instead of all the scenarios. This is justified because if the local constraints within a subproblem are feasible for the worst-case scenario, then it must also be feasible for all other scenarios.

With the proposed primal decomposition approach, it is important to note that the initial guess of the auxiliary variables $\boldsymbol{t}$ must be a feasible guess with respect to the nonlinear constraints. As described in 9.3, with the assumption of a feasible initial guess, the backtracking algorithm used here always ensures the feasibility of the nonlinear constraints in the scenario subproblems according to (9.10). One simple approach to get an initial feasible guess is by warm starting the auxiliary variable using the predicted control trajectory. The predicted control trajectory ranges from $\mathbf{u}_1$ to $\mathbf{u}_N$ for each scenario. The first control input $\mathbf{u}_1$

(which is the same for all the scenarios due to the non-anticipativity constraints) is implemented on the plant. For the next NMPC iteration, the auxiliary variable can be initialized using the predicted control trajectory starting from the second time step $\mathbf{u}_2$ to $\mathbf{u}_{N_r+1}$ in the prediction horizon corresponding to the worst-case scenario. By initializing the auxiliary variables using the predicted control input for the worst-case scenario, the initial guess will be feasible for all other scenarios as well. Additional back-off on predicted control input from the worst-case scenario may also be used when initializing the auxiliary variables at each time step to ensure a feasible initial guess.

9.6 Chapter summary

In this chapter, we proposed a primal decomposition approach (9.3) to solve the multistage scenario-based NMPC problems (cf. Algorithm 9.2). We showed that, primal decomposition enables real-time closed-loop implementation of the multistage approach, even in the case where the iterations between the master problem and subproblems are terminated prematurely, unlike the dual decomposition approach.

Furthermore, we also presented a novel feasibility-ensuring backtracking algorithm (cf. Algorithm 9.1) to suitably choose the size of the step length in the master problem update in Section 9.3.

A CSTR case study demonstrates the effectiveness of the proposed method. In addition to the CSTR process, this method was also tested on a gas-lift optimization case study which can be found in Appendix J.

Chapter 10

Improving Scenario Decomposition for Multistage NMPC using a Sensitivity-based Path-following Algorithm

This chapter considers the primal decomposition based distributed multistage scenario formulation from the previous chapter. Since the different scenarios differ only in the uncertain parameters, the distributed scenario NMPC problem can be cast as a parametric nonlinear programming (NLP) problem. By using the NLP sensitivity, we do not need to solve all the subproblems as full NLPs. Instead they can be solved exploiting the parametric nature by a path-following predictor-corrector algorithm that approximates the NLP. This further reduces the online computation time.

Based on the article published in IEEE Control systems letters [98] and 57th IEEE Conference on Decision and Control.

10.1 Introduction

In Chapter 9, we proposed a primal decomposition approach [94] for solving the multistage sceanrio-based NMPC problem by decomposing the problem into several subproblems, with each subproblem solving a discrete scenario. Both the dual and the primal decomposition approaches, involve solving each scenario independently and a master problem co-ordinates the different scenarios. Although performance improvements have been reported by decomposing the scenario decomposition approaches by Lucia et al. [117], Martí et al. [124] and Krishnamoorthy et al. [94], it still requires solving a nonlinear programming problem (NLP) for each scenario. Even with today's computing power, solving nonlinear dynamic optimization problems online can be computationally intensive.

In this chapter, we propose to further improve the scenario decomposition algorithms by using NLP sensitivity-based path-following approaches [109, 178]. From

(9.3) and Fig. 9.2 it can be clearly seen that the different scenario subproblems differ only in the uncertain parameters $\boldsymbol{p}_j$. Therefore, we propose to re-cast the scenario decomposition problem in the framework of parametric NLP and exploit the NLP sensitivites to improve the computation time. Here, using the solution of one full NLP, the subsequent scenario subproblems are solved by tracking the optimal path along the parameter range that leads to the scenarios by a sequence of predictor-corrector QPs. In simple terms, each of these QPs tells us how the optimal solution changes when the parameter changes by a small value, and the NLP solution change for a larger parameter change can be found by solving several QPs. We apply this idea to the distributed multistage scenario NMPC problem to compute how the optimal solution changes from one scenario to the other.

The main contribution of this chapter is the use of an NLP sensitivity-based path-following method to efficiently solve the distributed multistage scenario NMPC algorithm. The main result is presented as a Corollary of Theorem 10.2 and Algorithm 10.1. To the best of our knowledge, parametric optimization concepts have not been used previously to solve scenario decomposition problems for multistage scenario NMPC.

We consider the fully centralized problem (hereafter denoted as C_{NLP}), where the multistage scenario NMPC problem (8.1) presented in Chapter 8 is solved as a single optimization problem. We then consider the distributed multistage NMPC problem (hereafter denoted as D_{NLP}) where the scenario subproblems (9.3) are solved using primal decomposition, as presented in Chapter 9. Based on this, we are now ready to present the sensitivity-based distributed scenario NMPC with a path-following algorithm (main result).

10.2 Sensitivity-based distributed multistage scenario MPC

10.2.1 Sensitivity in Parametric NLP

To keep the presentation simple, we now reformulate each scenario subproblem (9.3) as a generic parametric NLP of the form,

$$\begin{aligned} \min_{\boldsymbol{X}} \quad & \mathcal{J}(\boldsymbol{X}, \boldsymbol{p}) \\ \text{s.t.} \quad & c_i(\boldsymbol{X}, \boldsymbol{p}) = 0, \qquad \forall i \in \mathbb{E} \\ & c_j(\boldsymbol{X}, \boldsymbol{p}) \leq 0, \qquad \forall j \in \mathbb{I} \end{aligned} \tag{10.1}$$

where $\boldsymbol{X} \in \mathbb{R}^{n_{\boldsymbol{X}}}$ denotes the optimization (primal) variables of (9.3), $\boldsymbol{p} \in \mathbb{R}^{n_p}$ is the vector of uncertain parameters and the objective function is denoted by $\mathcal{J} : \mathbb{R}^{n_{\boldsymbol{X}}} \times \mathbb{R}^{n_p} \to \mathbb{R}$. The equality and inequality constraints $c : \mathbb{R}^{n_{\boldsymbol{X}}} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_c}$ are denoted by the sets $\mathbb{E} = \{1, \ldots, v\}$ and $\mathbb{I} = \{v+1, \ldots, n_c\}$, respectively.

The Lagrangian of (10.1) is defined as

$$\mathcal{L}(\mathbf{X}, \boldsymbol{\lambda}, \boldsymbol{p}) := \mathcal{J}(\boldsymbol{X}, \boldsymbol{p}) + \boldsymbol{\lambda}^{\mathsf{T}} c(\boldsymbol{X}, \boldsymbol{p}) \tag{10.2}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers (dual variable). The Karush-Kuhn-Tucker (KKT) conditions for this problem can be stated as

$$\begin{aligned} \nabla_{\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}) &= 0, \\ c_i(\boldsymbol{X}, \boldsymbol{p}) &= 0,\ i \in \mathbb{E}, \\ c_i(\boldsymbol{X}, \boldsymbol{p}) &\leq 0,\ i \in \mathbb{I}, \\ \boldsymbol{\lambda}_i^T c_i(\boldsymbol{X}, \boldsymbol{p}) &= 0,\ i \in \mathbb{I}, \\ \boldsymbol{\lambda}_i &\geq 0,\ i \in \mathbb{I}. \end{aligned} \tag{10.3}$$

Definition 10.1 (KKT point)**.** Any point $(\boldsymbol{X}^*, \boldsymbol{\lambda}^*)$ that satisfies the KKT conditions (10.3) for a given parameter vector $\boldsymbol{p}$ is called a KKT point for $\boldsymbol{p}$.

The active inequality constraints in (10.1) are denoted by the set $\mathbb{A}(\boldsymbol{X}, \mathbf{p}) = \{c_i(\boldsymbol{X}, \mathbf{p}) = 0, i \in \mathbb{I}\}$ and the active set is then given by $\mathbb{E} \cup \mathbb{A}$. For a given KKT point $(\boldsymbol{X}^*, \boldsymbol{\lambda}^*)$, the active set $\mathbb{A}$ has two subsets, namely a weakly active set $\mathbb{A}_0(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}) = \{i \in \mathbb{A}(\boldsymbol{X}, \boldsymbol{p}) \mid \boldsymbol{\lambda}_i = 0\}$ and a strong active set $\mathbb{A}_+(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}) = \{i \in \mathbb{A}(\boldsymbol{X}, \boldsymbol{p}) \mid \boldsymbol{\lambda}_i > 0\}$. Consequently, the inactive set
$\mathbb{A}_-(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}) = \{c_i(\boldsymbol{X}, \mathbf{p}) < 0, i \in \mathbb{I}\}$ is the complement of set $\mathbb{A}$.

A constraint qualification is required to hold in order for the KKT conditions to be a necessary condition of optimality and in this work we consider the linear independence constraint qualification (LICQ) which is defined as follows.

Definition 10.2 (LICQ)**.** Given a vector $\boldsymbol{p}$ and a point $\boldsymbol{X}$, the linear independence constraint qualification (LICQ) holds at $(\boldsymbol{X}, \boldsymbol{p})$ if the set of vectors

$$\{\nabla_{\boldsymbol{X}} c_i(\boldsymbol{X}, \boldsymbol{p})\}_{i \in \mathbb{E} \cup \mathbb{A}(\boldsymbol{X}, \boldsymbol{p})}$$

are linearly independent.

Definition 10.3 (SSOSC)**.** The strong second order sufficient condition (SSOSC) holds at any KKT point $(\boldsymbol{X}^*, \boldsymbol{\lambda}^*)$, if $\boldsymbol{d}^T \boldsymbol{H}(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p})\, \boldsymbol{d} > 0$ for all $\boldsymbol{d} \neq 0$ such that $\nabla_{\boldsymbol{X}} c_i(\boldsymbol{X}, \boldsymbol{p})^T \boldsymbol{d} = 0$ for $i \in \mathbb{E} \cup \mathbb{A}_+$, where the Hessian of the Lagrangian (10.2) is given by

$$\boldsymbol{H}(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}) = \nabla^2_{\boldsymbol{X}\boldsymbol{X}} \mathcal{J}(\boldsymbol{X}, \boldsymbol{p}) + \sum_{i=1}^{n} \nabla^2_{\boldsymbol{X}\boldsymbol{X}} c_i(\boldsymbol{X}, \boldsymbol{p})\, \boldsymbol{\lambda}_i.$$

The LICQ and SSOSC guarantees that a KKT point is a strict local minimum.

Assumption 10.1: $\boldsymbol{X}^*$ satisfies the KKT conditions (10.3) for a given parameter vector $\boldsymbol{p}_0$ and the linear independence constraint qualification (LICQ) and strong second order sufficient condition (SSOSC) hold at $(\boldsymbol{X}^*, \boldsymbol{p}_0)$.

The reader is referred to Lemma 1 by Klintberg et al. [86] for detailed description on how the assumption of LICQ and positive definiteness of the Hessian translates to the multistage scenario MPC problem (9.3).

Theorem 10.1. *Let $\mathcal{J}, c$ be twice differentiable in $\boldsymbol{p}$ and $\boldsymbol{X}$ near a solution of (10.1) $(\boldsymbol{X}^*, \boldsymbol{p}_0)$ and let Assumption 10.1 hold, then the solution $(\boldsymbol{X}^*(\boldsymbol{p}), \boldsymbol{\lambda}^*(\boldsymbol{p}))$ is Lipschitz continuous in the neighborhood of $(\boldsymbol{X}^*, \boldsymbol{\lambda}^*, \boldsymbol{p}_0)$ and the solution $\boldsymbol{X}^*(\boldsymbol{p})$ is*

directionally differentiable. Additionally, the directional derivative uniquely solves the following quadratic problem (QP):

$$\begin{aligned}\min_{\Delta \boldsymbol{X}} \quad & \frac{1}{2}\Delta \boldsymbol{X}^T \nabla^2_{\boldsymbol{X}\boldsymbol{X}} \mathcal{L}\left(\boldsymbol{X}^*, \boldsymbol{p_0}, \boldsymbol{\lambda}^*\right) \Delta \boldsymbol{X} \\ & + \Delta \boldsymbol{X}^T \nabla_{\boldsymbol{X}\boldsymbol{p}} \mathcal{L}\left(\boldsymbol{X}^*, \boldsymbol{p_0}, \boldsymbol{\lambda}^*\right) \Delta \boldsymbol{p} \qquad (10.4)\\ \text{s.t.} \quad & \\ & \nabla_{\boldsymbol{X}} c_i\left(\boldsymbol{X}^*, \boldsymbol{p_0}\right)^T \Delta \boldsymbol{X} \\ & + \nabla_{\boldsymbol{p}} c_i\left(\boldsymbol{X}^*, \boldsymbol{p_0}\right)^T \Delta \boldsymbol{p} = 0 \qquad i \in \mathbb{A}_+ \cup \mathbb{E}, \\ & \nabla_{\boldsymbol{X}} c_i\left(\boldsymbol{X}^*, \boldsymbol{p_0}\right)^T \Delta \boldsymbol{X} \\ & + \nabla_{\boldsymbol{p}} c_i\left(\boldsymbol{X}^*, \boldsymbol{p_0}\right)^T \Delta \boldsymbol{p} \leq 0 \qquad i \in \mathbb{A}_0\end{aligned}$$

Proof. See [148] and [17, Section 5.2]. □

The theorem above implies that a quadratic programming (QP) problem (10.4), often referred as *pure-predictor QP*, can be solved instead of a full NLP problem, in order to compute an approximate solution of (10.1) in the neighborhood of perturbation $\boldsymbol{p}_0$. This is the core idea of the sensitivity-based approach that we now use to efficiently solve the distributed multistage scenario MPC problem.

10.2.2 Path-following predictor-corrector QP

A corrector term can be added to the objective function in (10.4) to improve the approximation accuracy, as shown by Suwartadi et al. [178]. With the assumption that the parameter enters linearly in the constraints, we can formulate the following QP.

$$\begin{aligned}\min_{\Delta \boldsymbol{X}} \quad & \frac{1}{2}\Delta \boldsymbol{X}^T \nabla^2_{\boldsymbol{X}\boldsymbol{X}} \mathcal{L}\left(\boldsymbol{X}^*, \boldsymbol{p_0} + \boldsymbol{\Delta p}, \boldsymbol{\lambda}^*\right) \Delta \boldsymbol{X} \\ & + \Delta \boldsymbol{X}^T \nabla_{\boldsymbol{X}\boldsymbol{p}} \mathcal{L}\left(\boldsymbol{X}^*, \boldsymbol{p_0} + \boldsymbol{\Delta p}, \boldsymbol{\lambda}^*\right) \Delta \boldsymbol{p} \\ & + \nabla_{\boldsymbol{X}} \mathcal{J}^T \Delta \boldsymbol{X} \qquad (10.5)\\ \text{s.t.} \quad & \\ & c_i\left(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p}\right) + \nabla_{\boldsymbol{p}} c_i\left(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p}\right)^T \Delta \boldsymbol{p} + \\ & \nabla_{\boldsymbol{X}} c_i\left(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p}\right)^T \Delta \boldsymbol{X} = 0 \ , i \in \mathbb{A}_+ \cup \mathbb{E}, \\ & c_i\left(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p}\right) + \nabla_{\boldsymbol{p}} c_i\left(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p}\right)^T \Delta \boldsymbol{p} \\ & + \nabla_{\boldsymbol{X}} c_i\left(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p}\right)^T \Delta \boldsymbol{X} \leq 0 \ , i \in \mathbb{A}_0.\end{aligned}$$

The QP formulation (10.5) is known as the *predictor-corrector QP*. It can be thought of a combination of a first-order sensitivity step and an SQP step towards the solution for the new parameter value. In the small neighborhood of $\boldsymbol{p}_0$, the predictor-corrector QP formulation was shown to provide good approximations of the NLP solution. However, the different models M used in the scenario optimization need not necessarily be in the small neighborhood of each other. Therefore,

in order to allow for large perturbations (i.e. large $\Delta \boldsymbol{p}$, we propose to apply a *path-following* approach [178], where we solve a series of QP problems sequentially.

Given an optimal solution $\boldsymbol{X}^*(\boldsymbol{p}_{j-1})$ for a parameter vector $\boldsymbol{p}_{j-1}$, we want to compute the optimal solution for a parameter vector $\boldsymbol{p}_j$. The path-following predictor-corrector QP then updates $\boldsymbol{X}$ for the parameter sequence $\boldsymbol{p}$ according to

$$\boldsymbol{p}(\nu_\kappa) = (1 - \nu_\kappa)\, \boldsymbol{p}_{j-1} + \nu_\kappa \boldsymbol{p}_j \tag{10.6}$$

where $\nu_0 = 0$ until it reaches $\nu_\kappa = 1$. In other words $\nu_0 = 0 < \nu_1 < \nu_2 < \cdots < \nu_\kappa = 1$. Given a sufficiently small step $\Delta\nu$, the path-following predictor-corrector QP, after solving a series of QP problems, provides the optimal solution $\boldsymbol{X}^*(\boldsymbol{p}_j)$ for a parameter vector $\boldsymbol{p}_j$[1]. For the sake of simplicity, we use a fixed step size $\Delta\nu = \nu_{\kappa+1} - \nu_\kappa$.

10.2.3 Sensitivity-based path-following distributed multistage scenario MPC

Based on these developments, we are now ready to formulate the sensitivity-based distributed multistage scenario MPC algorithm.

Assumption 10.2: There exists a continuous path of unique optimal solutions between the subproblems $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_{j-1})$ and $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j)$.

Corollary 10.2 (Main result)**.** *Let* $\left[\boldsymbol{X}^*(\boldsymbol{p}_{j-1}), \boldsymbol{\lambda}^*(\boldsymbol{p}_{j-1})\right]$ *be the solution for one scenario subproblem obtained by solving the NLP* $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_{j-1})$ *and let Assumptions* (10.1) *and* (10.2) *hold. Further, let* $\boldsymbol{p}_j$ *be in the neighborhood of* $\boldsymbol{p}_{j-1}$*, then the solution for all other scenario subproblems* $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j)$ *with the same set of auxiliary variables* $\boldsymbol{t}_l$ *is Lipschitz continuous in the neighborhood of* $\left[\boldsymbol{X}^*(\boldsymbol{p}_{j-1}), \boldsymbol{\lambda}^*(\boldsymbol{p}_{j-1})\right]$ *and can be obtained by repeatedly solving the predictor-corrector QP* (10.5)*.*

Proof. Since the only difference between the scenarios $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_{j-1})$ and $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j)$ is the parameter vector $\boldsymbol{p}_j$, it follows from Theorem 10.1 that the NLP problem $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j)$ can be approximated by repeatedly solving the QP problem (10.5) for a small parameter perturbation $\Delta \boldsymbol{p}$ along the path from $\boldsymbol{p}_{j-1}$ to $\boldsymbol{p}_j$. □

Corollary 10.2 above suggests that instead of solving S number of NLPs, the multistage scenario MPC problem can be solved using $M^{N_r - 1}$ number of NLPs and the remaining subproblems can be solved as QPs. The number of common nodes between two consecutive scenarios $n_{o,(j,j+1)}$ is used to check if the two scenarios have the same set of auxiliary variables $\boldsymbol{t}_l$.

The sensitivity-based distributed scenario MPC algorithm then consists of the following three steps.

[1] Note that path-following approaches were developed in the context of advance step MPC[74, 178], where the idea was to parameterize the initial condition constraint (i.e. $\boldsymbol{p} = \mathbf{x}_0$). Whereas in this chapter, we apply it to the distributed scenario MPC problem, where the parameter $\boldsymbol{p}$ are the uncertain parameter value used in each scenario subproblem.

1. For a given primal master variable $\boldsymbol{t}_l$, solve the NLP problem $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_{j-1})$ for one subproblem with the parameter vector $\boldsymbol{p}_{j-1}$ to obtain the optimal primal and dual variables $\boldsymbol{X}^*(\boldsymbol{p}_{j-1})$ and $\boldsymbol{\lambda}^*(\boldsymbol{p}_{j-1})$, respectively.

2. For the subsequent scenario subproblems with the same set of auxiliary variables, compute an approximation of the NLP problem $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j)$ using the QP (10.5) in a path-following manner as described in Section 10.2.2.

3. Using the computed Lagrange multipliers corresponding to the non-anticipativity constraints (9.3e) $\lambda \subset \boldsymbol{\lambda}$ from all the subproblems, update the primal master variable $\boldsymbol{t}_l$ according to (9.7).

A sketch of the proposed sensitivity-based multistage scenario MPC procedure is described in Algorithm 10.1.

Algorithm 10.1 Sensitivity-based distributed multistage scenario MPC

Define tolerance $\epsilon > 0$, $\Delta\nu \leq 1$.
Input: At each time step, initial state $\hat{\boldsymbol{x}}$, initial $\boldsymbol{t}_l^0$ and $\Delta\boldsymbol{t}_l > \epsilon$, initial α

while $\Delta\boldsymbol{t}_l > \epsilon$ **do**
  for $j = 1, 2, \ldots, S$ **do**
    if $(j = 1) \vee (n_{o,(j-1,j)} \leq N_r - 1)$ **then**
      $\left[\boldsymbol{X}^*(\boldsymbol{p}_j), \boldsymbol{\lambda}^*(\boldsymbol{p}_j)\right] \leftarrow$ solution NLP $\boldsymbol{\Phi}(\boldsymbol{t}_l, \boldsymbol{p}_j)$
    else ▷ Approximate NLP using QP (10.5).
      $\left[\Delta\boldsymbol{X}^*, \boldsymbol{\lambda}^*(\boldsymbol{p}_j)\right] \leftarrow$ QP_PF$(\boldsymbol{X}^*, \boldsymbol{\lambda}^*, \boldsymbol{p}_{j-1}, \boldsymbol{p}_j)$.
      Set $\boldsymbol{X}^*(\boldsymbol{p}_j) = \boldsymbol{X}^*(\boldsymbol{p}_{j-1}) + \Delta\boldsymbol{X}^*$.
    end if
  end for
  Update $\boldsymbol{t}_l^+ = \boldsymbol{t}_l + \alpha(\sum_{j=1}^{S} \lambda_j)$
  Update $\Delta\boldsymbol{t}_l = \|\boldsymbol{t}_l^+ - \boldsymbol{t}_l\|$
end while

function QP_PF$(\boldsymbol{X}^*(\boldsymbol{p}_{j-1}), \boldsymbol{\lambda}^*(\boldsymbol{p}_{j-1}), \boldsymbol{p}_{j-1}, \boldsymbol{p}_j)$
  Define $\mathbb{A}_+$.
  Set $\nu_\kappa = 0$.
  while $\nu_\kappa < 1$ **do**
    $[\Delta\boldsymbol{X}^*, \boldsymbol{\lambda}^*] \leftarrow$solution QP (10.5) with $\boldsymbol{p} = \boldsymbol{p}\,(\nu_\kappa)$
    $\boldsymbol{X}^* = \boldsymbol{X}^* + \Delta\boldsymbol{X}^*$
    $\nu_{\kappa+1} \leftarrow \nu_\kappa + \Delta\nu$
    $\boldsymbol{p}\,(\nu_\kappa) = (1 - \nu_\kappa)\,\boldsymbol{p}_{j-1} + \nu_\kappa \boldsymbol{p}_j$
  end while
  return $\Delta\boldsymbol{X}^*, \boldsymbol{\lambda}^*$
end function

Output: $\boldsymbol{X}^*(\boldsymbol{p}_j), \forall j \in \{1, \ldots, S\}$

10.3 Illustrative example

In this work, we consider the same CSTR example as in Chapter 9. We assume the concentration of component B in the feed stream is uncertain and consider five discrete realizations, namely, $C_{B,i} \in \{0, 0.05, 0.1, 0.15, 0.2\} mol/l$.

We use a multistage scenario NMPC with a prediction horizon of $T = 300$s divided equally into $N = 20$ samples. The NLP problem was solved using the `IPOPT` solver and the QP problems were solved using `TOMLAB MINOS` [132]. The optimization problem then consists,

1. $\mathbf{J}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) = \left(-2.009 C_B + (1.657 \times 10^{-3} T_i)^2\right)$,
2. discretized system model,
3. uncertain parameter $\boldsymbol{p} = C_{B_i}$ discretized into $M = 5$ finite models, namely, $C_{B_i} \in \{0, 0.05, 0.1, 0.15, 0.2\}$,
4. process constraints $\mathbf{g}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) = T - 425$, and
5. non-anticipativity constraints (8.1f).

We note that in the considered case study, the constraint $T \leq 425$K becomes active at steady-state only when $C_{B_i} \in \{0, 0.05\} mol/l$ and not when $C_{B_i} \in \{0.1, 0.15, 0.2\} mol/l$. Therefore the active constraint set changes between the different scenarios. The true realization of C_{B_i} used in the simulations changes from $C_{B_i} = 0.15 mol/l$ to $C_{B_i} = 0.05 mol/l$ at time $t = 300$s.

Simulation 1 - Robust horizon length = 1

In the first simulation we consider a robust horizon of $N_r = 1$ and hence we have $S = 5$ scenarios. We first compute the solution of a fully centralized approach (C_{NLP}), i.e. (8.1) to be used as a benchmark. The multistage scenario NMPC is then solved using the primal decomposition approach i.e. (9.3), where all the scenario subproblems are solved as NLP problems (D_{NLP}). We then solve the optimization problem using the proposed path-following QP (pf-QP), where the first scenario was solved as NLP problem and the subsequent four scenarios are solved using the path-following predictor-corrector QP (10.5) as described in Algorithm 10.1 with a fixed step size $\Delta\nu = 0.5$. Hence two QPs were solved to approximate each subproblem. For the distributed scenario approaches, the tolerance was chosen to be $\epsilon = 0.001$ and a feasibility ensuring backtracking algorithm was used to select a suitable step length α.

The closed-loop implemented solution for the proposed sensitivity-based distributed scenario NMPC are compared with the fully centralized scenario NMPC (C_{NLP}) and the distributed scenario NMPC solved using full NLPs (D_{NLP}) along with the corresponding absolute errors in Fig. 10.1.

Simulation2 - Robust horizon length = 2

In the second simulation we consider the same problem, but a robust horizon of $N_r = 2$ leading to a scenario tree with $S = 25$ scenarios. By using the path-following predictor-corrector QP (10.5), we solve 5 scenarios using NLPs and 20 scenarios were solved using path-following QPs. The closed-loop implemented solution for the

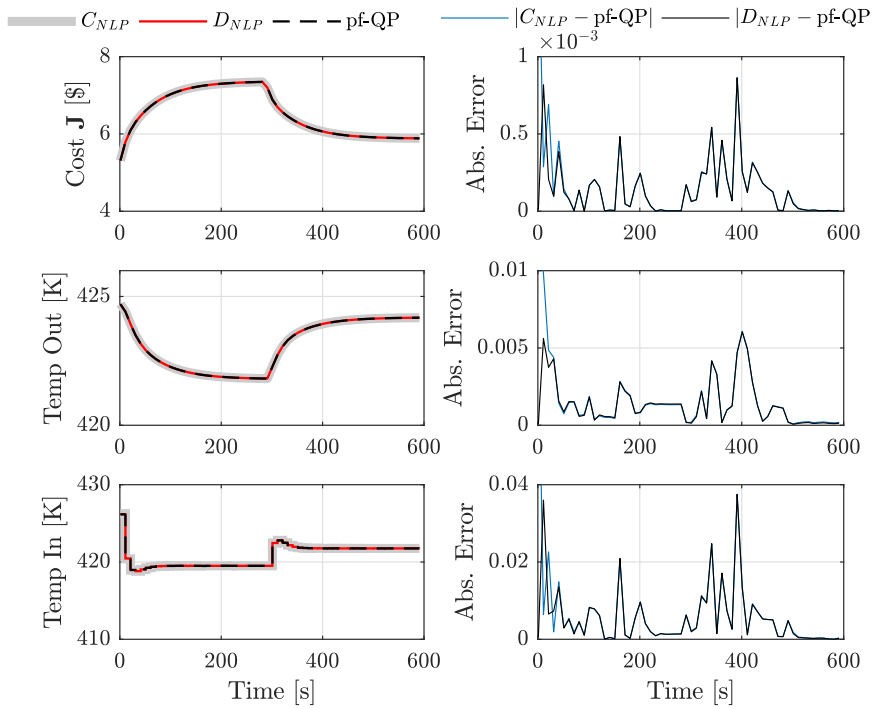


Figure 10.1: Closed-loop simulation results for fully centralized approach C_{NLP} (Thick gray lines), distributed approach with full NLP D_{NLP} (solid red lines) and the proposed path-following approach pf-QP (black dashed lines) for $N_r = 1$, $S = 5$ scenarios

proposed sensitivity-based distributed scenario NMPC (pf-QP) are compared with the fully centralized scenario NMPC (C_{NLP}) and the distributed scenario NMPC solved using full NLPs (D_{NLP}). The closed-loop results and the corresponding absolute errors are shown in Fig. 10.2.

The average CPU times for each subproblem for the two simulation cases are reported in Table 10.1 and graphically represented in Fig. 10.3 for Simulation 1 and 2. Note that the computation time depends heavily on the implementation and computation time of the QP may be further improved by using dedicated high performance QP solvers instead of an off-the-shelf solver.

The simulation results in Fig. 10.1 and Fig. 10.2 clearly demonstrates that the proposed sensitivity-based distributed Scenario NMPC is able to provide a very good approximation of the centralized scenario NMPC and full NLP distributed scenario NMPC from Chapter 9. The simulations also demonstrate that the proposed approach can handle changes in active constraint set between the different subproblems. Simulation 2 with 25 scenarios demonstrates scalability of the proposed approach.

Using the proposed approach as a foundation, one can then reduce the online computation time significantly, by combining it with the advanced step-framework

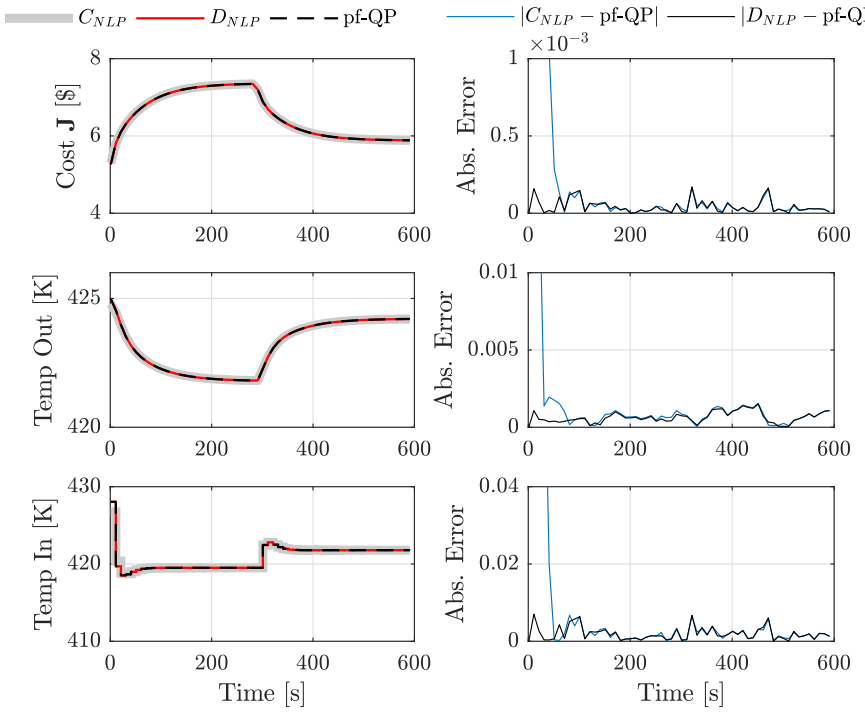
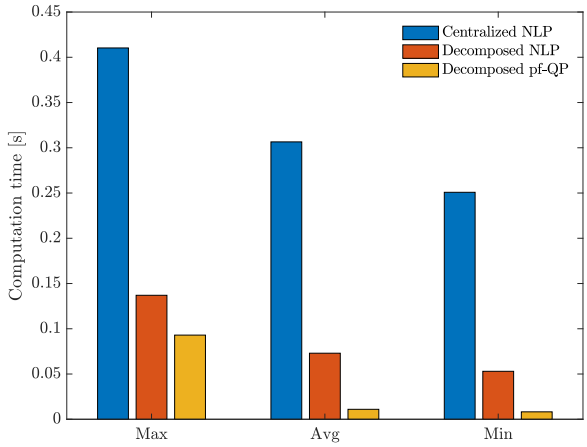


Figure 10.2: Closed-loop simulation results for fully centralized approach C_{NLP} (Thick gray lines), distributed approach with full NLP D_{NLP} (solid red lines) and the proposed path-following approach pf-QP (black dashed lines) for $N_r = 2$, $S = 25$ scenarios

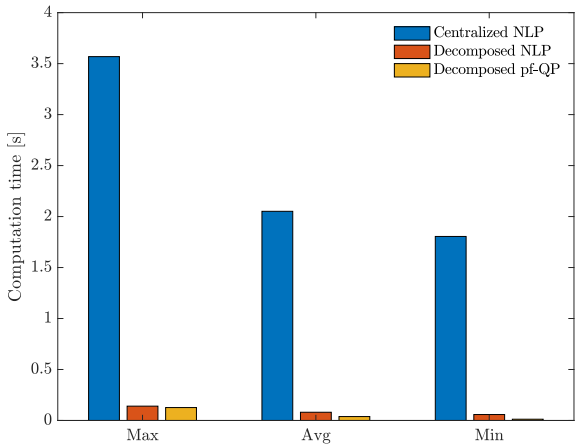
Table 10.1: CPU times (in sec)

	$N_r = 1$			$N_r = 2$		
	max	avg	min	max	avg	min
Centralized	0.4103	0.3065	0.2508	3.5868	2.0541	1.8048
Decomposed NLP	0.1368	0.0737	0.053	0.1405	0.0805	0.0582
Decomposed pf-QP	0.093	0.011	0.0082	0.1266	0.0383	0.0123

proposed in [189] and [178]. The main idea here will be to solve the M^{N_r-1} number of NLPs, offline using the respective predicted state $\hat{\boldsymbol{x}}_{t+1|t}$ as the initial condition. At time $t+1$, when $\hat{\boldsymbol{x}}_{t+1}$ is available, then the solution to the NLPs that was computed offline, can be updated in the same fashion as the advanced-step NMPC, by using the NLP sensitivity with respect to the initial condition $\hat{\boldsymbol{x}}_t$ [189]. The solution to the remaining scenarios can be updated from the M^{N_r-1} scenarios using the NLP sensitivity with respect to the scenarios $\boldsymbol{p}_j$ as proposed in this chapter, thereby avoiding the need to solve any NLPs online. This is an interesting future research direction that can be built upon the proposed approach.



(a)



(b)

Figure 10.3: Maximum, average and minimum CPU times in seconds for (a) simulation 1 (b) simulation 2.

10.4 Chapter summary

Earlier, in Chapter 9, we showed that multistage scenario NMPC can be decomposed using primal decomposition to reduce the online computation time. In this chapter, we have shown that the number of NLPs to be solved can be reduced by using a sensitivity-based path following approach presented in Algorithm 10.1 to solve the different scenario subproblems. Fig. 10.3 clearly shows the reduction in online the computation times by systematically decomposing the scenario subproblems and by exploiting the parametric nature of the subproblems.

Conclusion and Future Outlook

Part I - Summary

In part I of this thesis, we considered the steady-state optimal economic operation of the process. As mentioned in the introduction, one of the fundamental limitations of steady-state optimization is the steady-state wait time. Steady-state optimal operation can be tremendously improved if transient measurements can be used in the optimization problem. In part I, we presented different algorithms to online process optimization that answers the research question,

How can transient measurements be used to achieve optimal steady-state operation and avoid steady-state wait time?

- In Chapter 2, we presented the hybrid RTO approach, where dynamic models were used in the model update step, thereby avoiding the steady-state wait time. The corresponding steady-state model is then used in the optimization step. In Fig. 2.8, we showed that similar performance as dynamic RTO can be achieved using the proposed hybrid RTO at similar computation times as steady-state RTO.
- In Chapter 3, we proposed to convert the hybrid RTO approach into a feedback approach. Here, the steady-state gradient is estimated using transient measurements by linearizing the nonlinear dynamic model around the current operating point. The gradient is then estimated as $\hat{\mathbf{J}}_{\mathbf{u}} = -CA^{-1}B + D$, which is driven to a constant setpoint of zero using feedback control.
- In Chapter 4, we showed how classical feedback controllers, along with some simple logic blocks can be used to achieve optimal operation. A systematic guideline of designing classical feedback controllers for different operating cases and how to switch between the different operating regions was provided in Section 4.2.
- In Chapter 5, we considered the extremum seeking control approach for optimizing the unconstrained degrees of freedom. We showed that, by fixing the plant dynamics, transient measurements can be used, thus speeding up the convergence to the optimum by one order of magnitude compared to the classical extremum seeking control. The proposed method was also shown to be robust to neglected/unmodeled plant dynamics unlike the BI-approach by Bamberger and Isermann [7], which was quantified for a general case in Theorem 5.1. This approach also addresses the cost of developing accurate models (Challenge 1).

- In Chapter 6, we combined self-optimizing control with extremum seeking control. Due to the different time-scale at which these methods operate, these can be combined in a hierarchical fashion to exploit the combined advantages of the two methods. Self-optimizing control was able to provide quick reaction to the disturbances and keep the process in the near-optimal region and extremum seeking control was shown to fine tune the setpoints to the self-optimizing control layer to reduce the loss.

In addition, the methods presented in Chapters 4-6 avoids the need for rigorous nonlinear models, hence addressing Challenge 1.

The methods proposed in Chapters 3-6 also avoids the need to solve numerical optimization problems, hence addressing Challenge 3.

Future research directions

Distributed Optimization of Large-scale Processes

A natural extension to the Hybrid RTO approach presented in Chapter 2, is to develop the Hybrid RTO approach for large-scale distributed process systems with shared resources. Here, the optimization is performed for small clusters of process units and a master problem is used to co-ordinate the sub-problems. The subproblems optimize the local operations and report back the shadow prices to the master problem, which co-ordinates the subproblems by optimally allocating the shared resources. Existing literature on distributed RTO (such as [61, 185]) focuses more on the optimization part, rather than the online model update. The extension of the proposed hybrid RTO approach to a distributed optimization framework enables developing optimization routines for large-scale systems.

In particular, it enables industrial symbiosis and resource sharing in a circular economy setting, where different organizations share common resources and materials. The overall optimal operation of such a symbiotic industrial system involves sharing detailed information about the production network, in the form of models, real time measurements, local constraints and the objective function across the different organizations, which may not be desirable due to several reasons such as intellectual property rights and market competitiveness. Therefore, there is a clear need to optimize such process with limited sharing of information across the different organizations, in order to enable optimal operation in an industrial symbiotic setting. Preliminary results in this direction can be found in [105].

In the spirit of Industrial symbiosis, a very interesting future research direction could be to consider a distributed optimization framework, where the different organizations/subproblems use different RTO strategies to solve their local subproblems, which to the best of my knowledge, has not been studied before.

Gradient estimation algorithms

In chapters 3 - 6, we considered the problem of achieving optimal operation using feedback control, where the ideas was to control the steady-state cost gradient, or a combination of the cost gradients using the unconstrained degrees of freedom. Although the gradients are not measured variables, there exists a wide range of

model-based and model-free gradient estimation methods, including the two new gradient estimation schemes proposed in chapter 3 and 5 respectively. With different gradient estimation algorithms currently available in literature, there is a need for clear overview and understanding of the advantages and disadvantages of the different gradient estimation algorithms [2]. The different model-free gradient estimation algorithms have also been predominantly developed for single-input single-output (SISO) systems. Very little effort has been dedicated to multivariable extremum seeking control, except for a few works such as [56]. For large-scale processes with multiple inputs, computing the steady-state gradient from the different input channels to the scalar cost measurement poses additional challenges. Therefore, there is a clear need to develop gradient estimation algorithms for multivariable processes.

Hierarchical RTO

It is important to note that *there is no single approach to real-time optimization that addresses all the challenges* listed in Chapter 1 (cf. Section 1.1). The different approaches to online process optimization have varying degrees of complexity and flexibility, and the different methods work in different timescales and can handle different kinds of uncertainty. These methods have been developed and investigated rather independently and in general these methods are often seen as competitive to one another. However, as we showed in Chapter 6, the different methods are often complement each other. Based on the timescale separation, one can merge the different approaches in a hierarchical fashion, thereby reaping the potential rewards of the different approaches. Table 10.2 summarizes the advantages and disadvantages of different approaches to real-time optimization.

Table 10.2 is an important result in this thesis, as it aims to provide an overview and a clear understanding of the different RTO approaches.

To this end, we want to stress that the different methods proposed in this thesis are not a replacement of any other method but rather adds to the "toolbox" of available methods for economic process optimization. We summarize Part I with a future outlook on a hierarchical combination of various RTO approaches with three layers as shown in Fig. 10.4.

- **Lowest self-optimizing control layer**: The self-optimizing control acts in the fastest time scale providing immediate reaction to the disturbances and takes the process to the near optimal region, by maintaining the self-optimizing variable at a constant setpoint. By doing so, we keep the process operation in the near optimum region.
- **Middle model-based RTO layer**: We then propose that the traditional model-based RTO approaches (along with the modifications and developments proposed in this thesis) to adjust the setpoints to the self-optimizing variable in order to account for the nonlinearity in the process and reduce the optimality gap. These work in a slower time scale compared to the self-optimizing layer.

[2] Srinivasan et al. [172] provided such a brief overview of some of the gradient estimation algorithms.

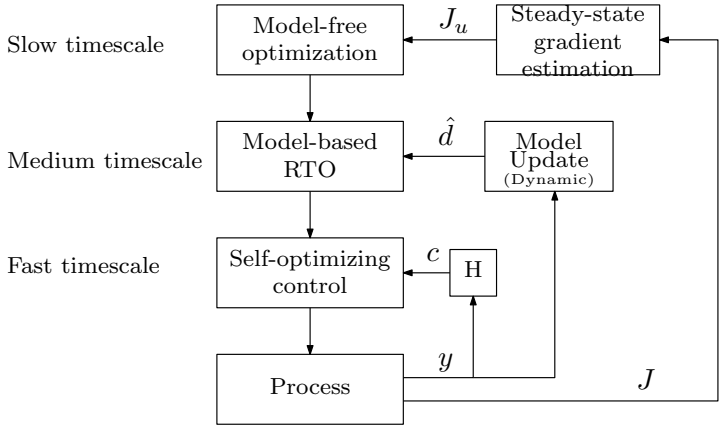


Figure 10.4: Schematic representation of a three layer Hierarchical RTO

- **Top model-free optimization layer**: The middle layer still requires accurate models and may still be subject to structural uncertainty (Challenge 1). Hence, we propose to account for this structural mismatch by estimating and exploiting the plant gradients directly from the cost measurement (model-free) in the slowest time scale, e.g. extremum seeking control and modifier adaptation.

Table 10.2: Comparison of different approaches to online process optimization

	self-optimizing control[a] (Ch. 6)	classical adv. control (Ch. 4)	extremum seeking (Ch. 5,6)[b]	Feedback RTO (Ch. 3)	steady-state RTO (Traditional)	Hybrid RTO (Ch. 2)	economic NMPC/ DRTO (Ch. 7–10)[c]	Modifier adaptation [120]
Cost Measured	No	No	Yes	No	No	No	No	Yes
Model	static model used offline	Only for unconstrained DOF[k]	Model-free	dynamic model used online	static model used online	static and dynamic model used online	dynamic model used online	static model used online
perturbation	No	No	Yes	No	No	No	No	Yes
Transient measurements	Yes	Yes	No/(Yes[h])	Yes	No	Yes	Yes	No
Long-term performance	near-optimal	Optimal	Optimal[e,i]	Optimal[d]	Optimal[d,f]	Optimal[d]	Optimal[d]	Optimal[e,i]
Convergence time	very fast	very fast	very slow[e,h]	fast	slow[f]	fast	fast[g]	very slow[e,h]
Handle change in active constraints	No	Yes[j]	No	No	Yes	Yes	Yes	Yes
Numerical solver	No	No	No	No	static	static	dynamic	static
Computational cost	very low	very low	very low	low	intermediate	intermediate	high	intermediate
Process size	small-scale	small-scale	small-scale	medium-scale	large-scale	large-scale	large-scale	medium-scale[l]

[a] SOC is complementary to the other methods that should ideally be used in combination.
[b] NCO tracking also has similar properties but can also track changes in active constraints in addition.
[c] Economic NMPC typically has non-economic control objectives in addition to the economic objectives in the cost function, whereas DRTO has only economic objectives. Studied in more detail in Part 2 of this thesis.
[d] Converges to model optimum. Converges to the true plant optimum only if model is structurally correct.
[e] requires time scale separation between system dynamics, dither and convergence. Sub-optimal operation for long periods following disturbances.
[f] Slow due to steady-state wait time. Sub-optimal operation for long periods following disturbances.
[g] limited by computation time.
[h] Transient measurements can be used for gradient estimation if local linear dynamics are included as proposed in Chapter 5, resulting in reduced convergence time by one order of magnitude.
[i] Converges to the true plant optimum and not the model optimum.
[j] Need additional logic blocks as presented in Chapter 4. Ok for smaller problems.
[k] Often not used in practice, instead engineering intuition is used to select suitable controlled variables for unconstrained DOF.
[l] limited by plant gradient estimation.

Part II - Summary

In part II of the thesis, we considered the dynamic optimization problem, with particular emphasize on handling uncertainty using the multistage scenario-based economic NMPC formulation. In Part II, we presented novel algorithms that answers the research question

How can computational issues be addressed in the dynamic optimization problem?

- In Chapter 8, we showed that the multistage economic NMPC formulation was able to provide less conservative solution than the worst-case approach, due to the "recourse". However, the problem size increases exponentially, and thus becomes computationally intensive.
- To address this computational issue, we presented a distributed multistage scenario NMPC approach based on primal decomposition in Chapter 9. We showed that the decomposition of the problem enables parallelization. In particular, we showed that primal decomposition always ensures feasibility of the non-anticipativity constraints (unlike dual decomposition), which is an important property for closed-loop implementation. Furthermore, a backtracking algorithm to choose the step length in the master problem was presented in Algorithm 9.1.
- In Chapter 10, we further improved the computation time by recasting the distributed multistage NMPC problem from Chapter 9 using parametric optimization, and showed that the different subproblems can be solved using sensitivity-based path-following approach, instead of solving the full NLP. Fig. 10.3 shows the reduction in the computation time by using Algorithm 9.2 and Algorithm 10.1.

Future research directions

Data analytic tools to understand the uncertainty

When considering optimization under uncertainty, a very important, and often overlooked, notion is to understand the uncertainty, in order to decide what uncertainties one must consider in the optimization problem, and how to handle them. For instance, a model parameter that has low/no effect on the optimal solution, need not be considered in the optimization problem.

Since the ultimate objective of optimization under uncertainty is to hedge against the variability, it is important to first understand and explain the variability, before one seeks robustness to it. Data analytic and statistical tools can be extremely useful in producing such valuable insights about the uncertainty, which can be used to then decide how to handle the uncertainty (if needed).

Most of the works in literature that study the multistage scenario-based approach assume that the uncertainty characteristics are known *a-priori*, and that the discrete scenarios are given, for example, based on engineering insight before the NMPC is designed. However, the issue of how to select the discrete realizations of the uncertainty for the scenario tree generation is an important practical aspect that has not been well studied in this area. Some initial results on how multivariate data analysis tools can be used to judiciously select the scenarios can be found in Appendix K.

Online update of uncertainty for time-invariant uncertainty

For problems with constant but unknown parameters, i.e. time-invariant uncertain parameters, it may be desirable to approach the problem from an adaptive framework rather than a robust framework [60]. An interesting future research direction is to update the uncertainty characteristics (i.e. compact set/ probability density function) instead of updating the parameters directly. The idea of updating the uncertainty characteristics instead of using a parameter estimator itself is not entirely new. Similar ideas of "adaptive-robust" approaches were also briefly explored by Guay et al. [60], Hanssen and Foss [64], Lucia and Paulen [115], where the uncertainty set containing all possible values of the uncertain parameters are estimated instead of adapting the parameters directly. In the context of multistage scenario MPC, we recently proposed an alternative approach using recursive Bayesian probability to update the scenario tree online, which can be found in Appendix L.

Machine learning for process optimization

Finally, the current enthusiasm and promise of machine learning and artificial intelligence is evident in the field of control and process systems engineering, with academics and practitioners alike. Currently, there are many applications that yield quick successful results. However, the greatest and perhaps the most intellectually interesting challenge in this direction lies in understanding and developing conceptual frameworks that address the different challenges of online process optimization. Often machine-learning algorithms are used to build surrogate models and digital twins starting with the assumption that no knowledge of the system is available. To address challenge 1, one must go beyond data-centric machine learning and combine machine-learning models with domain-specific knowledge. This leads to more efficient use of data focused on the unknown and uncertain aspects of the process.

Machine-learning algorithms and artificial intelligence have also been predominantly used from a computer science and statistics perspective. From online process optimization perspective, there is a clear need to tailor machine-learning algorithms, to develop purpose-built models for optimization. As mentioned earlier,

machine-learning algorithms try to learn the model with the objective of minimizing the model prediction error. However, from an optimization perspective, the objective is not to develop an excellent predictive model, but rather a model that will take the process close to its true optimum. A very good predictive model can still be a rather poor optimization model. Therefore, an important research direction for using machine learning in the context of online process optimization is to develop good optimization models. More specifically, the learning objective must be modified not only to minimize the prediction error, but one that also minimizes the error between the model gradients and the true plant gradients.

For many large-scale problems, solvers may take a long time to converge to the optimal solution or, in some cases, may even fail to converge to an optimal solution. For example, with today's computational power, numerical optimization solvers for very simple chemical processes running on standard workstations (for example, 2.6GHz processor and 16GB RAM or similar configuration) typically converge in the time scale of seconds to several minutes. This makes it very difficult to run such numerical computations for more complex processes on embedded platforms and cloud computing, due to their limited computation capacity. This challenge is escalated further with the inclusion of integer decision variables, leading to mixed-integer and combinatorial optimization problems, which are very common in the field of process systems engineering. Currently, the computation cost for solving such problems are prohibitively large for online decision-making.

In order to avoid solving numerical optimization problem online, one can approximate computationally expensive optimization problems using machine-learning algorithms. Instead of developing surrogate models that will be used in the optimizer, one can build "surrogate optimizers" that approximate the numerical optimization solvers. Similar ideas are recently explored in [80] in the context of nonlinear MPC, and in [88] in the context of real time optimization. However, it is worth noting that the training can be very expensive. Another alternative to approximating the numerical optimization solver is to train black-box models that predict the plant gradients, which can then be used to drive the process to its optimum (similar to extremum seeking control). The reader is referred to Appendix M for some preliminary discussions in this direction.

Appendices

Appendix A

Modeling of a gas lifted well network

Consider a network of gas-lifted wells producing to a common manifold as shown in Fig. A.1. Production from a cluster of $\mathcal{N} = \{1, \cdots, n_w\}$ gas lifted well can be described using differential and algebraic equations. The dynamics are include in the model due to the mass balances in each well which are described by the following differential equations.

$$\dot{m}_{ga_i} = w_{gl_i} - w_{iv_i} \tag{A.1a}$$

$$\dot{m}_{gt_i} = w_{iv_i} - w_{pg_i} + w_{rg_i} \tag{A.1b}$$

$$\dot{m}_{ot_i} = w_{ro_i} - w_{po_i} \qquad \forall i \in \mathcal{N} \tag{A.1c}$$

where, m_{ga_i} is the mass of gas in the annulus, m_{gt_i} is the mass of gas in the well tubing, m_{ot_i} is the mass of oil in the well tubing, w_{gl_i} is the gas lift injection rate, w_{iv_i} is the gas flow from the annulus into the tubing, w_{pg_i} and w_{po_i} are the produced gas and oil flow rates respectively and, w_{rg_i} and w_{ro_i} are the gas and oil flow rates from the reservoir for each well i. The mass balance in the riser for oil and gas phase is given by,

$$\dot{m}_{gr} = \sum_{i=1}^{n_w} w_{pg_i} - w_{tg} \tag{A.2a}$$

$$\dot{m}_{or} = \sum_{i=1}^{n_w} w_{po_i} - w_{to} \tag{A.2b}$$

where, m_{gr} is the mass of gas in the riser and m_{or} is the mass of oil in the riser and w_{tg} and w_{to} are the total gas and oil flow rates respectively. The densities ρ_{a_i} (density of gas in the annulus in each well) and ρ_{m_i} (fluid mixture density in the tubing for each well) and ρ_r (fluid mixture density in the riser) are given by,

$$\rho_{a_i} = \frac{M_w p_{a_i}}{T_{a_i} R} \tag{A.3a}$$

$$\rho_{w_i} = \frac{m_{gt_i} + m_{ot_i} - \rho_o L_{bh_i} A_{bh_i}}{L_{w_i} A_{w_i}} \tag{A.3b}$$

$$\rho_r = \frac{m_{gr} + m_{or}}{L_r A_r} \qquad \forall i \in \mathcal{N} \tag{A.3c}$$

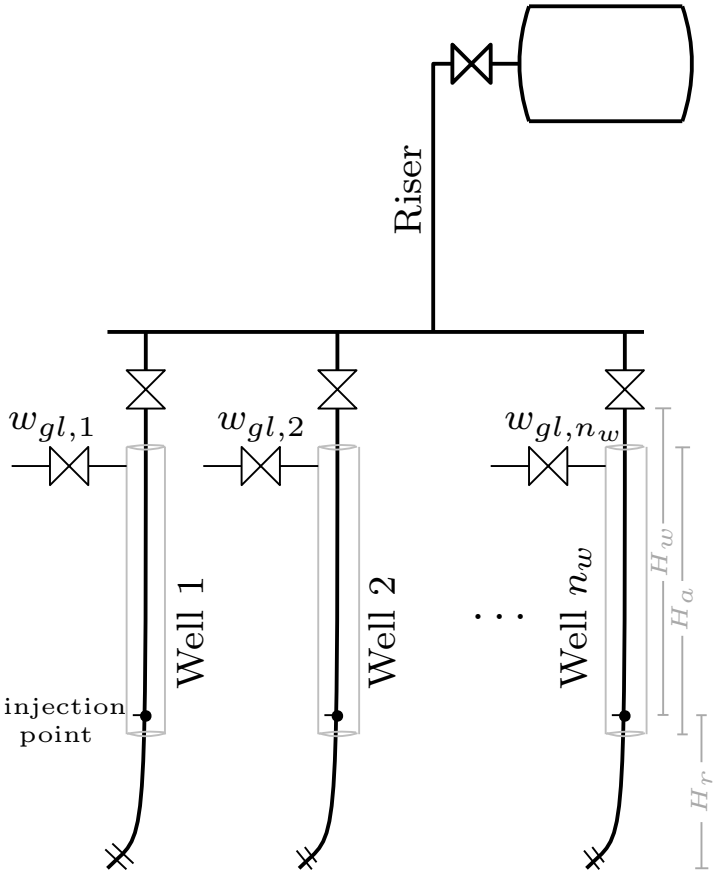


Figure A.1: Schematic representation of a production network with gas-lifted wells

where M_w is the molecular weight of the gas, R is the gas constant, T_{a_i} is the temperature in the annulus in each well, ρ_o is the density of oil in the reservoir, L_{bh_i} and L_{w_i} are the lengths of each well above and below the injection point respectively and A_{bh_i} and A_{w_i} are the cross-sectional area of each well above and below the injection point respectively. L_r and A_r are the length and the cross sectional area of the riser manifold. The annulus pressure p_{a_i}, wellhead pressure p_{wh_i}, well injection point pressure p_{iv_i} and the bottom hole pressure p_{bh_i} for each

well are given by,

$$p_{a_i} = \left(\frac{T_{a_i}R}{V_{a_i}M_w} + \frac{gH_{a_i}}{L_{a_i}A_{a_i}}\right) m_{ga_i} \tag{A.4a}$$

$$p_{wh_i} = \frac{T_{w_i}R}{M_w}\left(\frac{m_{gt_i}}{L_{w_i}A_{w_i} + L_{bh_i}A_{bh_i} - \frac{m_{ot_i}}{\rho_o}}\right) - \frac{1}{2}\left(\frac{m_{gt} + m_{ot}}{L_wA_w}gH_w\right) \tag{A.4b}$$

$$p_{wi_i} = p_{wh_i} + \frac{g}{L_{w_i}A_{w_i}}(m_{ot_i} + m_{gt_i} - \rho_o L_{bh_i}A_{bh_i})H_{w_i} + \Delta p^{\mathsf{T}}_{fric} \tag{A.4c}$$

$$p_{bh_i} = p_{wi_i} + \rho_{w_i}gH_{bh_i} + \Delta p^{bh}_{fric} \qquad \forall i \in \mathcal{N} \tag{A.4d}$$

where L_{a_i} and A_{a_i} are the length and cross sectional area of each annulus, T_{w_i} is the temperature in each well tubing, H_{r_i} and H_{w_i} are the vertical height of each well tubing below and above the injection point respectively and g is the acceleration of gravity constant. $\Delta p^{\mathsf{T}}_{fric}$ and Δp^{bh}_{fric} represents the frictional pressure drop in the well tubing above and below the gas injection point respectively. The manifold pressure p_m and the riser head pressure p_{rh} are given by,

$$p_{rh} = \frac{T_rR}{M_w}\left(\frac{m_{gr}}{L_rA_r}\right) \tag{A.5a}$$

$$p_m = p_{rh} + \rho_r gH_r + \Delta p^r_{fric} \tag{A.5b}$$

where T_r is the average temperature in the riser, H_r is the vertical height of the riser and Δp^r_{fric} is the frictional pressure drop in the riser. The flow through the downhole gas lift injection valve w_{iv_i}, total flow through the production choke w_{pc_i}, produced gas and oil flow rate, and the reservoir oil and gas flow rates are given by,

$$w_{iv_i} = C_{iv_i}\sqrt{max(0, \rho_{a_i}(p_{a_i} - p_{wi_i}))} \tag{A.6a}$$

$$w_{pc_i} = C_{pc_i}\sqrt{max(0, \rho_{w_i}(p_{wh_i} - p_m))} \tag{A.6b}$$

$$w_{pg_i} = \frac{m_{gt_i}}{m_{gt_i} + m_{ot_i}}w_{pc_i} \tag{A.6c}$$

$$w_{po_i} = \frac{m_{ot_i}}{m_{gt_i} + m_{ot_i}}w_{pc_i} \tag{A.6d}$$

$$w_{ro_i} = PI_i(p_{r_i} - p_{bh_i}) \tag{A.6e}$$

$$w_{rg_i} = GOR_i \cdot w_{ro_i} \qquad \forall i \in \mathcal{N} \tag{A.6f}$$

where, C_{iv_i} and C_{pc_i} are the valve flow coefficients for the downhole injection valve and the production choke for each well respectively, PI_i is the reservoir productivity index, p_{r_i} is the reservoir pressure and GOR_i is the gas-oil ratio for each well. The two wells produce to a common manifold, where the manifold pressure is denoted by p_m and the flow rates from the two well mixes together. The total flow through

Table A.1: List of well parameters and their corresponding values used in the results.

Parameter	units	Well 1	Well 2
L_w	[m]	1500	1500
H_w	[m]	1000	1000
D_w	[m]	0.121	0.121
L_{bh}	[m]	500	500
H_{bh}	[m]	500	500
D_{bh}	[m]	0.121	0.121
L_a	[m]	1500	1500
H_a	[m]	1000	1000
D_a	[m]	0.189	0.189
ρ_o	[kg m^{-3}]	800	800
C_iv	[m^2]	0.1E-3	0.1E-3
C_pc	[m^2]	2E-3	2E-3
p_r	[bar]	150	155
PI	[kg s^{-1} bar^{-1}]	0.7	0.7
T_a	[°C]	28	28
T_w	[°C]	32	32
GOR	[kg/kg]	0.1±0.05	0.12±0.02

the riser head choke w_{rh}, the total produced oil and gas rates are then given by,

$$w_{rh} = C_{rh}\sqrt{\rho_r(p_{rh} - p_s)} \tag{A.7a}$$

$$w_{tg} = \frac{m_{gr}}{m_{gr} + m_{or}} w_{rh} \tag{A.7b}$$

$$w_{to} = \frac{m_{or}}{m_{gr} + m_{or}} w_{rh} \tag{A.7c}$$

where C_{rh} is the valve flow coefficient for the riser head valve and p_s is the separator pressure, which is assumed to be held at a constant value.

As seen from (A.1a) - (A.7c), the gas lifted well is modeled as a semi-explicit index-1 DAE system of the form

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) \tag{A.8a}$$

$$\mathbf{G}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) = 0 \tag{A.8b}$$

where $\mathbf{F}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d})$ is the set of differential equations (A.1a) - (A.2b) and $\mathbf{G}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) = 0$ is the set of algebraic equations (A.3a) - (A.7c), $\mathbf{x} \in \mathbb{R}^{3n_w+2}$ are the set of differential variables, $\mathbf{z} \in \mathbb{R}^{12n_w+6}$ are the set of algebraic variables, $\mathbf{u} \in \mathbb{R}^{n_w}$ are the set of control inputs and $\mathbf{d} \in \mathbb{R}^{n_w}$ are the set of uncertain parameters.

Table A.2: List of riser parameters and their corresponding values used in the results.

Parameter	**units**	**Riser**
L_r	[m]	500
H_r	[m]	500
D_r	[m]	0.121
$C_r h$	[m^2]	10E-3
p_s	[bar]	20
T_r	[°C]	30
M_w	[g mol^{-1}]	20
R	[J mol^{-1} K^{-1}]	8.314

Appendix B

Dynamic model adaptation using extended Kalman filter

In this thesis, dynamic model adaptation was used in Chapters, 2, 3 and 8 among others, for online state and parameter estimation. This appendix details the use of extended Kalman filter for parameter estimation using an augmented state vector $\mathbf{x}' = [\mathbf{x}^\mathsf{T}, \mathbf{d}^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{n_x+n_\mathbf{d}}$ constructed using the states and the uncertain parameters.

The set of uncertain variables $\mathbf{d}$ is modeled using an integrated noise term

$$\mathbf{d}_{k+1} = \mathbf{d}_k + \mathbf{w}_{\mathbf{d},k} \tag{B.1}$$

where, $\mathbf{w}_{\mathbf{d},k} \sim \mathcal{N}(0, \mathbf{Q}_\mathbf{d})$ is a small artificial noise with zero mean and covariance $\mathbf{Q}_\mathbf{d}$ term that allows the Kalman filter to adjust the estimate the of the parameter [164]. The augmented system is then given by

$$\begin{aligned} \mathbf{x}'_{k+1} = \begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{d}_{k+1} \end{bmatrix} &= \mathbf{f}'(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k) + \mathbf{w}'_k \\ \mathbf{y}_{meas,k} &= \begin{bmatrix} \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{d}_k \end{bmatrix} + \mathbf{v}_k \end{aligned} \tag{B.2}$$

where $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R})$ is the normally distributed measurement noise with zero mean and covariances $\mathbf{R}$ and the augmented system model $\mathbf{f}'(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k)$ is constructed as shown,

$$\mathbf{f}'(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k) = \begin{bmatrix} f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k) + \mathbf{w}_k \\ \mathbf{d}_k + \mathbf{w}_{\mathbf{d},k} \end{bmatrix} \tag{B.3}$$

where $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q})$ is the normally distributed process noise with zero mean and covariances $\mathbf{Q}$.

Remark B.1. The maximum dimension of the disturbance $\mathbf{d}$ that one can choose, such that the augmented system remains detectable is equal to the number of measurements (i.e. $n_d \leq n_y$) [144].

The discrete-time extended Kalman filter for the augmented system is then given by [164],

$$\hat{\mathbf{x}}'_{k|k-1} = \mathbf{f}'(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, \hat{\mathbf{d}}_{k-1|k-1}) \tag{B.4a}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^{\mathsf{T}} + \mathbf{Q}'_{k-1} \tag{B.4b}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathsf{T}}(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k + \mathbf{R}_k)^{-1} \tag{B.4c}$$

$$\hat{\mathbf{x}}'_{k|k} = \hat{\mathbf{x}}'_{k|k-1} + \mathbf{K}_k(\mathbf{y}_{meas,k} - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}, \mathbf{u}_k)) \tag{B.4d}$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1} \tag{B.4e}$$

where P is the covariance of the state and parameter estimates, K is the Kalman gain, $\mathbf{F}$ and H depicts the linearized system around the current estimate, given by

$$\mathbf{F} = \left.\frac{\partial \mathbf{f}'}{\partial \mathbf{x}'}\right|_{\mathbf{x}'=\hat{\mathbf{x}}'} \quad \mathbf{H} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}'}\right|_{\mathbf{x}'=\hat{\mathbf{x}}'} \tag{B.5}$$

and the augmented covariance $\mathbf{Q}'$ is given by,

$$\mathbf{Q}' = \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & \mathbf{Q_d} \end{bmatrix} \tag{B.6}$$

Appendix C

Application Examples of feedback RTO

This appendix contains papers showing three application examples of the Feedback RTO approach proposed in Chapter 3.

- Application to oil production optimization - Paper published in2018 IFAC International Workshop on Automatic Control in Offshore Oil and Gas Production (OOGP), Esbjerg, Denmark*.
- Application to three bed ammonia reactor - Paper published in 2018 Computer Aided Chemical Engineering (ESCAPE 28), Graz, Austria.
- Application to Evaporator process - Paper published in 2019 PSE Asia, Bangkok, Thailand.

* This paper received the IFAC-ABB Best paper award.

Gas-lift Optimization by Controlling Marginal Gas-Oil Ratio using Transient Measurements ⋆

Dinesh Krishnamoorthy Esmaeil Jahanshahi Sigurd Skogestad

Dept. of Chemical Engineering, Norwegian Univ. of Science & Technology , NO-7491 Trondheim , (e-mail: dinesh.krishnamoorthy@ntnu.no, esmaeil.jahanshahi@hotmail.com, skoge@ntnu.no).

Abstract: This paper presents the application of a steady-state gradient control using transient measurements to a gas-lift optimization problem. Optimal operation of a gas-lifted field involves controlling the marginal gas-oil ratio (mGOR), which is the steady-state gradient of the oil rate with respect to the gas lift rate. In this paper, we apply a novel method to estimate the marginal GOR online using a dynamic model and transient measurements, without the need for additional perturbation. The proposed method is based on linearizing the dynamic model around the current operating point to estimate the marginal GOR, which is then controlled using simple feedback controllers to achieve optimal operation. In case of disturbances, the proposed method is able to adjust fast to the new optimal point, without the need to solve computationally expensive optimization problems. By using transient measurements, it does not need to wait for the process to reach steady-state to update the model. The proposed method was tested in simulations and was shown to provide similar performance as an economic MPC.

Keywords: Production optimization, Measurement-based optimization, real-time optimization, plant-wide control

1. INTRODUCTION

In many mature oil and gas production fields, when the reservoir pressure is not sufficient to lift the fluids economically to the surface, artificial lift methods are used to boost the production. Gas-lift is one such commonly used artificial lift method, where compressed gases are injected into the well tubing to reduce the hydrostatic pressure drop and hence increase production. Injecting too much gas also has a detrimental effect on the oil production due to increased frictional pressure losses in the well tubing. Each well then has an optimal gas-lift injection rate that optimizes the production. This paper deals with the problem of finding the optimal gas lift injection rates for the different wells.

Daily production optimization is an important aspect of operating an oil and gas production network. Many different approaches are available in literature to optimize the production from a gas-lifted well network. Traditionally, production engineers use commercially available steady-state multiphase simulators to generate the so-called *gas-lift performance curves*, which gives the static mapping between the oil production rate and the gas lift injection rate (Rashid, 2010). Nonlinear steady-state optimization tools may then be used to compute the optimal gas lift injection rates. However, a common approach to optimizing production from a gas-lifted well network is to use the gas-lift performance curves directly. The optimal allocation of gas-lift rate is known to occur when the marginal gas-oil ratio is equal for all the wells. Marginal gas-oil ratio or simply known as marginal GOR, is a quantity that describes the increase in oil rate per unit change in the gas-lift injection rate. In other words, marginal GOR is given by the slope of the gas-lift performance curves (Bieker et al., 2007).

The principle of marginal GOR has been proven to be the optimal solution for any parallel unit such as the gas lift production network (Downs and Skogestad, 2011), and was also used by Urbanczyk et al. (1994) and Kanu et al. (1981). This was also later shown to fulfill the necessary condition of optimality (Sharma and Glemmestad, 2013). Therefore, the simplest approach to optimal gas lift allocation is by controlling the marginal GOR to be equal for all the wells.

Recently, the use of centralized dynamic optimization solutions such as economic NMPC has been gaining popularity in the process control literature. The use of economic NMPC for the gas lift optimization was considered by Codas et al. (2016) and Krishnamoorthy et al. (2016a). There is no doubt that theoretically optimal performance can be achieved by using such centralized optimizing controllers, however, solving a numerical optimization problem may be computationally intensive and can potentially lead to computational delays. Moreover controller tuning and pro-

⋆ The authors gratefully acknowledge the financial support from SFI SUBPRO, which is financed by the Research Council of Norway, major industry partners and NTNU.

longed maintenance over time is crucial to ensure good performance (Skogestad and Postlethwaite, 2007).

There have also been developments in other optimization methods, where instead of solving a numerical optimization problem, optimal operation is achieved via feedback control. Such methods are classified as *direct input adaptation* methods (Chachuat et al., 2009), where the optimization problem is converted into a feedback control problem.

Self-optimizing control is one example of such a method. It involves finding the right controlled variable which when kept constant leads to near optimal operation (i.e. minimum loss). Alstad (2005) demonstrated the application of self-optimizing control using nullspace method for gas lift optimization. This method however is based on local linearization around the nominal operating point and may lead to steady-state losses if the disturbances moves the operation of the process far away from the nominal operating point. The ideal self-optimizing variable for the gas-lift problem would indeed be the marginal GOR. However, the major challenge is that the marginal GOR is not a readily available measurement for control.

Model-free methods such as extremum seeking control has been applied for gas lift wells by Peixoto et al. (2015) and Krishnamoorthy et al. (2016b). Extremum seeking control involves estimating the steady-state gradient (the marginal GOR) directly using the measurements. The estimated marginal GOR is then controlled using simple integral action. The main advantage of these methods is that it does not require a model. However, to estimate the marginal GOR accurately, constant perturbations of the manipulated inputs are required, which may be undesirable in many oil production wells. It also requires direct measurement of the cost function.

More importantly, the use of transient measurements in such model-free methods leads to erroneous gradient estimation. Therefore, such methods often require clear time scale separation between the plant dynamics , excitation signal and the convergence to the optimum, such that the plant can be approximated as a static map (Krstić and Wang, 2000). This results in very slow convergence to the optimum. Gas-lift wells typically have long settling times due to compressibility of the gas in the annulus and transport time inside the well tubing. Therefore, the time scale separation can make such model-free methods prohibitively slow for gas lift optimization. Additionally, abrupt disturbances may cause undesired responses during the transients, which was motivated in Krishnamoorthy et al. (2016b) using a gas-lift optimization problem.

In this paper, we propose to use a new model-based steady-state gradient estimation method to drive the process to optimal operation (Krishnamoorthy et al., 2018). It uses available transient measurements along with a dynamic model online to estimate the exact steady-state gradient around the current operating point without any additional perturbation. Consequently, it converges to the new optimum point in the fast time scale following a disturbance. The proposed method also does not require the need to measure the cost directly. Furthermore, the proposed method is computationally cheap, since the optimization is done via feedback.

The main contribution of this paper is the application of the new steady-state gradient estimation method using transient measurements to the gas-lift optimization problem, which is demonstrated using a simple case example with two gas lifted wells connected to a common manifold.

The reminder of the paper is organized as follows. Section 2 introduces the proposed method and its application for both, unconstrained and constrained gas-lift optimization cases. Simulation results for both the cases are provided in section 3 and compared with economic NMPC. Some discussions are provided before concluding the paper in section 4.

2. PROPOSED METHOD

In this paper we consider a gas lifted well network with n_w wells. The gas lifted well network can be modelled as a dynamic model,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) \tag{1a}$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)) \tag{1b}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ is the vector of differential variables, $\mathbf{u} \in \mathbb{R}^{n_u}$ is the vector of manipulated variables, $\mathbf{d} \in \mathbb{R}^{n_d}$ is the vector of process disturbances and $\mathbf{y} \in \mathbb{R}^{n_y}$ is the vector of measurements. $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$ is the set of differential equations and $\mathbf{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ is the measurement model. The reader is referred to Krishnamoorthy et al. (2016a) for detailed description of the model.

In a gas lifted well, the flow from the wells are predominantly controlled by the gas lift injection rates under normal operating conditions. Typically, well head chokes are used only under unusual conditions such as to dampen slug flow or to control casing-heading etc., which are not considered in this work for the sake of simplicity. Therefore, in this work, the manipulated inputs are set to be the gas lift injection rates $\mathbf{u} = [w_{gl,1}, \ldots, w_{gl,n_w}]^\mathsf{T}$.

Let the total oil production rate w_{to} be given by,

$$w_{to} = \sum_{i=1}^{n_w} w_{po,i} = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \tag{2}$$

where, $\mathbf{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $w_{po,i}$ is the oil production rate from the i^{th} well..

Marginal GOR is defined as the change in oil rate per unit change in the gas lift rate which is equivalent to the steady-state gradient of (2) with respect to the control inputs. Let the marginal GOR be represented by the symbol ν

$$\nu_i = \frac{\partial w_{to}}{\partial w_{gl,i}} \quad \forall i \in \{1, \ldots, n_w\} \tag{3}$$

In order to control the marginal GOR, we propose a new approach to estimating the marginal GOR using transient measurements. The proposed method uses a dynamic model (1) online to estimate the states and the disturbances using any state estimator such as the extended Kalman filter (EKF) (Simon, 2006). The dynamic model from the total oil production rate (2) to the manipulated variables $\mathbf{u}$ is then linearized around the current operating point to get a local linear dynamic model approximation of the form,

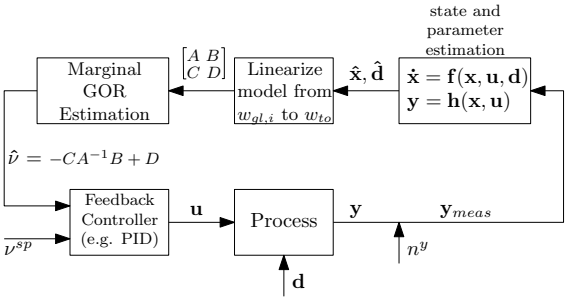


Fig. 1. The proposed method to estimate and control the steady-state gradient (marginal GOR) using transient measurements.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{4a}$$
$$w_{to} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{4b}$$

where,

$$\mathbf{A} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{x=\hat{x}} \quad \mathbf{B} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{x=\hat{x}}$$
$$\mathbf{C} = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right|_{x=\hat{x}} \quad \mathbf{D} = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right|_{x=\hat{x}}$$

The corresponding local linear steady-state model is given by setting $\dot{\mathbf{x}} = 0$. Consequently, the estimated steady-state marginal GOR is given by,

$$\hat{\boldsymbol{\nu}} = -\mathbf{C}\mathbf{A}^{-1}\mathbf{B} + \mathbf{D} \tag{5}$$

where, $\hat{\boldsymbol{\nu}} = [\nu_1 \ \dots \ \nu_{n_w}]^\mathsf{T}$ is the vector of estimated marginal GOR values.

The estimated marginal GOR can then be controlled using any feedback controller to a constant setpoint to achieve optimal operation. This is schematically represented in Fig.1. In the following subsections, we consider simple control structures to control the marginal GOR for different cases.

2.1 Unconstrained Optimization problem

In the unconstrained optimization case, we assume that there is an unlimited supply of gas available for gas lift, and the objective is to maximize the total oil production by computing the optimal gas lift injection rates $w_{gl,i}$ for all the wells i.

$$\min_{\mathbf{u}} \quad J = -\sum_{i=1}^{n_w} w_{po,i} \tag{6a}$$
s.t.
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) \tag{6b}$$

$$\hat{\mathbf{J}}_{\mathbf{u}} = \frac{\partial J}{\partial \mathbf{u}} = \begin{bmatrix} \dfrac{\partial w_{to}}{\partial w_{gl,1}} \\ \vdots \\ \dfrac{\partial w_{to}}{\partial w_{gl,n_w}} \end{bmatrix} = \begin{bmatrix} \hat{\nu}_1 \\ \vdots \\ \hat{\nu}_{n_w} \end{bmatrix} \tag{7}$$

The estimated marginal GOR can then be controlled to drive the system to its optimum using any feedback controller such as a PI controller. To maximize the total oil production in the unconstrained case, the marginal GOR

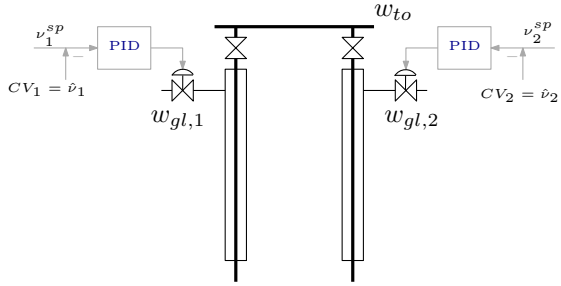


Fig. 2. A gas lifted field with $n_w = 2$ wells and the proposed controller design in the case of unlimited gas lift supply.

of each well is driven to a constant setpoint of $\nu_i^{sp} = 0$ (thus satisfying the necessary conditions of optimality).

In many gas lifted fields, the objective is not only to maximize the oil production, but also minimize the usage of gas lift, due to the costs associated with compressing the gas. The modified cost function J' then has additional terms that penalizes input usage.

$$\min_{\mathbf{u}} \quad J = -c_o \sum_{i=1}^{n_w} w_{po} + c_{gl} \sum_{i=1}^{n_w} w_{gl,i} \tag{8a}$$
s.t.
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) \tag{8b}$$

where c_o and c_{gl} are the value of oil and cost of gas compression respectively. In this case, the steady-state gradient of the modified cost function is given by

$$\hat{\mathbf{J}}_{\mathbf{u}} = \frac{\partial J}{\partial \mathbf{u}} = \begin{bmatrix} -c_o\hat{\nu}_1 + c_{gl} \\ \vdots \\ -c_o\hat{\nu}_{n_w} + c_{gl} \end{bmatrix} \tag{9}$$

At the optimum $\hat{\mathbf{J}}_{\mathbf{u}} = 0$. Therefore the estimated marginal GOR $\hat{\nu}_i$ in (5) is now controlled to a constant setpoint of $\nu_i^{sp} = c_{gl}/c_o$ to achieve optimal operation.

For a gas lifted field with n_w wells, we can use n_w decentralized PI controllers to control the marginal GOR as shown in Fig.2.

$$w_{gl,i} = \left(Kp_i + \frac{K_{I_i}}{s}\right)(\nu_i^{sp} - \hat{\nu}_i) \ , \forall i = \{1, \dots, n_w\} \tag{10}$$

2.2 Constrained Optimization problem

In many cases, the total gas available for gas lift is limited due to the limited compression capacity. In such cases, the total available gas lift must be optimally allocated among the different wells. The optimization problem can be written as,

$$\min_{\mathbf{u}} \quad J = -c_o \sum_{i=1}^{n_w} w_{po} + c_{gl} \sum_{i=1}^{n_w} w_{gl,i} \tag{11a}$$
s.t.
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) \tag{11b}$$
$$\sum_{i=1}^{n_w} w_{gl,i} \leq w_{gl}^{max} \tag{11c}$$

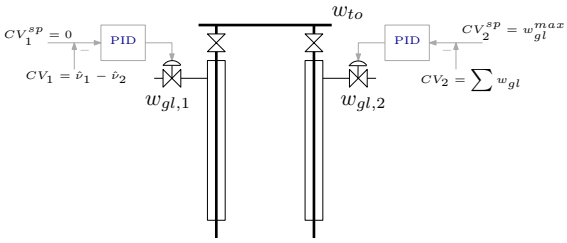


Fig. 3. A gas lifted field with $n_w = 2$ wells and the proposed controller design in the case of limited gas supply.

where w_{gl}^{max} is the maximum gas lift rate.

When the total available gas lift rate is insufficient to operate all the wells at their local optimum, then the optimum occurs when all the available gas is optimally allocated among the different wells, i.e. the maximum gas lift rate becomes active at the optimal operation. Therefore, according to good plant-wide control practice, one of the wells is used to control this active constraint tightly (Skogestad, 2000). The remaining $(n_w - 1)$ unconstrained degrees of freedom are used to maintain the marginal GOR of the wells to be equal. This is because the optimal operation of any parallel unit happens when the marginal cost of the different units (wells) are equal (Downs and Skogestad, 2011). The $(n_w - 1)$ self-optimizing controlled variables would then be

$$\nu_i - \nu_{i+1} \quad \forall i \in \{1, \ldots, n_w - 1\} \tag{12}$$

In such a case, $(n_w - 1)$ feedback controllers are used to maintain equal marginal GOR and 1 controller is used to control the active constraint tightly. For example, in a production network with two wells, if well 2 is used to control the active constraint tightly, then well 1 maintains the marginal GOR of all the wells to be equal, i.e. the controlled variable is set to $\nu_1 - \nu_2$ and controlled to a constant setpoint of zero. This is schematically represented in Fig.3.

3. SIMULATION RESULTS

In this section, we demonstrate the proposed method for gas lift optimization using a case study with two gas lifted wells. For the state and parameter estimation, we use a discrete time extended Kalman filter with a sampling time of 1s as used by Krishnamoorthy et al. (2017). Decentralized PID controllers were used to control the estimated marginal GOR. The PI controllers were tuned using the SIMC tuning rules (Skogestad, 2003). The plant simulator was implemented using `IDAS` integrator.

The proposed method was compared with an centralized optimizing control structure. An economical nonlinear model predictive control was implemented with a sampling time of 5 min and a prediction horizon of 60 samples. The continuous time model was discretized using a third order collocation. The reader is referred to Krishnamoorthy et al. (2016a) for more detailed description on this.

3.1 Unconstrained case

In this subsection, we consider that the total gas available for gas lift is unlimited. In the first simulation, we want to maximize the oil production from the two wells using the cost function (6). The estimated marginal GOR for each well is then controlled to a constant setpoint of zero. The PI controller gains were tuned using the SIMC tuning rules and the resulting PI tuning values used for the two controllers are shown in Table 1. The disturbance enters in the form of step changes in gas-oil ratio (GOR) from the reservoir. The GOR for well 1 increases from 0.1 to 0.12 at time $t = 2h$ and the GOR of well 2 decreases from 0.12 to 0.1 at $t = 3h$. The simulation results using the proposed method compared with economic NMPC is shown in Fig.4.

Table 1. PI controller tunings.

		K_p	K_I
Unconstrained case	Well 1	7.0934	0.0149
	Well 2	11.1111	0.0214
Constrained case	Well 1	7.0934	0.0149
	Well 2	0	1

It can be clearly seen that the computed optimal gas lift rates by the proposed method converges to the same solution as the economic NMPC.

We then simulate the same problem, but now the objective is to maximize the oil production and at the same time minimize the costs associated with gas compression as shown in (8). The value of oil $c_o = 1\$$ and the cost of compression $c_{gl} = 0.5\$$. Therefore the marginal GOR of both wells are now controlled to a constant setpoint of $c_{gl}/c_o = 0.5$ instead of 0. The PI controller tunings were the same as shown in Table 1. We consider the same disturbances in GOR as in the previous simulation case. The simulation results are shown in Fig.5 and are compared with the solution provided by the economic NMPC. It can be clearly seen that the proposed method is able to provide similar performance as that of an economic MPC.

3.2 Limited Gas lift case

In this simulation case, we now consider that the total available gas for gas lift is limited to $w_{gl}^{max} = 4kg/s$. Well 1 is used to control the marginal GOR of the two wells to be equal, whereas well 2 tightly controls the active constraint. The proposed method estimates the marginal GOR of both the wells and a PI controller is used to control $\hat{\nu}_2 - \hat{\nu}_1$ to a constant setpoint of zero. By doing so, we ensure that the marginal GOR of the two wells would be equal. The PI controller tuning is shown in Table 1. We consider the same disturbances in GOR as in the previous simulation case. The simulation results are shown in Fig.6 and compared with the optimal solution provided by economic NMPC. It can be clearly seen that the proposed method is able to provide a similar performance as that of an economic MPC.

4. DISCUSSION AND CONCLUSIONS

In this paper, we presented some simple plant-wide control structure design for optimal operation of gas lifted wells.

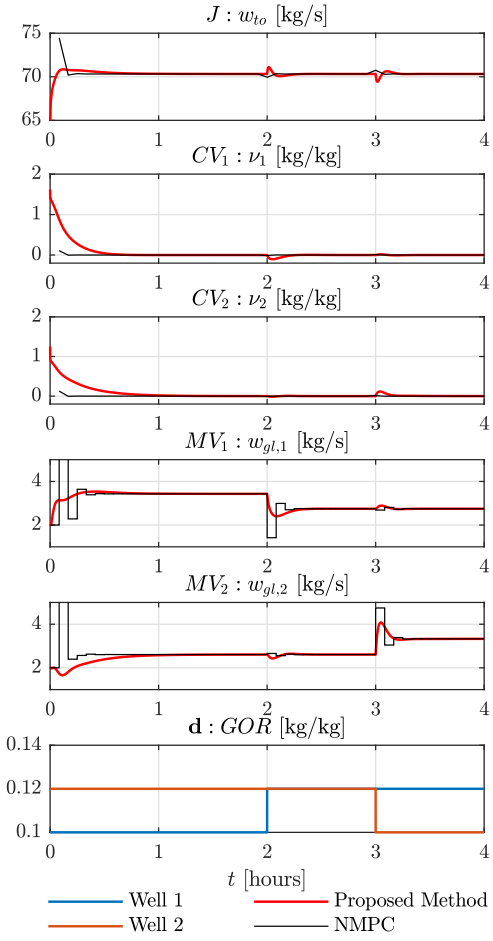


Fig. 4. Simulation results for the unconstrained case when the cost function is given by (6). The proposed method is shown in red lines and the economic NMPC is shown in thin black lines.

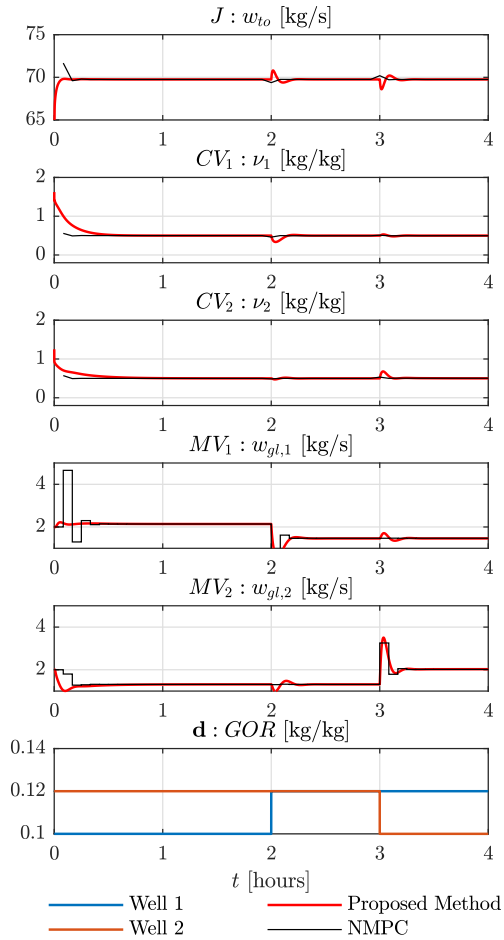


Fig. 5. Simulation results for the unconstrained case when the cost function is given by (8). The proposed method is shown in red lines and the economic NMPC is shown in thin black lines.

We showed that optimal operation can be achieved by using a dynamic model online to estimate the marginal GOR using transient measurements and control the marginal GOR to constant setpoints. The performance of the proposed control strategy was compared to economic NMPC and was shown to provide similar response as the economic NMPC. The proposed method is based on feedback control and hence is computationally cheap to implement. The average computation times for the case study used here were $0.005s$ for the proposed method as opposed to $0.924s$ for the economic NMPC. The proposed control structure is easier for the operators to understand. Additionally, the PI controllers are also easier to tune and maintain than the economic NMPC solution.

It is however important to note that when the active constraint set changes, then the control structure design is different as shown in Fig.2 and Fig.3. This may require redesign and re-tuning of the PI controllers. The economic NMPC can however easily handle changes in the active constraint set.

Rashid (2010) noted that, in practice, the wells are often considered independently neglecting the back-pressure effects imposed by interconnected wells. Optimization based on marginal GOR from individual gas lift performance curves may only lead to pseudo-steady-state solutions. The proposed method can include the interaction terms as well, thereby overcoming this limitation.

In terms of plant model mismatch, since the model used in the proposed method and the economic NMPC are the same, both the methods are equally affected by the plant-model mismatch. Model-free slow optimizing controllers such as extremum seeking control or NCO-tracking control

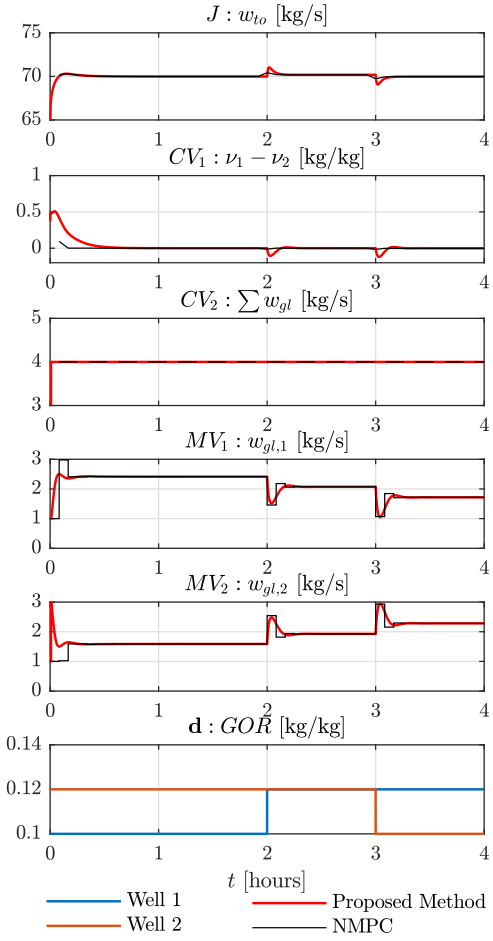


Fig. 6. Simulation results when the total available gas for injection is limited to $w_{gl}^{max} = 4kg/s$. The proposed method is shown in red lines and the economic NMPC is shown in thin black lines.

can be employed on top of the proposed method to account for any plant-model mismatch (Jäschke and Skogestad, 2011; Straus et al., 2017). In the simulation case study shown here, the same model structure was used to estimate the marginal GOR and in the plant simulator. A more realistic case would be to test with the plant modelled in advanced multiphase simulators such as OLGA which is an ongoing work.

REFERENCES

Alstad, V. (2005). *Studies on selection of controlled variables*. Ph.D. thesis, Norwegian University of Science and Technology (NTNU).

Bieker, H.P., Slupphaug, O., Johansen, T.A., et al. (2007). Real-time production optimization of oil and gas production systems: A technology survey. *SPE Production & Operations*, 22(04), 382–391.

Chachuat, B., Srinivasan, B., and Bonvin, D. (2009). Adaptation strategies for real-time optimization. *Computers & Chemical Engineering*, 33(10), 1557–1567.

Codas, A., Jahanshahi, E., and Foss, B. (2016). A two-layer structure for stabilization and optimization of an oil gathering network. *IFAC-PapersOnLine*, 49(7), 931–936.

Downs, J.J. and Skogestad, S. (2011). An industrial and academic perspective on plantwide control. *Annual Reviews in Control*, 35(1), 99–110.

Jäschke, J. and Skogestad, S. (2011). NCO tracking and self-optimizing control in the context of real-time optimization. *Journal of Process Control*, 21(10), 1407–1416.

Kanu, E.P., Mach, J., Brown, K.E., et al. (1981). Economic approach to oil production and gas allocation in continuous gas lift (includes associated papers 10858 and 10865). *Journal of Petroleum Technology*, 33(10), 1–887.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2017). Gas lift optimization under uncertainty. *Computer Aided Chemical Engineering*, 40, 1753–1758.

Krishnamoorthy, D., Jahanshahi, E., and Skogestad, S. (2018). A feedback rto strategy using transient measurements. *Journal of Process Control*, In preparation.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2016a). Real time optimization under uncertainty - applied to gas lifted wells. *Processes*, 4(4). doi:10.3390/pr4040052.

Krishnamoorthy, D., Pavlov, A., and Li, Q. (2016b). Robust extremum seeking control with application to gas lifted oil wells. *IFAC-PapersOnLine*, 49(13), 205–210.

Krstić, M. and Wang, H.H. (2000). Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36(4), 595–601.

Peixoto, A.J., Pereira-Dias, D., Xaud, A.F., and Secchi, A.R. (2015). Modelling and extremum seeking control of gas lifted oil wells. *IFAC-PapersOnLine*, 48(6), 21–26.

Rashid, K. (2010). Optimal allocation procedure for gas-lift optimization. *Industrial & Engineering Chemistry Research*, 49(5), 2286–2294.

Sharma, R. and Glemmestad, B. (2013). On generalized reduced gradient method with multi-start and self-optimizing control structure for gas lift allocation optimization. *Journal of Process Control*, 23(8), 1129–1140.

Simon, D. (2006). *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons.

Skogestad, S. (2000). Plantwide control: the search for the self-optimizing control structure. *Journal of process control*, 10(5), 487–507.

Skogestad, S. (2003). Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control*, 13(4), 291–309.

Skogestad, S. and Postlethwaite, I. (2007). *Multivariable feedback control: analysis and design*. Wiley New York, 2 edition.

Straus, J., Krishnamoorthy, D., and Skogestad, S. (2017). Combining self-optimizing control and extremum seeking control- applied to ammonia reactor.

Urbanczyk, C.H., Wattenbarger, R.A., et al. (1994). Optimization of well rates under gas coning conditions. *SPE Advanced Technology Series*, 2(02), 61–68.

Anton Friedl, Jiří J. Klemeš, Stefan Radl, Petar S. Varbanov, Thomas Wallek (Eds.)
Proceedings of the 28th European Symposium on Computer Aided Process Engineering
June 10th to 13th, 2018, Graz, Austria.

Control of the Steady-State Gradient of an Ammonia Reactor using Transient Measurements

Harro Bonnowitz[a], Julian Straus[b], Dinesh Krishnamoorthy[b], Esmaeil Jahanshahi[b] and Sigurd Skogestad[b*]

[a]*Department of Chemistry, Technische Universität Berlin, Berlin, Germany*
[b]*Department of Chemical Engineering, Norwegian University of Science and Technology, Trondheim, Norway*
sigurd.skogestad@ntnu.no

Abstract

This paper presents the application of a steady-state real-time optimization strategy using transient measurements to an ammonia synthesis reactor case. We apply a new method for estimating the steady-state gradient of the cost function based on linearizing a dynamic model at the present operating point. The gradient is controlled to zero using a standard feedback controller, for example, a PI-controller. The applied method is able to adjust fast to the new optimal operation in case of disturbances. The advantage compared to standard steady-state real-time optimization is that it reaches the optimum much faster and without the need to wait for steady-state to update the model. It is significantly faster than classical extremum-seeking control and does not require the measurement of the cost function and additional process excitation. Compared to self-optimizing control, it allows the process to achieve the true optimum.

Keywords: Optimal Control, Extremum-Seeking Control, Reactor Control, Measurement Based Optimization

1. Introduction

The general aim of a process plant is to operate at the economic optimum. Different approaches are available in the literature for driving a process to its optimal operation point. The traditional approach is steady-state real-time optimization (RTO) in which a rigorous steady-state model is used for computing optimal setpoints. The necessary model reconciliation requires however that the plant is at steady-state before each reoptimization. This is a fundamental limitation of RTO as it may lead to subopimal operation most of the time (Darby et al., 2011).

Self-optimizing control (SOC) (Skogestad, 2000) alleviates this problem through keeping the operation close to the optimum at all times by controlling selected controlled variable at a constant setpoint. Therefore, it can be used for close to optimal operation while waiting for the steady-state. The implementation is very fast and simple, but in case of unknown or large disturbances, the setpoints need to be updated using some other approach.

An alternative to RTO is a data-based approach, e.g. extremum-seeking control (ESC), which uses the plant measurements to drive the process to its optimal operation (Krstić and Wang, 2000). This is achieved by estimating the steady-state gradient from the input to the cost and using a small I-controller to drive the gradient to zero. Closely related approaches are the "hill-climbing" controller of Shinskey used recently by Kumar and Kaistha (2014) and the NCO-tracking approach

of Bonvin and coauthors (Franois et al., 2005). Their main advantage is that they are model free. The main challenge in these methods is the accurate estimation of the steady-state gradient from dynamic measurements. This normally requires constant excitations that are slow enough such that the dynamic system can be approximated as a static map (Krstić and Wang, 2000). As a result the convergence to the optimum is usually very slow. In the presence of abrupt disturbances, extremum-seeking control also causes unwanted deviations as discussed by Kumar and Kaistha (2014) andKrishnamoorthy et al. (2016).

A newer method for optimal operation is economic nonlinear model predictive control (E-NMPC), which handles the dynamic process behaviour, operational constraints, and leads to the optimal inputs for multivariable processes. Nevertheless, solving the optimization problem for a large-scale problem is computationally intensive and can potentially lead to computational delay.

In this paper, a new model-based dynamic gradient estimation (Krishnamoorthy et al., 2018, in preparation) is applied to drive the process to optimal operation. In contrast to standard ESC, the exact steady-state gradients is estimated based on the dynamic model of the process and hence no excitations are required. For the proposed method there is no need to measure the cost directly. Moreover, reoptimization is done by feedback control and solving the optimization problem is not necessary.

Heat-integrated processes, like the ammonia synthesis reactor (Morud and Skogestad, 1998) considered in this paper, are widespread in case of exothermic reactions to utilize the reaction heat. However, limit-cycle behaviour and reaction extinction may occur in the case of disturbances due to the positive feedback imposed through the heat integration (Morud and Skogestad, 1998). Straus and Skogestad (2017) proposed the application of E-NMPC for optimal control of the ammonia reactor to avoid this behaviour. In this paper, the application of the new feedback method is suggested to drive this process to optimal operation without the need of nonlinear dynamic optimization as is the case with E-NMPC or dynamic RTO.

2. Steady-state gradient control using transient measurements

We consider a process that can be modelled as a nonlinear dynamic system of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{d}) \tag{1}$$

$$\mathbf{y} = \mathbf{h}_{\mathbf{y}}(\mathbf{x}, \mathbf{u}) \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$, $\mathbf{u} \in \mathbb{R}^{n_u}$, $\mathbf{d} \in \mathbb{R}^{n_d}$, and $\mathbf{y} \in \mathbb{R}^{n_y}$ are the states, available control inputs, disturbances, and measurements. The cost does not need to be directly measured. In the proposed method, a state estimator such as an extended Kalman filter (EKF) (Simon, 2006) is applied to estimate the states $\mathbf{x}$ of the system by using the measurements and the dynamic model, given in Eqs. (1) and (2).

Let the cost be modelled as $J = h_J(\mathbf{x}, \mathbf{u})$ with $h_J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$. A local linear state-space model, given by the Eqs. (3) and (4), can be obtained through linearization around the current operation point, as shown by Krishnamoorthy et al. (2018, in preparation).

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{3}$$

$$J = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{4}$$

where $\mathbf{A} = \partial \mathbf{f}/\partial \mathbf{x}$, $\mathbf{B} = \partial \mathbf{f}/\partial \mathbf{u}$, $\mathbf{C} = \partial h_J/\partial \mathbf{x}$, and $\mathbf{D} = \partial h_J/\partial \mathbf{u}$. In order to derive the steady state gradient, we set $\dot{\mathbf{x}} = 0$ and can derive in deviation variables

$$\Delta J = \left(-\mathbf{C}\mathbf{A}^{-1}\mathbf{B} + \mathbf{D}\right)\Delta \mathbf{u} \tag{5}$$

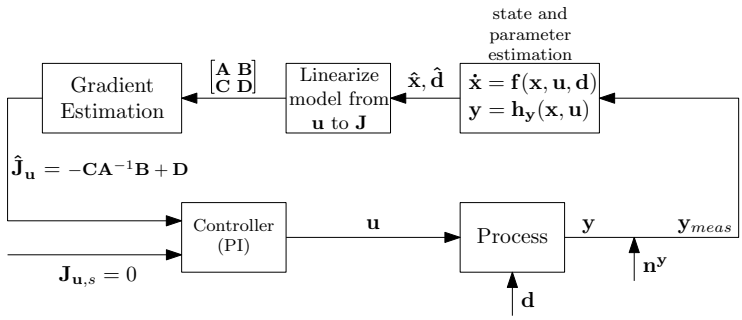


Figure 1: Block diagram of the proposed method.

which, since $\Delta J = \mathbf{J}_{\mathbf{u}}\Delta\mathbf{u}$, gives the following estimate or prediction of the steady-state gradient:

$$\hat{\mathbf{J}}_{\mathbf{u}} = -\mathbf{C}\mathbf{A}^{-1}\mathbf{B} + \mathbf{D} \tag{6}$$

We want to drive the system to an optimal steady-state where $\mathbf{J}_{\mathbf{u}} = 0$, so even if the system is not at steady-state, we can use feedback control with $\mathbf{y} = \hat{\mathbf{J}}_{\mathbf{u}}$ as "measurements" to drive the system to the optimal steady-state and by that satisfying the necessary conditions of optimality (Krishnamoorthy et al., 2018, in preparation). Any feedback controller, such as a PI controller, can be used to bring the gradient to zero. It is important to note that by using a nonlinear state estimator and a dynamic model for estimating the steady-state gradient $\mathbf{J}_{\mathbf{u}}$, we can use transient measurements, without the need to wait for steady-state, as in traditional RTO. The scheme of the proposed method is shown in Figure 1. The disturbances can be estimated as well through the extension of Eq. (1) to an augmented system (Simon, 2006).

3. Model and problem formulation

The model of the ammonia reactor and all the model assumptions are based on Morud and Skogestad (1998)'s stability analysis. The process, shown in Figure 2, consists of 3 sequential reactor beds and the feed is split into 4 streams. The model is a differential algebraic system, where the differential equations describe the temperature evolution in the reactor beds and the algebraic equations represent the corresponding mass fraction of ammonia. There are 3 split-ratios or $\mathbf{u}_0 = \begin{bmatrix} u_{0,1} & u_{0,2} & u_{0,3} \end{bmatrix}^{\mathsf{T}}$ which are controlled by local temperature controllers. This is necessary for stabilizing the process. The temperature controllers are incorporated into the model in continous time increasing the number of states by 3. This leads to $\mathbf{u} = \begin{bmatrix} T_{In,1}^{sp} & T_{In,2}^{sp} & T_{In,3}^{sp} \end{bmatrix}^{\mathsf{T}}$. The temperature controllers are modelled as single-input single-output integrator controllers, as the response can be approximated as a proportional process. The SIMC rules (Skogestad and Grimholt, 2012) were applied for the slave controllers tuning.

The state estimation is performed using an EKF (Simon, 2006). To this end, the model was reformulated as a system of ordinary differential equations. Each reactor bed in the model consists of n discrete volumes, which can be modelled as a CSTR cascade. This leads for each reactor bed to a total of $2n$ state variables per time step. For any CSTR reactor j in the CSTR cascade, the differential equations for the ammonia weight fractions $w_{\mathrm{NH}_3,j}$ can be formulated as seen in Eq. (7), in which $\alpha = 0.33$ represents the bed void fraction, $\rho_g = 50$ kg/m^3 the density of the gas,

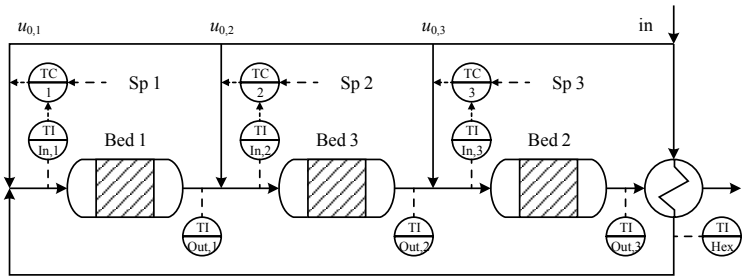


Figure 2: Heat-integrated 3 bed ammonia synthesis reactor with cascade control. The setpoint of the slave temperature loop is given by the proposed method.

and $V_j = V_{bed}/n$ the volume of each CSTR reactor j (Morud, 1995).

$$\frac{\mathrm{d}w_{\mathrm{NH}_3,j}}{\mathrm{d}t} = \frac{\dot{m}_{j-1} w_{\mathrm{NH}_3,j-1} - \dot{m}_j w_{\mathrm{NH}_3,j} + m_{cat,j} r_{NH_3,j}}{V_j \rho_g \alpha} \tag{7}$$

To summarize, we can write, $\mathbf{x} \in \mathbb{R}^{6n+3}$, $\mathbf{u} \in \mathbb{R}^3$ in the system, given in Eqs. (1) and (2).

In contrast to Straus and Skogestad (2017), full state knowledge is not assumed in this paper. The measurement set for state estimation is given by the inlet and outlet temperature of each reactor as well as the outlet temperature of the heat exchanger (see Figure 2). In real plants the catalyst activity is changing over time, which is difficult or impossible to measure and leads to a plant-model mismatch. To take into account industrial applicability, we assume, that the catalyst activity is not measured, but included in the model as an uncertain parameter. Hence, the states and the uncertain parameter are combined to the augmented states with $d = [a_{cat}]$, what results in an augmented system. To optimize the operation, we want to maximize the (mass) extent of reaction.

$$\xi = \dot{m}_{in}\left(w_{\mathrm{NH}_3,3n} - w_{\mathrm{NH}_3,in}\right) \tag{8}$$

This results in a cost function $J = -\xi$. In this case, a cascade control is used, where the master controllers drive the three gradients to zero by giving new set points to three slave control loops. The EKF and the proposed method were implemented in discrete time. The controller of the proposed method are single-input single-output controllers. The continuous time process model, given in Eq. (1), was modelled using CasADi (Andersson, 2013) and integrated with CVODES, which is part of the SUNDIALS package (Hindmarsh et al., 2005)

4. Results

In the following section, we consider a disturbance in the feed flow and a plant-model mismatch, given by a mismatch in the catalyst activity. In all cases, we have three inner stabilizing temperature loops as indicated by the letter "T" on the plots. In addition, the results are compared to pure self-optimizing control (SOC) and extremum-seeking control (ESC) in the optimization layer. The integrated cost difference (loss) J_{int} is used for comparison of the different methods,

$$J_{int}(t) = \int_0^t \left[\xi_{opt,SS}(t') - \xi(t')\right] \mathrm{d}t' \tag{9}$$

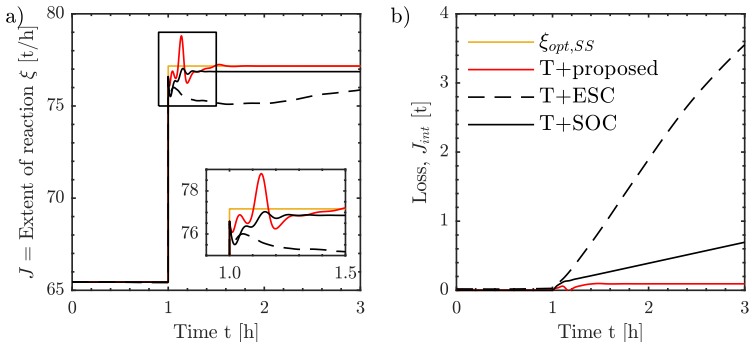


Figure 3: Responses of the alternative methods in a) the extent of reaction and b) the integrated loss to a disturbance in the feed flowrate of $\Delta\dot{m}_{in} = -15$ kg/s at time $t = 1$ h. $\xi_{opt,SS}$ represents the steady-state optimal extent of reaction.

First we simulate a disturbance change in the inlet flowrate $\dot{m}_{in}$ to evaluate the performance of the control structure. The results for a decrease in the feed flowrate of $\Delta\dot{m}_{in} = -15$ kg/s at time $t = 1$ h are presented in Figure 3. The new proposed method gives fast disturbance rejection and settles down at the new optimal operation after about 30 min, as seen Figure 3 a). SOC is equally fast, but it does not quite reach the new optimum. This leads to a continous increase in the integrated loss for SOC as seen in Figure 3 b). If we compare the proposed method to ESC as optimizing control, the proposed method is much faster and therefore causes a lower integrated cost difference of $J_{int}(t_{end}) = 0.1$ t. This is because the data-based gradient estimation takes longer time for accurate gradient estimation and the controller gain has to be small to satisfy stability. The application of extremum-seeking controllers does not converge to the steady-state optimum in the investigate time frame and requires 13 h.

In the second simulation, we consider plant-model mismatch. The results for a decreased catalyst activity Δa_{cat} by 20 % at time $t = 1$ h, which normally occurs slowly over a longer period of time, are presented in Figure 4. Therefore, the activity of the catalyst, or more specifically the

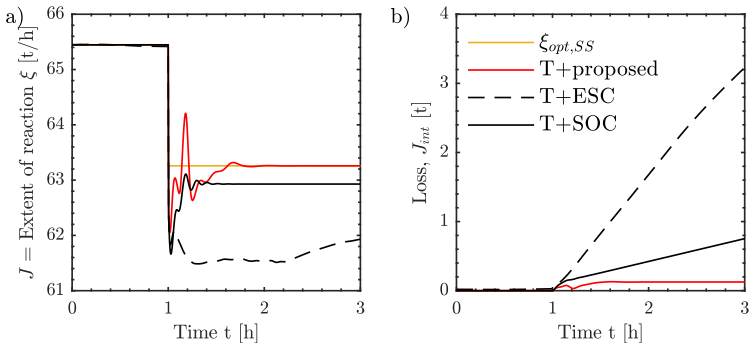


Figure 4: Responses of the alternative methods in a) the extent of reaction and b) the integrated loss to a plant-model mismatch of $\Delta a_{cat} = -20$ % at time $t = 1$ h. $\xi_{opt,SS}$ represents the steady-state optimal extent of reaction.

pre-exponential factor of the Arrhenius equation, spontaneously changes between the model used for the simulations and the model for the state estimation. The simulation shows that the proposed method is performing well even in the presence of a plant-model mismatch. This is because we are able to estimate the real value of the catalyst activity using the augmented EKF framework. About 1.5 minutes after the activity change, the mismatch as well as the states are estimated correctly. The proposed method is much faster than T+ESC, which in turn results in a lower total loss as seen in Figure 4 b). Again, SOC is equally fast, but the real optimum is not reached. The proposed method causes a total integrated cost difference of about $J_{int}\left(t_{end}\right) = 0.12$ t of ammonia for the considered case with plant-model mismatch.

5. Conclusion

We have applied a new method of utilizing transient measurements and a dynamic estimator to estimate the steady-state gradient and then using a simple PI controller for driving the process to its optimal operation. For an ammonia synthesis reactor with both disturbances and plant-model mismatch, the proposed method outperforms comparable control strategies. The industrial applicability is conceivable due to the usage of only seven measurements of the process besides the used dynamic model. An extended Kalman filter (EKF) allows the estimation of the steady-state gradients, even in case of plant-model mismatch by including unmeasured but modelled parameters in the estimator.

References

J. Andersson, October 2013. A General-Purpose Software Framework for Dynamic Optimization. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium.

M. L. Darby, M. Nikolaou, J. Jones, D. Nicholson, 2011. RTO: An overview and assessment of current practice. Journal of Process Control 21 (6), 874 – 884.

G. Franois, B. Srinivasan, D. Bonvin, 2005. Use of measurements for enforcing the necessary conditions of optimality in the presence of constraints and uncertainty. Journal of Process Control 15 (6), 701 – 712.

A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, C. S. Woodward, 2005. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Transactions on Mathematical Software (TOMS) 31 (3), 363–396.

D. Krishnamoorthy, E. Jahanshashi, S. Skogestad, 2018, in preparation. A feedback RTO strategy using transient measurements.

D. Krishnamoorthy, A. Pavlov, Q. Li, 2016. Robust extremum seeking control with application to gas lifted oil wells. IFAC-PapersOnLine 49 (13), 205–210.

M. Krstić, H.-H. Wang, 2000. Stability of extremum seeking feedback for general nonlinear dynamic systems. Automatica 36 (4), 595–601.

V. Kumar, N. Kaistha, 2014. Hill-climbing for plantwide control to economic optimum. Industrial & Engineering Chemistry Research 53 (42), 16465–16475.

J. Morud, Dec. 1995. Studies on the Dynamics and Operation of integrated Plants. PhD thesis, The Norwegian Institute of Technology, Department of Chemical Engineering, NTNU, Trondheim.

J. C. Morud, S. Skogestad, 1998. Analysis of instability in an industrial ammonia reactor. AIChE Journal 44 (4), 888–895.

D. Simon, 2006. Optimal state estimation: Kalman, H infinity, and nonlinear approaches. John Wiley & Sons.

S. Skogestad, 2000. Plantwide control: the search for the self-optimizing control structure. Journal of Process Control 10 (5), 487 – 507.

S. Skogestad, C. Grimholt, 2012. The SIMC Method for Smooth PID Controller Tuning. Springer London, London, pp. 147–175.

J. Straus, S. Skogestad, June 2017. Economic NMPC for heat-integrated chemical reactors. In: 2017 21st International Conference on Process Control (PC). pp. 309–314.

A feedback real-time optimization strategy applied to an evaporator process

Dinesh Krishnamoorthy[a], **Esmaeil Jahanshahi**[b], **and Sigurd Skogestad**[c,*]

Department of Chemical Engineering, Norwegian University of Science and Technology (NTNU), Trondheim, Norway
E-mail: [a]dinesh.krishnamoorthy@ntnu.no, [b]esmaeil.jahanshahi@hotmail.com , [c]skoge@ntnu.no (Corresponding author)

Abstract. This paper presents the application of a feedback based real time optimization (RTO) strategy using transient measurements for optimal operation of an evaporator process. The proposed method is based on estimating the steady-state gradient of the cost function by linearizing the nonlinear dynamic model around the current operating point. Any feedback controller can then be used to drive the estimated steady-state gradient to zero to achieve optimal operation. Since the proposed method uses transient measurements, it avoids the steady-state wait time which is a limitation with traditional steady-state RTO. Since the optimization is achieved via feedback, it does not require to solve numerical optimization problems, as in conventional steady-state RTO or dynamic RTO. Compared to self-optimizing control, the proposed method does not have any steady-state losses when operated far away from the nominal optimal point. Compared to model-free methods such as extremum seeking control, it is significantly faster and does not require external process excitation for steady-state gradient estimation. The performance of the proposed method for the evaporator process is compared with traditional static RTO, dynamic RTO, hybrid RTO, self-optimizing control and extremum seeing control.
Keywords: Real-time optimization, Optimal Control, Measurement Based Optimization

1. Introduction

Real-time optimization of chemical processes traditionally involves solving a steady-state optimization problem using rigorous steady-models of the process. Following any changes in the disturbance or the operating conditions, the optimization problem must be solved to re-compute the new optimal points. However, before the new optimal points can be recomputed, it is necessary to wait for the process to settle to a steady-state operating point. This steady-ate wait time is a fundamental limitation of the traditional steady-state optimization of chemical processes [1]. Dynamic real-time optimization does not suffer from the steady-state wait time. However, the computation cost is prohibitively expensive in many applications even with today's computing power.

In order to address the steady-state wait time issue of the traditional static RTO and the computation cost of dynamic RTO, we recently proposed a hybrid RTO strategy, where the model adaptation is dine using dynamic models and the numerical optimization is solved using the corresponding steady-state model. The hybrid RTO approach thus requires the maintenance of both the static and the dynamic models.

Recently, there has been an increasing interest in the so-called *direct input adaptation* methods, where optimal operation is achieved by means of feedback control. Self-optimizing control, extremum seeking control, NCO-tracking etc. are some well known approaches that belong to such a category. In self-optimizing control we control a combination of measurements to a constant setpoint such that the economic losses are minimized. However when the process is operated far from the nominal optimal region, this leads to large steady-state losses [2].

Model-free approaches such as extremum seeking control and NCO-tracking on the other hand are based on estimating the steady-state gradient directly from the measurements and controlling them to a constant

setpoint of zero. However, the main disadvantage of these methods is that it has a very slow convergence to the optimum. In addition, these methods also require additional perturbation for accurate gradient estimation [3]. Extremum seeking like approaches are also known to provide unwanted deviations in the presence of abrupt disturbances as motivated by [4].

In this paper, we apply a recently developed feedback-based RTO approach for the evaporator process, which is based on converting the hybrid RTO into a feedback control problem. The proposed method is based on estimating the steady-state gradient of the cost function by linearizing the nonlinear dynamic model around the current operating point. The estimated steady-state gradient can then be controlled to a constant setpoint of zero. Since the proposed method uses nonlinear dynamic models, it can use transient measurements and hence avoid the issue of steady-state wait time. It also does not require any additional dither to estimate the steady-state gradient.

2. Feedback RTO using steady-state gradient control

Consider a nonlinear dynamic system of the form,

$$\begin{aligned} \dot{x} &= f(x,u,d) \\ y &= g(x,u) \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, $d \in \mathbb{R}^{n_d}$ and $y \in \mathbb{R}^{n_y}$ are the states, inputs, disturbances and the measured outputs respectively. Note that the n_u control inputs (manipulated variables) considered here are the unconstrained degrees of freedom. In the proposed feedback RTO method, any state estimator such as an extended Kalman filter (EKF) [5] can be applied to estimate the states x of the system by using the measurements and the nonlinear dynamic model (1).

Let the cost be modelled as,

$$J = h(x,u) \tag{2}$$

with $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$. Note that the cost does not need to be measured in the proposed method. Using the updated states, the nonlinear model from the inputs to the cost can be linearized around the current operating point to get a local linear state-space model, given by (3) as described in [6].

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{3}$$

where the system matrices A, B, C and D are the Jacobians of the non-linear functions f, and h from (1) and (3), evaluated around the current operating point,

$$A = \left.\frac{\partial f}{\partial x}\right|_{x=\hat{x}} \quad B = \left.\frac{\partial f}{\partial u}\right|_{x=\hat{x}} \quad C = \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}} \quad D = \left.\frac{\partial h}{\partial u}\right|_{x=\hat{x}} \tag{4}$$

The steady state gradient can then be obtained by setting $\dot{x} = 0$ to get,

$$\Delta J = \left(CA^{-1}B + D\right)\Delta u \tag{5}$$

The estimate of the steady-state gradient around the current operating point is then given by,

$$\hat{J}_u = CA^{-1}B + D \tag{6}$$

since $\Delta J = J_u \Delta u$.

To optimize the operation of the process, the estimated steady-state gradient is driven to a setpoint of $\hat{J}_u = 0$ by using any feedback controller thus satisfying the necessary conditions of optimality [6].

It is important to note that by using a nonlinear state estimator and a dynamic model for estimating the steady-state gradient $\hat{J}_u$, we can use transient measurements, without the need to wait for steady-state, as in traditional RTO. The proposed method is schematically shown in Fig. 1.

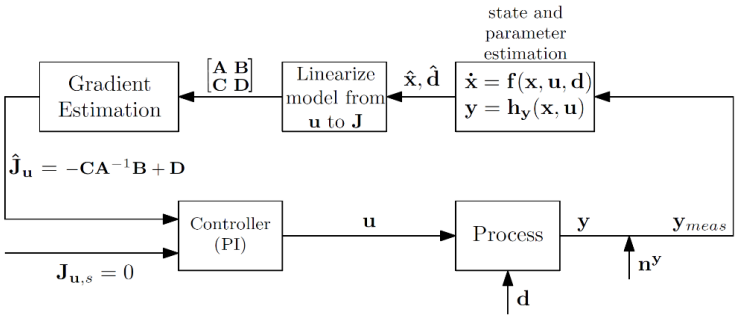


Fig. 1. Block diagram of the Feedback RTO approach

3. Optimal operation of an evaporator process

We now apply the proposed feedback RTO method on the evaporator process shown in Fig. 2 and compare it with the traditional static RTO (SRTO), dynamic RTO (DRTO) and the recently developed hybrid RTO (HRTO). In addition, the proposed method is compared with two other direct-input adaptation methods, namely, self-optimizing control (SOC) and extremum seeing control (ESC). The purpose of the evaporator process is to increase the concentration of the dilute liquor by evaporating the feed solvent F_1 while the liquor is circulated through the heat exchanger. The model is the same as the one used in [7]and for more detailed information on the model equations and the model parameters, the reader is referred to [7].

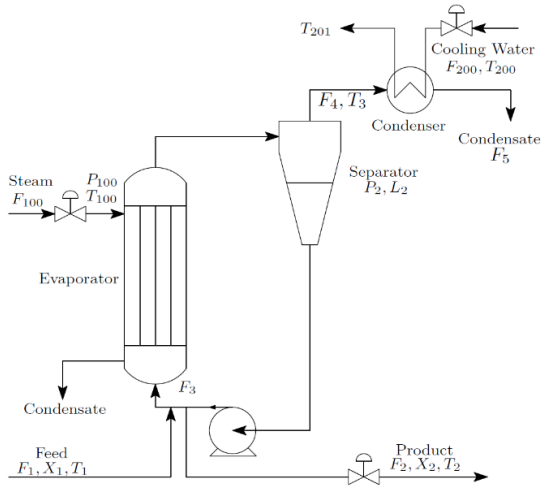


Fig. 2. Evaporator process

The objective is to minimize the operation cost, and the optimization problem is formulated as,

$$\begin{aligned} \min_{u} \; J &= 600F_{100} + 0.6F_{200} + 1.009\left(F_2 + F_3\right) \\ \text{s.t.} \quad \dot{x} &= f(x,u,d) \\ X_2 &\geq 35.5\% \\ P_{100} &\leq 400\text{kPa} \end{aligned} \tag{7}$$

There are four dynamic degrees of freedom, namely, F_2, P_{200}, F_3 and F_{200}.At the optimal point, both the MV inequality constraints are active, namely $X_2 = 35.5\%$ and $P_{100} = 400\text{kPa}$. Therefore, these two active constraints are tightly regulated using PI controllers. The input F_2 is used to control X_2 at a constant

setpoint of $X_2 = 35.5\%$, and P_{100} which is an input itself, is maintained at a constant value of $P_{100} = 400\text{kPa}$. In addition, the separator level which does not have any steady-state effect is controlled by F_3. Therefore, F_{200} is the only remaining unconstrained degree of freedom that can be used for optimization. In other words, we only have one steady-state degree of freedom, i.e. $u = F_{200}$. By using F_{200} for control, we optimize P_2, thus optimizing the evaporator process. We assume there are four unmeasured disturbances $d = \begin{bmatrix} F_1 & X_1 & T_1 & T_{200} \end{bmatrix}$ and the available measurements are $y = \begin{bmatrix} F_2 & F_{100} & T_{200} & F_3 \end{bmatrix}$.

As mentioned earlier, the proposed method uses a state estimator. In this work, we use an extended Kalman filter (EKF) for combined state and disturbance estimation by augmenting the unmeasured disturbances with the states, e.g. see [5]. The steady state gradient of the cost is then estimated according to (6). In this work, we use a PI controller to control the estimated gradient to a constant setpoint of zero. The process is simulated for a total simulation time of 10h with variations in the unmeasured disturbances as shown in Fig.3 [7]. The measurements are assumed to be available with a sampling rate of 1s.

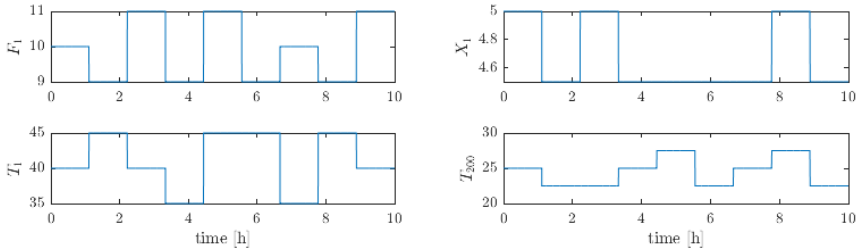


Fig. 3. Disturbance trajectories affecting the evaporator process.

3.1. Comparison with optimization based approaches

First, we compare the proposed feedback RTO method with the traditional steady state RTO (SRTO), where the disturbances are estimated using a static model of the system. The system is subject to disturbances as shown in Fig. 3. In traditional SRTO, the parameter estimation uses only the steady-state operating points, and a steady-state detection algorithm as described in [8] was used. The steady-state wait time due to the steady-state detection is a fundamental limitation of the traditional SRTO and the plant is operated sub-optimally for significant periods before the model can be updated.

We also compare the newly proposed hybrid RTO (HRTO) approach and with dynamic RTO (DRTO) using the same extended Kalman filter as the one used in the proposed feedback RTO approach. As mentioned earlier, the hybrid RTO approach uses the *dynamic* model and the EKF to update the model using the transient measurements, but uses the corresponding updated *static* model to solve a *static* numerical optimization problem. The only difference between the hybrid RTO and the proposed feedback RTO is that the hybrid RTO solves a numerical optimization problem, whereas in feedback RTO, optimization is achieved via feedback control.

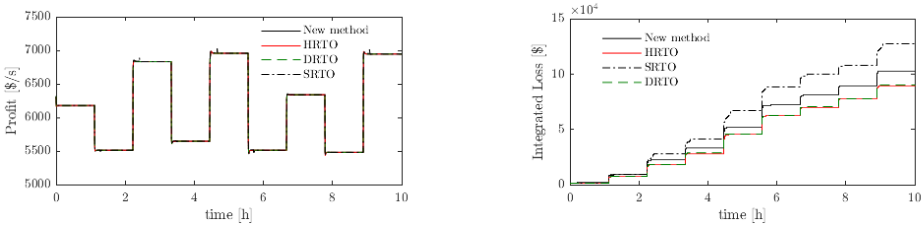
For the dynamic RTO, the updated dynamic model is directly used to solve a dynamic optimization problem with a prediction horizon of 30 min and a sampling time of 1s to compute the optimal input trajectory. As mentioned earlier, the main challenge with DRTO is the computation time.

The performance of the proposed feedback RTO (black solid lines) is compared with the traditional static RTO (black dash-dotted lines), dynamic RTO (green dashed lines) and hybrid RTO (red solid lines) in Fig.4 The cost function J is shown in Fig.4a along with the integrated loss in Fig.4b, which is given by the expression,

$$L_{int} = \int_0^t J_{opt,SS}(t) - J(t)\ \mathrm{d}t \tag{8}$$

3.2. Comparison with self-optimizing control

For comparison with self-optimizing control, null-space method was used to compute the optimal selection matrix. The resulting self-optimizing variable $c = 0.002F_2 - 0.0976F_{100} - 0.0081T_{201} + 0.0125F_3$ is controlled to a constant setpoint of $c_s = -0.9951$, as described in [7]. The simulations were performed with

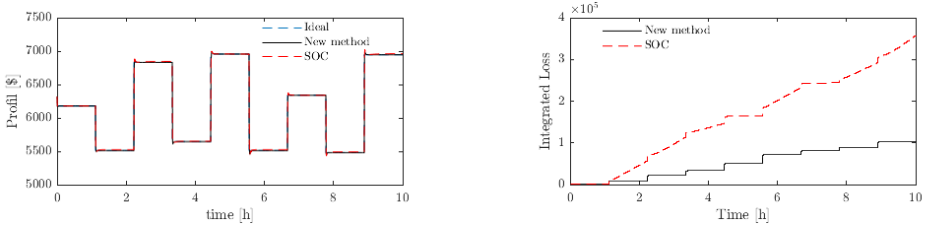


(a) (b)

Fig. 4. Comparison of the proposed method with traditional static RTO, dynamic RTO and hybrid RTO (a)Profit (b) Integrated Loss given by (8)

the same disturbances as in the previous case. The performance of the proposed feedback RTO (red solid lines) is compared with self-optimizing control (red dashed lines) in Fig.5, where the cost function is shown in Fig.5a and the integrated loss is shown in Fig.5b

It can be clearly seen that when the disturbances move the operating point of the system away from the nominal optimal point, self-optimizing control leads to steady-state losses. This is not the case with the proposed feedback RTO. This is because, in the proposed method, the nonlinear model is linearized around the current operating point as opposed to linearizing around a nominal optimal point in self-optimizing control.



(a) (b)

Fig. 5. Comparison of the proposed method with self-optimizing control (a) Profit (b) Integrated loss given by (8)

3.3. Comparison with extremum seeking control

In this subsection, we compare the performance of the proposed method with extremum seeking control, which is also based on estimating and controlling the steady-state gradients directly from measurements. In this section we use the least square based extremum seeking control as introduced in [9]. In this method, the steady-state gradient is estimated purely based on the measurements by constantly perturbing the system around the current operating point. The estimated steady state gradient is then driven to zero using integral action.

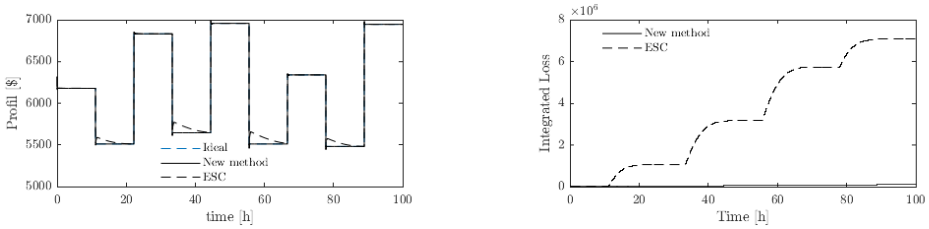
Since the steady-state gradient are estimated directly from the measurements, it requires time scale separation between the system dynamics, perturbation and the convergence to the optimum. Therefore, the convergence of the extremum seeking control is prohibitively slow in many cases.

Due to the slow convergence, the process is simulated with a total simulation time of 100h (10 times longer than the previous simulation cases). In this simulation, an integral gain of $K_{ESC} = 0.02$ was chosen. In this simulation, the disturbances vary over a period of 100h instead of 10h. The simulation results are shown in

Fig.6. From the simulation results, it can be seen that the proposed method converges significantly faster than the extremum seeking control.

4. Conclusion

We have proposed a new method of utilizing transient measurements and a dynamic estimator to estimate the steady-state gradient and then using a simple PI controller for driving the process to its optimal operation. For an ammonia synthesis reactor with both disturbances and plant-model mismatch, the proposed method outperforms comparable control strategies. The industrial applicability is conceivable due to the usage of only seven measurements of the process besides the used dynamic model. An extended Kalman filter (EKF) allows the estimation of the steady-state gradients, even in case of plant-model mismatch by including unmeasured but modelled parameters in the estimator.



(a) (b)

Fig. 6. Comparison of the proposed method with extremum seeking control (a) Profit (b) Integrated loss given by (8)

5. Acknowledgements

The authors acknowledge support from SFI SUBPRO which is financed by the research council of Norway, NTNU and major industry partners.

References

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

[2] S. Skogestad, "Self-optimizing control: The missing link between steady-state optimization and control," *Computers and Chemical engineering,* pp. 569-575, 2000.

[3] M. Krstic and H. Wang, "Stability of Extremum Seeking feedback for general nonlinear dynamic systems," *Automatica,* vol. 36, pp. 595-601, 2000.

[4] D. Krishnamoorthy, A. Pavlov and Q. Li, "Robust extremum seeking control with application to gas lifted oil wells," *IFAC-papersOnline,* pp. 205-210, 2016.

[5] D.Simon, Optimal state estimation: Kalman, H infinity, and nonlinear approaches., John Wiley & Sons, 2006.

[6] D. Krishnamoorthy, E. Jahanshashi and S. Skogestad, "Feedback Real time optimization approach using transient measurements," *Ind. Eng. Chem. Res,* (submitted).

[7] L. Ye, Y. Cao, Y. Li and Z. Song, "Approximating Necessary Conditions of Optimality as Controlled Variables," *Industrial & Engineering Chemistry Research,* vol. 2, no. 52, pp. 798-808, 2012.

[8] M. Câmara, A. Quelhas and J. Pinto, "Performance Evaluation of Real Industrial RTO Systems," *Processes,* vol. 4, no. 4, p. 44, 2016.

[9] B. G. B. Hunnekens, M. A. M. Haring, N. van de Wouw and H. Nijmeijer, "A dither-free extremum-seeking control approach using 1st-order least-squares fits for gradient estimation," in *53rd IEEE Conference on Decision and Control (CDC)*, 2014.

Appendix D

Optimal operation of William-Otto reactor using simple control structures

In this Chapter 4, we proposed a generalized framework for selecting *"what to control"* in order to achieve optimal economic operation. An optimization problem can be converted into a feedback control problem by controlling

- $\mathbf{g}_{\mathbb{A}} \to 0$
- $\mathbf{c} = \mathbf{N}^{\mathsf{T}} \nabla_{\mathbf{u}} J \to 0$, where $\mathbf{N}$ is chosen such that $\mathbf{N}^{\mathsf{T}} \nabla_{\mathbf{u}} \mathbf{g}_{\mathbb{A}} = 0$

This will be demonstrated using a benchmark William-Otto reactor in this appendix.

D.1 Optimal CV selection

Consider the benchmark William-Otto reactor example, where the raw materials A and B are converted to useful products P and E through a series of reactions

$$A + B \to C \qquad k_1 = 1.6599 \times 10^6 e^{-6666.7/T_r} \tag{D.1}$$

$$B + C \to P + E \qquad k_2 = 7.2177 \times 10^8 e^{-8333.3/T_r} \tag{D.2}$$

$$C + P \to G \qquad k_3 = 2.6745 \times 10^{12} e^{-11111/T_r} \tag{D.3}$$

The feed stream F_A with pure A component is a disturbance to the process and the manipulated variables are the feed stream F_B with pure B component and the reactor temperature T_r as shown in Fig. D.1. The objective is to maximize the production of valuable products P and E, subject to some purity constraints on G and A in the product stream,

$$\min_{T_r, F_B} \; -1043.38 x_P (F_A + F_B) - 20.92 x_E (F_A + F_B) + 79.23 F_A + 118.34 F_B \tag{D.4}$$

$$\text{s.t.} \quad x_G \le 0.08, \qquad x_A \le 0.12$$

Since we have two constraints, we can have at most $2^2 = 4$ active constraint regions, namely, 1) x_A and x_G active, 2) only x_G active, 3) only x_A active, and

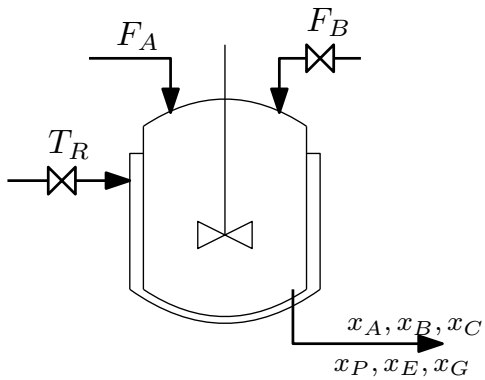
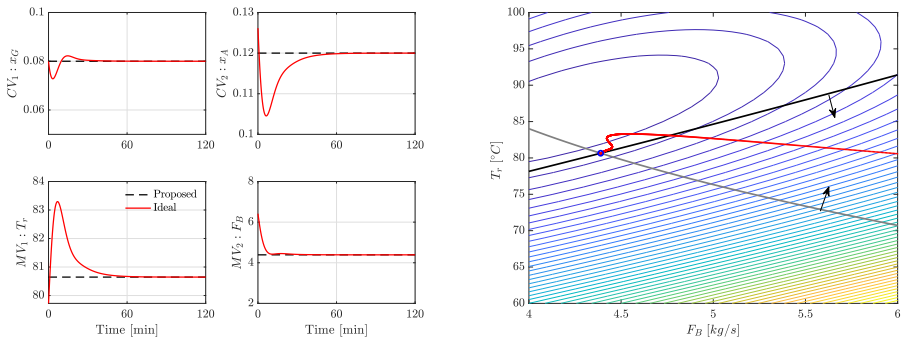


Figure D.1: Simulation results using the linear gradient combination as the self-optimizing variable

4) unconstrained. However, the max limit on x_G is so low that x_G will always be active. Therefore, we only need to choose CVs for regions 1 and 2 only. In region 1, we simply control the concentration of x_A to its limit of $0.12kg/kg$ and x_G to its limit of $0.08kg/kg$. In region 2, we control x_G to its limit of $0.08kg/kg$, and control the linear gradient combination $c := 0.9959\nabla_{F_B}J + 0.0906\nabla_{T_r}J$ to a constant setpoint of zero.

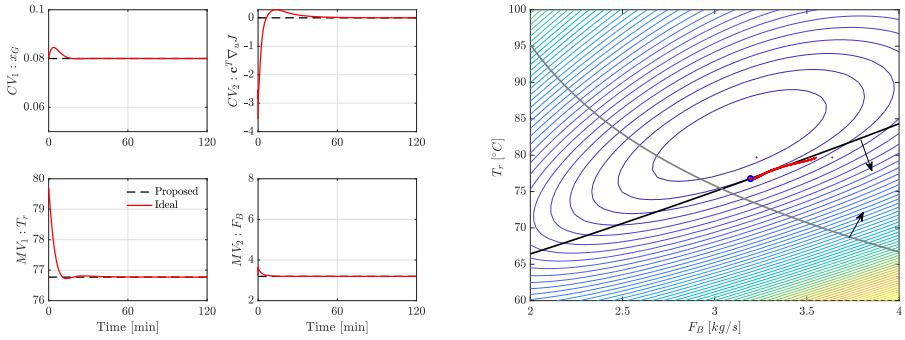
Region 1 ($F_A = 1.8275kg/s$) - When the disturbance is $F_A = 1.8275kg/s$, we are operating in region 1, with both the constraints active. This is the simplest case, where optimal operation is achieved using active constraint control.



(a) (b)

Figure D.2: Region 1: Simulation results using the proposed CVs when $F_A = 1.8275kg/s$.

Region 2 ($F_A = 1.3kg/s$) - When the disturbance is $F_A = 1.3kg/s$, we are operating in region 2, with only x_G constraint active. We use the reactor temperature T_r



(a) (b)

Figure D.3: Region 2: Simulation results using the proposed CVs when $F_A = 1.3kg/s$.

to control this constraint tightly and use F_B to control the linear gradient combination $c := 0.9959\nabla_{F_B} J + 0.0906\nabla_{T_r} J$. In this case, we use a model-based gradient estimation method proposed in Chapter 3. The simulation results are shown in Fig. D.3, where it can be seen that the proposed CVs are able to drive the process ot its true optimum.

Switching between x_A and c can automatically be achieved using a max selector block according to Theorem 4.2. Fig.D.4 shows the simulation results with varying disturbance and measurement noise, and the switching between the active constraint regions using a max selector.

In region 2, we now compare the performance of the proposed linear gradient combination with linear measurement combination as self-optimizing CV. Based on the optimal sensitivity around the nominal operating point of $F_A = 1.3kg/s$, x_B and x_C were the least sensitive to the disturbance. Hence we compare the loss when x_B, x_C and a linear combination of x_B and x_C is chosen as the self-optimizing variable. For the linear combination, we use the nullspace method by [2] which gives $c := 0.6305x_B + 0.7762x_C$, controlled to setpoint of 0.2656. The loss for the three candidate CVs, along with the proposed linear gradient combination is summarized in Table...

Table D.1: Steady-state loss for the candidate CV with respect to disturbances.

	$F_A = 1$	$F_A = 1.1$	$F_A = 1.2$	$F_A = 1.3$	$F_A = 1.4$
x_B	3.6892	2.0482	0.8355	0.0026	0.4832
x_C	3.8021	2.0994	0.8460	1.225e-5	0.4742
$h_1 x_B + h_2 x_C$	3.6858	2.0468	0.8325	9.715e-5	0.4861
$\eta_1 \nabla_{F_B} J + \eta_2 \nabla_{T_r} J$	0	0	0	0	0

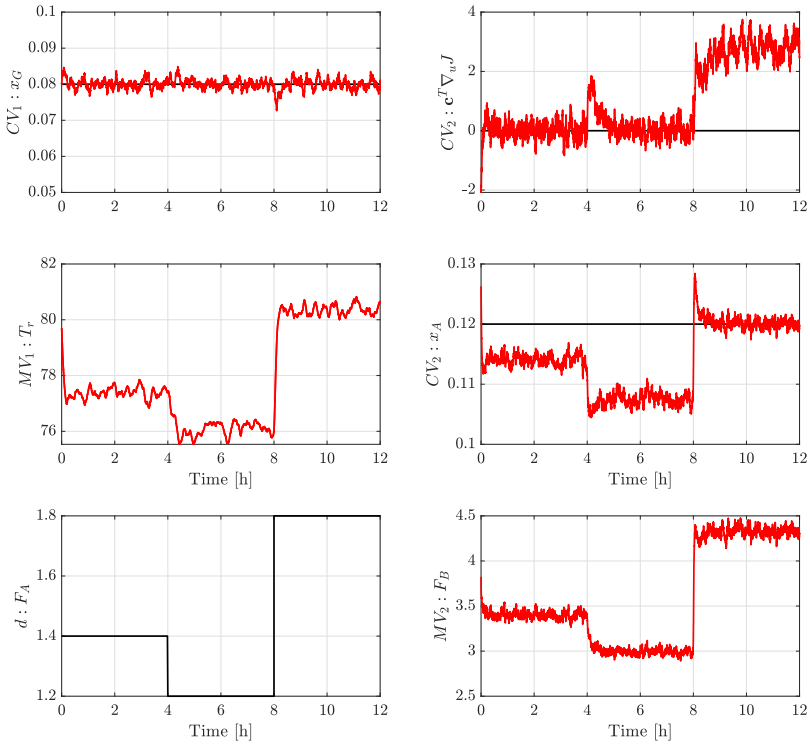


Figure D.4: Simulation results showing the switching between different active constraint regions.

D.2 Optimal operation of parallel operating units

In this section, we show how the proposed linear gradient combination framework can be used to choose the CVs for optimal operation of parallel operating units. Often in practice, when a plant capacity expands, this is done by simply adding new units in parallel to the existing units. The parallel units often share common resources such as feed, hot water etc. The different units may have different capacities, different equipment condition and different efficiencies.

Consider the optimal operation of p parallel units each with a cost function $\ell_i(u_i)$ and a given total feed U^{max}. The optimization problem is given as

$$\min_{u_i} \; J = \sum_{i=1}^{p} \ell_i(u_i) \quad \text{s.t.} \; \sum_{i=1}^{p} u_i - U^{max} = 0 \tag{D.5}$$

In this case, $\nabla_{\mathbf{u}}\mathbf{g}_{\mathbb{A}} = \mathbb{1}^p$ and $\mathbf{N} \in \mathbb{R}^{(p-1)\times p}$ such that

$$\sum_{j=1}^{p} \eta_{i,j} \nabla_{\mathbf{u}_j} J, \qquad \sum_{j=1}^{p} \eta_{i,j} = 0 \quad \forall i = 1, \ldots, p-1 \tag{D.6}$$

where $\eta_{i,j}$ is the i and j^{th} element in $\mathbf{N}$. This implies $\nabla_{\mathbf{u}_i} \ell_i = \nabla_{\mathbf{u}_j} \ell_j$ for all $i \neq j$. That is the optimal operation of parallel units occur when the marginal cost is the same for all the units, which was also proved by [37] and commonly used in practice.

To illustrate this, consider a process with $p = 3$ parallel units. Using the nullspace of $\nabla_{\mathbf{u}}\mathbf{g}_{\mathbb{A}} = [1, 1, 1]^\mathsf{T}$, we get

$$\begin{aligned} c_1 &: -0.5774 \nabla_{\mathbf{u}_1} J + 0.7887 \nabla_{\mathbf{u}_2} J - 0.2113 \nabla_{\mathbf{u}_3} J = 0 \\ c_2 &: -0.5774 \nabla_{\mathbf{u}_1} J - 0.2113 \nabla_{\mathbf{u}_3} J + 0.7887 \nabla_{\mathbf{u}_3} J = 0 \end{aligned}$$

Adding $c_1 + c_2$ yields $-\nabla_{\mathbf{u}_2} J + \nabla_{\mathbf{u}_3} J = 0$. Substituting this in c_1 gives $-0.5774 \nabla_{\mathbf{u}_1} J + 0.5774 \nabla_{\mathbf{u}_2} J = 0$, which results in $\nabla_{\mathbf{u}_1} J = \nabla_{\mathbf{u}_2} J = \nabla_{\mathbf{u}_3} J$.

Appendix E

Optimal operation of ESP lifted wells using simple PID controller and logics

In Chapter 4, we used gas lift as the artificial lift technology. In this appendix, we show the use of simple PID controllers with logics to achieve optimal operation of Electrical Submersible Pumps (ESP). [1] ESPs are especially common in heavy oil fields, where the reservoir oil is very viscous and does not flow naturally. ESPs are multistage centrifugal pumps that are placed several meters below inside the well tubing [179] as depicted in Fig. E.1.

The ESP lifted wells are operated by adjusting the rotational speed of the pump (denoted by ω) and the production choke (denoted by z_c), i.e. there are two manipulated variables for each well. Offshore oil fields may have several ESP lifted wells producing to a common manifold, such that the operation of one pump affects the operation of the other ESP lifted wells due to the coupling via the manifold pressure p_m. In addition, the fluid viscosity, reservoir inflow conditions and the available power may also change, which affects the optimal operation.

Traditionally, the operation of an ESP lifted well is decided by the operators together with an external ESP expert team (typically from the ESP vendor). However it may be challenging to operate several ESP lifted wells simultaneously, especially when the wells are highly coupled. It is important to operate the ESP lifted well properly in the presence of disturbances in order to avoid pump failures and to extend the pump life time, since ESP failures can be expensive both in terms of the replacement costs and lost production.

Automatic control of an ESP lifted well can contribute to safe and optimal operation [135]. The objective is to compute the optimal ESP speed and the production choke opening such that the :

- ESP intake pressure is maintained at desired setpoint
- ESP power consumption is minimized

[1] The experimental results presented in this appendix was carried out in 2013 before I started my PhD.

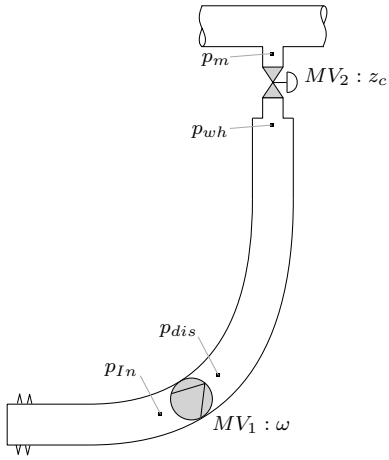


Figure E.1: Schematic representation of an ESP lifted well.

- ESP operation is maintained within a desired operating envelop.

The main objective in controlling an ESP lifted well is to maintain the ESP intake pressure at a desired setpoint, since this gives direct control over the production rate from the reservoir [90]. In addition, it is desirable to achieve this setpoint using minimal power consumption. For example, this can be achieved by reducing the ESP speed to its minimum limit and manipulate the production choke suitably to achieve the intake pressure setpoint. Reduced ESP frequency directly translates to reduced power consumption, since the power consumed increases cubically with increasing speed, as given by the pump affinity laws [179],

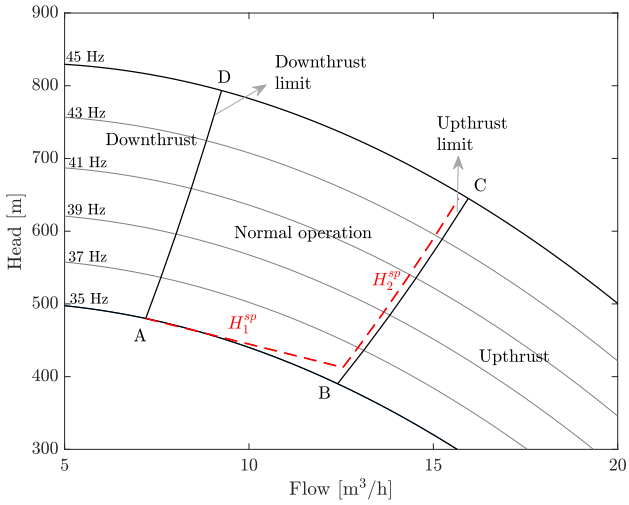
$$P = P_0 \left(\frac{\omega}{\omega_0} \right)^3 \tag{E.1}$$

where P_0 is the power consumed at some nominal ESP speed ω_0 and P is the power consumed at frequency ω.

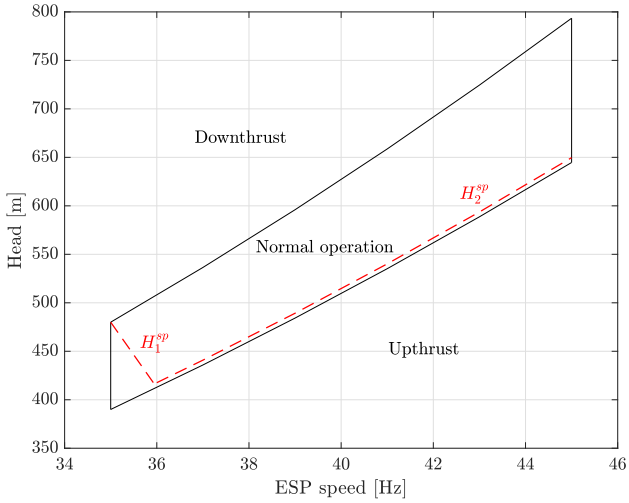
The pump must also be operated within a pre-defined operating envelop, which is constrained by the minimum and maximum ESP speed (ω_{min} and ω_{max}) and two constraints known as upthrust and downthrust limits as shown in Fig. E.2. Upthrust and downthrust regions correspond to unbalanced thrust forces either in the upwards or downwards direction, leading to mechanical degradation of the pump. Therefore, it is undesirable to operate in these regions. These are often represented on the pump performance curves, which are usually provided by the ESP vendor. A typical ESP envelope is shown in Fig. E.2a as a function of flowrate and pump head. The pump head is given by,

$$H = \frac{(p_{dis} - p_{In})}{\rho g} \tag{E.2}$$

where p_{In} and p_{dis} are the pump intake and discharge pressures respectively (see Fig. E.1) and ρ is the mixture density and g is the acceleration of gravity. In



(a)



(b)

Figure E.2: ESP operating envelope shown on (a) head versus flow map (b) head versus ESP speed map.

the pump performance curves shown in Fig. E.2a, the normal operating envelope is shown by the edges A-B-C-D, where A-B and C-D are the minimum and maximum ESP speed respectively. B-C and A-D are the upthrust and downthrust limits respectively.

The operational envelope in Fig. E.2a can be translated to the envelope in Fig. E.2b which is plotted as a function of head and ESP speed. Since each line representing the pump characteristics for different frequencies in Fig. E.2a, the maximum and minimum head corresponding to the upthrust and downthrust limits respectively can be retrieved to translate the envelope in terms of head and ESP speed [2]. The main motivation for translating the pump envelope into head and ESP speed in Fig.E.2b is because this envelope is not sensitive to viscosity unlike the flow vs head envelope in Fig. E.2a. As the water and oil fractions vary, the water and oil mixture forms emulsions, which will lead to significant viscosity changes, especially for heavy oil wells. For most centrifugal pumps, the head is not very sensitive to viscosity up to 500cP, unlike flow rate, power and pump efficiency. Hence formulating the upthrust and downthrust constraints in terms of head and ESP speed makes it insensitive to viscosity changes.

E.0.1 Control structure design

In order to achieve these objectives, optimizing controllers such as model predictive control (MPC) have been proposed by Krishnamoorthy et al. [90], Pavlov et al. [135]. In this appendix chapter, we show how simple feedback controllers can be used to achieve the same objective.

In order to reduce the power consumption, we want to minimize the ESP speed, hence the low limit on the ESP speed in Fig.E.2b is potentially an active MV constraint. In the presence of disturbances, the pump operation maybe pushed towards the upthrust operating region. In such cases, the upthrust limit becomes active. To handle these different operating conditions, we use a split range controller to control the intake pressure at its setpoint as shown in Fig. E.3. The output of this PID control (with nominal range 0-100%) enters a split range logic. When the output signal u is below a chosen value u_{SR} (e.g. 50%), the ESP speed is kept close its minimum limit $\omega_{min} + \epsilon$ (e.g. between 35Hz and 36Hz) to minimize the power consumption. The well head choke is controlled via a lower layer controller that controls the pump head. The corresponding setpoint for the head H_2^{sp} is given by a linear value from maximum to minimum head for the lowest ESP speed.

$$H_2^{sp}(u) = H_{\omega_{min}}^{dn} - \frac{H_{\omega_{min}}^{dn} - H_{\omega_{min}}^{up}}{u_{SR}} u \tag{E.3}$$

where, $H_{\omega_{min}}^{dn}$ and $H_{\omega_{min}}^{dn}$ are the downthrust and upthrust head corresponding to the minimum ESP speed ω_{min} that are obtained from the ESP operating envelope in Fig. E.2. The head measurement is computed using the pump intake and discharge pressure as shown in (E.2). Head reduction in this case will reduce the ESP intake pressure. Therefore, in this case, the ESP speed is kept close to its

[2] Note that upthrust corresponds to low head and downthrust corresponds to high head in Fig. E.1

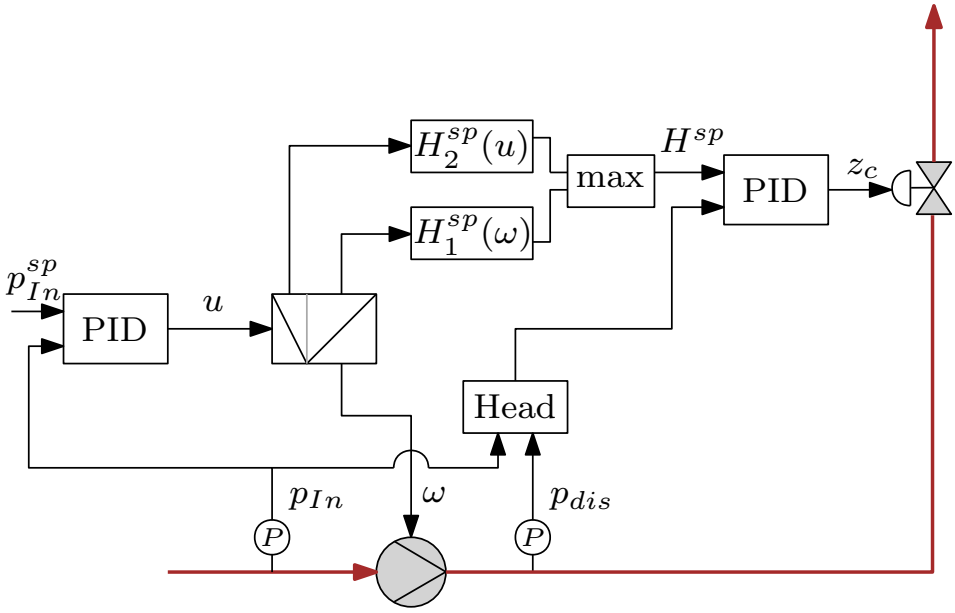


Figure E.3: Proposed control structure design for optimal operation of an ESP lifted well.

minimum ω_{min} and the ESP intake pressure is driven to its setpoint using the well head choke z_c indirectly via the head control (E.3).

When the first control output gets higher than u_{SR}, then the pump speed increases to maintain the ESP intake pressure at its setpoint. In this case, the upthrust constraint becomes active. Therefore, the head is controlled using the well head choke to maintain it at its upthrust limit and the corresponding setpoint is given as the upthrust head, which is a function of the ESP speed,

$$H_1^{sp}(\omega) = c_{up}\omega^2 \tag{E.4}$$

The head setpoint in this case is designed such that there is a margin to the actual upthrust limit. Here, the head setpoint essentially follows the upthrust limit (plus a user defined margin) as shown in the operating envelope in Fig. E.2b. Note that the hear scales quadratically with ESP speed as given by the pump affinity laws [179], hence the square term on the ESP speed in (E.4). In this case, the intake pressure is controlled using the ESP speed and the wellhead choke in used to control the head at its upthrust limit[3]. To automatically switch between the two head setpoints, (E.3) and (E.4), we use a maximum selector block $H^{sp} = \max(H_1^{sp}, H_2^{sp})$, as shown in Fig. E.3. Therefore, the operation of the ESP is maintained along the red dashed lines that are shown in the ESP envelope in Fig. E.2. The ESP speed and the head

[3]Depending on the optimization objective, one may also choose to follow the best efficiency point (BEP) instead of the upthrust limit for increased ESP lifetime instead of increased production

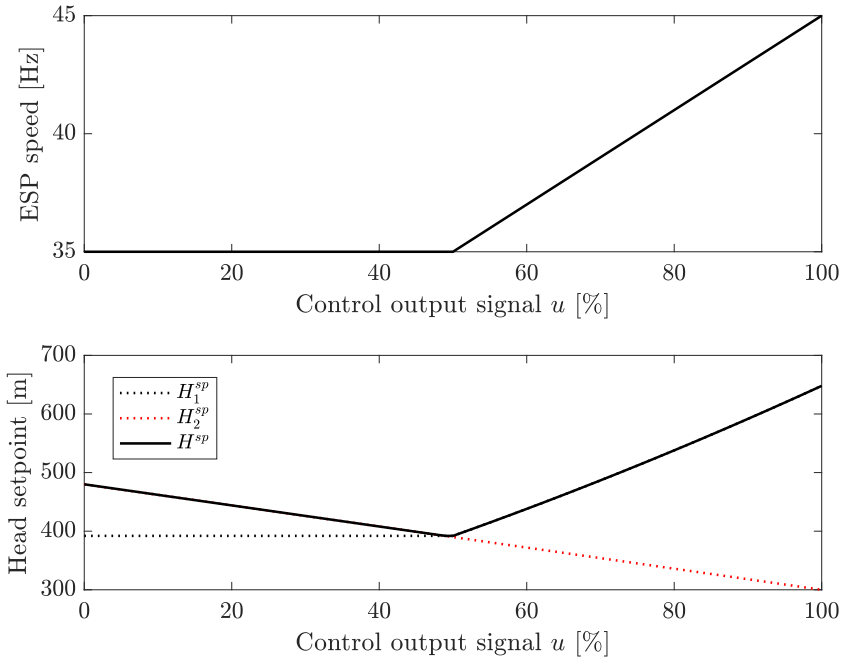


Figure E.4: The ESP speed and the head setpoint as a function of the intake pressure controller output signal.

setpoint as a function of the intake pressure control output u are shown in Fig. E.4.

E.0.2 Experimental results

The proposed control system was tested in a large-scale multi-phase test facility at Equinor research center in Porsgrunn, Norway. The test facility is designed to use live viscous crude oil (crude oil in real well conditions). The ESP installed in the test facility is a full-scale multistage horizontal centrifugal pump with 84 stages. The pump is a 538-P75 SXD model from Baker Hughes and is equipped with a variable speed drive to control the pump speed.

The layout of the multiphase flow loop is shown in Fig. E.5. The oil and water stream from the main separator is fed to the ESP with the help of feeder pumps. The heat exchanges allow the fluids to be cooled to a desired temperature. The oil feeder pump and the corresponding split stream are controlled such that the flow rate is inversely proportional to the ESP intake pressure. The water split stream is controlled using a ratio controller to get the desired watercut. As such, the main separator along with the oil and water feed pumps emulate the reservoir inflow in to the well. Water and oil is mixed upstream the ESP and enters the ESP. The fluid mixture is then directed through a 2" test section that is 200m long. This section

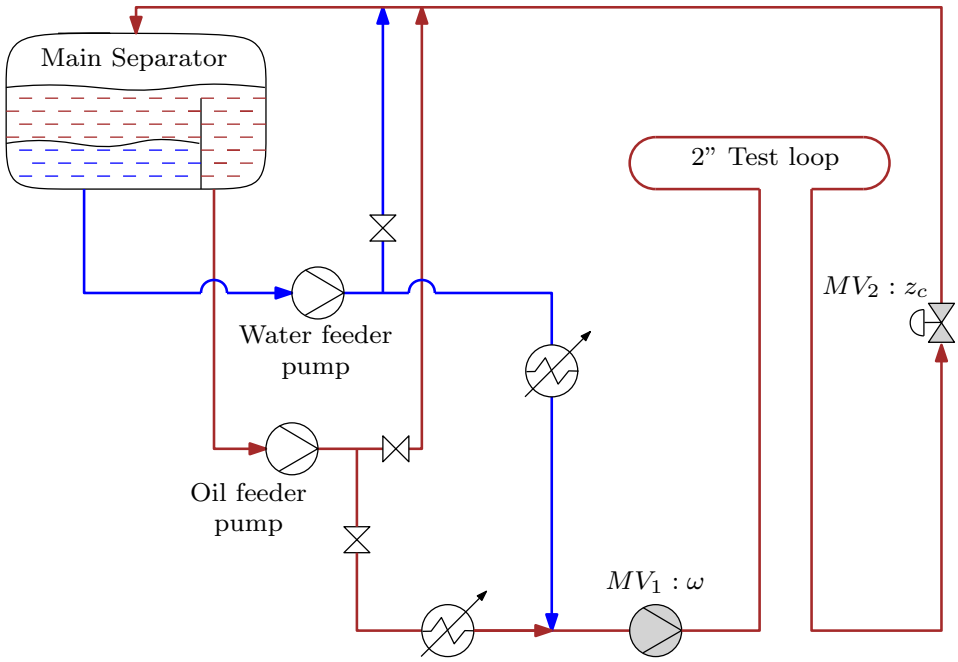


Figure E.5: Schematic representation of the large-scale experimental test facility used to test the proposed control structure.

emulates the well tubing. A choke downstream this 2" test section is used as the production choke as indicated in Fig. E.5. The fluid mixture is then redirected back to the main separator. The specifications of the test facility are summarized in Table. E.1.

The PID controllers, logic blocks and the calculation blocks to compute the head setpoint were all implemented in the process control system from Kongsberg AIM process control system that is normally used for operation of the test facility. Adjustment of setpoints, controller parameters and the signal range are carried out from the same user interface that the operators are familiar with. The PID controllers were tuned using trial and error method and are shown in Table. E.2.

For the ESP in the test facility, the minimum and maximum ESP speed were 35Hz and 45Hz respectively. The split range controller used to control the ESP intake pressure was designed with $u_{SR} = 50\%$, where the ESP speed is kept between 35Hz and 36Hz when the controller output is below $u_{SR} = 50\%$ and the head is kept at its upthrust limit when the controller output is above $u_{SR} = 50\%$. The head setpoints were computed according to (E.4) and (E.3) for the regions below and above $u_{SR} = 50\%$. The different parameter values used to compute the head setpoints are shown in Table. E.3.

The proposed control structure was tested for a total of 2 hours (from 9:00 to 11:00). The experiment starts with 0% watercut. The controllers were turned on at time 09:05 and we see that the intake pressure is driven to its setpoint as shown in Fig. E.6. The setpoint for the ESP intake pressure and the head are shown in

Table E.1: Test facility specifications

No. of phases	3 (oil, water and gas)
Fluids	Saline water, crude oil, natural gas
Oil flow rate	0-15m^3/h
Water flow rate	0- 10m^3/h
Liquid flow rate	0 - 25m^3/h
Max pressure at ESP	175 bar
Temperature range	4 - 110°C
nominal oil viscosity	70cP (at $\sim$ 40°C)
Pipe internal diameter	0.05248 - 0.079m
Total liquid volume	8.3m^3
Material	Duplix stainless steel 22Cr

Table E.2: Controller tuning parameters used in Fig. E.3.

	K_P	K_I
Intake pressure control	1	0.05
Head control	0.05	2.5e-3

Table E.3: Parameters used in the control structure design

Parameter	value
$H^{dn}_{\omega_{min}}$	480m
$H^{up}_{\omega_{min}}$	390m
c_{up}	0.32
u_{SR}	50%

solid red lines and the measured intake pressure and the head are shown in solid black lines. The upthrust and downthrust constraints are shown in red dotted lines. The ESP speed and the well head choke are also shown in solid black lines and the maximum and minimum MV limits are shown in red dotted lines, as clearly marked in Fig. E.6. The ESP head is also plotted on the head Vs speed envelope which shows that the proposed control structure design is able to maintain the ESP operation inside the safe operating envelope as shown in Fig. E.7. It is important to note that when large disturbances such as flow inversion happens, the pump violates the upthrust limit only dynamically, which is acceptable.

At time 09:13, the ESP intake pressure was changed from 39bar to 37bar and the new intake pressure setpoint is achieved by the controllers as shown in Fig. E.6. It can also be seen that this is achieved by keeping the ESP speed close to its minimum (i.e. between 35Hz and 36Hz in this case).

As mentioned earlier, one of the most common disturbance to an ESP lifted well is the manifold pressure (due to the coupling between the different wells). To emulate this disturbance, the pressure downstream the production choke was varied between times 09:20 and 09:55 as shown in the bottom-left subplot in Fig. E.6. When this disturbance happens, it can be seen that the intake pressure controlled

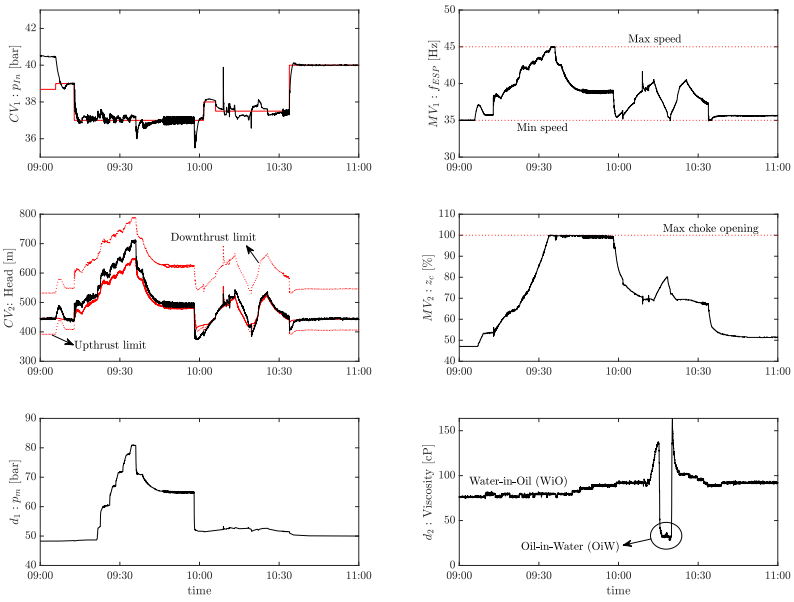


Figure E.6: Experimental Results using the proposed control structure. Measured values are shown in solid black lines, setpoints are shown in solid red lines and constraints are shown in red dotted lines.

output increases above $u_{SR} = 50\%$ and the ESP speed is increased to maintain the intake pressure setpoint. Consequently, the head setpoint now coincides with the upthrust limit as shown in Fig. E.6 and the production choke is increased to follow the head setpoint.

At time 09:45, the watercut starts to increase slowly and consequently, the mixture viscosity starts to increase due to water-in-oil (WiO) emulsion formation. This can be seen in the bottom right subplot in Fig. E.6. As the watercut increases, the ESP speed increases to maintain the intake pressure at its setpoint and the head setpoint coincides with the upthrust limit. At 10:15, the watercut is high enough to cause an inversion from water-in-oil emulsion[4] to oil-in-water emulsion[5]. When this happens, there is a significant drop in viscosity from ∼140cP to ∼30cP. At 10:20, the water cut is reduced again and we see that the emulsion inverts back to water-in-oil emulsion. During the flow inversions, there is a significant change in the viscosity and it can be seen that the proposed control structure is able to maintain the intake pressure at its setpoint and maintain the ESP operating within the operating envelop.

Several other tests were conducted at the Equinor test facility in Q1 2013 to

[4]water droplets suspended in oil medium
[5]oil droplets suspended in water medium

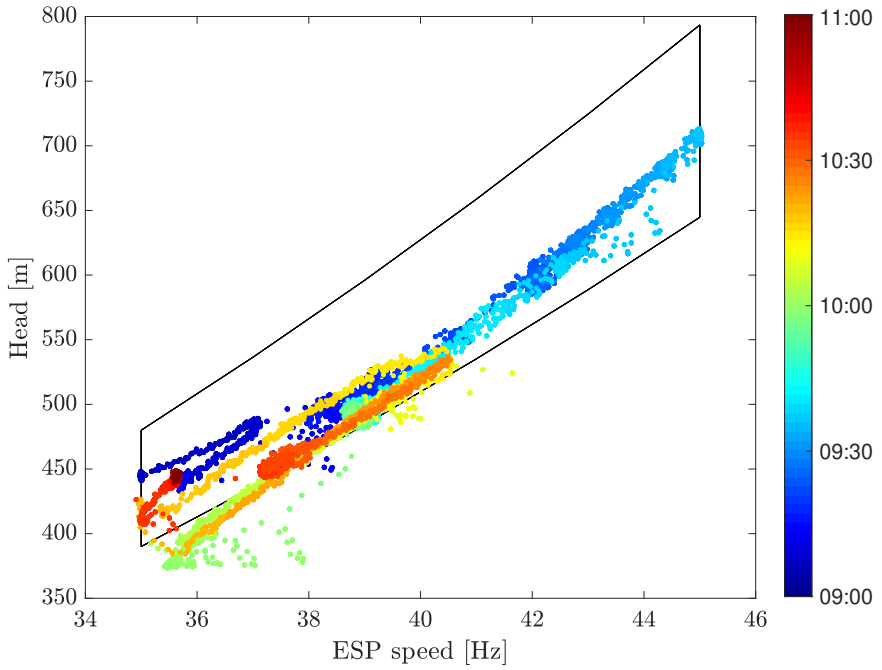


Figure E.7: Experimental Results using the proposed control structure. Measured head plotted on the operating envelope.

validate the use of simple PID control structures for optimal operation of the ESP lifted well. These lasted over 6 days. The test campaign also involved repeating the tests several times to ensure repeatability and reproducability of the test results. The overall conclusion from the test campaign was that PID controllers were able to consistently achieve the control objectives and simple logics such as split range and selectors were sufficient to handle changes in different operating regions. For the sake of brevity, only a few test points are presented in this thesis. Based on the test results, the use of automatic control of ESP lifted wells were qualified for first-use in Equinor operated fields.

Appendix F

Handling active constraint changes with MV-MV switching

Online process optimization using simple PID control structures was presented in Chapter 4, where the main focus was on how to handle CV-CV switching. This appendix contains the paper on handling active constraint changes with MV-MV switching.

- Paper published in 2018 Computer Aided Chemical Engineering (ESCAPE 28), Graz, Austria.

This paper was selected as a Keynote presentation. The reader is also referred to [146] for further reading on this topic.

Anton Friedl, Jiří J. Klemeš, Stefan Radl, Petar S. Varbanov, Thomas Wallek (Eds.)
Proceedings of the 28th European Symposium on Computer Aided Process Engineering
June 10th to 13th, 2018, Graz, Austria.

Changing between Active Constraint Regions for Optimal Operation: Classical Advanced Control versus Model Predictive Control

Adriana Reyes-Lúa[a], Cristina Zotică[a], Tamal Das[a], Dinesh Krishnamoorthy[a] and Sigurd Skogestad[a*]

[a]*Norwegian University of Science and Technology, Department of Chemical Engineering, Sem Sælands vei 4, 7491 Trondheim, Norway*
**sigurd.skogestad@ntnu.no*

Abstract

Control structures must be properly designed and implemented to maintain optimality. The two options for the supervisory control layer are Advanced Control Structures (ACS) and Model Predictive Control (MPC). To systematically design the supervisory layer to maintain optimal operation, the constraints that can be given up when switching active constraint regions should be prioritized. We analyze a case study in which we control the temperature and the flow in a cooler with two degrees of freedom (DOF) represented by two valves, one for each of the two streams. Either valve can saturate and make a constraint active, forcing other constraints to be given-up, and thus changing the set of active constraints. We show that optimal or *near*-optimal operation can be reached with both ACS and MPC. We do a fair comparison of ACS and MPC as candidates for the supervisory layer, and provide some guidelines to help steer the choice.

Keywords: Process control, supervisory control, PID control, MPC, optimal control, active constraints

1. Introduction

On a time-scale basis, the overall control problem of a process plant can be decomposed into different layers. The upper layers are explicitly related to slow time scale economic optimization, which sends economic setpoints to the lower and faster control layer. The control layer is divided into supervisory layer and regulatory layer. The latter follows the set-points given by the former and stabilizes the plant. Most process are operated under a set of constraints, which can be operational limitations, quality specifications, or safety and environmental requirements. “Active constraints” are related to variables that should be kept at their limiting value to achieve optimality. These can be either Manipulated Variables (MVs) or Controlled Variables (CVs). The MVs correspond to the dynamic (physical) DOF used by the control system, and a typical MV constraint is the maximum opening of a valve. An example of CV constraint is the maximum pressure in a distillation column. Every process is subject to disturbances, such as changes in feed rate or product specification. It is the task of the supervisory or “advanced” control layer to maintain optimal operation despite disturbances. The supervisory control layer has three main tasks (Skogestad, 2012):

1. Switch between the set of CVs and control strategies when active constraint changes occur due to disturbances.

2. Supervise the regulatory layer, avoiding saturation of the MVs used for regulatory control.
3. Follow economic objectives by using the setpoints to the regulatory layer as MVs .

The supervisory control layer could be designed using classical ACS with PID controllers, or using MPC, which achieves optimal operation and handles constraints and interactions by design. With ACS, we refer to PID-based structures such as split range control (SRC), input resetting (valve positioning), and use of selectors, to name a few.

2. Changes in active constraint regions and optimal operation

When a disturbance occurs, the process might start operating in a different active constraint region. If the supervisory layer is well-designed, it is possible to maintain optimal operation by using ACS with PID controllers, or by using MPC.

2.1. Optimal control in the presence of active constraint changes

Regardless of whether we choose ACS or MPC, the first step to systematically design the supervisory control layer is to identify and prioritize all constraints. It is useful to visualize how disturbances may cause new constraints to become active. In some cases, we can generate a plot showing the active constraint regions (optimal operation) as a function of variations in important disturbances by solving a series of optimization problems. This may be very time consuming and, in some cases, difficult due to the lack of an appropiate model. Moreover, it can also be difficult to visualize for more than two variables. Alternatively, we can use process knowledge and engineering insight to minimize the need for numerical calculations (Jacobsen and Skogestad, 2011). This information is useful regardless of the type of controller used in the supervisory layer.

Prioritization of constraints has been implemented in a few industrial MPC applications (Qin and Badgwell, 2003). Reyes-Lúa et al. (2018) propose a guideline to generate a priority list of constraints that can be used also for ACS. Under this scheme, the constraints with the lowest priority should be the first given-up when it is not feasible to fulfill all constraints. This way, controlling a high priority constraint will never be sacrificed in order to fulfill a low priority constraint.

2.2. Advanced control structures in the supervisory layer

ACS requires a choice of pairings, which can become challenging with changing active constraints. When implementing ACS, Reyes-Lúa et al. (2018) propose to start designing the control system for the nominal point, with few active constraints and with most of the priorities satisfied. Then, to minimize the need for reassignment of pairings when there are changes in active constraints, we should pair MVs with CVs according to the *Pairing Rule* (Minasidis et al., 2015): *An important controlled variable (CV) (which cannot be given up) should be paired with a manipulated variables (MV) that is not likely to saturate*.

When a disturbance occurs and the process starts operating in a different active constraint region, two types of constraints might be reached:

- MV constraint: we must give up controlling the corresponding CV. If the pairing rule is followed, this MV is paired with a low priority CV, which can be given-up. However, if it is not possible to follow the *pairing* rule, the high priority CV must be reassigned to an MV which is controlling a low priority CV. This requires the use of ACS such as input resetting (valve position control) or SRC combined with a selector block.
- CV constraint: we should give up controlling a CV with a lower priority. We can do this using a *min/max* selector.

2.3. Model predictive control in the supervisory layer

MPC uses an explicit process model to predict the future response of the plant and, by computing a sequence of future MV adjustments, optimizes the plant behavior. The first input of the sequence is applied to the plant, and the entire calculation is repeated at every sampling time (Qin and Badgwell, 2003).

The main challenge when using MPC is that expertise and a good model is required. This is either difficult to have ready at startup, or the modelling effort is too expensive. To achieve a truly optimal operation, the model would need to be perfect, and all the measurements would need to be available and reliable, which is unrealistic from a practical point of view. There are methods to circumvent this, but there is no universal solution and this analysis is out of the scope of this paper.

When an application lacks DOF to meet all control specifications, standard text-book MPC does not handle changes in active constraints effectively. The standard approach is to use weights in the objective function to assign the priorities. Having weights in the objectives function implies a trade-off between the control objectives. An optimal weights selection can assure that a CV is completely given-up, or that the solution will lie at the constraint, as explained in Section 3.4.2. However, there is no systematic way of choosing the weights, as there are no tuning rules for MPC, and this has to be done by trial and error.

An alternative approach consists of implementing a two-stage MPC with a priority list. The first stage has the purpose of finding the solution of a sequence of local steady-state optimization problems (LPs and/or QPs). In this sequence constraints are added in order of priority. The resulting information regarding feasibility is used in the formulation of the dynamic optimization problem for the MPC in the second stage (Qin and Badgwell, 2003).

3. Case study

We study a cooler in which the main control objective is to keep the outlet temperature in the hot stream to a desired setpoint ($T_H = T_H^{sp}$) by using cooling water (F_C). Additionally, the setpoint for the flow of the hot stream (F_H^{sp}) can be changed.

There are two MVs, one corresponding to the cooling water (F_C) and another to the hot stream (F_H). Desired operation is at maximum throughput, with $F_H^{sp} = F_H^{max}$. The primary input (F_C) may saturate for a large disturbance (T_c^{in}). This case is an extension of what is presented by Reyes-Lúa et al. (2018).

3.1. Process model

We consider a countercurrent cooler, represented by the dynamic lumped model in Eq. (1). The cooler is discretized in space into a series of $n = 10$ cells, as depicted in Fig. 1. Incompressible fluids and constant heat capacities are assumed. The boundary conditions are: $T_{H_0} = T_{H_{in}}$ for cell $i = 1$ (inlet), and $T_{C_{11}} = T_{C_{in}}$ for cells $i = 10$ (outlet). The energy balance for cell $i = 1 \dots n$ is:

$$\frac{dT_{C_i}}{dt} = \frac{F_C}{\rho_C V_{C_i}}(T_{C_{i+1}} - T_{C_i}) + \frac{UA_i(T_{H_i} - T_{C_i})}{\rho_C V_{C_i} c_{p_C}} \tag{1a}$$

$$\frac{dT_{H_i}}{dt} = \frac{F_H}{\rho_H V_{H_i}}(T_{H_{i-1}} - T_{H_i}) + \frac{UA_i(T_{H_i} - T_{C_i})}{\rho_H V_{H_i} c_{p_H}} \tag{1b}$$

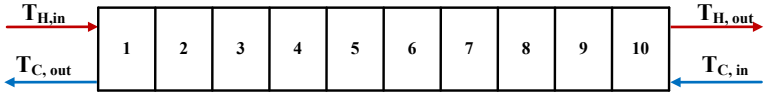


Figure 1: Lumped model for the studied cooler.

3.2. List of priorities

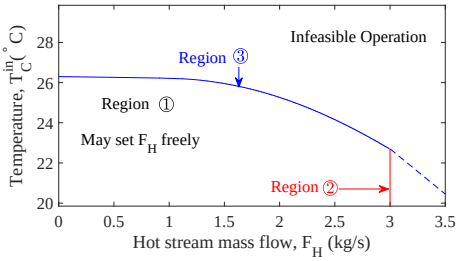
Table 1 shows the priority list of constraints for the cooler we are analyzing.

Table 1: Constraints for the studied cooler.

Priority level	Description	Constraints
1	MV inequality constraints which define the feasibility region	$F_H \leq F_H^{max}$ $F_C \leq F_C^{max}$
2	MV or CV equality constraints, which is the control objective	$T_H = T_H^{sp}$
3	Desired throughput	$F_H = F_H^{sp}$
4	CV inequality constraints or self optimizing variables	*none*

3.3. Active constraint regions for cooler

As we want to keep $T_H = T_H^{sp}$, this constraint is always active and only one DOF remains. With one DOF and three potential constraints we have three possible active constraint regions, which are shown as a function of the throughput (F_H) and the disturbance (T_c^{in}) in Fig. 2.



Active constraint in each region:

- Region 1: $F_H = F_H^{sp}$
- Region 2: $F_H = F_H^{max}$
- Region 3: $F_C = F_C^{max}$

Figure 2: Active constraint regions for the cooler

3.4. Design of the supervisory layer for the cooler

We consider nominal operation in Region 2 ($F_H^{sp} = F_H^{max}$). According to the priority list, when T_c^{in} is so high that $F_C = F_C^{max}$, the controller should give up controlling $F_H = F_H^{max}$ and reduce F_H to keep $T_H = T_H^{sp}$, thus switching to Region 1. We design both an ACS and an MPC for this case.

3.4.1. Classical advanced control structures for optimal operation

To design the supervisory layer using ACS, we implement SRC with a *min* selector block, as in Fig. 3. The controller is tuned by fitting a first order plus delay model obtained from the open-loop step response of the process, and applying the SIMC rule (Skogestad, 2003) with $\tau_c = 80\ s$ and $K_c = -0.06$. To account for the different gains that F_C (negative) and F_H (positive) have on T_H, the MVs were respectively multiplied with a gain of 1 and -2.

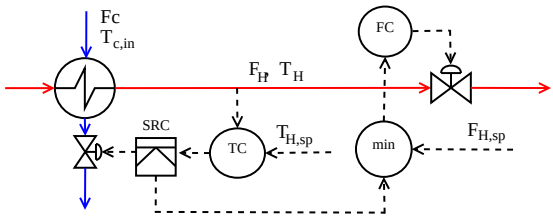


Figure 3: Split range control structure for cooler.

3.4.2. Model predictive control for optimal operation

The optimal control problem is discretized into a finite dimensional optimization problem divided into $N = 40$ control intervals. We use a third order direct collocation scheme for a polynomial approximation of the system dynamics for each time interval.

The dynamic optimization problem is setup in CasADi (Andersson, 2013), which is an algoritmic differentiation tool. According to Eq. 1, the dynamic model is non-linear. The resulting NLP problem is thus solved using IPOPT (Wächter and Biegler, 2005). The prediction horizon is set to 400 s with a sampling time of $\Delta t = 10$ s. We assume we have full state feedback and the disturbance, T_c^{in}, is measured.

In this paper, we chose to implement the standard MPC formulation given by Eq. 2, and to assign different weights for the two control objectives. A high weight is assigned to the high priority CV (T_H) and a low weight is assigned to the low priority CV (F_H). The values $\omega_1 = 3$ and $\omega_2 = 0.1$ are used. These were found by trial and error. In addition, the MVs are restricted to a rate of change of 10 % of F_H^{max} and F_C^{max} respectively.

$$
\begin{aligned}
&\min \sum_{k=1}^{N} \left(\omega_1 \left\| \left(T_{H_k} - T_H^{sp}\right) \right\|^2 + \omega_2 \left\| \left(F_{H_k}^{max} - F_{H_k}\right) \right\|^2 \right) \\
&\text{s.t.} \\
&\left.\begin{aligned}
&T_{k,i} = f(T_{H_k,i}, T_{H_k,i-1}, T_{C_k,i}, T_{C_k,i+1}, F_{H_k}, F_{C_k}) \\
&0 \le F_{H_k} \le F_H^{max} \\
&0 \le F_{C_k} \le F_C^{max}
\end{aligned}\right\} \quad \forall k \in \{1,\ldots,N\} \\
&\left.\begin{aligned}
&0 \le \Delta F_{H_k} \le 0.1 F_H^{max} \\
&0 \le \Delta F_{C_k} \le 0.1 F_C^{max}
\end{aligned}\right\} \quad \forall k \in \{1,\ldots,N-1\}
\end{aligned}
\tag{2}
$$

where $\Delta F_k = F_k - F_{k-1}, \forall k \in \{1,\ldots,N-1\}$. For $k = 1$, F_{k-1} represents the flow at the nominal operation point.

3.4.3. Simulation results

Fig. 4 shows the simulation results for the case study. T_H^{sp} is 26.3°C. MPC and SRC structures are tested for the same step disturbances in T_C^{in}: +2°C at $t = 10\,s$, and an additional +4°C at $t = 1000\,s$. Both MPC and SRC follow the priority list and reach optimal operation at steady state. Once $F_C = F_C^{max}$, the control structure gives-up controlling $F_H = F_H^{max}$, and F_H is used as MV to maintain $T_H = T_H^{sp}$.

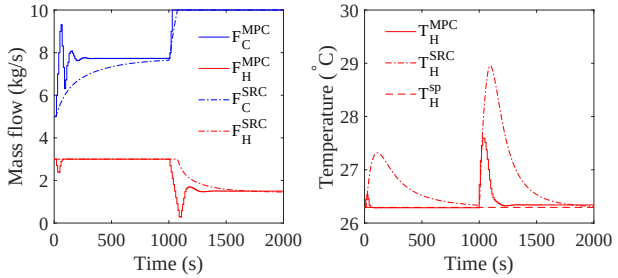


Figure 4: Simulation results for MPC and SRC.

4. Discussion and conclusion

The supervisory control layer can be designed using MPC or ACS. MPC uses the manipulated variables to achieve optimal operation by design, but it requires expertise and a model, which may be difficult to obtain. Well designed ACS can also maintain optimal operation, require much less model information, and are usually easier to implement and tune. In our example, SRC efficiently switches the MVs, achieving optimal operation. Compared to ACS, MPC implementation requires more effort as the tuning of weights in the objective function is more challenging because it is done by trial and error. As it is seen in Fig 4, for a short transient time during the first disturbance in the MPC implementation, $F_H \neq F_H^{max}$. This could be improved by increasing ω_2 relative to ω_1. This would however be at the expense of having an offset for T_H from T_H^{sp}, as its weight in the objective function would be smaller. Therefore, we should point out that a different MPC implementation or tuning could have better performance, especially on the input usage.

We recommend to use priority lists as a tool for analyzing and designing the supervisory layer. Understanding the process is an important step to decide which controller should be implemented. Both ACS and MPC have advantages and disadvantages, and the designer of the control layer should be aware of these. While in simple cases such as the presented case study, ACS seems better fitted due to achieving optimality with less implementation effort, in multivariable systems with more interactions, MPC should be considered as the most convenient alternative.

5. Acknowledgements

This work was partly supported by the Norwegian Research Council under HighEFF (Energy Efficient and Competitive Industry for the Future) and SUBPRO (Subsea Production and Processing).

References

J. Andersson, 2013. A General Purpose Software Framework for Dynamic Optimization. Phd thesis, KU Leuven.

M. G. Jacobsen, S. Skogestad, 2011. Active Constraint Regions for Optimal Operation of Chemical Processes. Industrial & Engineering Chemistry Research 50 (19), 11226–11236.

V. Minasidis, S. Skogestad, N. Kaistha, 2015. Simple Rules for Economic Plantwide Control. In: K. V. Gernaey, J. K. Huusom, R. Gani (Eds.), PSE 2015 and ESCAPE-25. Elsevier Science Direct, pp. 101–108.

S. Qin, T. A. Badgwell, 2003. A survey of industrial model predictive control technology. Control Engineering Practice 11 (7), 733–764.

A. Reyes-Lúa, C. Zotica, S. Skogestad, 2018. Optimal Operation with Changing Active Constraint Regions using Classical Advanced Control. In: 10th ADCHEM. IFAC, Shenyang, China.

S. Skogestad, 2003. Simple analytic rules for model reduction and PID controller tuning. Journal of Process Control 13, 291–309.

S. Skogestad, 2012. Economic Plantwide Control. In: G. P. Kariwala, Vinay Rangaiah (Ed.), Plantwide Control. Wiley, Ch. 11, pp. 229–251.

A. Wächter, L. T. Biegler, Apr. 2005. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106 (1), 25–57.

Appendix G

Plantwide control of an oil production network

This appendix contains the paper showing the comparison of Economic NMPC, self-optimizing control and feedback RTO based on dynamic simulations using the high-fidelity OLGA simulator.

- Preprint submitted to Computers and Chemical Engineering.

Plantwide control of an oil production network

Esmaeil Jahanshahi[a], Dinesh Krishnamoorthy[a], Andrés Codas[b], Bjarne Foss[b], Sigurd Skogestad[a]

[a] *Department of Chemical Engineering, Norwegian Univ. of Science & Technology (NTNU), NO-7491 Trondheim*
[b] *Department of Engineering Cybernetics, Norwegian Univ. of Science & Technology (NTNU), NO-7034 Trondheim*

Abstract

In this paper, we consider Real-Time Optimization (RTO) and control of an oil production system. We follow a systematic plantwide control procedure. The process consists of two gas-lift oil wells connected to a pipeline-riser system, and a separator at the topside platform. When the gas injection rates are low, the desired steady flow regime may become unstable and change to slug flow due to the casing-heading phenomenon. Therefore, a regulatory control layer is required to stabilize the desired two-phase flow regime. To this end, we propose a new control structure using two pressure measurements, one at the well-head and one at the annulus. For the optimization layer, we compare the performance of nonlinear Economic Model Predictive Control (EMPC), dynamic Feedback-RTO (FRTO) and Self-Optimizing Control (SOC). Based on dynamic simulations using the realistic OLGA simulator, we find that SOC is the most practical approach.

Keywords: Oil production, plant-wide control, Self-Optimizing Control, unstable systems

1. Introduction

Modeling, estimation, control, and optimization methodologies are becoming exceedingly important in the upstream petroleum industries. Concepts from Advanced Process Control (APC) are being developed and deployed in offshore oil and gas production (Campos et al., 2015). There are however numerous remaining challenges (Foss, 2012).

One standard approach for control and optimization of multi-input multi-output processes is centralized model-based control (*e.g.*, Nonlinear Model Predictive Control, NMPC) which simultaneously uses all the inputs and outputs of the system (Engell, 2007). In theory, such a control scheme can optimally handle the dynamic interactions between different input/output pairings, and provide inputs for optimal operation of the system. However, the success of this solution depends on obtaining a good process model and the ability to update it. In addition, optimization using detailed dynamic models (with hundreds of state variables) is computationally demanding and often not suitable for real-time applications. Campos et al. (2015) says that many numerical issues need to be addressed before dynamic optimizers can be widely used in the offshore oil and gas production. Instead, fast local controllers can be used for stabilization while slower centralized optimizers may be used for long-term optimization (Skogestad, 2004).

In our study, we initially attempted to solve an optimal oil production control problem using a centralized NMPC approach, that is, with a single optimizing controller. However, we were not successful. This was, partly, because the plant is unstable and, also because of the plant-model mismatch. We used the Olga simulator as the "real" process and a simplified dynamic model for the control design. Willersrud et al. (2013) successfully applied a centralized control structure (NMPC) to another oil and gas production system. However, they assumed no model-plant mismatch; that is, they considered the same model for both optimization and simulation. Thus, robustness against modeling errors was neglected. Moreover, this single-layer centralized strategy was not tested in closed-loop with unmeasured disturbances.

Complex industrial processes require a structured control architecture for their operation. Skogestad (2004); Saputelli et al. (2006); Foss (2012) propose to decompose the control and optimization problem on different time scales. Luyben et al. (1997) and Skogestad (2004) propose systematic procedures for design of such plantwide control systems.

The resulting multi-layer plantwide control structure in Fig. 1 (Skogestad, 2004) is well established in the process industry, see *e.g.*, Campos et al. (2015). The lower control layers are fast and do not affect the optimization of the process. In practice, the slow Real-Time Optimization (RTO) layer is designed based on steady-state models. These models are usually detailed physical models, but they may also contain empirical nonlinear equations. Such models can be described, for example, by piecewise-defined numeric functions in the optimization problem formulation (Gunnerud and Foss, 2010).

The fastest regulatory layer typically controls levels, flow rates, and pressures. There is also a growing interest in introducing a slower secondary advanced (supervisory) control layer (APC), for example using model predictive control (MPC), for coordination purposes and for taking

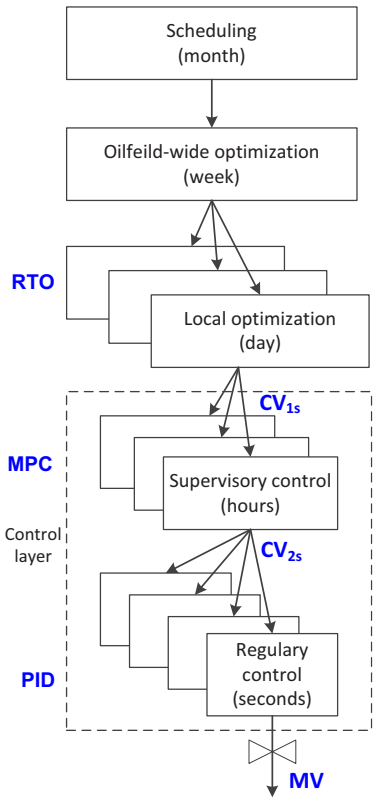


Fig. 1: Schematic presentation of multi-layer control structure

into account the constraints and interactions (Foss, 2012).

The plantwide control structure design method presented by Skogestad (2004) is divided into a Top-down analysis and Bottom-up design (Skogestad, 2004, see Table 1). The Top-down analysis starts with the definition of operational objectives, then the identification of the manipulated variables and degrees of freedom (DOF), optimization and finally selecting the primary variables for control (CV_1 in Fig. 1). In contrast, the Bottom-up design starts with the stabilizing control layer, then supervisory controller, and ends up with providing the integration with the optimization layer. The idea is that the control layer should contribute to the optimization.

For the top-down step a key decision is the selection of economic controlled variables (CV_1). For the bottom-up step a key desicion is the selection of stabilizing controlled variables (CV_2). Often the CV_1 variables are active constraints and they are sometimes moved into the fast regulatory layer (as part of CV_2).

Many papers on oil and gas production optimization focus only on the optimization layer, assuming a perfect regulatory control (Krishnamoorthy et al., 2016). However, gas-lifted wells and multiphase risers may become dynamically unstable, in particular at the optimum economic point (Di Meglio et al., 2012). The usual remedy to these instabilities is to use more lift-gas, which may be costly or not available. Moreover, injecting more gas increases the friction pressure loss which reduces the production rate. Another passive solution is to choke a downstream valve, which may lead to sub-optimal operation due to an increased back-pressure. Stabilizing the desired non-slug flow regime by feedback control is shown as an optimal solution to gas-lifted well instabilities (Dalsmo et al., 2002). Both robustness and tracking performance are necessary for the stabilizing regulatory control layer. The regulatory control layer must remain stable despite the variations in the process gain due to change of the operating point.

The primary objective of this study is to apply the plantwide control design procedure on the gas-lift production system. We compare three online optimizing control strategies, namely, 1) Economic NMPC, 2) Self-Optimizing Control, and 3) Direct input adaptation using a new model-based feedback-RTO (Krishnamoorthy et al., 2019). Moreover, we study the effect of unmeasured disturbances in reservoir pressure and gas-oil ratio. We use the Olga simulator as the "real" plant and use a simplified dynamic ODE model for state estimation and optimization. This approach allows us to study the effect of modeling errors.

The article is organized as follows. In Section 2, we introduce the oil production network. The top-down analysis is explained in Section 3. The bottom-up design procedure is presented in Section 4 where the design and the robustness properties of the regulatory control layer are presented. In Section 5, we consider the optimization layer including the Economic NMPC, self-optimizing control and the dynamic Feedback-RTO. Dynamic simulations are presented in Section 6. After discussing the findings and challenges of this work in Section 7, the concluding remarks are given in Section 8.

2. Oil production netwok

The oil production network is simulated using the Olga simulator and is represented in Fig. 2. Olga is the industry-standard tool for simulating dynamic multiphase flow (Schlumberger, 2018). In this work, the Olga model consists of two wells operated by gas-lift. The oil wells feed a common pipeline-riser connected to a topside separator. The system has seven control inputs (Manipulated Variables, MVs):

- Gas injection mass flow rate at the annulus top of each well (MV_{1A}, MV_{1B})
- Production choke valve opening of each well (MV_{2A}, MV_{2B})
- Top-side valve opening (MV_3)
- Two valves for separator control (MV_4, MV_5)

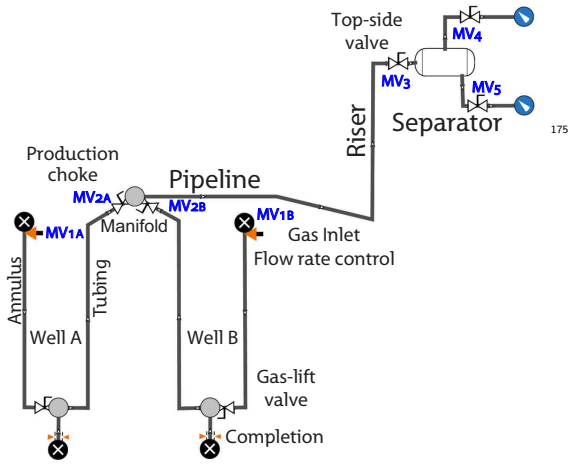


Fig. 2: Production network model in Olga simulator (open-loop process without controllers)

We consider fourteen measurements:

- Pressure measurements (7) at the riser top, at the inlet of the pipeline, at the wellheads, at the top of each annulus and in the separator pressure
- Mass flow rate measurements (6) gas and liquid rates at the two wellheads and at the riser top
- Level measurement (1) in the separator

The two vertical wells are assumed to be geometrically identical with a tubing and annulus length of 2048 m. The inner diameter of the tubing is 0.124 m, and the annulus is modeled by a cylindrical (not annular) pipe with 0.2 m diameter. The roughness of the pipes is set to 4.5E-5 m. The reservoir temperature is assumed to be 108 °C. The well inflow relation is assumed to be linear (*i.e.*, $w_{\text{res}} = \text{PI}\,(P_{\text{res}} - P_{\text{bh}})$) with a Productivity Index (PI) of 0.247 kg/s/bar. Nominally, the feed from the reservoir is oil for the present case study, that is, the produced gas-oil ratio (GOR) and water-cut are assumed to be negligible at nominal conditions. The two reservoir pressures are considered to be different; the nominal values are 160 bar for well A and 170 bar for well B.

The pipeline length is 4300 m, where the last 2300 m has a negative inclination of 1°. The pipeline goes to a riser with height 300 m. The pipeline and riser have a diameter of 0.2 m and roughness of 2.8E-5 m. The separator operates at a constant pressure of 5 bar.

In Olga, the fluid properties can be specified by a black-oil model or as PVT Tables. We use PVT tables generated by PVTSim©. For a given feed composition, these tables contain all fluid properties such as viscosity, gas density, oil density, and gas mass fraction as functions of pressure and temperature. The viscosity of the oil considered in this paper ranges from 0.2 to 1 cP, which is not sufficient to classify it as a heavy oil. In Fig. 3, we show the oil density and gas mass fraction as a function of pressure at 88 °C. The produced fluid is at 'undersaturated condition' and does not have free gas in the range of bottom-hole pressures considered. With these fluid conditions and the low reservoir pressure, the wells considered in this work are not *naturally flowing*. Therefore, gas-lift is required to assist the production.

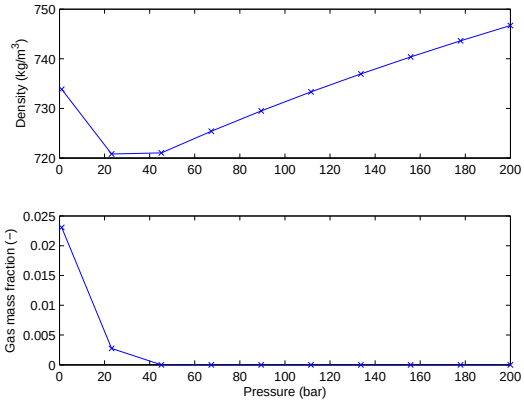


Fig. 3: Fluid Properties

The two gas-lifted oil wells operate in the casing-heading instability region when the injected gas rates are low. As shown in Fig. 4, the theoretically optimal steady-state operating point is in this region (Aamo et al., 2005). The casing-heading instability causes slugging flow regimes which are characterized by large flow and pressure oscillations in the form of a limit cycle behavior. The slugging causes safety problems and inadequate separation of oil and gas. Moreover, as seen from Fig. 4, the average production under slugging is less than the production with non-slug flow. To avoid slugging, stabilizing control is required. The stabilization is performed by low-level controllers in the regulatory layer.

2.1. Models

Three different models of the oil production network are used in this work as described next.

2.1.1. Reference model in the Olga simulator

The Olga model of the oil production network is used as the "real" process in the subsequent dynamic simulations. When the development of Olga was initiated in the 1980's at NTNU and Sintef, one of the motivations was to study the dynamics of slow flow regimes for offshore oilfields (Bendiksen et al., 1991). The simulator includes empirical correlations which have been fine-tuned by data from the oil-fields in the North Sea.

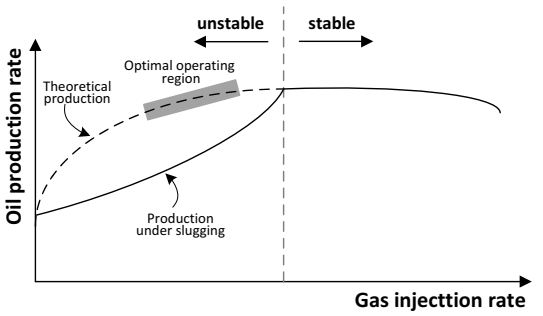


Fig. 4: Gas-lift performance curve (adapted from Aamo et al. (2005)).

However, for the user, the simulator is a black box and the user cannot see the model equations. This means it is difficult to use the commercially available Olga simulator for production optimization.

2.1.2. Spline surrogate model

With the Olga simulator, it is possible to find the global optimal operating point and optimal cost. To this end, we generated data points from simulations using Olga and constructed a surrogate spline model (Jahanshahi et al., 2016). The surrogate spline model consists of piecewise third-order or fourth-order polynomials. These models have the necessary accuracy and smoothness (two times differentiable) for global optimization solvers, and are then used to find the optimal operating points for different disturbance scenarios. The optimal cost values are used as a benchmark to compare the different optimizing control methodologies.

2.1.3. Simplified dynamic model

A simplified nonlinear dynamic model of the process is used for the state and parameter estimation and for solving the dynamic optimization problem (NMPC). The state equations of the model are the mass balances in different parts of the network. The gas-lift well model is similar to the one used by Jahanshahi et al. (2012) with some modifications, and the pipeline-riser model is as presented by Jahanshahi and Skogestad (2014).

The dynamic model can be represented as a set of ordinary differential equations:

$$\dot{x} = f\left(x, u, d\right) \tag{1a}$$
$$y = h\left(x, u, d\right) \tag{1b}$$

where x represents the dynamic states, *i.e.*, the mass of liquid and gas in the pipeline segments, u represents the manipulated variables (MV), *i.e.*, the valve openings and the mass injection of lift-gas, and d represent the disturbances, *e.g.* accounting for variations in the reservoir pressure and GOR. The outputs y includes variables of particular interests, such as pressure measurements and constrained outputs.

We fine-tuned the simplified dynamic models to match the Olga simulator. The valve coefficients were used as the tuning parameters for this purpose.

3. Top-down design procedure

3.1. Definition of operational objectives

The primary goal is to optimize an economic objective which in most cases corresponds to recover as much oil as possible. The objective can be seen from a long-term perspective, where the reservoir dynamics play an essential role (Jansen et al., 2009), or from a short-term perspective (commonly known as daily production optimization), where the reservoir is considered to be static (Kosmidis et al., 2005; Gunnerud and Foss, 2010; Codas et al., 2012; Grimstad et al., 2016). This work considers the short-term perspective and therefore disregards the reservoir dynamics. The short-term optimization of gas-lift wells is often constrained by the maximum amount of available lift-gas (Camponogara et al., 2009).

In this work, we minimize the operation cost, J = injected gas cost - produced oil value. This is the negative of the economic profit. The constraints include the maximum injection rates and the minimum valve pressure drops. The latter are for controllability purposes. The constraints are controlled in the regulatory layer as explained in the bottom-up design, section 4.2. Ideally, the constraints should be controlled at their corresponding physical bounds, but a back-off is introduced to ensure feasibility and stable operation.

More precisely, the steady-state optimization problem is:

$$\min_{u} \mathrm{J} = \alpha_g(w_{\mathrm{inj,A}} + w_{\mathrm{inj,B}}) - \alpha_o(w_{\mathrm{o},A} + w_{\mathrm{o},B}) \tag{2a}$$
subject to
$$0 = f(x, u, d) \tag{2b}$$
$$y = h(x, u, d) \tag{2c}$$
$$w_{\mathrm{inj,A}} + w_{\mathrm{inj,B}} \leq w_{gl}^{max} \tag{2d}$$
$$b_{\mathrm{l}}^{y} \leq y \leq b_{\mathrm{u}}^{y} \tag{2e}$$
$$b_{\mathrm{l}}^{u} \leq u \leq b_{\mathrm{u}}^{u} \tag{2f}$$

The five decision variables $u = (w_{\mathrm{inj,A}}, w_{\mathrm{inj,B}}, v_A^{\mathrm{w}}, v_B^{\mathrm{w}}, v^{\mathrm{p}})$ are the two gas injection rates three and valve openings. α_o, α_g are oil and gas prices ($\$/kg$). Equation (2d) is the constraint on the total available gaslift, (2e) are the constraints on selected outputs (*e.g.* valve pressure drops), and (2f) are the constraints on the five decision variables.

3.2. Manipulated variables and degrees of freedom

As mentioned, the degrees of freedom for the steady-state optimization are the three valve openings $(v_A^{\mathrm{w}}, v_B^{\mathrm{w}}, v^{\mathrm{p}})$ and the two gas injections rates $(w_{\mathrm{inj,A}}, w_{\mathrm{inj,B}})$. There are

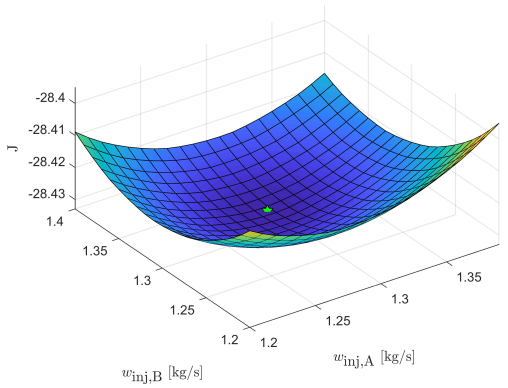


Fig. 5: Cost as a function of gas injection rates of two wells, generated by surrogate models.

actually five valves, but it is assumed that two downstream valves are used to keep constant separator level and pressure. The degrees of freedom resulting from the three valves are replaced by the valve pressure drop setpoints ($\text{CV}_{1\text{s}}$). This transformation does not have any impact on the control performance, but it is essential for optimization purposes due to the uncertainty of the valve models.

3.3. Optimal operation

The steady-state real-time optimizer uses the surrogate spline model to represent the gas-lift system. The two injection rates are the steady-state degrees of freedom for the optimization.

To obtain the surrogate model, we used the built-in parametric study tool in Olga, and we applied 0.01 kg/s steps in the two gas injection rates which gave 420 points for each disturbance scenario. Fig. 5 shows a plot of the cost as a function of the two gas injection rates with other parameters at their nominal values.

In addition to the nominal operating point, we also optimize the process for disturbances in the two reservoir pressures (D1, D2) and their gas-oil (mass) ratios (D3, D4). The nominal reservoir pressures are 160 and 170 bars, and the nominal gas-oil mass ratios are 0. Table 1 summarizes the optimal inputs and corresponding cost for various disturbances.

3.4. Active constraints (CV_1)

The primary economic controlled variables (CV_1) consist of the active constraints plus self-optimizing controlled variables. In general, it is always economically optimal to open the valves as much as possible. A maximum valve opening is equivalent to a minimum pressure drop of the valve. That is, the three valve pressure drop constraints on the three valves are always active and these are thus selected as primary controlled variables (CV_1). As a result, only the two gas injection rates are unconstrained degrees of freedom for the steady-state optimization.

Table 1: Optimal gas injection rates for various disturbance scenarios

		Nom.	D1	D2	D3	D4
DVs	$P_{\text{res,A}}$	160	155	155	160	160
	$P_{\text{res,B}}$	170	170	165	170	170
	GOR_A	0	0	0	0.03	0.03
	GOR_B	0	0	0	0	0.03
u_{opt}	$w_{\text{inj,A}}$	1.296	1.279	1.277	0.847	0.846
	$w_{\text{inj,B}}$	1.325	1.321	1.307	1.315	0.830
	J_{opt}	-28.433	-27.878	-27.313	-28.827	-29.239

3.5. Primary controlled variables CV_1 for Self-Optimizing Control

In this section, we apply the self optimizing control (Skogestad, 2004) to select the two associated controlled variables (CV_1) to be kept at constant setpoint. These controlled variables are regulated by the two unconstrained DOFs (gas injection rates). The objective is to achieve an acceptable loss with constant setpoints when disturbances occur without re-optimizing the process for the disturbances. The setpoint values are chosen as the nominal optimal values. The loss relative to the ideal (reoptimized) costs for each disturbance are shown for seven candidate controlled variables in Table 2. These losses are obtained from simulations in the Olga simulator without any simplification. We also considered the open-loop case (Alternative 0) where the two gas injection rates are kept constant.

We conclude that the well-head gas flow rates (Alternative 7) are the best controlled variables (CV_1) for self-optimizing control. Note that the well-head gas flow rate is the sum of the injection rate, the produced gas from the reservoir, and the gas flashed from the petroleum at the lower pressure of the well-head compared to the reservoir pressure. Controlling the well-head liquid flow rates (Alternative 6) gives the worst result, even worse than the open-loop case (Alternative 0). Constant GOR at the well-head also gives small losses (Alternative 1).

3.6. Production rate

The location of the throughput manipulator (TPM) is not considered in this work, because it is optimal to minimize the production cost, and the production rate is set indirectly by the optimization. The optimal production rate is affected by the prices of injected gas and produced oil, as well as the different disturbances. However, in general, the throughput manipulator should be located close to the bottleneck. For example, it should be located close to the receiving facilities if the maximum capacity of processing units (e.g., separation) is the bottleneck.

Table 2: Loss compared to optimal (J_{opt}) for four disturbance scenarios. The well-head gas mass flow rates (Alternative 7) are selected for self-optimizing control.

				Disturbances				
				Nominal	D1	D2	D3	D4
	Controlled Variables	Description	J_{opt}	−28.433	−27.878	−27.313	−28.827	−29.239
0	Open-loop	No control	Loss	0.000	0.001	0.001	0.274	0.480
1	$GOR_{\mathrm{wh,A}}, GOR_{\mathrm{wh,B}}$	Well-head GORs		0.000	0.004	0.005	0.001	0.001
2	$P_{\mathrm{wh,A}}, P_{\mathrm{wh,B}}$	Well-head pressures		0.000	0.042	0.018	0.000	0.000
3	P_{inl}	Pipeline inlet pressure		0.000	0.005	0.018	0.166	0.001
4	Z_{top}	Topside valve opening		0.000	0.004	0.013	0.165	0.005
5	$Z_{\mathrm{wh,A}}, Z_{\mathrm{wh,B}}$	Well-head valve openings		0.000	0.019	0.022	0.014	0.022
6	$w_{\mathrm{Lwh,A}}, w_{\mathrm{Lwh,B}}$	Well-head liquid mass flow rate		0.000	1.639	2.551	0.007	0.033
7	$w_{\mathrm{Gwh,A}}, w_{\mathrm{Gwh,B}}$	Well-head gas mass flow rate		0.000	0.001	0.001	0.000	0.000

4. Bottom-up design procedure

4.1. Selection for regulatory controlled variables CV2

From a controllability point of view, the bottom-hole pressure is the preferred Controlled Variable (CV2) for stabilizing control of gas-lift oil wells (Jahanshahi et al., 2012). However, the bottom-hole pressure is often not available, and if it is available, it has long sampling time intervals only suitable for monitoring purposes. Bottom-hole sensors can also easily fail and are generally not replaced in case of failure.

Aamo et al. (2005) proposed to apply a nonlinear observer to estimate the bottom-hole pressure from a combination of top-side measurements. These topside measurements are the tubing head pressure, casing head pressure, and the multi-phase fluid density. Jahanshahi et al. (2012) have carried out a controllability analysis on unstable gas lift oil wells, and have concluded that a combination of topside measurements results in a robust stabilizing control system.

Codas et al. (2016) considered combining the tubing head pressure (CV_1) and the casing head pressure (CV_2) by applying cascade control. Both CVs are located at the seabed and are relatively robust and easy to measure. Figure 6 shows such a control structure (CS1) where the annulus pressure (CV_2) is controlled in the inner loop (slave) and the tubing pressure (CV_1) is controlled by the outer loop (master). Here, the production choke valve opening is used as the Manipulated Variable (MV) for the regulatory control, and the tubing pressure setpoints (CV_{1s}) are available as a DOFs for the optimization layers.

Alternatively, the pressure drops over the two wellhead valves (CV_1) can be controlled by the master controller. This is reasonable because, as noted, the minimum pressure drop is an active constraints and should be selected as (CV_1). Such a control structure is shown as Control Structure 2 (CS2) in Figure 6. A similar control structure is applied to control the pipeline-rise subsystem (see Figure 7).

The master loops of the cascade controllers are here classified as part of the supervisory control. The master loop helps for the stabilization by avoiding drift from the controller design point (*i.e.* keeping the system in the linear region), and at the same time track the optimal setpoints given by optimization layer. The tracking performance of the master control loops is important for optimal operation.

4.2. Setpoints for active constraints

In this work, we have used 2 bar pressure as the minimum pressure drop ΔP for the wellhead valves, and 0.7 bar for the riser choke.

For optimal operation, it is desired to minimize the pressure drop in the network. Therefore, these ΔP constraints will always be active, and should be selected as primary controlled variables (CV_1). The proposed control structure (CS2 in Figure 6) allows for controlling the pressure drops at their constraints.

4.3. Regulatory control layer

A block diagram of the regulatory controllers and the disturbances are shown in Figure 8. The regulatory layer must keep the process stable during the transition along the optimal path provided by the supervisory controllers, and reject disturbances on a fast time scale. The reservoir pressure and gas-oil-ratio (GOR) are uncertain parameters which are considered as unknown disturbances in this work. Besides, the MVs used by the layers above are disturbances to the regulatory layer. For example, the gas injection rates used for as the DOFs for optimization are disturbances to the regulatory controllers.

4.4. Robustness

The main consideration for tuning the regulatory controllers is robustness. The robustness is evaluated at two operating points, at the initial conditions and at the steady-state optimal point. The gas injection rates to the wells are initially 1 kg/s, with the three valves 50% open. PI tunings, gain margin and phase margin for six pressure controllers are shown for the two operating points in Table 3. The gain margins are larger than 2 for all cases. That is, if the process gain increases by a factor 2 or decreases to half due to nonlinearity, the system remains stable.

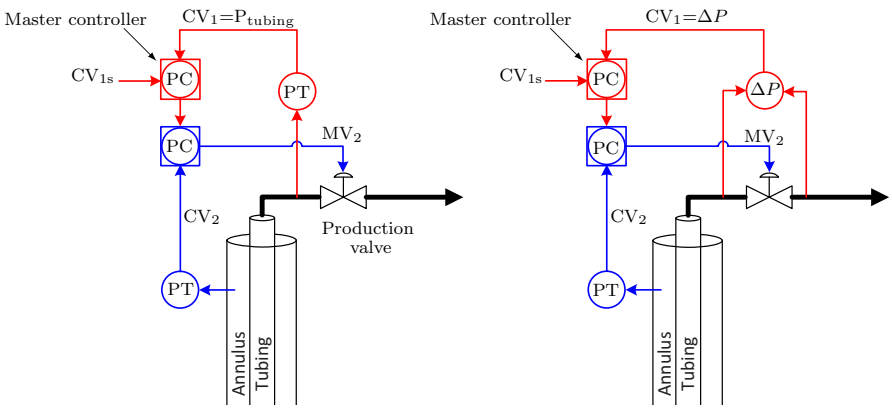


Fig. 6: Cascade control structures used for regulatory control of oil wells

Table 3: PI tuning and robustness measures of controllers. Initial operating point is $w_{\text{inj,A}} = 1$, $w_{\text{inj,B}} = 1$, $Z_{\text{wh,A}} = 0.5$, $Z_{\text{wh,B}} = 0.5$, $Z_{\text{top}} = 0.5$

design point	Controller	K_c	$T_i[s]$	GM*	PM*	DM [s]*	$\|S\|$**	$\|T\|$**
Initial operating point	$PC_{an,A}$	-2×10^{-5}	600	2.39	20.67	13.69	–	–
	$PC_{wh,A}$	0.016	100	3.85	66.19	2988	1.42	1.00
	$PC_{an,B}$	-2×10^{-5}	600	2.57	19.67	11.90	–	–
	$PC_{wh,B}$	0.016	100	3.90	66.83	3193	1.42	1.00
	PC_{inl}	-2×10^{-6}	300	3.86	53.24	377.87	–	–
	PC_{top}	0.4	300	–	105.02	1224	1.00	1.00
Optimal point	$PC_{an,A}$	-3×10^{-5}	600	3.32	32.33	20.56	–	–
	$PC_{wh,A}$	0.016	100	4.81	72.28	2926	1.26	1.00
	$PC_{an.B,}$	-3×10^{-5}	600	3.11	32.14	20.37	–	–
	$PC_{wh,B}$	0.016	100	4.66	72.03	2945	1.27	1.00
	PC_{inl}	-2×10^{-6}	300	6.49	80.65	10.39	–	–
	PC_{top}	0.4	300	–	100.90	1051	1.00	1.00

* larger is better ** smaller is better

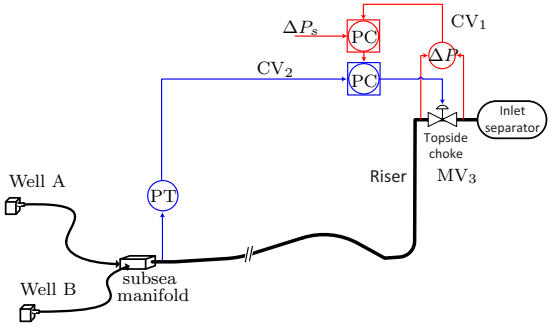


Fig. 7: Cascade control structure used for regulatory control to stabilize pipeline-riser subsystem (CS4)

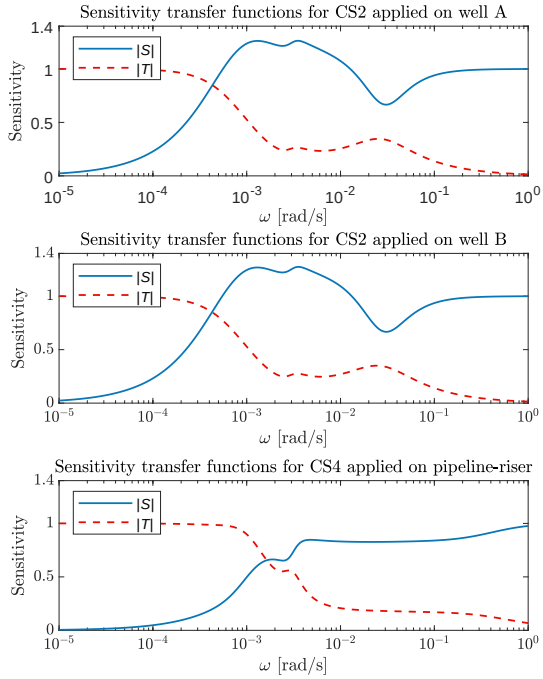


Fig. 9: Sensitivity transfer functions for regulatory control loops at the optimal operating point

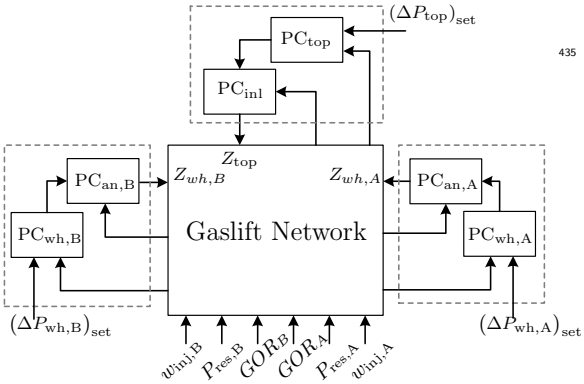


Fig. 8: Block diagram of the decentralized regulatory control structure used to stabilize the gaslift network

Figure 9 shows the sensitivity transfer functions $S = (I+GC)^{-1}$ and $T = I - S$ as functions of frequency at the optimal operating point. Here, S is the transfer function from a output disturbance to CV_1, and T is the transfer function from $\text{CV}_{1\text{s}}$ to CV_1. These are for the master controllers. The peaks of sensitivity transfer functions (M_s-value) are less that 1.4 which indicates good robustness (Skogestad and Grimholt, 2012).

5. Optimization layer

5.1. Economic nonlinear model predictive control

In this section, we apply nonlinear economic model predictive control (EMPC) as defined by the following optimization problem.

$$\min_{u} \left[\sum_{k}^{k+n_p} \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) + \sum_{k}^{k+n_p} \Delta \mathbf{u}_k^T R \Delta \mathbf{u}_k \right] \tag{3a}$$

subject to

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k) \tag{3b}$$

$$\underline{\phi} \leq \phi(\mathbf{x}_k, \mathbf{u}_k) \leq \overline{\phi} \tag{3c}$$

$$\underline{\mathbf{u}} \leq \mathbf{u}_k \leq \overline{\mathbf{u}} \tag{3d}$$

$$\underline{\Delta \mathbf{u}} \leq \Delta \mathbf{u}_k \leq \overline{\Delta \mathbf{u}} \tag{3e}$$

The first term in (3a) sums the economic cost J (2a) over the prediction horizon $\{t_k, t_{k+n_p}\}$ where $\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) =$ J, and the second summation term penalizes large control movements to regularize the problem and make the control action smooth. Note that this second term does not have any steady-state effect, because at steady-state the input change $\Delta\mathbf{u}_k$ is zero. The weight matrix R is a tuning parameter which is chosen as a diagonal matrix. The continuous-time model (1a) is discretized using the direct single shooting method (Biegler (2010)), and the resulting discretized nonlinear dynamic process model gives the equality constraints (3b). Additionally, general box constraints are imposed in (3c). Input constraints and rate of input constraints are imposed in (3d) and (3e), respectively. An extended Kalman filter is also implemented to give full state-feedback for the EMPC implementation.

5.2. Steady-state gradient control (Feedback RTO)

We also consider the recently proposed dynamic feedback based RTO approach (Krishnamoorthy et al., 2019), which is based on estimating the steady-state gradient using process measurements y_{meas} and a nonlinear model. A state estimation scheme is used to estimate the states $\hat{\mathbf{x}}$ and the unmeasured disturbances $\hat{\mathbf{d}}$. In this paper, for the sake of demonstration, we use an augmented extended Kalman filter (EKF) for combined state and parameter estimation, see Simon (2006) for a detailed description.

Once the states and unmeasured disturbances are estimated to get an updated nonlinear model (1a), the model is linearized to obtain a *local linear dynamic model* from the inputs $\mathbf{u}$ to the objective function J. This linear model is given in state-space form by the matrices A, B, C and D.

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{4a}$$

$$\mathbf{J} = C\mathbf{x} + D\mathbf{u} \tag{4b}$$

The steady-state gradients can then be estimated as follows (Krishnamoorthy et al., 2019).

$$\hat{\mathbf{J}}_{\mathbf{u}} = -CA^{-1}B + D \tag{5}$$

The process can be driven to its optimum by controlling the estimated steady-state gradient to a constant setpoint of zero using any feedback controller, for example a PI controller. The idea is illustrated in Fig. 10.

Note that the steady-state gradient is obtained from a dynamic model and not from the steady-state model as would be the conventional RTO approach (François et al., 2012). The use of a dynamic model means that we can use the transient measurements, and thus avoid the wait time for the process to reach steady-state, as required for conventional RTO.

5.3. Self-optimizing control (SOC)

We have in Section 3.5 found that the well head gas rate is a very promising self-optimizing variable (see Table 2). Therefore, in addition to economic MPC and the feedback RTO approach, we also study the performance of self-optimizing control, by keeping the wellhead gas rate constant at its nominal optimal value.

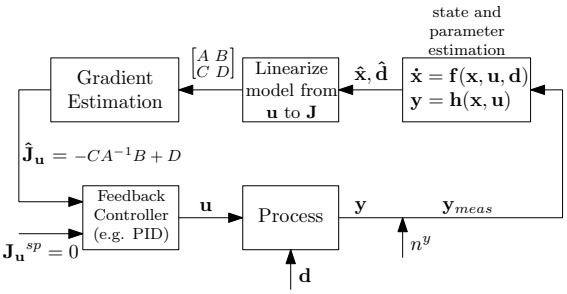


Fig. 10: Block diagram of the Dynamic Feedback-RTO method

6. Simulation results

In this section, we compare the three alternative optimization approaches by dynamic simulations.

6.1. Implementation and computation time

The Olga simulator is treated as the "real" process. The controllers (including the optimization) are implemented in Python using the simplified dynamic model of the process. The simplified dynamic model of the process is implemented in Modelica. The Modelica compiler generates a *functional mock-up unit* (FMU), which is a standard model component that can be shared with other applications. The resulting model was imported to CasADi (Andersson et al., 2019) which includes efficient automatic differentiation techniques. NMPC and EKF were implemented using the CasADi verion 2.0.0. The communication between the Olga Simulator and the controllers was done by OPC Data Access where the Olga OPC Server is a built-in module of the simulator and the OPC client is coded in Python.

The sampling interval for the state estimation using EKF is 10 sec. The control interval of the EMPC is set to 1 hour. Both the prediction and control horizon of EMPC are set to 16 hours. The dynamic model used for the EKF and the optimization includes 21 state variables and five inputs. Therefore, for a single shooting formulation of the EMPC, there are $5 \times 16 = 80$ optimization variables.

Each EMPC optimization takes about 2 minutes of computation time to solve which is significantly larger than the sampling interval of 10 sec. On the other hand, the Feedback-RTO computation time is less than the sampling interval. As a result, the feedback-RTO updates the optimal control settings (injection rates and pressure setpoints) at each sampling interval (10 sec).

Time scale separation is necessary for the regulatory controllers to settle to the new optimal setpoints before we perform a new re-optimization (Foss, 2012; Saputelli

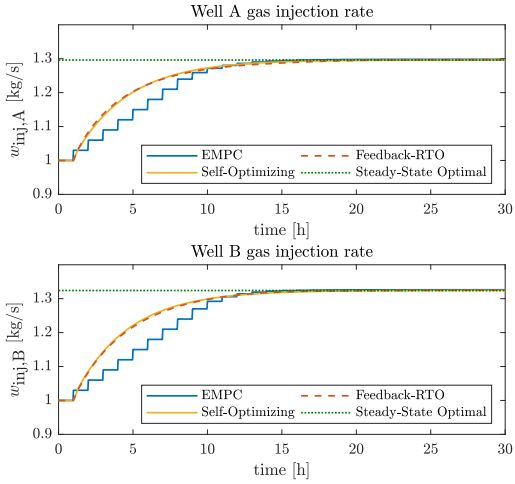


Fig. 11: Gas injection rates for nominal conditions

et al., 2006). One should notice that the time-scale separation concerns the closed-loop time constant of the upper control layer, not the sampling time interval of the data acquisition and the control signal. In general, a more frequent MV signal update leads to a smoother operation when the closed-loop time constant (adjusted by the controller gain) is kept constant.

6.2. Optimization at nominal conditions

Figures 11 and 12 compare the performance of EMPC, self-optimizing control and dynamic Feedback-RTO when taking the system from an initial operating point to the optimal steady-state with no disturbance.

We fine-tuned the simplified dynamic model so that its optimal operating point is very close to that of the Olga model. Therefore, the dynamic optimization converges to the same values as the steady-state RTO based on the surrogate model.

Fig. 11 shows the two gas injection rates and Fig. 12 shows the cost and two gradients. The gradients for EMPC and self-optimizing control are shown for comparison purposes, as the gradients are used for control only by Feedback-RTO. For self-optimizing control, we show in Fig. 13 the well-head gas flow rates which are the self-optimizing CV_1s. The optimal setpoints for the self-optimizing control are provided by the steady-state optimizer explained in Section 3.3. As expected, the estimated gradients for the self-optimizing control approach to zero as the gas flow rates settle to their optimal setpoints.

Figures 14, 15 and 16 show the optimal setpoints and the performance of the low-level pressure controllers for EMPC, self-optimizing control and Feedback-RTO, respectively. As discussed earlier, the Feedback-RTO updates the optimal settings more frequently, and the process response is smoother than with EMPC. The regulatory con-

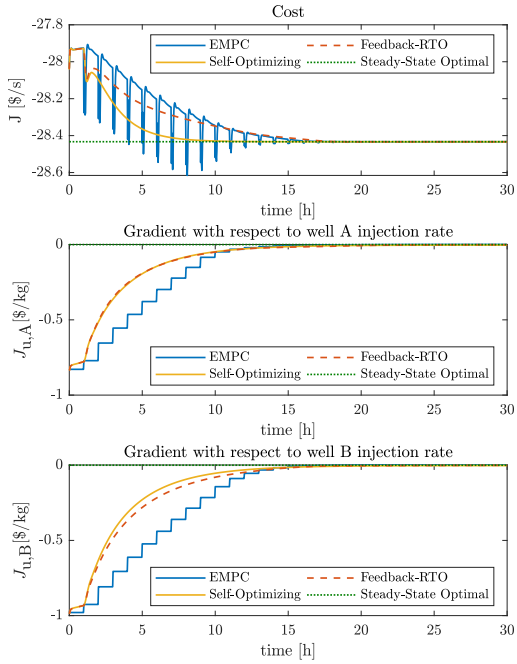


Fig. 12: Cost function and gradients for nominal conditions

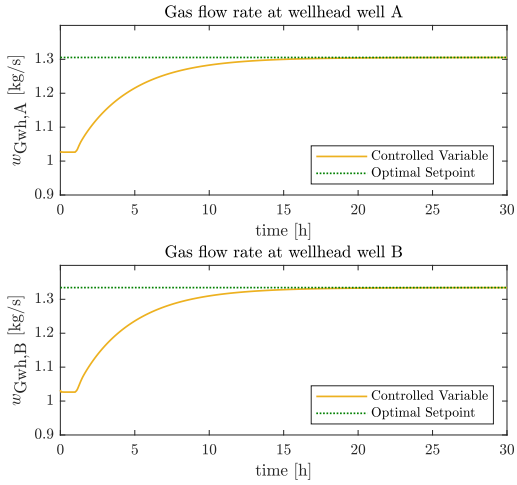


Fig. 13: Self-optimizing control: gas flow rates at well-heads (CV_1) at nominal conditions

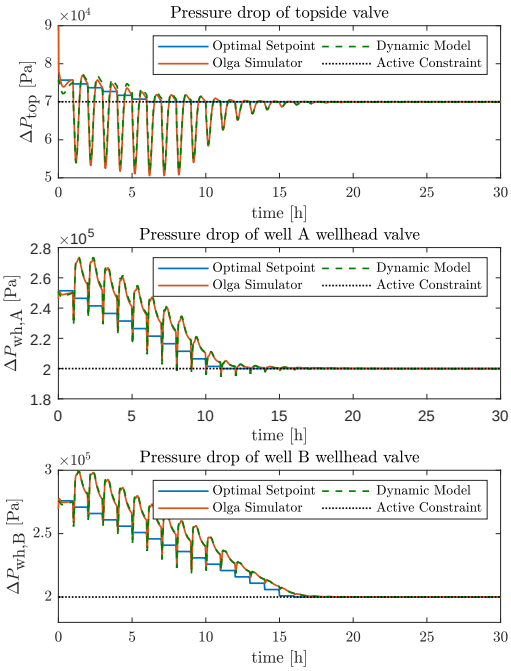


Fig. 14: EMPC: Optimal setpoints and performance of low-level pressure controllers

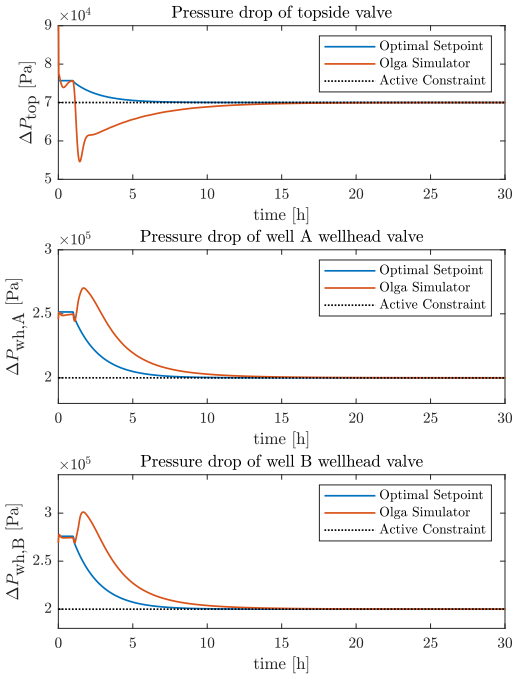


Fig. 15: Self-optimizing control: Performance of low-level pressure controllers

trollers for the self-optimizing control and Feedback-RTO show only one overshoot when the optimization is turned on (at $t = 1$ hour), whereas there is one overshoot for every 1 hour for EMPC (Fig. 14). The overshoots are because of the inverse response of the process to the step changes of the pressure setpoints and the gas injection rates with 1-hour intervals. The process with the ΔP outputs is *non-minimum phase*.

For the self-optimizing control, we have a decentralized control layer with five CVs and five MVs. The two unconstrained DOFs (gas injection rates) are used to control the self-optimizing CV_1s (gas flow rates at the well-heads shown in Fig. 13). The remaining DOFs are the three CV_1s setpoints used to control the three ΔP (CV_1) on their constraints (Fig. 15).

6.3. Optimization in presence of disturbances

The three optimizing control methods are next compared for two disturbances scenarios. The first scenario is a 5 bar decrease in the reservoir pressure of both wells (D1, D2), and the second disturbance scenario (D3, D4) is a 3 % increase the mass gas-oil ratio of the fluid coming from the reservoir. The disturbances are ramped up and down within 10 hours. we do not use step changes because they are not typical for a real process, and also because a large step change crashes the numerical simulation.

Figures 17 and 18 show the cost and loss from the ideal for the two disturbance scenarios, respectively. As expected, reduced reservoir pressures have a negative effect on the production and increases the cost, whereas increased gas-oil ratio decreases (improves) the cost.

Figures 19 and 20 show the corresponding gas injections. The increase in the gas-oil ratio has a larger effect than the reservoir pressure on the manipulated variables. The gas injection decreases significantly to compensate for the extra gas coming from the reservoir. The extra gas causes an increase in the fluid velocity and the friction in the well which leads to more pressure drop. Hence, the optimizer decreases the gas injection.

Both the EMPC and the Feedback-RTO rely on the joint state and parameter estimation by EKF. The disturbances in the reservoir pressure (D1, D2) and the gas oil ratio (D3, D4) are estimated as parameters. Figures 21 and 22 show the performances of EKF for estimating the disturbances when the well-head pressures and flow rates are used as the available measurements. The accuracy of this estimation depends on the model used by the EKF. As shown in the Figures 21 and 22, there are almost 10% estimation errors compared to the actual values of the Olga simulator.

The losses are compared in Table 4. For self-optimizing control, we use the variable that is found in Table 2, namely, the gas mass flow rates at wellheads. The loss values in

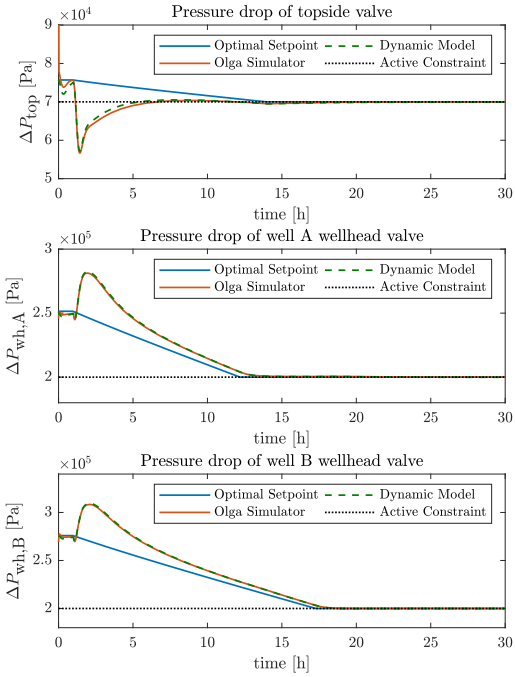


Fig. 16: Feedback RTO: Optimal setpoints and performance of low-level pressure controllers

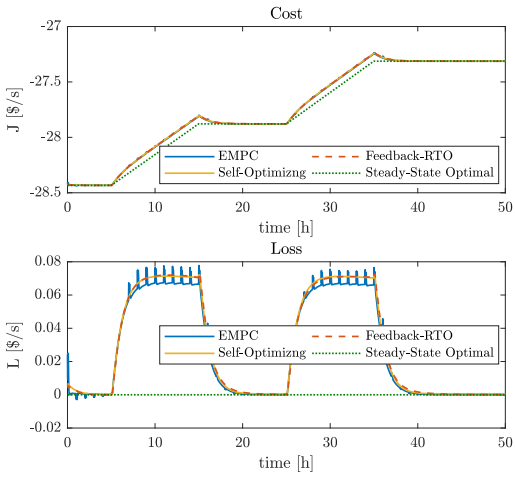


Fig. 17: Cost and loss from ideal for disturbance in reservoir pressures (D1, D2)

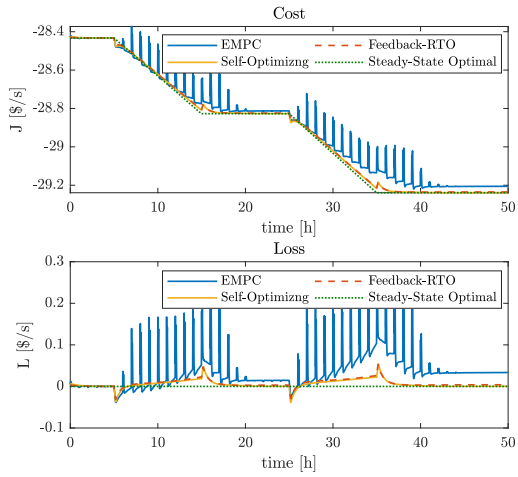


Fig. 18: Cost and loss from ideal for disturbance in gas-oil ratio (D3, D4)

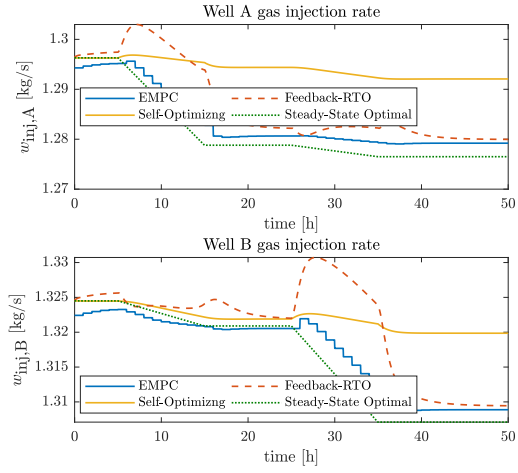


Fig. 19: Optimal gas injection for disturbance in reservoir pressures (D1, D2)

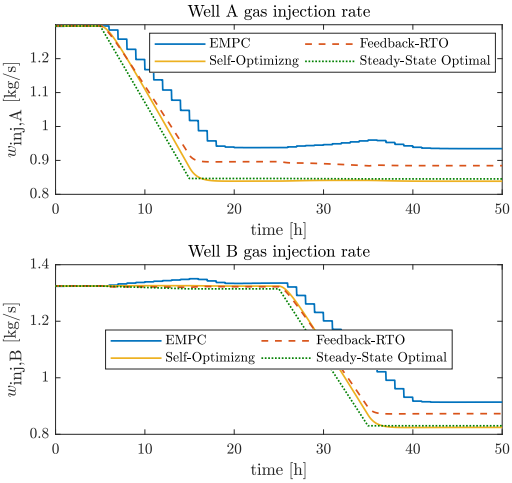


Fig. 20: Optimal gas injection for disturbance in gas-oil ratio (D3, D4)

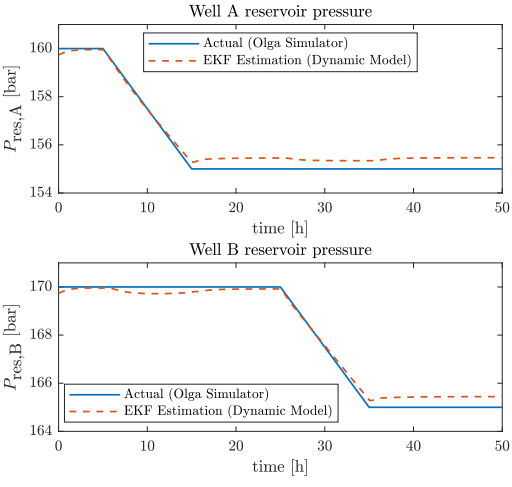


Fig. 21: Performance of EKF for estimating the disturbances in reservoir pressure (D1, D2) by measuring well-head pressures and flow rates

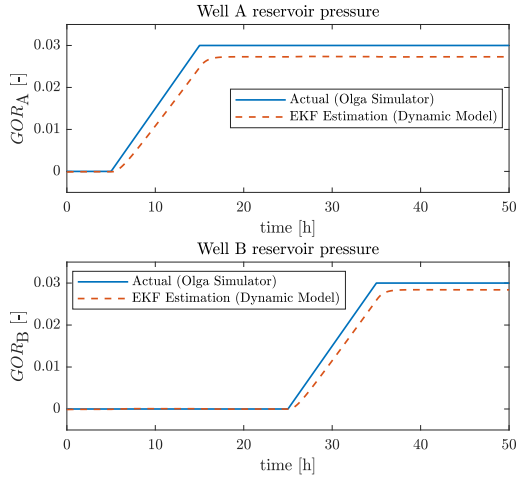


Fig. 22: Performance of EKF for estimating the disturbances in gas-oil ratio (D3, D4) by measuring well-head pressures and flow rates

Table 4 show that for the disturbances in the reservoir pressures (D1, D2), none of the controllers have any significant advantage over the open-loop (gas injections kept at their nominal optimal values). However, this is dependent on the regulatory layer control design. The losses with the self-optimizing control are the lowest overall, and the loss values of the Feedback-RTO are lower than those for the Economic NMPC for the disturbances in the gas-oil ratios (D3, D4). Since the EMPC and Feedback-RTO are based on the same dynamic model and the same EKF, this is a surprising result that is discussed in the next section. Regardless of the dynamics, we expected they settle to the same steady-state.

7. Discussion

7.1. Effect of regulatory controllers on optimization

For the disturbance in the reservoir pressure (D1, D2), we do not obtain any significant benefit from the optimizing control over the open-loop situation (*i.e.*, the gas injection rates are kept at their nominal optimal values). However, it depends on the control structure of the regulatory layer. In this work, we chose to control the pressure drop of the wellhead valves because their constraints are active, and in practice, they will be controlled on a constant setpoint. Hence, it becomes easier to apply the self-optimizing control. If we control the wellhead pressures (CS1) instead of the pressure drops (CS2), we will get poor results for the disturbances in the reservoir pressures. Obviously, it is not optimal to keep the wellhead pressure constant when the reservoir pressure decreases.

The Feedback-RTO gives lower losses than the EMPC (Table 4), although they are designed based on the same dynamic model. The reason is the smoother behavior of

Table 4: Loss from optimal for different control designs and different disturbances

	Disturbances				
	Nominal $J_{opt} = -28.433$	D1 $J_{opt} = -27.878$	D2 $J_{opt} = -27.313$	D3 $J_{opt} = -28.827$	D4 $J_{opt} = -29.239$
Open-loop	0.000	0.001	0.001	0.274	0.480
Self-Optimizing Control	0.000	0.001	0.001	0.000	0.000
Economic NMPC	0.000	0.000	0.000	0.015	0.033
Dynamic Feedback-RTO	0.000	0.000	0.000	0.003	0.004

the regulatory controllers for the Feedback-RTO simulation. The inverse responses of the process to the setpoint changes decreases the control performance and prevents the controllers from tracking the optimal setpoints. Therefore, the limitation of the regulatory control performance affects the optimization. As an experiment, we decreased the control interval of the EMPC from 1 hour to 20 minutes so that the regulatory controller have a smoother operation to track the optimal setpoints. As a result, the loss values of EMPC decreased by about 40%. However, by reducing the control interval to one third, the number of optimization variables increases to three times (240 variables), and the computation time for optimization increases from 2 minutes to 18-20 minutes.

7.2. Importance of robust regulatory control

Dealing with an unstable plant is a challenge for long-term dynamic optimization. Explicitly considering the regulatory controllers in the model for dynamic optimization seems necessary, for accuracy and completeness. However, this approach requires proper tuning and possibly gain-scheduling for robustness. If the system becomes unstable, the optimization problem cannot be solved because of invalid Jacobians. This is related to the fact that when the system becomes unstable, the inputs saturate that is $\Delta u = 0$, and note that elements of the Jacobians matrix are $(\delta y_i)/(\delta u_j)$.

Excluding the regulatory dynamics from the optimization is possible only when the two control layers operate at completely different time scales, such that the regulatory dynamics do not affect the optimization. However, this is not the case for our system.

7.3. Consideration of active constraints in MPC

As explained above, reducing the control interval of EMPC increases the number of the optimization variables. We know that three of the input constraints (related to pressure drop setpoints) are active at the optimal point. We used this information for the disturbance rejection simulations and removed these three variables from the optimization problem by replacing the inequality constraints by equality constraints. This reduces the number of optimization variables from 240 to 96, saving a significant amount of computation time.

7.4. Computation time for steady-state models

To obtain the data for the surrogate spline models, we run each simulation for 20 hours to reach the steady state. It takes about 15 minutes to finish each simulation on a laptop with a quad-core CPU running at 3.7 GHz. It took about five days to run 420 simulations and generate the surface shown in Fig. 5.

By using surrogate models, it is possible to incorporate the commercially available simulation models (*e.g.*, Olga, LedaFlow, K-Spice) into the optimization problem formulation. However, constructing the surrogate models based on simulation data involves extensive offline computations to obtain the steady-state data. Nevertheless, this approach is becoming a viable solution with the availability of faster computers and using cloud computing with high (parallel) processing power.

8. Conclusion

To our knowledge, this is the first publication on the complete control structure design (including both the regulatory and the optimization layer) applied to an oil production network and tested on the Olga simulator.

We compared three approaches for optimization layer. The dynamic Feedback-RTO and self-optimizing control are able to steer the operation to the optimal point more smoothly compared to economic MPC (EMPC). In case of unknown disturbances, self-optimizing control results in the lowest loss compared to EMPC and Feedback-RTO.

We conclude that the self-optimizing control is the most practical method for our case study. However, this depends on the right choice of the controlled variables and the control structures throughout the control hierarchy. In this way, we found the gas mass flow rate at the well-head as the best controlled variable for self-optimizing control. Also, controlling the pressure drop over the valves (*i.e.* supervisory layer) is necessary to control the process on the active constraints.

References

Aamo, O., Eikrem, G., Siahaan, H., Foss, B., 2005. Observer design for multiphase flow in vertical pipes with gas-lift - theory and experiments. Journal of Process Control 15, 247 – 257.

Andersson, J., Gillis, J., Horn, G., Rawlings, J., Diehl, M., 2019. Casadi — a software framework for nonlinear optimization and optimal control. Mathematical Programming Computation 11.

Bendiksen, K.H., Malnes, D., Moe, R., Nuland, S., 1991. Dynamic two-fluid model olga. theory and application. SPE Production Engineering 6, 171–180.

Biegler, L.T., 2010. Nonlinear programming: concepts, algorithms, and applications to chemical processes. SIAM.

Camponogara, E., Plucenio, A., Teixeira, A.F., Campos, S.R.V., 2009. An automation system for gas-lifted oil wells: Model identification, control, and optimization. Journal of Petroleum Science and Engineering 70, 157–167. doi:10.1016/j.petrol.2009.11.003.

Campos, M., Meien, O., Neto, S., Stender, A., Takahashi, T., Ashikawa, F., 2015. Anti-slug advanced control for offshore production platforms. OTC Brasil doi:10.4043/26243-ms.

Codas, A., Campos, S., Camponogara, E., Gunnerud, V., Sunjerga, S., 2012. Integrated production optimization of oil fields with pressure and routing constraints: The Urucu field. Computers & Chemical Engineering 46, 178–189. doi:10.1016/j.compchemeng.2012.06.016.

Codas, A., Jahanshahi, E., Foss, B., 2016. A two-layer structure for stabilization and optimization of an oil gathering network. IFAC-PapersOnLine 49, 931 – 936. doi:https://doi.org/10.1016/j.ifacol.2016.07.317. 11th IFAC Symposium on Dynamics and Control of Process SystemsIncluding Biosystems DYCOPS-CAB 2016.

Dalsmo, M., Halvorsen, E., Slupphaug, O., 2002. Active feedback control of unstable wells at the brage field, in: SPE Annual Technical Conference and Exhibition, Society of Petroleum Engineers (SPE). doi:10.2118/77650-ms. *SPE no.* 77650.

Di Meglio, F., Petit, N., Alstad, V., Kaasa, G.O., 2012. Stabilization of slugging in oil production facilities with or without upstream pressure sensors. Journal of Process Control 22, 809 – 822. doi:10.1016/j.jprocont.2012.02.014.

Engell, S., 2007. Feedback control for optimal process operation. Journal of Process Control 17, 203 – 219. doi:10.1016/j.jprocont.2006.10.011. special Issue {ADCHEM} 2006 Symposium.

Foss, B., 2012. Process control in conventional oil and gas fields — challenges and opportunities. Control Engineering Practice 20, 1058 – 1064. doi:10.1016/j.conengprac.2011.11.009. 4th Symposium on Advanced Control of Industrial Processes (ADCONIP).

François, G., Srinivasan, B., Bonvin, D., 2012. Comparison of six implicit real-time optimization schemes. Journal Européen des Systèmes Automatisés 46, 291–305.

Grimstad, B., Foss, B., Heddle, R., Woodman, M., 2016. Global optimization of multiphase flow networks using spline surrogate models. Computers & Chemical Engineering 84, 237–254. doi:10.1016/j.compchemeng.2015.08.022.

Gunnerud, V., Foss, B., 2010. Oil production optimization—a piecewise linear model, solved with two decomposition strategies. Computers & Chemical Engineering 34, 1803 – 1812. doi:10.1016/j.compchemeng.2009.10.019.

Jahanshahi, E., Grimstad, B., Foss, B., 2016. Spline fluid models for optimization. IFAC-PapersOnLine 49, 400–405. doi:https://doi.org/10.1016/j.ifacol.2016.07.374. 11th IFAC Symposium on Dynamics and Control of Process SystemsIncluding Biosystems DYCOPS-CAB 2016.

Jahanshahi, E., Skogestad, S., 2014. Simplified dynamic models for control of riser slugging in offshore oil production. SPE Oil and Gas Facilities 3, 64–79.

Jahanshahi, E., Skogestad, S., Hansen, H., 2012. Control structure design for stabilizing unstable gas-lift oil wells, in: Advanced Control of Chemical Processes, Singapore. pp. 93–100.

Jansen, J.D., Brouwer, R., Douma, S., 2009. Closed loop reservoir management, in: Proceedings of SPE Reservoir Simulation Symposium, Society of Petroleum Engineers, The Woodlands, Texas, USA. doi:10.2118/119098-MS.

Kosmidis, V.D., Perkins, J.D., Pistikopoulos, E.N., 2005. A mixed integer optimization formulation for the well scheduling problem on petroleum fields. Computers & Chemical Engineering 29, 1523–1541. doi:10.1016/j.compchemeng.2004.12.003.

Krishnamoorthy, D., Foss, B., Skogestad, S., 2016. Real-time optimization under uncertainty applied to a gas lifted well network. Processes 4. doi:10.3390/pr4040052.

Krishnamoorthy, D., Jahanshahi, E., Skogestad, S., 2019. Feedback real-time optimization strategy using a novel steady-state gradient estimate and transient measurements. Industrial & Engineering Chemistry Research 58, 207–216. doi:10.1021/acs.iecr.8b03137.

Luyben, M.L., Tyreus, B.D., Luyben, W.L., 1997. Plantwide control design procedure. AIChE Journal 43, 3161–3174. doi:10.1002/aic.690431205.

Saputelli, L., Nikolaou, M., Economides, M.J., 2006. Real-time reservoir management: A multiscale adaptive optimization and control approach. Computational Geosciences 10, 61–96. doi:10.1007/s10596-005-9011-5.

Schlumberger, 2018. Olga dynamic multiphase flow simulator. URL https://www.software.slb.com/products/olga. Accessed: 2018-12-26.

Simon, D., 2006. Optimal State Estimation, Kalman, H and Nonlinear Approches. Wiley-Interscience, Hoboken, New Jersey.

Skogestad, S., 2004. Control structure design for complete chemical plants. Computers & Chemical Engineering 28, 219 – 234. doi:10.1016/j.compchemeng.2003.08.002. escape 12.

Skogestad, S., Grimholt, C., 2012. The SIMC Method for Smooth PID Controller Tuning. Springer, London. chapter 5. pp. 147–175. doi:10.1007/978-1-4471-2425-2_5.

Willersrud, A., Imsland, L., Hauger, S.O., Kittilsen, P.l., 2013. Short-term production optimization of offshore oil and gas production using nonlinear model predictive control. Journal of Process Control 23, 215–223. doi:10.1016/j.jprocont.2012.08.005.

Appendix H

Open-loop versus closed-loop optimization

In Chapter 7, we considered the optimization problem formulation under uncertainty, where we provided some useful discussions on open-loop optimization and closed-loop optimization. In this appendix, we provide some more detailed discussions that complements Chapter 7.

Model predictive control differs from conventional control, in the sense that it does not use a pre-computed control law $u = \kappa(x)$. Instead, MPC obtains a set of control actions by solving an open-loop optimal control problem for the current initial state. The first control action is implemented on the plant and the OCP is re-optimized for the new system state at each sampling time. This is an effective implementation of the dynamic programming (DP) solution, since the optimal control u obtained by the MPC in the deterministic case, satisfies $u = \kappa(x)$ for the current state x, whereas DP solves a feedback version of the same OCP, yielding the receding horizon control law $\kappa(\cdot)$ that can be used for any state [144].

In the absence of uncertainty, the solution obtained by the open-loop control is equivalent to the feedback version of the OCP. However, in the presence of uncertainty, closed-loop optimization is far superior to open-loop optimization. We will now demonstrate this using a simple example, and show how the multistage MPC problem approximates the DP solution.

H.1 Dynamic programming solution

We first provide a brief introduction to dynamic programming (DP) to solve a finite horizon linear quadratic problem of the form,

$$\begin{aligned} min \sum_{k=0}^{N-1} \ell(x(k), u(k)) + \ell_N(x(N)) \\ \text{s.t. } x(k+1) = Ax(k) + Bx(k) \end{aligned} \tag{H.1}$$

where the stage cost is defined as $\ell(x, u) := 0.5(x^\mathsf{T} Q x + u^\mathsf{T} R u)$ and the terminal stage cost $\ell_N(x(N)) := 0.5 x^\mathsf{T} P_N x$. This is solved using the backward DP, where

the problem is first solved at the last stage, and then move to the second last stage of the DP recursion, and then to the next stage and so on and so forth, as clearly explained in [144]. The optimal control policy at each stage is given by,

$$u_k(x) = K(k)x \qquad k = N-1, N-2, \ldots, 0 \tag{H.2}$$

with the optimal gain $K(k)$

$$K(k) = -(B^\mathsf{T}\Pi(k+1)B + R)^{-1}B^\mathsf{T}\Pi(k+1)A \qquad k = N-1, N-2, \ldots, 0 \tag{H.3}$$

and the Riccati matrix

$$\Pi(k+1) = Q + A^\mathsf{T}\Pi(k)A - A^\mathsf{T}\Pi(k)B(B^\mathsf{T}\Pi(k)B + R)^{-1}B^\mathsf{T}\Pi(k)A \tag{H.4}$$
$$k = N, N-1, \ldots, 1$$

with the terminal condition $\Pi(N) = P_N$.

H.2 Illustrative example

Consider a linear dynamic system with an additive disturbance w

$$x(k+1) = x(k) + u(k) + w(k)$$

where w is known to lie in the compact set $[-1, 1]$. The objective is to solve the finite horizon OCP problem with the stage cost $\ell(x, u) := 0.5(x^2 + u^2)$ and the terminal stage cost $0.5x^2$ [1].

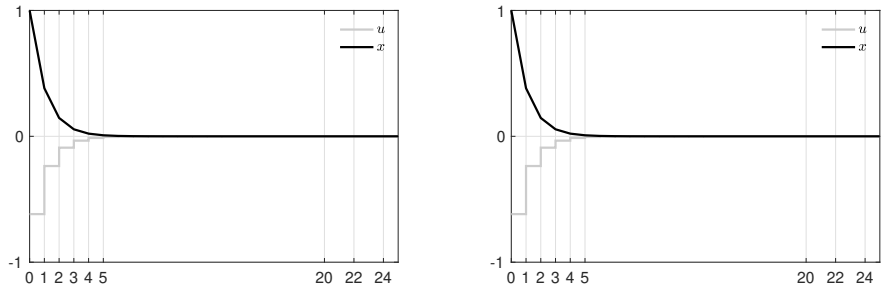
No uncertainty First let us consider the case with no uncertainty, i.e. $w = 0$[2] . In this case, the open-loop optimal control and state sequence $\mathbf{u}^0(x(0))$ and $\mathbf{x}^0(x(0))$ with the initial condition $x(0) = 1$ for $N = 25$ are shown in Fig. H.1a. The closed-loop/feedback optimization using DP recursions (H.3) and (H.4), yields a feedback policy

$$\boldsymbol{\mu}^0 := (\mu_0^0(\cdot), \mu_1^0(\cdot), \ldots, \mu_{N-1}^0(\cdot))$$

which is a sequence of control laws, rather than a sequence of control actions. For the state at time i, the control is given by $\mu_i^0(x_i)$, which in the deterministic case is simply $u(i)$. The DP solution is shown in Fig. H.1b, which is identical to the open-loop optimal trajectories in Fig. H.1a.

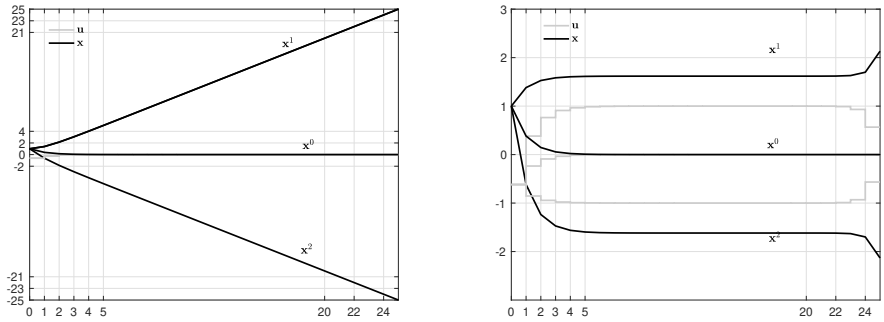
With uncertainty However, in the presence of uncertainty, if we apply the same sequence of open-loop optimal *control actions* $\mathbf{u}^0$ determined earlier, it does not yield the same result as applying the sequence of feedback *control policies* $\boldsymbol{\mu}^0$ determined using DP recursions. We now compare the results for three different realizations of the uncertainty: $\mathbf{w}^0 := \mathbf{0}^N$, $\mathbf{w}^1 := \mathbb{1}^N$, and $\mathbf{w}^2 := -\mathbb{1}^N$. The corresponding state trajectories are denoted by $\mathbf{x}^0$, $\mathbf{x}^1$ and $\mathbf{x}^2$ respectively. Fig. H.2a shows the open-loop state trajectories when applying sequence of control actions $\mathbf{u}^0$,

[1] This is the same illustrative example used in [144].
[2] The notation $(\cdot)^0$ denotes the nominal case.



(a) (b)

Figure H.1: No uncertainty: (a) Open-loop optimization (b) Closed-loop optimization computed using DP recursion.



(a) (b)

Figure H.2: With uncertainty: (a) Open-loop optimization (b) Closed-loop optimization computed using DP recursion.

and Fig. H.2b shows the feedback state trajectories when applying the sequence of control laws $\boldsymbol{\mu}^0$. It can be seen that the *spread* of the trajectories $|x^2(N) - x^1(N)| = 2N$ in the case of open-loop optimization, whereas for the closed-loop optimization case using DP recursion, $|x^2(N) - x^1(N)| \to 3.24$ as $N \to \infty$, which was also demonstrated in [144].

The obvious conclusion from this is that, in the presence of uncertainty, optimizing over control policies (closed-loop optimization) is better than optimizing over control actions (open-loop optimization). However, solving a closed-loop optimization problem using DP recursions is rather complex.

H.3 Comparison of multistage MPC with DP

In chapter 7, we presented the multistage MPC approach, which is a closed-loop optimization approach that explicitly takes into account the feedback. In this sec-

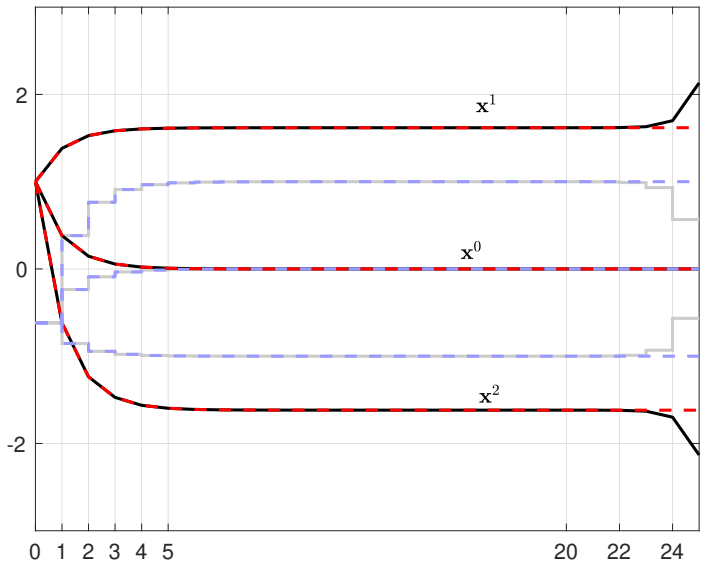


Figure H.3: Comparison of state trajectories (solid black) and control trajectories (solid gray) from DP and state trajectories (dashed red) and control trajectories (dashed blue) from multistage MPC for $\mathbf{w}^0 := \mathbf{0}^N$, $\mathbf{w}^1 := \mathbb{1}^N$, and $\mathbf{w}^2 := -\mathbb{1}^N$.

tion, we now compare the solution of the multistage MPC with that of the DP solution, using the same example as above.

We solve the multistage MPC problem with three discrete realizations of the uncertainty $\{-1, 0, 1\}$ and a robust horizon of $N_r = 1$, leading to 3 discrete scenarios, as described in chapter 8. The non-anticipativity constraints ensures that the first control input is the same, which is implemented on the plant in a receding horizon fashion (cf. chapter 9). Fig. H.3 shows the results of the multistage MPC (dashed lines) implemented in a receding horizon fashion, along with the DP solution computed earlier, where it can be seen that the multistage MPC implemented in a receding horizon fashion (even with robust horizon = 1) approximates the DP solution.

To summarize,

- **No uncertainty**: Open-loop optimization with closed-loop implementation (conventional MPC) approximates the DP solution.
- **With uncertainty**: Closed-loop optimization with closed-loop implementation (e.g. multistage MPC) approximates the DP solution.

Appendix I

Primal decomposition

Consider the optimization problem

$$\min_{x_i} \sum_{i=1}^{S} \ell_i(x_i) \tag{I.1a}$$

$$s.t.$$

$$\sum_{i=1}^{S} g_i(x_i) = 0 \tag{I.1b}$$

$$x_i \in \mathcal{X}_i \quad \forall i \in \{1, \ldots, S\} \tag{I.1c}$$

(I.1) can be decomposed using primal decomposition by introducing auxiliary variables t_i, where each subproblem is posed as,

$$\mathbf{\Phi}(t_i) := \min_{x_i} \ \ell_i(x_i) \tag{I.2a}$$

$$s.t.$$

$$g_i(x_i) = t_i \tag{I.2b}$$

$$x_i \in \mathcal{X}_i \tag{I.2c}$$

for a given t_i. Clearly, original problem (I.1) is equivalent to the following problem, which is known as the master problem,

$$\min_{t_i} \sum_{i=1}^{S} \mathbf{\Phi}(t_i) \tag{I.3a}$$

$$s.t. \sum_{i=1}^{S} t_i = 0 \tag{I.3b}$$

The simplest way to find a solution to the master problem is by using the gradient descent method (also known as subgradient method [20]), where a step is taken along the descent direction σ_i

$$t_i^{(v+1)} = t_i^{(v)} + \alpha\sigma_i \qquad \forall i$$

with a suitable step size α. It can be seen that the descent direction of the master problem is given by the gradients of the subproblem.

The Lagrangian of the subproblems can be written as

$$\mathbf{\Phi}(t_i) := \min_{x_i} \ \ell_i(x_i) + \lambda_i^{\mathsf{T}}(g_i(x_i) - t_i) \tag{I.4a}$$

$$s.t.\ x_i \in \mathcal{X}_i \tag{I.4b}$$

Consequently,

$$\sigma_i := \nabla_{t_i} \mathbf{\Phi}(t_i) = -\lambda_i$$

For the descent direction to be feasible $\sum_{i=1}^{S}(t_i) = 0$ [111]. This can be ensured by introducing only $S-1$ auxiliary variables instead of S auxiliary variables and the last auxiliary variable is simply set as

$$t_S = -\sum_{i=1}^{S-1} t_i \tag{I.5}$$

The gradient descent problem is then given as

$$t_i^{(v+1)} = t_i^{(v)} + \alpha\sigma_i \qquad \forall i = 1, \ldots, S-1$$

with $t_S(v) = -\sum_{i=1}^{S-1} t_i(v)$.

We now show that the problem (I.1) can be equivalently solved using primal decomposition.

Assumption I.1: For $i = 1, \ldots, S$, we assume that

- $\mathcal{X}_i$ is nonempty, compact and convex
- $\ell_i(x_i)$ is convex and differentiable on $\mathcal{X}_i$
- $g_i(x_i)$ is convex and differentiable on $\mathcal{X}_i$
- (I.1) has a feasible solution denoted by $\overline{x}^*$

Theorem I.1. *Given assumption I.1, if t_i^* is the minimizer of the master problem (I.3), and $x_i^*(t_i)$ is the minimizer of the i^{th} subproblem (I.2), then $x^* := [x_1^*(t_1^*), \ldots, x_S^*(t_S^*)]$ is the minimizer of the original problem (I.1).*

Proof. Since (I.1) is equivalent to (I.3), feasibility of (I.1) implies that (I.3) is feasible. Updating t_i iteratively using gradient descent method moves t_i in a direction of improvement of the overall objective [19, 111] and (I.5) ensures that the descent direction is feasible. For a differentiable system, it can be shown by using the gradient descent algorithm [12]

$$\lim_{v\to\infty} t_i \to t_i^*$$

Moreover continuity of ℓ_i and g_i, along with the compactness of $\mathcal{X}_i$ imply that the subproblems have an optimal solution whenever it is feasible [111]. Therefore,

$$[x_1^*(t_1^*), \ldots, x_S^*(t_S^*)] = \overline{x}^*$$

□

Appendix J

Distributed Multistage NMPC applied to oil production optimization

This appendix contains the paper showing the application of the distributed scenario NMPC proposed in Chapter 9 to an oil production optimization problem.

- Paper published in 2018 IFAC International Symposium on Advanced Control of Chemical Processes (ADCHEM), Shenyang, China.

This paper was selected as a Keynote presentation and was a finalist for the IFAC Young Author award.

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-18 (2018) 351–356

IFAC Papers Online CONFERENCE PAPER ARCHIVE

A Distributed Algorithm for Scenario-based Model Predictive Control using Primal Decomposition ⋆

Dinesh Krishnamoorthy * **Bjarne Foss** ** **Sigurd Skogestad** *

* *Dept. of Chemical Engineering, Norwegian Univ. of Science & Technology (NTNU) , NO-7491 Trondheim , (e-mail: dinesh.krishnamoorthy@ntnu.no, skoge@ntnu.no).*
** *Dept. of Engineering Cybernetics, Norwegian Univ. of Science & Technology (NTNU), NO-7491 Trondheim , (e-mail: bjarne.foss@ntnu.no)*

Abstract: In this paper, we consider the decomposition of scenario-based model predictive control problem. Scenario MPC explicitly considers the concept of recourse by representing the evolution of uncertainty by a discrete scenario tree, which can result in large optimization problems. Due to the inherent nature of the scenario tree, the problem can be decomposed into each scenario. The different subproblems are only coupled via the non-anticipativity constraints which ensures that the first control input is the same for all the scenarios. This constraint is relaxed in the dual decomposition approaches, which may lead to infeasibility of the non-anticipativity constraints if the master problem does not converge within the required time. In this paper, we present an alternative approach using primal decomposition which ensures feasibility of the non-anticipativity constraints throughout the iterations. The proposed method is demonstrated using gas-lift optimization as case study.

Keywords: Scenario Optimization, Primal decomposition, Uncertainty, Distributed optimization

1. INTRODUCTION

Model predictive control (MPC) has proven to be a highly successful control methodology in the process control industry due to its ability to handle large and complex multivariable systems, subject to process and operating constraints. MPC typically uses models that represents the system and computes an optimal input trajectory based on model predictions in order to minimize a certain cost function over the prediction horizon. Recently, there has been an increasing trend in the use of *Economic NMPC*, where the economic objectives are incorporated into the MPC problem.

The presence of plant-model mismatch or process variations can easily lead to constraint violations or suboptimal operation. Different approaches have been proposed in the literature to handle uncertainty in the MPC problem, such as min-max MPC (Campo and Morari, 1987), which computes an optimal input trajectory that minimizes the cost of the worst-case realization of the uncertainty. This, however, leads to a very conservative solution, since the optimization is performed in an open-loop fashion. It ignores the fact that new information will be available and a new control trajectory will be re-computed in the future. In other words, min-max MPC ignores one of the important aspect of uncertainty handling, namely, *feedback*. Feedback min-max MPC scheme was proposed by Scokaert and Mayne (1998) to overcome the limitations of the open-loop min-max MPC. Feedback min-max MPC is a closed-loop optimization scheme, where the notion of feedback is explicitly taken into account by optimizing over different control policies rather than a single control trajectory by representing the evolution of the uncertainty by a scenario tree. This approach was later studied in detail for nonlinear systems in the context of multistage NMPC problem and was shown to reduce the conservativeness at the cost of computational time (Lucia et al., 2013a).

One of the main challenges of this method is that the computational size of the problem grows exponentially with 1) length of the prediction horizon, 2) number of uncertain parameters and disturbances and 3) number of discrete models for each uncertain variable that is considered in generating the different scenarios. This poses a challenge for real-time implementation, despite advancements in computational power and efficient numerical solvers.

One solution to this problem is to stop the branching after a certain number of samples in the prediction horizon (known as robust horizon) in order to curb the number of scenarios as described in Lucia et al. (2013a). Another solution is to exploit the fact that each scenario can be written as an independent subproblem except for the so-called non-anticipativity constraints. Hence decomposition methods can be employed by solving the subproblems

⋆ The authors gratefully acknowledge the financial support from SUBPRO, which is financed by the Research Council of Norway, major industry partners and NTNU.

10.1016/j.ifacol.2018.09.325

independently and later use a master problem to coordinate the individual subproblems iteratively.

Scenario decomposition using dual decomposition was proposed by Lucia et al. (2013b) and Martí et al. (2015). Dual decomposition (also known as Lagrangian decomposition) method solves the subproblems by relaxing the coupling constraints. A master problem then co-ordinates the individual subproblems iteratively. The previously relaxed constraints are feasible only upon convergence. Martí et al. (2015) indicates that such methods require a relatively large number of iterations between the master problem and the subproblem to converge, leading to challenges with practical implementation. The use of augmented lagrangian methods can help improve the convergence properties, however this makes the problem non-separable (Boyd et al., 2011).

The risk of dual decomposition is then that the master problem may not converge within the required time. This leads to infeasibility of the non-anticipativity constraints, the implications of which are that the different subproblems may give different control inputs at the first sample time in the prediction horizon. This is not acceptable for real-time closed-loop implementation. In this paper, we propose an alternative approach to scenario decomposition using the primal decomposition approach which ensures the non-anticipativity constraints are always feasible. This is because, in contrast to dual decomposition, primal decomposition produces a primal feasible solution with monotonically decreasing objective value at each iteration.

The key challenge in any real-time implementation of optimizing controllers such as MPC is clearly, how best to deal with time. Quoting Kerrigan et al. (2015), "*The correctness of a computation is a function of time*". The late-arrival of a solution in many cases may simply not be acceptable. In real-time optimization, *approximate solution now is better than an accurate solution tomorrow.* This strategy is adopted in many optimization algorithms (Kerrigan et al., 2015). This is also the motivation to use primal decomposition as opposed to dual decomposition for the scenario MPC problem.

The paper is organized as follows. The framework of scenario MPC is introduced in section 2. The decomposition algorithm is presented in section 3. The proposed methodology is verified using a case study in section 4 before concluding the paper in section 5.

2. SCENARIO MPC

Consider a discrete-time nonlinear system of the form,

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

where, $\mathbf{x}_k \in \mathbb{R}^{n_x}$ denotes the state vector at time step k, $\mathbf{u}_k \in \mathbb{R}^{n_u}$ is the vector of control inputs and $\mathbf{d}_k \in \mathbb{R}^{n_d}$ represents the uncertain parameters and disturbances. Let us assume that the uncertainty belongs to a known distribution such that $\mathbf{d}_k \in \mathcal{U}\ \forall k$.

If the model (1) is *perfect*, the predicted state trajectory is given by $\mathbf{x}_{[k,k+N]}$ for the open-loop implementation of the corresponding input trajectory $\mathbf{u}_{[k,k+N-1]}$ over the prediction horizon $[k, k+N]$. However, in the presence of plant-model mismatch, $\mathbf{u}_{[k,k+N-1]}$ must be associated

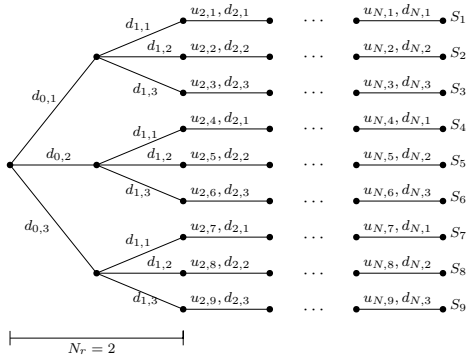


Fig. 1. Scenario Tree for $M = 3$ and $N_r = 2$.

with a cone of state trajectories $\{\mathbf{x}_{[k,k+N]}\}_{\mathcal{U}}$ depending on the realization of the uncertain variables (Guay et al., 2015). Optimizing over a single control trajectory (open-loop optimization) disregards feedback. In other words, it disregards the fact that new information will be available in the future and the control trajectory will be re-optimized. It may be prudent to optimize over different control policies rather than a single control trajectory, see Mayne (2014) and Mayne (2015). In other words, the optimization problem should compute a cone of possible control trajectories $\{\mathbf{u}_{[k,k+N-1]}\}_{\mathcal{U}}$ instead of a single control trajectory. A simple approach to solve this problem is to discretize the uncertainty space and represent the cone of trajectories as discrete scenarios. This is the basic principle behind scenario MPC. Scenario MPC (also known as multistage MPC or feedback min-max MPC) is thus a closed-loop optimization approach, where the evolution of the uncertainty is explicitly taken into account by modelling a tree of discrete scenarios as described by Scokaert and Mayne (1998). By doing so, we can considerably reduce the conservativeness of the solution compared to min-max methods that optimize over a single control trajectory (Lucia et al., 2013a).

To formulate the scenario MPC mathematically, the discrete-time nonlinear system (1) reads as,

$$\mathbf{x}_{k+1,j} = \mathbf{f}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \mathbf{d}_{k,j}) \tag{2}$$

where, the subscript $(\cdot)_{k,j}$ represents the j^{th} scenario at time step k.

The first step to building a scenario tree is to discretize the uncertainty space $\mathcal{U}$ to get M discrete realizations. A common practice is to consider a combination of nominal and extreme values to cover the overall uncertainty space, which has been shown to give good results in many different application examples, see Lucia et al. (2013a), Krishnamoorthy et al. (2017) and the references therein.

From the discrete realizations of the uncertainty, a scenario tree is generated as shown in Fig.1. Each scenario is defined as the path from root node to the leaf node. The number of scenarios resulting from the branching at each time step leads to exponential growth of the problem. A simple strategy to curb this is to stop the branching after a certain period of time N_r (known as robust horizon) as justified in Lucia et al. (2013a). The total number of scenarios is then given by $S = M^{N_r}$.

The resulting optimization problem is then written as,

$$\min_{\mathbf{x}_{k,j},\mathbf{u}_{k,j}} \sum_{j=1}^{S}\left[\omega_j \sum_{k=1}^{N} \mathbf{J}(\mathbf{x}_{k,j},\mathbf{u}_{k,j})\right] \tag{3a}$$

s.t

$$\mathbf{x}_{k+1,j} = \mathbf{f}(\mathbf{x}_{k,j},\mathbf{u}_{k,j},\mathbf{d}_{k,j}) \tag{3b}$$

$$\mathbf{g}(\mathbf{x}_{k,j},\mathbf{u}_{k,j}) \leq 0 \tag{3c}$$

$$\sum_{j=1}^{S} \bar{\mathbf{E}}_j \mathbf{u}_j = 0 \qquad \forall j \in \{1,\cdots,S\} \tag{3d}$$

where ω is the probability or weight for each scenario, $\mathbf{J}(\mathbf{x}_{k,j},\mathbf{u}_{k,j})$ is the cost function, $\mathbf{f}(\mathbf{x}_{k,j},\mathbf{u}_{k,j},\mathbf{d}_{k,j})$ is the system model, $\mathbf{g}(\mathbf{x}_{k,j},\mathbf{u}_{k,j})$ represents the nonlinear constraints. The constraints in (3d) are known as *non-anticipativity* or *causality* constraints which impose the fact that the future control inputs cannot anticipate the realization of the uncertainty. This implies that the states that branch at the same parent node, must have the same control input. Note that $\mathbf{u}_j$ here represents the sequence of optimal control input for the j^{th} scenario, i.e. $\mathbf{u}_j = \left[\mathbf{u}_{0,j}^T \cdots \mathbf{u}_{N-1,j}^T\right]^T \in \mathbb{R}^{n_u N}$. To explain the notation of $\bar{\mathbf{E}}$, we first introduce the notation:

$$p = n_u \sum_{j=1}^{S-1} n_{c,(j,j+1)} \tag{4}$$

where $n_{c,(j,j+1)}$ represents the number of common nodes for two consecutive scenarios j and $j+1$ in the scenario tree (Klintberg et al., 2016). The matrices $\bar{\mathbf{E}}_j \in \mathbb{R}^{p \times n_u N}$ can then be given as,

$$\begin{aligned}\bar{\mathbf{E}} &= \left[\begin{array}{c|c|c|c|c} E_{1,2} & -E_{1,2} & & & \\ & E_{2,3} & -E_{2,3} & & \\ & & \ddots & \ddots & \\ & & & E_{S-1,S} & -E_{S-1,S} \end{array}\right] \\ &= \left[\,\bar{\mathbf{E}}_1 \middle| \bar{\mathbf{E}}_2 \middle| \cdots \middle| \bar{\mathbf{E}}_S\,\right]\end{aligned} \tag{5}$$

where

$$E_{j,j+1} = \left[\begin{array}{ccc|c} I_{n_u} & & & \\ & \ddots & & \mathbf{0} \\ & & I_{n_u} & \end{array}\right] \in \mathbb{R}^{n_u n_{c,(j,j+1)} \times n_u N} \tag{6}$$

and $\mathbf{0} \in \mathbb{R}^{n_u n_{c,(j,j+1)} \times n_u (N-N_r)}$ is a zero matrix. Using such a chain structure for the non-anticipativity constraints results in sparse structures, which can be an added advantage (Klintberg et al., 2016).

3. SCENARIO DECOMPOSITION

As described above, the different scenarios are independent except for the non-anticipativity constraints, which couple the different scenarios together. To this end, the different scenarios are easily separable. Different decomposition strategies exists that facilitates efficient solutions of such large scale optimization problems by decomposing them into smaller subproblems. This way the different subproblems can be parallelized. A master problem is then employed to co-ordinate the coupling constraints, (Bertsekas, 1999).

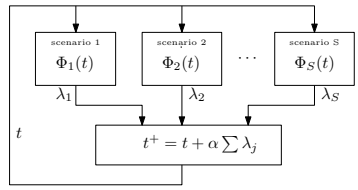


Fig. 2. Block diagram showing the structure information flow between the subproblems and the master problem.

3.1 Lagrangian Decomposition

In Lagrangian decomposition, the dual variables λ corresponding to the non-anticipativity constraints are used to define the Langrange function,

$$\mathcal{L}(\mathbf{x},\mathbf{u},\lambda) = \sum_{j=1}^{S}\left[\omega_j \sum_{k=1}^{N} \mathbf{J}(\mathbf{x}_{k,j},\mathbf{u}_{k,j})\right] + \lambda^T \sum_{j=1}^{S} \bar{\mathbf{E}}_j \mathbf{u}_j \tag{7}$$

Since $\mathcal{L}(\mathbf{x},\mathbf{u},\lambda)$ is separable in $\mathbf{x}$ and $\mathbf{u}$, each scenario can be solved independently by relaxing the non-anticipativity constraints as shown (Klintberg et al., 2016),

$$\mathcal{L}_j(\mathbf{x}_j,\mathbf{u}_j,\lambda) = \omega_j \sum_{k=1}^{N} \mathbf{J}(\mathbf{x}_{k,j},\mathbf{u}_{k,j}) + \lambda^T \bar{\mathbf{E}}_j \mathbf{u}_j \tag{8}$$

Note that (7) and (8) are also subject to the system model (3b) and nonlinear constraints (3c) for all $j \in \{1,\ldots,S\}$. The master problem iterates on λ and the non-anticipativity constraints become feasible only upon convergence of λ. Different forms of augmented Lagrangian decomposition methods have also been introduced in Martí et al. (2015), where an additional quadratic penalty term is added to (7) to improve the convergence properties. However, this makes the problem not separable in $\mathbf{x}$ and $\mathbf{u}$. The subproblems must then be solved sequentially using the Alternating Directions Method of Multipliers (ADMM) approach (Boyd et al., 2011). Solving the subproblems sequentially can then make the computation time longer for problems with large number of scenarios.

The relaxation of the non-anticipativity constraints in Lagrangian decomposition poses a challenge for real time implementation. In an MPC framework, the optimization problem is solved to compute the optimal control trajectory and the first control move in implemented in the plant in a receding horizon fashion. In scenario MPC, the non-anticipativity constraints ensure that the first control input is the same for all the scenarios. However, if λ fails to converge within the required sampling time, infeasibility of the non-anticipativity constraints would mean that the first control move provided by the different scenario subproblems may be different. This is not acceptable for closed-loop implementation. We therefore, provide an alternative approach using primal decomposition framework which always produces a primal feasible point with monotone-decreasing objective value at each iteration.

3.2 Primal Decomposition

Primal decomposition iterates directly on the shared variables (Bertsekas, 1999). This ensures that the non-anticipativity constraints are always feasible at any point

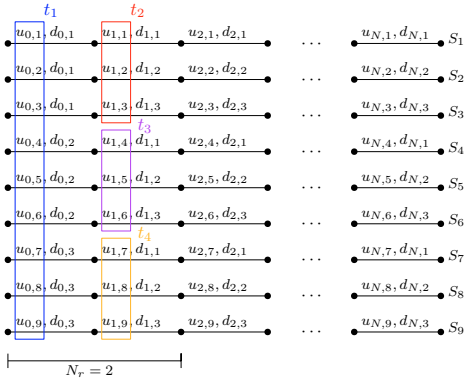


Fig. 3. Parallalized representation of the scenario tree for an uncertainty with 3 discrete realizations (M = 3) and a robust horizon of 2 samples ($N_r = 2$). The non-anticipativity constraints that couple the different scenarios are marked in different coloured boxes.

in time. Thus the first control move provided by all the scenario subproblems will be the same, enabling closed-loop implementation.

The subproblem for each scenario can be written by introducing a new auxiliary variable t_l,

$$\mathbf{\Phi}_j(t_l) = \min_{\mathbf{x}_{k,j}, \mathbf{u}_{k,j}} \omega_j \sum_{k=1}^{N} \mathbf{J}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) \tag{9a}$$

s.t

$$\mathbf{x}_{k+1,j} = \mathbf{f}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \mathbf{d}_j) \tag{9b}$$

$$\mathbf{g}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) \leq 0 \tag{9c}$$

$$\bar{\mathbf{E}}_j \mathbf{u}_j = \bar{\mathbf{t}}_j \tag{9d}$$

where $\bar{\mathbf{t}}$ has a similar structure to $\bar{\mathbf{E}}_j$ as shown below,

$$\bar{\mathbf{t}} = \left[\begin{array}{c|c|c|c|c} t_{1,2} & -t_{1,2} & & & \\ & t_{2,3} & -t_{2,3} & & \\ & & \ddots & \ddots & \\ & & & t_{S-1,S} & -t_{S-1,S} \end{array}\right] \tag{10}$$

$$= [\,\bar{\mathbf{t}}_1 | \bar{\mathbf{t}}_2 | \cdots | \bar{\mathbf{t}}_S\,]$$

$t_{j,j+1} \in \mathbb{R}^{n_u n_{c,(j,j+1)}}$ is composed of auxiliary variables $t_l \in \mathbb{R}^{n_u}$. The index l can be given by the expression,

$$l \in \{1, \ldots, \sum_{m=1}^{N_r} M^{m-1}\} \tag{11}$$

The master problem is then written as,

$$\min_{t_l} \sum_{j=1}^{S} \Phi_j(t_l) \tag{12}$$

which simplifies to updating each t_l using the corresponding lagrange multipliers from the different subproblems as shown in Fig.2.

The generation of $t_{j,j+1}$ and the master problem update is illustrated using an example with $M = 3$ and $n_R = 2$. The corresponding scenario tree is shown in Fig.1 and the decomposed tree is shown in Fig.3. For such a tree, $l = 4$ and $t_l = \{t_1^T, \cdots, t_4^T\}$. Table 1 shows the $t_{j,j+1}$ or each scenario pair.

Each t_l is then updated in the master problem as shown,

$$t_1^+ = t_1 + \alpha_1(\lambda_{1,1} + \cdots + \lambda_{1,9}) \tag{13}$$

$$t_2^+ = t_2 + \alpha_2(\lambda_{2,1} + \cdots + \lambda_{2,3}) \tag{14}$$

$$t_3^+ = t_3 + \alpha_3(\lambda_{2,4} + \cdots + \lambda_{2,6}) \tag{15}$$

$$t_4^+ = t_4 + \alpha_4(\lambda_{2,6} + \cdots + \lambda_{2,9}) \tag{16}$$

where the subscripts of $\lambda_{k,j}$ represents the lagrange multiplier at sample instant k for the j^{th} scenario and α is a suitable step length. A simple stopping criteria for the iterations between the master problem and the subproblems could be when the change in t between two consecutive iterations is less than some small user-defined tolerance ϵ.

By introducing the auxiliary variables t_l, the first control input for all the scenarios is $u_{1,j} = t_1$. The master problem iterates to drive t_1 to the optimal input. In the case, where the master problem does not converge to the optimum within the required sampling time, the non-anticipativity constraints are still feasible, thus enabling closed-loop implementation. By warm-starting t_l in the subsequent time steps, the optimization problem is expected to eventually converge to the true optimum.

4. ILLUSTRATIVE EXAMPLE

4.1 Process description

The primal decomposition approach proposed above is implemented on an oil and gas production optimization problem. We consider a gas lifted well network consisting of 2 wells producing to a common manifold and a riser as shown in Fig.4. More detailed description of the system can be found in Krishnamoorthy et al. (2016) and the references therein.

The objective of the optimization problem is to find the optimum gas lift injection rates for the two wells such that the the profits from the oil production is maximized and the cost of gas compression is minimized. The gas-oil-ratio GOR_i for each well $i \in \{1, \cdots, n_w\}$, is assumed to be uncertain. The nominal value GOR_{0_i} and the variance σ_i are assumed to be known a-priori.

$$\min_{w_{gl}} \sum_{k=1}^{N} \left[-\$_o \sum_{i=1}^{n_w} w_{po,i_k} + \$_{gl} \sum_{i=1}^{n_w} w_{gl,i_k} \right] \tag{17a}$$

s.t.

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, GOR_{i_k}) \tag{17b}$$

$$GOR_i \in \{GOR_{0_i} \pm \sigma_i\} \tag{17c}$$

$$\forall i \in \{1, \cdots, n_w\}, k \in \{1, \cdots, N\}$$

where w_{po} is the oil production rate from each well, w_{gl} is the gas lift injection rate for each well, $n_w = 2$ is the number of wells, $\$_o$ and $\$_{gl}$ are the value of produced

Table 1. Construction of $t_{j,j+1}$ for the scenario tree in Fig.3.

$(j, j+1)$	$n_{c,(j,j+1)}$	$t_{j,j+1}$
(1,2)	2	$[t_1^T, t_2^T]^T$
(2,3)	2	$[t_1^T, t_2^T]^T$
(3,4)	1	$[t_1^T]$
(4,5)	2	$[t_1^T, t_3^T]^T$
(5,6)	2	$[t_1^T, t_3^T]^T$
(6,7)	1	$[t_1^T]$
(7,8)	2	$[t_1^T, t_4^T]^T$
(8,9)	2	$[t_1^T, t_4^T]^T$

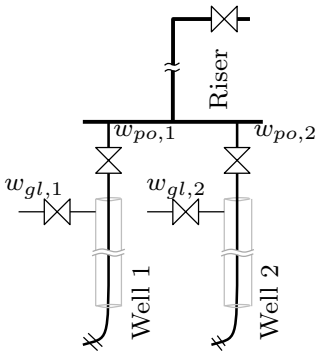


Fig. 4. Schematic of a gas lifted well network with 2 wells producing to a common riser manifold.

Table 2. The discrete realizations of GOR used in the optimizer

GOR well 1	0.08	0.10	0.12
GOR well 2	0.10	0.12	0.14

oil and cost of gas compression respectively. The system constraints are enforced in (17b).

The continuous time differential equations are discretized into (17b) using a third order direct collocation scheme in CasADi v3.0.1 (Andersson, 2013) using the MATLAB R2017a programming environment. The NLP problem is then solved using IPOPT version 3.12.2 running with mumps linear solver.

The dynamic optimization problem was solved with a prediction horizon of $N = 15$ and a sampling time of $T_s = 5min$. A robust horizon of $n_R = 2$ was chosen. M =3 discrete realizations of the uncertain parameter GOR chosen are shown in Table.2. For the scenario decomposition approach, the step length was fixed at $\alpha = [0.0001, 0.0002, 0.0002, 0.0002]^T$. The stopping criteria was defined as when the change in t between two consecutive iterations is less than $\epsilon = 0.001$.

4.2 Results and Discussion

In this section, the performance of the centralized approach and the distributed approach using primal decomposition as proposed above is compared using the case study described above. For the comparison of scenario MPC with nominal and worst case MPC for this problem, the reader is referred to Krishnamoorthy et al. (2017).

In the first simulation, we compare the centralized solution with the decomposed solution. The true realization of GOR for the cases is as shown in Fig.5. The total produced oil for the centralized and decomposed case are shown in top left subplot in Fig.5. The error between the centralized and decomposed solution is shown in top right subplot. The control input (gas lift injection rates for wells 1 and 2) for centralized and decomposed solution is plotted in the middle left subplot and the corresponding error is plotted in the middle right subplot. The number of iterations required for the scenario decomposition to converge at each time step is plotted in the bottom right subplot. From the simulation results, it can be seen that the

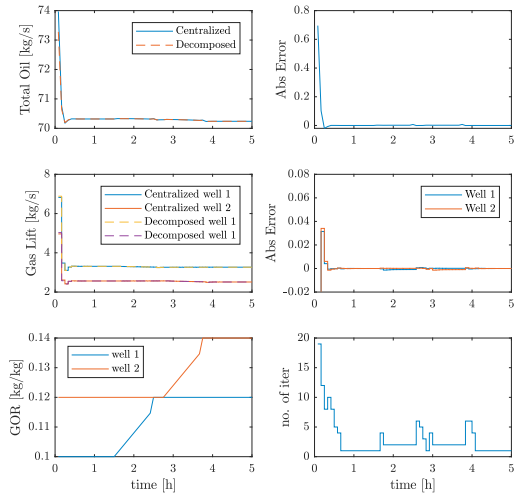


Fig. 5. Comparison of centralized approach and decomposition approach.

primal decomposition approach provides similar solution as the centralized approach. Warm starting the problem at subsequent time steps reduced the number of iterations required to converge in the subsequent time steps. The average computation time for each subproblem was around 1s, as opposed to 11s for the centralized problem.

As mentioned earlier, the main advantage of primal decomposition over dual decomposition methods is when the master problem does not converge within a required sample time. This will lead to violation of the non-anticipativity constraints in dual decomposition, thus leading to closed-loop implementation issues. However, primal decomposition always ensures the feasibility of non-anticipativity constraints. From the results in Fig.5, it was seen that the number of iterations varied between 1 and 19 to converge. To emulate the case where the master problem has to be terminated before it converges fully, the number of iterations is capped at 5. The simulation setup is the same as the previous case. In the case of dual decomposition, prematurely stopping the iterations as done in this simulation will result in an infeasible solution, which causes implementation issues.

The results are shown in Fig.6. It can be clearly seen that the error between the centralized and decomposed approach is much larger during the first hour compared to the results in Fig.5. It can also be seen that the error becomes smaller over time, clearly showing the benefits of warm starting the master problem. The number of iterations required is also reduced to 1 when the change in GOR is constant for a period. This shows that if the the disturbance is not varying too much, the primal decomposition is able to converge to the true optimal solution despite terminating the master problem prematurely. A close look of the first 1 hour of simulation comparing the simulation with the number or iteration uncapped (Fig.5) and capped (Fig.6) is shown in Fig.7.

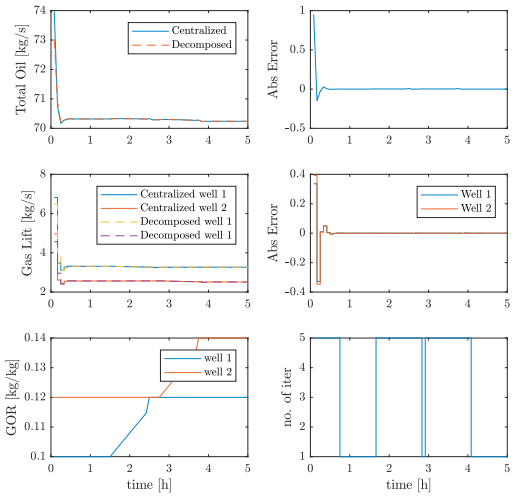


Fig. 6. Comparison of centralized approach and decomposition approach with the maximum number of iterations capped at 5.

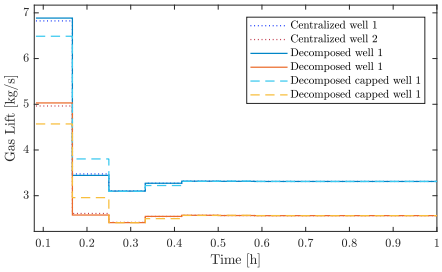


Fig. 7. Closer look at the first one hour of simulation to compare the centralized, decomposed, and decomposed with max iterations capped at 5.

5. CONCLUSION

In this paper, we presented an alternative approach to scenario decomposition using primal decomposition. The primal decomposition approach always ensures the feasibility of the non-anticipativity constraints, hence enabling closed-loop implementation, unlike dual decomposition methods. Warm-starting the master problem eventually leads to convergence over time. Primal decomposition approach may thus be an useful way to decompose scenario MPC for applications with higher sampling rates. The proposed method was tested on a gas lift optimization case study. The simulation results clearly demonstrates the benefit of primal decomposition approach for scenario decomposition. Simulation results show that the primal decomposition eventually converges to the solution of the centralized counterpart despite being terminated prematurely.

6. ACKNOWLEDGEMENTS

Fruitful discussions with Prof. Eduardo Camponogara, Dr.Thiago Lima and Marco Aguiar from the Federal University of Santa Catarina are gratefully acknowledged.

REFERENCES

Andersson, J. (2013). *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering and Optimization in Engineering Center.

Bertsekas, D.P. (1999). *Nonlinear programming*. Athena scientific Belmont.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1), 1–122.

Campo, P.J. and Morari, M. (1987). Robust model predictive control. In *American Control Conference, 1987*, 1021–1026. IEEE.

Guay, M., Adetola, V., and DeHaan, D. (2015). *Robust and Adaptive Model Predictive Control of Nonlinear Systems*. Institution of Engineering and Technology.

Kerrigan, E.C., Constantinides, G.A., Suardi, A., Picciau, A., and Khusainov, B. (2015). Computer architectures to close the loop in real-time optimization. In *Decision and Control (CDC), 2015 IEEE 54th Conference on*, 4597–4611. IEEE.

Klintberg, E., Dahl, J., Fredriksson, J., and Gros, S. (2016). An improved dual newton strategy for scenario-tree mpc. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, 3675–3681. IEEE.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2017). Gas lift optimization under uncertainty. *Computer Aided Chemical Engineering*, 40, 1753–1758.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2016). Real time optimization under uncertainty - applied to gas lifted wells. *Processes*, 4(4). doi:10.3390/pr4040052.

Lucia, S., Finkler, T., and Engell, S. (2013a). Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9), 1306–1319.

Lucia, S., Subramanian, S., and Engell, S. (2013b). Non-conservative robust nonlinear model predictive control via scenario decomposition. In *Control Applications (CCA), 2013 IEEE International Conference on*, 586–591. IEEE.

Martí, R., Lucia, S., Sarabia, D., Paulen, R., Engell, S., and de Prada, C. (2015). Improving scenario decomposition algorithms for robust nonlinear model predictive control. *Computers & Chemical Engineering*, 79, 30–45.

Mayne, D.Q. (2014). Model predictive control: Recent developments and future promise. *Automatica*, 50(12), 2967–2986.

Mayne, D. (2015). Robust and stochastic mpc: Are we going in the right direction? *IFAC-PapersOnLine*, 48(23), 1–8.

Scokaert, P. and Mayne, D. (1998). Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic control*, 43(8), 1136–1142.

Appendix K

Data-driven scenario selection for multistage NMPC

A main assumption in many works considering multistage model predictive control is that discrete realizations of the uncertainty are chosen a-priori and that the scenario tree is given. In this chapter, we focus on choosing the scenarios which is an important practical aspect of multistage scenario-based NMPC. In particular, we show how data analytic tools such as principle component analysis (PCA) can be used to select the scenarios.

This appendix contains the paper that shows how to select the scenarios using PCA and its application to a thermal storage system with time-varying uncertainty.

- Paper published in 2018 IFAC Conference on Nonlinear Model Predictive Control (NMPC) 2018, Madison, WI, USA.
- Paper published in 2019 IFAC International Symposium on Dynamic Control of Process Systems (DYCOPS), Florianopolis, Brazil.

Data-driven Scenario Selection for Multistage Robust Model Predictive Control ⋆

Dinesh Krishnamoorthy, Mandar Thombre, Sigurd Skogestad, Johannes Jäschke

Department of Chemical Engineering, Norwegian University of Science & Technology , NO-7491 Trondheim , (e-mail: dinesh.krishnamoorthy@ntnu.no, mandar.thombre@ntnu.no, skoge@ntnu.no, johannes.jaschke@ntnu.no).

Abstract: A main assumption in many works considering multistage model predictive control (MPC) is that discrete realizations of the uncertainty are chosen a-priori and that the scenario tree is given. In this work, we focus on choosing the scenarios, which is an important practical aspect of scenario-based multistage MPC. In many applications, the distribution of the uncertain parameters is not available, but instead a finite set of data samples are available. Given this finite set of data samples, we present a data-driven approach to selecting the scenarios using principal component analysis (PCA). Using this approach, the scenarios are carefully selected such that the conservativeness of the solution can be reduced while still maintaining robustness towards constraint feasibility. The effectiveness of the proposed method is demonstrated using a simple example.

Keywords: Multistage MPC, Big data analysis, principal component analysis, MPC under uncertainty

1. INTRODUCTION

Model predictive control under uncertainty is an active research area that has received tremendous attention in the recent past, with developments in several different approaches to robust and stochastic MPC in the control literature. Many of these approaches solve an open loop optimization problem to determine the optimal control sequence, taking into account the uncertainty. However, this may not be optimal, since efficient handling of uncertainty requires feedback. In a recent review paper, Mayne (2014) notes that a better strategy would be to optimize over different control trajectories (closed-loop optimization) rather than a single control trajectory (open-loop optimization). One such closed-loop optimization strategy is the multistage scenario MPC also known as feedback min-max MPC or scenario-tree MPC (Scokaert and Mayne, 1998; Lucia et al., 2013).

In this approach, the evolution of the uncertainty in the prediction horizon is described by a scenario tree generated using discrete realizations of the uncertainty. By computing different control trajectories for the different scenarios, the notion of feedback, also known as recourse, is explicitly taken into account in the receding horizon implementation. This was later extended to nonlinear model predictive control by Lucia et al. (2013) in the framework of robust multistage MPC. The approach has since then received a lot of interest and has been applied to several chemical process systems (Lucia and Engell, 2013; Martí et al., 2015), autonomous vehicles (Klintberg et al., 2016), energy systems including power systems, oil and gas (Krishnamoorthy et al., 2016; Verheyleweghen and Jäschke, 2017), building climate control (Maiworm et al., 2015) etc. to name a few.

Most of these works assume that the uncertainty characteristics are known *a-priori* and that the discrete scenarios are given, for example, based on engineering insight before the MPC is designed. However, the issue of how to select the discrete realizations of the uncertainty for the scenario tree generation is an important practical aspect that has not been well studied in the control literature. Nevertheless, the problem has recently been considered in the operations research community under the topic of multistage stochastic optimization and is usually applied only for convex multistage optimization problems assuming full recourse. For example, Monte-Carlo sampling methods were considered in Shapiro (2003) and moment matching methods of the probability density functions (PDF) were used in Høyland et al. (2003). Lucia et al. (2013) also noted that the issue of how to generate the scenario tree for MPC applications is an important future research direction that must be addressed to enable practical implementation of such methods. Recently, a quadrature-based scenario tree generation was proposed using sparse grids by Leidereiter et al. (2014).

In many real applications, the probability distribution function (PDF) or the uncertainty set for the uncertain parameters is not readily available, but only a finite num-

⋆ D.K, S.S and J.J gratefully acknowledge the financial support from SFI SUBPRO. M.T, S.S and J.J gratefully acknowledge the financial support from FME HighEFF. Corresponding author: J.J.

ber of data samples may be available. Classical stochastic MPC frameworks make use of such data indirectly to infer the probability distribution of the uncertain problem parameters by means of statistical estimation methods. The estimated probability distribution function is then subsequently used in the optimization problem (Parys et al., 2016). Thus classical stochastic MPC problem is based on this two-step approach:

(1) estimate the PDF from the finite data samples
(2) use the estimated PDF in the optimization problem.

The main issue with this two step approach is that the estimation step often aims to achieve maximum prediction accuracy without tailoring it to the optimization problem. Hence, the estimated probability distribution function itself may be uncertain as noted by Parys et al. (2016) (leading to recent developments in the so-called distributionally robust optimization). In multistage MPC, the scenario tree is generated using a finite number of uncertainty representations. Given finite data samples, the uncertainty representations may be chosen directly from this data set, thus releasing the assumption of the uncertainty having any particular distribution.

Therefore in this paper, we propose a data-driven multistage scenario MPC problem that avoids the estimation of probability distribution functions and selects discrete realizations of the uncertainty from the finite set of data samples using Big data analytics.

In the case of multi-dimensional parametric uncertainty, the scenario tree becomes large. In such cases, careful selection of scenarios becomes very important to reduce the conservativeness and keep the computation cost low. Given a finite set of data for the different parameters, the use of univariate statistical analysis may fail to detect the relationship between the different parameters. Consequently, this often leads us to choose the scenarios assuming that the parameters are independent of one another. The resulting scenario tree may then span over an unnecessarily large uncertainty space leading to conservative solutions. Big data analytics can examine such large and varied data sets to uncover hidden correlations and can help us choose the scenarios. Therefore in our approach, the relationship between the different parameters is exploited to carefully choose only those combinations of parameters that are likely to be the true realization of the uncertain parameters.

In this paper, we address the issue of how to choose the discrete scenarios from a finite number of data samples and propose the use of multivariate data mining tools such as principal component analysis (PCA) to judiciously choose the scenarios in order to reduce the conservativeness. We use an example to motivate and demonstrate the use of PCA in choosing the scenarios for the multistage MPC formulation.

Methods such as principal component analysis have long since been used together with model predictive control. Among these, the two main application areas combining MPC and PCA has been 1) online performance monitoring (Loquasto and Seborg, 2003; Qin and Yu, 2007; AlGhazzawi and Lennox, 2009) and 2) model reduction (Maurath et al., 1988; Wang et al., 2002; Drgoňa et al., 2018). In an another interesting approach by Liu et al. (2006), the MPC framework is used to control the score space of the PCA to reduce variations in product specifications.

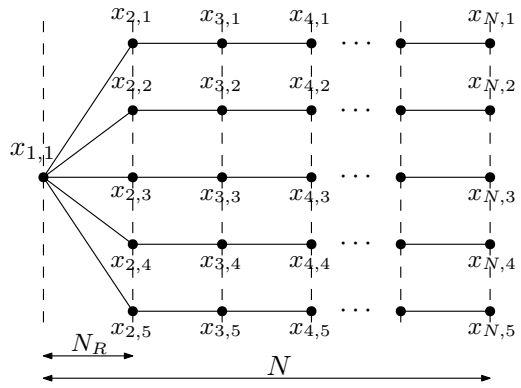


Fig. 1. Schematic representation of a scenario tree generated for $M = 5$ models and a robust horizon of $N_r = 1$.

The remainder of the paper is organized as follows. We introduce the multistage MPC problem in Section 2. Using a simple example, Section 3 motivates the need for data-mining techniques for choosing the discrete scenarios and describes the proposed data-driven multistage scenario MPC using principal component analysis (PCA). Simulation results for the corresponding multistage scenario MPC are provided in Section 4 as a proof-of-concept. Section 5 provides some useful discussions and future research directions towards Big data optimization with respect to multistage MPC before concluding the paper in Section 6.

2. MULTISTAGE MPC

Consider a discrete time nonlinear dynamic system

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

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$ and $\mathbf{u}_k \in \mathbb{R}^{n_u}$ denotes the states and inputs at time step k respectively and $\mathbf{p} \in \mathbb{R}^{n_p}$ denotes the vector of constant but uncertain parameters. The objective is to minimize a performance function $\mathbf{J}(\mathbf{x}_k, \mathbf{u}_k) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ while satisfying constraints $\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k) \leq 0$ using an MPC with a prediction horizon of length N.

In multistage MPC, branching of the scenarios at each sample makes the problem size to grow exponentially over the prediction horizon. In order to curb the problem size, the scenario tree branching is stopped after a certain number of samples $N_r < N$ known as robust horizon as justified by Lucia et al. (2013).

Given M discrete realizations of the uncertainty and a robust horizon of length N_r, we then have $S = M^{N_r}$ discrete scenarios in the scenario tree as shown in Fig. 1. The resulting multistage MPC problem can be formulated as,

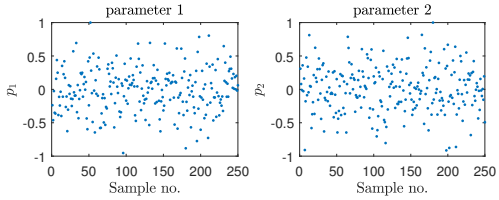


Fig. 2. Raw data of the two parameters p_1 (left subplot) and p_2 (right subplot).

$$\min_{\mathbf{x}_{k,j},\mathbf{u}_{k,j}} \quad \sum_{j=1}^{S} \omega_j \sum_{k=1}^{N} \mathbf{J}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) \tag{2a}$$

s.t.

$$\mathbf{x}_{k+1,j} = \mathbf{f}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \mathbf{p}_j) \tag{2b}$$

$$\mathbf{g}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) \leq 0 \tag{2c}$$

$$\sum_{j=1}^{S} \mathbf{E}_j \mathbf{u}_j = 0 \tag{2d}$$

$$\forall k \in \{1, \ldots, N\}, \forall j \in \{1, \ldots, S\}$$

where the subscript $(\cdot)_{k,j}$ denotes the time step k and scenario j and ω_j represents the weight given to each scenario. (2d) represents the non-anticipativity constraints with $\mathbf{u}_j = \left[\mathbf{u}_{0,j}^T \cdots \mathbf{u}_{N-1,j}^T\right]^T \in \mathbb{R}^{n_u N}$. The non-anticipativity constraints enforce the fact that all the decisions that branch at the same parent node are the same. This captures the real-time decision process correctly, since the control inputs cannot anticipate the future realization of the uncertainty, see Krishnamoorthy et al. (2018a,b) for more details on the structure of $\mathbf{E}_j$.

In this paper, we consider a constrained optimization problem under uncertainty, where the constraint feasibility must be ensured for any given realization of the uncertainty at the cost of conservativeness. Multistage MPC was shown to provide robust constraint feasible solutions whilst being less conservative than min-max approaches (Lucia et al., 2014).

In the next section, we will present a method for analyzing the data and choosing appropriate M discrete realizations of the uncertain parameters $\mathbf{p}$ given a finite set of data samples representing the uncertainty. Note that we require no knowledge on how the data is distributed, however, we assume that discrete historical data samples are available for the different uncertain parameters.

3. DATA-DRIVEN MULTISTAGE MPC

3.1 Motivating example

For the sake of simplicity, let us consider a system with two parameters ($n_p = 2$) and the finite data samples for each of the parameters are available as shown in Fig. 2. At first glance, the data samples tell us that each of the parameters vary in $[-1, 1]$. With no additional information, one often tends to assume that the parameters are uncorrelated, and assumes for example, a box uncertainty set. Consequently, the discrete realizations of the uncertainty from the four corners of the uncertainty set and the nominal value may be chosen, namely, $\mathbf{p}_j \in \{(-1,-1), (1,-1), (0,0), (-1,1), (1,1)\}$, to get a good representation of the uncertainty as shown in Fig. 3 (in gray shaded area). It is also often argued that a combination of extreme and nominal values of the all the parameters must be part of the scenario tree (Lucia et al., 2013).

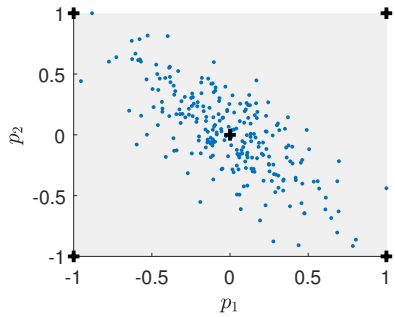


Fig. 3. Multivariate plot of the two parameters. The points $\mathbf{p}_j \in \{(-1,-1), (1,-1), (0,0), (-1,1), (1,1)\}$ are represented by the black '+' and the gray shaded area represents the univariate limits of the two parameters.

However, plotting the two parameters against each other gives us more information about the relationship between the two parameters, as shown in Fig. 3. One can immediately see that $\mathbf{p}_j$ selected using independent parameter variations includes parameter combinations that are unlikely to be the true realization of the uncertainty. Seeking robustness against parameter combinations that are not likely can lead to very conservative and hence suboptimal operation. The information from the simple multivariate plot in Fig. 3 gives us more information into the data's hidden structure which can be exploited to choose the different uncertainty realizations that are more likely. This simple two parameter example already motivates the need for multivariate data analysis methods when choosing the discrete scenarios for multistage MPC.

When the number of parameters and the number of data points increases, it can be cumbersome and time consuming, if not impossible, to plot two parameters at a time to find out the hidden structures in the data simply due to information overload and the effort required to make each plot. Multivariate data mining approaches that attempt to find the hidden structure in big data sets can thus lead to more information that would not have been otherwise discovered. This can directly be exploited in the scenario tree generation to select and include only those parameter combinations that are likely to be the true realization of the uncertainty.

3.2 Data mining using principal component analysis

Principal component analysis (PCA) is a universal data mining tool for extracting useful information hidden in massive amounts of data (Seber, 1984). Principal component analysis attempts to explain the variability in a given set of data by separating the data into so-called *principal components* (PC) where each PC contributes to explaining the total variability of the data. More specifically, PCA uses an orthogonal transformation to convert

a set of (possibly correlated) data into a set of linearly uncorrelated principal components. This transformation is such that the first principal component explains the largest variance in the data set and the other PCs are ordered in decreasing component variance. A principal component therefore points out which variables contribute most to the observed variability in the data and finds the relationship between the different variables (Rao, 1964). In simplest terms, PCA can be thought of as fitting a multi-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component (Hotelling, 1933).

Consider a data set with n_o number of observations for each parameter and the data is represented by a data matrix $\mathbf{P} \in \mathbb{R}^{n_o \times n_p}$. It is important to note that PCA is sensitive to the scaling of the variables and hence the data must be scaled. In addition, mean-centering is also necessary to ensure that the first principal component describes the direction of maximum variance (Jolliffe, 1986).

Therefore, let $\mathbf{P}_0 \in \mathbb{R}^{n_o \times n_p}$ be the scaled and mean-centered data matrix corresponding to $\mathbf{P}$. PCA returns the bilinear model

$$\mathbf{P}_0 = \mathbf{\Lambda}\mathbf{C}^T \tag{3}$$

where the matrix $\mathbf{\Lambda} \in \mathbb{R}^{n_o \times n_p}$ contains the so-called scores (left-hand eigenvectors). The scores represent the distance of the different data points from the mean along the direction of the principal components. The matrix $\mathbf{C} \in \mathbb{R}^{n_p \times n_p}$ contains the coefficients of the principal components which represents the weight by which each original data point should be multiplied to get the component score.

The principal components, scores and coefficients are useful means of understanding the correlation between the different parameters. This information can be exploited in choosing the scenarios as explained in the section below.

In the following subsection, we show how PCA can be used to select scenarios for the multistage robust MPC framework, which to the best of our knowledge has not been used before.

3.3 Scenario generation using data

We now describe how the scores and the coefficients from the principal component analysis can be used to select the discrete realizations of the uncertain parameters. The variance in the scores along the different principal components can be used to describe the uncertainty set instead of using the univariate parameter data. To do this, we pick the data points corresponding to the maximum and minimum scores along the directions of the different principal components that explains the variability with sufficient component variance. Using the coefficients of the principal components, we can then transform this to the original parameter space. These points now form the discrete realizations of the uncertainty that represents the uncertainty space.

This is further illustrated using the data set for two parameters shown in Fig. 2 and Fig. 3. The score plot for this data set is shown in Fig. 4, where the data points corresponding to the maximum and minimum scores along first and second principal component directions are shown

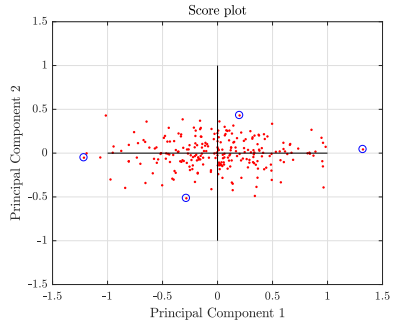


Fig. 4. Score plot along the two principal component directions. The data points corresponding to the maximum and minimum scores along the two PC directions are shown in blue circles.

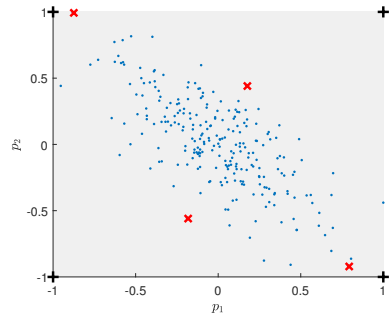


Fig. 5. Data plot in the original space, with realizations corresponding to maximum scores on first and second principal components marked by red '×'. The black '×' correspond to realizations picked by simply taking the combination of extreme values of the parameters.

in blue circles. These are then transformed into the original parameter space as shown in Fig. 5 using a red '×'. We can see that the discrete realizations selected using principal component analysis captures the parameter variations more tightly than the ones chosen by looking at the parameter variations independently.

The proposed approach can thus be summarized by the following steps,

(1) Scale and mean center the data set $\mathbf{P}$ to obtain $\mathbf{P}_0$.
(2) Perform PCA to compute the principal components and the scores for each of the data points $\mathbf{\Lambda}$ and the corresponding co-efficient matrix $\mathbf{C}$ of the principal components.
(3) Pick out the maximum and minimum scores along the direction of the different principal components that sufficiently explain the total variance of the data.
(4) Using the coefficient matrix, re-transform the selected scores from step 3 to the original data space.
(5) Generate the scenario tree based on the discrete realizations of the uncertainty selected in step 4.

4. ILLUSTRATIVE EXAMPLE

In this section, we now compare the effect of the discrete realizations of the uncertainty on the performance of the multistage scenario MPC. We consider a simple example, where the system is given by a model with two states $\mathbf{x} = [x_1, x_2]^T$ and one control input $\mathbf{u} = u$ and two uncertain parameters $\mathbf{p} = [p_1, p_2]^T$ as shown below,

$$\dot{x}_1 = \frac{1}{\tau}\left(-3.5u^2 + 30u - x_1\right) \tag{4}$$
$$\dot{x}_2 = \frac{1}{\tau}\left(4u + 2p_1 + 4p_2 + 10 - x_2\right)$$

with $\tau = 5s$ being the time constant.

The objective is to maximize x_1 while satisfying constraints on x_2 despite the uncertainty in p_1 and p_2. We apply the multistage scenario MPC approach (2) with,

- stage cost $\mathbf{J}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) = -x_1$,
- system model (4) discretized using third order direct collocation,
- inequality constraint (2c): $x_2 \leq 20$,
- uncertain parameters discretized into $M = 5$ realizations of the uncertain parameter
- non-anticipativity constraints (2d).

We choose a prediction horizon of T=1 min divided equally into $N = 60$ samples and a robust horizon of $N_r = 2$ samples (25 scenarios). The true realization of the parameters for the simulation was chosen to be at its nominal value $(0, 0)$. The resulting multistage MPC was implemented in `MATLAB` using `CasADi` algorithmic differentiation tool (Andersson, 2013) version 3.1.0, and `IPOPT` solver (Wächter and Biegler, 2006) was used to solve the resulting nonlinear programming problem.

We first simulate the multistage scenario MPC using $\mathbf{p}_j$ selected using the parameters variations independently, as shown in Table.1 (Simulation 1) and in Fig. 5 using black '+'. This corresponds to using the corner points of the box $[-1 \quad 1] \times [-1 \quad 1]$. The points at the boundaries that constitute a combination of the minimum and maximum values of the uncertain parameters along with the nominal point (0,0) have been selected to get a good representation of the uncertainty based on the time series (univariate) data in Fig.2. This simulation is used as a benchmark.

We then solve the same problem but by replacing $\mathbf{p}_j$ which is now selected using principal component analysis as shown in Table.1 (Simulation 2) and Fig. 5 using red '×'. The simulation results are compared in Fig. 6. The left subplot shows x_1 which has to be maximized, the right subplot shows x_2 which must be maintained below its maximum value of 20. It can be clearly seen from the simulation results that the scenarios chosen using principal component analysis is much less conservative than the scenarios chosen using the parameter data independently. This is because, in the proposed approach, we do not consider scenarios in the scenario tree that are not likely to be the true realization of the uncertainty.

We then simulated the system for 30 runs with different randomly chosen realizations of the uncertain parameters in the plant simulator as shown in Fig. 7 (right subplot). To evaluate the performance, we also plot the integrated objective (left subplot), which is the objective function J

Table 1. Discrete realizations of $\mathbf{p}$ used in the simulations

j	Simulation 1		Simulation 2	
	p_1	p_2	p_1	p_2
1	1	1	0.18	0.44
2	1	-1	0.79	-0.92
3	0	0	0	0
4	-1	1	-0.87	0.99
5	-1	-1	-0.18	-0.56

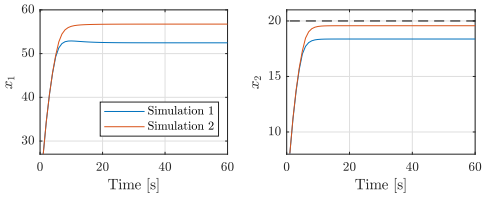


Fig. 6. Simulations results with two different set of scenarios.

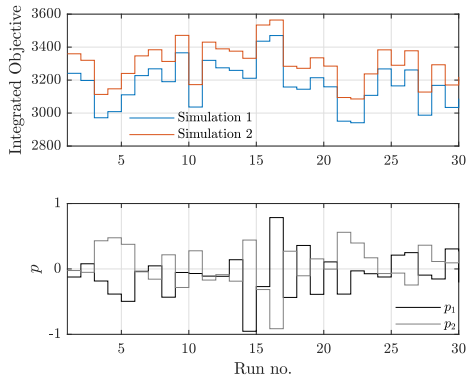


Fig. 7. Monte Carlo Simulations results with different realizations of the uncertain parameters.

integrated over the entire simulation time of $t = 60$ min for each simulation, i.e. integrated objective

$$J_{int} = \int_{t=0}^{t=60} J(t)\,\mathrm{d}t.$$

It can be clearly seen that by using the scenarios selected using the PCA method, we are able to improve the performance for different realizations of the uncertainty from the given data set whilst being robust feasible.

5. DISCUSSION AND FUTURE WORK

In this paper, we proposed to use data mining approaches to select the scenarios based on a finite set of data samples. Note that we have purposefully used a simple example with two uncertain parameters to clearly demonstrate the concept to readers of any level of expertise with such methods. Indeed, the full potential of such data-mining techniques is realized for large data sets with multidimensional parameters, where it may be difficult to select the scenarios purely based on engineering intuition and univariate analysis. For example, consider the building

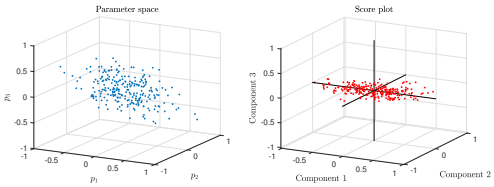


Fig. 8. Data samples for three parameters and the corresponding score plot.

climate control problem, where many uncertain parameters such as temperature, humidity, cloud cover, solar radiation, building occupancy level etc. affect the building climate control problem. Multistage MPC for the building climate control was shown to be a promising approach in Maiworm et al. (2015). One or more of these parameters affecting the climate control problem may be correlated, such as cloud cover, solar radiation and temperature. Using historical data of the weather conditions and building occupancy levels, multivariate data mining techniques can be used to appropriately choose the scenarios to reduce the conservativeness, instead of picking the scenarios based on a combination of maximum and minimum values of the different parameters. By doing so, one can potentially reduce the number of scenarios or the span of the scenario tree to be used in the multistage MPC problem. The application of the proposed data-based scenario selection approach for a building climate control problem is an ongoing work.

5.1 Scenario reduction using variability explanation

Methods such as principal component analysis also provides the percentage of variability explained by the different principal components. This information can in addition, be used to discard scenarios that do not sufficiently explain the variability in the data, hence reducing the number of scenarios that must be considered in the multistage scenario MPC problem. This can help reduce the problem size. For example, consider a different data set for three parameters as shown in Fig. 8. The PCA for this data set returns three principle components, where the first principle component explains 72.5% and the second principal component explains 26.9% of the variability in the data. The third principal component explains only 0.48% of the variability. Based on this, we can then select the maximum and minimum scores along the direction of the first and second principle components and discard the scenario combinations along principal component 3, since it does not sufficiently explain the variability of the data. This helps in reducing the number of scenarios to be included in the scenario tree.

5.2 Weighting in the MPC cost function

The different scenarios can be weighted in the optimization problem as shown in (2a). The results from principal component analysis can not only be used to select the scenarios from the data, but also provide a weight for the selected scenario.

As mentioned earlier, the scores provided by the PCA represent how far a data point is from the mean along the direction of the principal components. Since the data matrix is mean-centered, the data points with large scores are far away from the mean and the vice versa. The weight given to a data point that is far away from the mean (i.e. large score) must be low, compared to the weight given to a data point that is closer to the mean (i.e. low score). Therefore, the weights for the discrete scenarios selected by the PCA method are chosen to be inversely proportional to its score.

5.3 Online update of scenarios

In this paper, we assumed that a finite set of data samples are available which was used to select the scenarios offline using principal component analysis. As more data points become available, PCA can also be used online to continuously adapt the scenarios to reflect the most recent data points. This can be especially useful when the uncertain parameters are time varying in nature. As more data points become available, this information can be included to update the different scenarios in the multistage MPC formulation.

5.4 Other data analytic methods

It must be noted here that PCA does have its limitations, although it works well with the example considered in this work. PCA aims to find hidden linear correlations within the data set, and is thus lacking when data has inherently nonlinear correlations. Further, it only finds PCs that are orthogonal to each other, whereas the projections within the data with highest variance may be nonorthogonal in nature.

For data that is not linearly separable, other data classifiers such as the nonlinear support vector machines (SVM) may be used. The nonlinear SVM maps the given data into a higher-dimensional space using so-called kernel functions, and the transformed data is then linearly separable. Another avenue for further research in improving upon the proposed methodology would be to use advanced data mining techniques for outlier detection. This would be helpful in eliminating the selection of parameters that correspond to 'unlikely' scenarios and help reducing the conservativeness of the solution.

In the previous section, we selected the scenarios corresponding to the maximum scores along the different PC directions. This was done in order to ensure robust constraint feasibility for any realization of the uncertain parameters from the given data set. Alternatively, the scenarios can be chosen based on the scores that falls within some user-defined percentile along the different PC directions to further reduce the conservativeness by trading off on the constraint satisfaction. For example, the scenarios can be chosen using the scores that fall within the 90th percentile in order to reduce the conservativeness to ensure constraint satisfaction with a given probability (analogous to using a chance constrained MPC formulation). Alternatively, one may also use the associated probabilities of the data points to appropriately choose the scenarios. Note that more rigourous analysis must be carried out to get an equivalent performance as using a chance constrained optimization, which is another useful future research direction.

6. CONCLUSION

To conclude, we have motivated the need to develop methods to appropriately choose the scenarios based on a finite sample of historical data. Using a simple example we have demonstrated the concept of how data mining techniques such as principal component analysis can be used to uncover hidden structures in the data, which can then be exploited in choosing the necessary scenarios and discarding the scenarios that need not be considered in the optimization problem. This leads to a less conservative solution as demonstrated in the simulation example. We have also provided some discussions on possible research avenues towards using data mining techniques for scenario selection and hope to stimulate further research in this direction.

REFERENCES

AlGhazzawi, A. and Lennox, B. (2009). Model predictive control monitoring using multivariate statistics. *Journal of Process Control*, 19(2), 314–327.

Andersson, J. (2013). *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, KU Leuven.

Drgoňa, J., Picard, D., Kvasnica, M., and Helsen, L. (2018). Approximate model predictive building control via machine learning. *Applied Energy*, 218, 199–216.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6), 417.

Høyland, K., Kaut, M., and Wallace, S.W. (2003). A heuristic for moment-matching scenario generation. *Computational optimization and applications*, 24(2-3), 169–185.

Jolliffe, I.T. (1986). Principal component analysis and factor analysis. In *Principal component analysis*, 115–128. Springer.

Klintberg, E., Dahl, J., Fredriksson, J., and Gros, S. (2016). An improved dual newton strategy for scenario-tree mpc. In *IEEE 55th Conference on Decision and Control (CDC), 2016*, 3675–3681. IEEE.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2016). Real time optimization under uncertainty - applied to gas lifted wells. *Processes*, 4(4). doi:10.3390/pr4040052.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2018a). A distributed algorithm for scenario-base model predictive control using primal decomposition. *IFAC ADCHEM 2018 (In-Press)*.

Krishnamoorthy, D., Suwartadi, E., Foss, B., Skogestad, S., and Jäschke, J. (2018b). Improving scenario decomposition for multistage mpc using a sensitivity-based path-following algorithm. *IEEE Control Systems Letters*, 2(4). doi:10.1109/LCSYS.2018.2845108.

Leidereiter, C., Potschka, A., and Bock, H.G. (2014). Quadrature-based scenario tree generation for nonlinear model predictive control. *IFAC Proceedings Volumes*, 47(3), 11087–11092.

Liu, X., Chen, X., Wu, W., and Zhang, Y. (2006). Process control based on principal component analysis for maize drying. *Food control*, 17(11), 894–899.

Loquasto, F. and Seborg, D.E. (2003). Model predictive controller monitoring based on pattern classification and pca. In *American Control Conference, 2003. Proceedings of the 2003*, volume 3, 1968–1973. IEEE.

Lucia, S. and Engell, S. (2013). Robust nonlinear model predictive control of a batch bioreactor using multi-stage stochastic programming. In *Control Conference (ECC), 2013 European*, 4124–4129. IEEE.

Lucia, S., Finkler, T., and Engell, S. (2013). Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9), 1306–1319.

Lucia, S., Paulen, R., and Engell, S. (2014). Multi-stage nonlinear model predictive control with verified robust constraint satisfaction. In *IEEE 53rd Annual Conference on Decision and Control (CDC), 2014*, 2816–2821. IEEE.

Maiworm, M., Bäthge, T., and Findeisen, R. (2015). Scenario-based model predictive control: Recursive feasibility and stability. *IFAC-PapersOnLine*, 48(8), 50–56.

Martí, R., Lucia, S., Sarabia, D., Paulen, R., Engell, S., and de Prada, C. (2015). Improving scenario decomposition algorithms for robust nonlinear model predictive control. *Computers & Chemical Engineering*, 79, 30–45.

Maurath, P.R., Laub, A.J., Seborg, D.E., and Mellichamp, D.A. (1988). Predictive controller design by principal components analysis. *Industrial & engineering chemistry research*, 27(7), 1204–1212.

Mayne, D.Q. (2014). Model predictive control: Recent developments and future promise. *Automatica*, 50(12), 2967–2986.

Parys, B.P.G.V., Kuhn, D., Goulart, P.J., and Morari, M. (2016). Distributionally robust control of constrained stochastic systems. *IEEE Transactions on Automatic Control*, 61(2), 430–442. doi: 10.1109/TAC.2015.2444134.

Qin, S.J. and Yu, J. (2007). Recent developments in multivariable controller performance monitoring. *Journal of Process Control*, 17(3), 221–227.

Rao, C.R. (1964). The use and interpretation of principal component analysis in applied research. *Sankhyā: The Indian Journal of Statistics, Series A*, 329–358.

Scokaert, P. and Mayne, D. (1998). Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43(8), 1136–1142.

Seber, G. (1984). Multivariate analysis of variance and covariance. *Multivariate observations*, 433–495.

Shapiro, A. (2003). Monte carlo sampling methods. *Handbooks in operations research and management science*, 10, 353–425.

Verheyleweghen, A. and Jäschke, J. (2017). Framework for combined diagnostics, prognostics and optimal operation of a subsea gas compression system. *IFAC-PapersOnLine*, 50(1), 15916–15921.

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.

Wang, P., Litvak, M., and Aziz, K. (2002). Optimization of production operations in petroleum fields. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers.

Data-driven Online Adaptation of the Scenario-tree in Multistage Model Predictive Control ⋆

Mandar Thombre, Dinesh Krishnamoorthy, Johannes Jäschke

Department of Chemical Engineering, Norwegian University of Science & Technology , NO-7491 Trondheim , (e-mail: mandar.thombre@ntnu.no, dinesh.krishnamoorthy@ntnu.no, johannes.jaschke@ntnu.no).

Abstract: Multistage model predictive control (MPC) is based on the enumeration of scenarios that represent the uncertainty in the system. Scenario selection is important in multistage MPC since the choice of scenarios determines the degree of conservativeness of the optimal solution. We propose a data-driven approach based on principal component analysis (PCA) to dynamically select the scenarios, leading to reduced conservativeness. When time-varying uncertainty is considered, PCA can be performed online to select new scenarios whenever the uncertainty data is updated. The results of the approach are demonstrated for a two-plant system with a thermal storage tank. The solution obtained is less conservative than with standard multistage MPC. This is because the online PCA-based approach accounts for the most recent, and thus more representative, uncertainty realizations.

Keywords: Multistage MPC, online PCA, scenario selection, parametric uncertainty

1. INTRODUCTION

A lot of energy in chemical process plants is lost in the form of industrial waste heat. In industrial clusters of multiple process plants, energy resources such as steam, cooling water, raw materials, etc. are often shared. Optimal energy efficiency in such a system demands flexible operation of the plants, so that surplus energy from one plant is transferred to another plant in need of it. The plants in such clusters can act both as sources of heat (those having surplus heat) and sinks of heat (those having energy demands). Examples of industrial clusters in Scandinavia include Mo Industripark in Norway and Kalundborg Symbiosis in Denmark. The surplus heat streams from different sources are available at varying temperatures. Moreover, supply of surplus heat and energy demand is often asynchronous, while also operating at differing time scales. To mitigate some of these issues, employing a thermal energy storage system is an attractive option. Such a system creates a buffer between the energy suppliers and users. For example, domestic thermal energy storage in buildings was considered by de Oliveira et al. (2016).

Process plants often exhibit highly nonlinear dynamics. In addition, they also have to contend with disturbances in process parameters like temperatures and flow rates during operation. In the context of energy exchange, this consequently results in an uncertain yield/consumption of surplus heat on the supply/demand side. The existence of such uncertainties presents a significant challenge for operating these plants at an optimal point.

⋆ M.T and J.J gratefully acknowledge the financial support from FME HighEFF. D.K and J.J gratefully acknowledge the financial support from SFI SUBPRO. Corresponding author: J.J.

Model predictive control (MPC) is a powerful tool for operational optimization that is widely used in the process industry. However, nominal MPC does not explicitly model the uncertainties in the system. To rigorously account for uncertainty, methods classified under "Robust MPC" have been receiving some attention in control literature. Based on the theory of robust optimization, the so called min-max MPC approach (first proposed by Campo and Morari (1987)) minimizes the cost of the worst-case realization of the uncertainty. The notion of feedback in min-max MPC was first introduced by Scokaert and Mayne (1998). In this approach, the closed-loop optimization is sought over different sequences of control inputs for different realizations of the uncertainty. This idea was further extended by Lucia et al. (2013) for multistage nonlinear MPC, based on the concepts of stochastic programming. Here, the evolution of uncertainty is assumed to be modeled by a scenario tree, which is generated from the discrete realizations of the uncertainty. By optimizing over each branch of this scenario tree, the idea that new information will be available in future stages is explicitly accounted for.

The convention in multistage MPC is to assume that the uncertainty information is known *a-priori*, that it is known perfectly and that it is discrete. However, the selection of scenarios that build the scenario tree is very important in the practical implementation, and has not been sufficiently addressed in the context of multistage MPC. Traditionally, scenario-based stochastic programming methods involve a two step process: estimating a probability distribution function (PDF) from a finite data-set, and subsequently discretizing the PDF to generate scenarios (Birge and Louveaux, 2011). Another approach is to skip the PDF

estimation step and go directly from data to scenarios, i.e. the discrete scenarios can directly be chosen from the available data samples. After all, the different data samples represent the discrete measurements of uncertainty in the system. Ideally, then, any set of selected scenarios should be a subset of this data set for the best representation of uncertainty.

Having decided on selecting scenarios directly from available data, the next question to consider is *which* data samples to select as scenarios. The size of the multistage MPC problem increases exponentially with increasing number of scenarios. Hence it is important to capture maximum uncertainty information with minimum number of scenarios, in order to be computationally efficient.

Uncertain parameters in a system often exhibit correlations. Sampling methods like the Monte Carlo or the Latin hypercube sampling emphasize randomness of sampling to maximize information, but ignore correlations. Therefore, these scenario selection methods may not be the best if we want to exploit the correlation to reduce the number of scenarios. To overcome this, multivariate data-analysis methods like the principal component analysis (PCA) can be used to detect any hidden correlations within the available data samples. The scenarios chosen using these multivariate methods explicitly take into account the interdependence between the parameters. Dimensionality reduction methods such as PCA explain the parametric variation in a data set in fewer dimensions - referred to as principal components. This means that lesser number of scenarios are able to effectively describe the parametric variation in the system, leading to a compact scenario tree formulation.

In this paper, we propose an online PCA-based approach for systems with time-varying uncertainty, that can be performed dynamically in the multistage MPC implementation. The idea is to select new scenarios online whenever new uncertainty data becomes available, and to systematically reformulate the optimization problem in anticipation of a predicted change in uncertainty. This can provide an additional hedge against uncertainty, since the "latest" data is constantly being used to select the scenarios. Moreover, we propose that if the parametric variation can be explained by a small number of "dominant" principal components (shown by PCA), it suffices to select the scenarios only along these components to sufficiently explain the uncertainty. The proposed approach is applied to a simple thermal energy storage model.

The paper is structured as follows. Section 2 describes the formulation of the multistage MPC problem and the dynamic scenario selection strategy using PCA. Section 5 illustrates the modeling for energy storage used as a case study for the demonstration of the multistage MPC. Simulation results are presented in Section 6 and the conclusions and recommendations are stated in Section 7.

2. PRELIMINARIES - MULTISTAGE MPC

Consider sets $\mathcal{S} = \{1, \ldots, S\}$ and $\mathcal{J} = \{0, \ldots, N-1\}$, where S is the total number of scenarios and N is the length of the prediction horizon. The scenario-based multistage MPC problem can be formulated as follows:

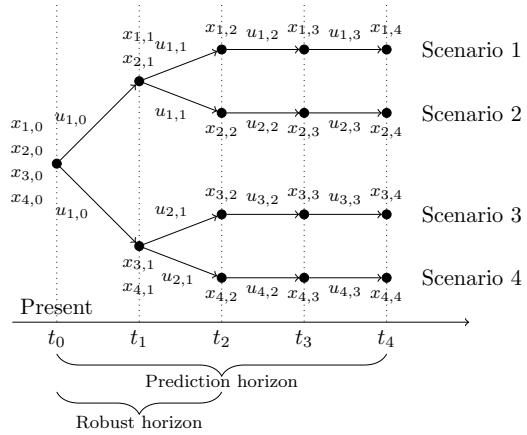


Fig. 1. Illustration of the scenario tree at time t_0 with $M = 2$ realizations of uncertainty. The prediction horizon $N = 4$ and the robust horizon $N_r = 2$.

$$\min_{\mathbf{x}_{i,j}, \mathbf{u}_{i,j}} \quad \sum_{i \in \mathcal{S}} \omega_i \sum_{j \in \mathcal{J}} \phi(\mathbf{x}_{i,j}, \mathbf{u}_{i,j}) \tag{1a}$$

s.t.

$$\mathbf{x}_{i,0} = \mathbf{x}^{init}, \qquad \forall i \in \mathcal{S} \tag{1b}$$

$$\mathbf{x}_{i,j+1} = \mathbf{f}(\mathbf{x}_{i,j}, \mathbf{u}_{i,j}, \boldsymbol{\pi}_i), \qquad \forall j \in \mathcal{J}, \forall i \in \mathcal{S} \tag{1c}$$

$$\mathbf{g}(\mathbf{x}_{i,j}, \mathbf{u}_{i,j}) \leq 0, \qquad \forall j \in \mathcal{J}, \forall i \in \mathcal{S} \tag{1d}$$

$$\sum_{i \in \mathcal{S}} \mathbf{E}_i \mathbf{u}_i = 0, \qquad \forall j \in \mathcal{J}, \forall i \in \mathcal{S} \tag{1e}$$

The states and inputs for the ith scenario and jth time step are denoted by $\mathbf{x}_{i,j} \in \mathbb{R}^{n_x}$ and $\mathbf{u}_{i,j} \in \mathbb{R}^{n_u}$ respectively. The uncertain parameters for the ith scenario are denoted by $\pi_i \in \mathbb{R}^{n_\pi}$. Equation (1a) represents the cost to minimize, where ω_i is the weight assigned to the ith scenario and $\phi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is the objective function. Equation (1b) represents the initial condition constraints, with $\mathbf{x}^{init}$ being the vector of starting points for the states. Equation (1c) represents the model of the nonlinear dynamic system described by the vector of state equations $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\pi} \to \mathbb{R}^{n_x}$, and Equation (1d) represents the constraints in the system, denoted by $\mathbf{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$.

Equation (1e) represents the non-anticipativity constraints, which enforce that all control inputs applicable to branches of the same node, are equal. This is because, in real applications, the uncertainty is realized *after* the control input is applied. In other words, one cannot anticipate how the scenario tree is going to branch out at a node before a decision is taken at that node. The reader is referred to Krishnamoorthy et al. (2018a) for details on the construction of the non-anticipativity matrix $\mathbf{E_i}$.

To curb the rapid growth of the scenario tree, the uncertainty is assumed to be constant after a certain point in the prediction horizon so as to reduce the computational cost, as justified by Lucia and Engell (2013). This point represents the so-called robust horizon of the problem, with a length N_r. The scenario tree evolution showing the prediction horizon and the robust horizon is illustrated in Fig. 1. Therefore if we have M discrete realizations of the uncertainty, then this results in $S = M^{N_r}$ scenarios.

3. DYNAMICALLY ADJUSTING THE SCENARIO-TREE

Many systems have to contend with uncertainty that is time-varying. In this paper, we propose to update the scenario-tree dynamically to acknowledge this time-varying uncertainty. Our approach takes into account that the considered scenarios may change during the operation of the system. That is, the uncertainty may have very different characteristics during different points in time.

For example, consider the intra-day variation of energy demand in a district heating network, with high peaks in the morning and the afternoon. Clearly, the scenarios depicting the demand during the peak hours will be different from the scenarios during low and medium demand periods.

To reflect these changes, the scenario-tree can be updated during operation as new uncertainty data becomes available. We propose two update strategies for the scenario-tree:

(1) Adjust the scenarios, i.e the parameter values, at every time step as new information becomes available.
(2) Extend the length of the robust horizon at the current time step if a change in uncertainty information at a future time step can be anticipated.

The first strategy recognizes that multistage MPC is performed on a moving horizon, where the optimization problem is repeatedly solved at every time step with an updated initial condition. We propose, in addition, to also update the scenarios themselves at every time step in recognition of newly available uncertainty information.

For instance, assume that at time t_0, the scenario-tree is as given in Fig. 1. This tree is used in the multistage-MPC controller until updated information is available. If at time t_2 new information about the uncertain parameters becomes available, the scenario-tree for the optimization problem will be updated using the new data. This updated scenario-tree will be used in the multistage MPC implementation from time t_2 until newer information about the scenarios becomes available.

With respect to the second strategy, if it is known a-priori that uncertainty data will be updated at a future time step in the horizon, the robust horizon can be accordingly modified to take this into account. Farther way from the point of update, a shorter robust horizon can be used to reduce computational burden. As the predicted point of update comes closer, the robust horizon can be extended to include the new scenarios reflecting the update.

Consider again the scenario tree shown in Fig. 1. Here $N_r = 2$ at time t_0. However, if it known at time t_0 that new uncertainty information is available at at time t_2, then the robust horizon can be increased from $N_r = 2$ to $N_r = 3$ at time t_0, in order to accommodate this extra branching at time t_2. This would thus lead to consideration of additional scenarios (in this case, $2^3 = 8$ scenarios).

These updates of scenarios and the robust horizon can be done dynamically within multistage MPC. The procedures can be performed via online computations at any time step which presents new information about the uncertainty.

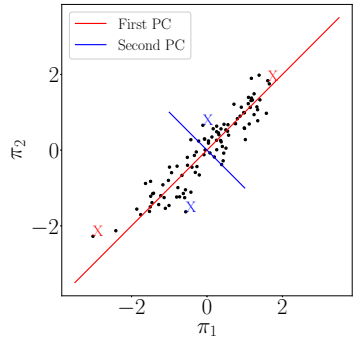


Fig. 2. Selected scenarios, marked by 'X', correspond to the maximum and minimum scores along the two principal components.

4. DATA-DRIVEN SCENARIO SELECTION

Given a data set representing the uncertain parameters, the naive approach of selecting scenarios would be to take the combinations of the minimum and maximum values of each parameter to maximize uncertainty information. Similar to random sampling methods, this approach ignores parameter correlation. PCA can be leveraged to judiciously choose those samples from the set that incorporate information about correlations between the parameters. This is especially true for large data sets with many parameters where detecting correlations between parameters, if they exist, is impractical via simple tools such as univariate analysis.

PCA employs a mathematical procedure that transforms a data set with multiple, possibly correlated, variables into a lesser number of uncorrelated variables, known as principal components. Essentially, it is an orthogonal linear transformation of the data set into a new coordinate system with each new axis representing a principal component. The first principal component points in the direction of maximum variance within the data set. Subsequent principal components account for as much of the remaining variance as possible, in decreasing order. This dimensionality reduction helps explain the parametric variation in the data using smaller number of components.

Consider a data matrix $\mathbf{X} \in \mathbb{R}^{n_o} \times \mathbb{R}^{n_\pi}$, where rows of the matrix represent observations and columns represent the (possibly correlated) parameters. To remove arbitrary biases from the measurements, the data is mean-centered and scaled, resulting in the data matrix $\mathbf{X}_{sc} \in \mathbb{R}^{n_o} \times \mathbb{R}^{n_\pi}$. Performing PCA on $\mathbf{X}_{sc}$ results in the output $\mathbf{Y} \in \mathbb{R}^{n_o} \times \mathbb{R}^{n_{\pi'}}$, with $\pi' \leq \pi$, according to:

$$\mathbf{Y} = \mathbf{X}_{sc}\mathbf{P} \tag{2}$$

where, $\mathbf{P} \in \mathbb{R}^{n_\pi} \times \mathbb{R}^{n_{\pi'}}$ is the projection matrix with each column representing a principal component. In other words, each column of $\mathbf{P}$ contains the coefficients that project the original data point to the new coordinate system ($\mathbf{Y}$) of π' principal components. These are also referred to as *loadings*. The matrix $\mathbf{Y}$ is called the *scores* matrix. The score of a data point along a principal component represents the distance of that data point from

the mean along the direction of that principal component. For an overview of PCA and its algorithms, the reader is referred to Jolliffe and Cadima (2016).

Scenarios can be chosen by leveraging information from this transformed data set. Since the principal components are orthogonal to each other, scenarios can be chosen along the direction of these principal components to obtain maximum uncertainty information, as demonstrated in our previous paper (Krishnamoorthy et al., 2018b). For example, the chosen scenarios could correspond to the maximum and minimum scores along the dominant principal components that explain the variations in the data sufficiently, as shown in Fig. 2.

The PCA may result in principal components such that some components dominate over the others, in terms of how much data variability they explain. We propose to select scenarios only along these dominant principal components, since the chosen scenarios can then account for maximum variation in the uncertainty.

For instance, Fig. 2 shows scenarios (marked by red and blue "X"s) selected along both the principal components. Instead, since it can be seen clearly that the first principal component is dominant [1], scenarios can be chosen only in that direction (marked by red "X"s). Thus, instead of choosing 4 scenarios, only 2 scenarios can encompass most of the parametric variation in the data shown in Fig. 2, without any significant loss in explained variability. Reducing the number of scenarios in this manner can thus make the size of resulting multistage MPC problem significantly smaller, reducing the computational effort.

5. CASE STUDY: SIMPLE ENERGY STORAGE SYSTEM

We consider a simple two-plant thermal storage system, with one plant being the supplier of heat (plant A) and the other being the consumer (plant B). A thermal storage tank acts as a buffer between the two plants to facilitate the energy exchange. The tank interacts with the two plants via heat exchangers, as shown in Fig. 3.

Further, the tank can directly be heated up via a local heating source. For example, in the context of industrial clusters, surplus heat from the flue gases that result from various chemical processes can be used to heat up the tank. Thus, the local heating source is considered inexpensive. If the energy in the tank is insufficient to meet the energy requirements on the demand side, the plant has to purchase the excess energy from the market. Energy from the market is usually much more expensive than the local heating source. The objective is to operate the system such that the total cost of energy purchase is minimized.

5.1 Process model

The heat exchangers are modeled as devices with two chambers representing the hot side and the cold side. Both chambers of the heat exchanger, as well as the tank itself, are considered to have the same temperature throughout their volumes. Thus the temperatures exiting these volumes are considered to be same as those inside the volumes. The driving force for the heat exchange between the two chambers is the difference in the temperatures of the two chambers. Further, we consider hot water as the fluid for heat exchange.

[1] For the data shown in Fig. 2, the first principal component explains 96.4% of the variance.

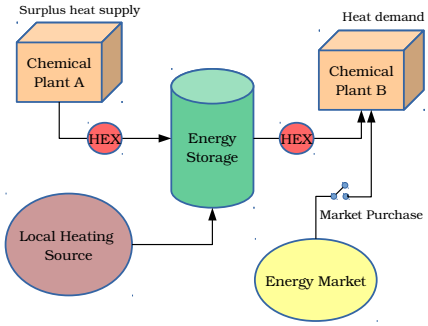


Fig. 3. Illustration of a simple energy storage system.

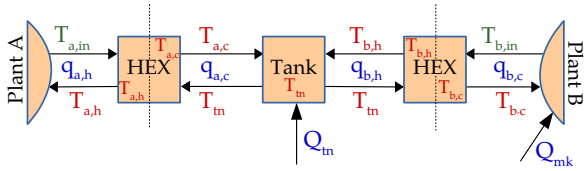


Fig. 4. Schematic of the model. The states, inputs and disturbances are shown in red,blue, and green respectively.

Table 1. Model parameters

Symbol	Value	Symbol	Value
V_{tn}	100 m^3	V	0.5 m^3
A_{tn}	100 m^2	A	300 m^3
U_{tn}	0.5 kW/m^2K	U	0.5 kW/m^2K
$(q_{ah})_{max}$	1 m^3/s	$(q_{ah})_{min}$	0 m^3/s
$(q_{ac})_{max}$	1 m^3/s	$(q_{ac})_{min}$	0 m^3/s
$(q_{bh})_{max}$	1 m^3/s	$(q_{bh})_{min}$	0 m^3/s
$(q_{bc})_{max}$	1 m^3/s	$(q_{bc})_{min}$	0 m^3/s
$(T_{tn})_{max}$	100 $°C$	$(T_{tn})_{min}$	30 $°C$
$(Q_{tn})_{max}$	5000 kW	$(Q_{tn})_{min}$	0 kW
T_{amb}	20 $°C$	Q_{demand}	5000 or 10000 kW
P_{tn}	5 units/kW	P_{mk}	1000 units/kW
ρ	1000 kg/m^3	C_p	4.18 kJ/kgK

Both heat exchangers have an area A, volume V and heat transfer coefficient U. The tank has a volume V_{tn}, a surface area of A_{tn}, and experiences heat loss with a heat transfer coefficient U_{tn}. The density and specific heat capacity of water are denoted by ρ and C_p respectively. Temperatures on the hot and cold sides of heat exchangers on both sides (plant A and plant B) are $T_{a,h}$, $T_{a,c}$, $T_{b,h}$ and $T_{b,c}$ respectively. The tank temperature is T_{tn}, whereas the ambient temperature is T_{amb}. Inlet temperatures from plant A and plant B are $T_{a,in}$ and $T_{b,in}$ respectively The flow rates on either side of the heat exchangers are $q_{a,h}$, $q_{a,c}$, $q_{b,h}$ and $q_{b,c}$ respectively. Local heat supply is denoted by Q_{tn} and the energy purchased from the market is denoted by Q_{mk}. The model parameters are shown in Table 1.

Fig. 4 shows the schematic with the different states, inputs and disturbances in the model. The exit temperatures from the chambers of heat exchangers and the tank are the states $\mathbf{x}$. Flow rates on either sides of the heat exchanger, along with the local and market heat supply, are the inputs $\mathbf{u}$. The inlet temperatures from the two plants are the disturbances. The energy balances on the hot and cold chambers of the heat exchangers, and on the tank, become:

$$\frac{dT_{a,h}}{dt} = \frac{1}{V}\left\{ q_{a,h}(T_{a,in} - T_{a,h}) - \frac{UA(T_{a,h} - T_{a,c})}{\rho C_p} \right\} \quad (3a)$$

$$\frac{dT_{a,c}}{dt} = \frac{1}{V}\left\{ q_{a,c}(T_{tn} - T_{a,c}) + \frac{UA(T_{a,h} - T_{a,c})}{\rho C_p} \right\} \quad (3b)$$

$$\frac{dT_{b,h}}{dt} = \frac{1}{V}\left\{ q_{b,h}(T_{tn} - T_{b,h}) - \frac{UA(T_{b,h} - T_{b,c})}{\rho C_p} \right\} \quad (3c)$$

$$\frac{dT_{b,c}}{dt} = \frac{1}{V}\left\{ q_{b,c}(T_{b,in} - T_{b,c}) + \frac{UA(T_{b,h} - T_{b,c})}{\rho C_p} \right\} \quad (3d)$$

$$\frac{dT_{tn}}{dt} = \frac{1}{V_{tn}}\left\{ q_{a,c}(T_{a,c} - T_{tn}) - q_{b,h}(T_{tn} - T_{b,h}) + \frac{Q_{tn}}{\rho C_p} - \frac{U_{tn}A_{tn}(T_{tn} - T_{amb})}{\rho C_p} \right\} \quad (3e)$$

5.2 Uncertainty description

We consider uncertainties in the form of disturbances in the system, $T_{a,in}$ and $T_{b,in}$, the inlet temperatures from the two plants. The temperature data from various streams in a process is usually logged. A period of 24 hours is considered for this uncertainty data. Further, we consider that the inlet temperature distributions are different for three distinct phases of the day (12 AM to 8 AM, 8 AM to 4 PM and 4 PM to 12 AM). We assume that historic data is available for the temperature distributions for these phases.

The plants operate at higher temperatures during the day phase and lower temperatures during the evening and night phases. Further, it is reasonable to expect that these plant temperatures are correlated to each other. This is because the periods of high and low activity in plants in an industrial cluster are similar. The scatter plot of the two inlet temperatures is shown in Fig. 5, for a series of historic data.

5.3 Formulating the multistage MPC problem

For a given energy demand profile for plant B, the economic objective is to minimize the cost of energy. In context of the multistage MPC formulation (1), the cost function for ith scenario and jth time step can then be stated as:

$$P_{tn}(Q_{tn})_{i,j} + P_{mk}(Q_{mk})_{i,j}$$

where P_{tn} and P_{mk} are the prices of the local energy supply and the market supply respectively.

The starting values of all temperatures are imposed as the constraints (1b). The model equations (1c) of the multistage MPC formulation are obtained by discretizing

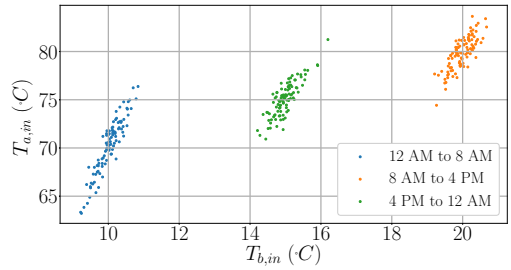


Fig. 5. Uncertainty data - inlet temperatures from the plants for different phases of the day. It can be seen that the temperatures are correlated differently throughout each phase of the day.

the energy storage model equations (3) using collocation on finite elements. These form the equality constraints of the problem. Bounds on the temperatures, flow rates, and energy supplies are imposed for each scenario and time step, and these form the inequality constraints (1d). In addition, the non-anticipativity constraints (1e) are also imposed.

Recall that plant B purchases energy from the market to satisfy the demands in excess of what can be satisfied solely through the storage system. For the ith scenario and jth time step, this can be formulated as the constraint:

$$(Q_{mk})_{i,j} + \rho C_p q_{b,c}((T_{b,c})_{i,j} - (T_{b,in})_{i,j}) \geq Q_{demand}$$

where Q_{demand} is the total energy demand of plant B. This imposes the condition that the energy purchased from the market in addition to the energy from the storage system must at least be equal to energy demand from the plant.

6. RESULTS

The multistage MPC problem is formulated with $N = 24$ hours (finite elements) and $N_r = 1$ hour, with control action changing every hour. `JuMP` (version 0.18.2) (Dunning et al., 2017), a modeling tool within the framework of `Julia` (version 0.6.2) (Bezanson et al., 2017) programming language, is used to implement the multistage MPC problem. The resulting nonlinear optimization problem is solved using `Ipopt` (Wächter and Biegler, 2006). The results are divided into the following two parts:

(1) Comparison of a scenario selection using a data-driven PCA and a conventional "BOX" approach, with a constant $N_r = 1$.
(2) Studying the effect of dynamically adjusting N_r closer to an anticipated change in the uncertainty data, while using PCA for scenario selection.

6.1 Data-driven vs conventional scenario selection

For comparison, the dynamic scenario selection is done with two methods. In the first method, a conservative approach is used, selecting scenarios as the four corner points from the box that encompasses all the uncertainty data over each 8-hour phase during the day. Essentially, these scenarios represent the combinations of the minimum and maximum of the data set along each dimension. A

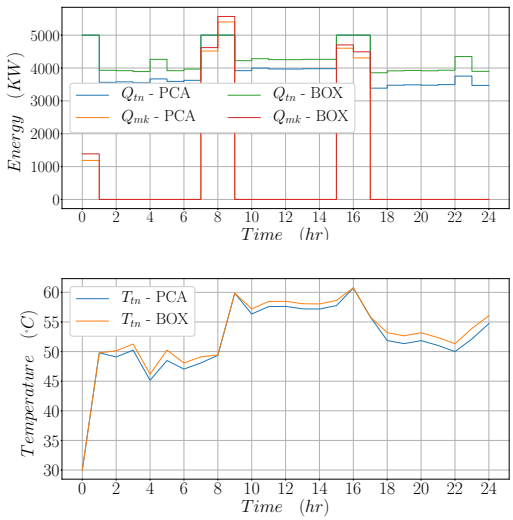


Fig. 6. The energy supplies and the tank temperature across the 24-hour period, for the two methods.

fifth scenario is chosen to represent the mean value along each dimension. This approach is referred to as the "BOX" method. The BOX method does not account for any correlations between the uncertain inlet temperatures.

In the second, "PCA" method, the scenarios are chosen by performing PCA dynamically over the data sets that are relevant for the corresponding phases of the day (shown by different colors in Fig. **??**).The PCA results in two principal components for each data set, with the first principal component explaining 96.51% of the total variability for the 12 AM to 8 AM data, 89.72% for the 8 AM to 4 PM data, and 91.84% for the 4 PM to 12 AM data. Since the first principal components explain a large fraction of the variance in the data, two scenarios are selected corresponding to the minimum and maximum scores along this principal component. A third scenario is chosen to represent the mean value along each dimension.

We consider the uncertainty to be time-varying over the MPC horizon, where the "true" realization of the uncertainty in the simulator is chosen randomly for each hour from the corresponding data set. The simulation is considered for the 24 hour horizon from 12 AM to 12 AM. The demand is constant at 5000 kW throughout the day, except for 7 AM to 9AM and 3 PM to 5 PM, when there is a peak demand of 10,000 kW.

The results of the optimization are shown in Fig. 6. It can be clearly seen that for the BOX method, the solution is more conservative. The tank is heated to a higher temperature for satisfying the same demand profile across the day. Similarly, the heat supplied to the tank is also more compared to the PCA method. Also, the data-driven approach leads to lower purchases of the expensive energy from the market during peak demands.

Moreover, the simulations were repeated 30 times for the time-varying uncertainty case. The "true"" set of uncer-

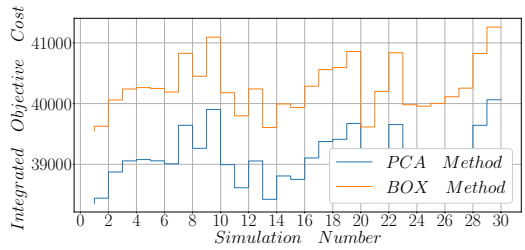


Fig. 7. The averaged integrated cost for 30 different simulation runs.

tain parameters in each simulation was a randomly chosen subset of the available data set. The performance for each simulation run was evaluated based on the integrated objective function, which sums up the values of the objective cost for all stages and scenarios. The results are presented in Fig. 7, where it can be seen that the PCA method outperforms the BOX method with a lower cost. Note that the integrated objective costs are divided by the respective number of scenarios chosen for each method for a fairer comparison.

6.2 Dynamically adjusting the robust horizon

Here, the simulations were run such that the multistage MPC was implemented using two N_r cases. In the first case, the robust horizon was kept constant throughout the simulation at $N_r = 1$. This is referred to as the "NR1" case.

In the second case, the robust horizon was dynamically adjusted (switched) from $N_r = 1$ to $N_r = 2$, *one hour before* the night-to-morning and morning-to-evening phase changes. This change was implemented only for the corresponding next one time step, and subsequently reduced back to $N_r = 1$ for later time steps. Thus, $N_r = 2$ was used for the MPC time steps at 7 AM and 3 PM. This is because it is known that, at these times, the uncertainty data will be updated in one hour due to change in phase; and the robust horizon length of 2 hours reflects this. This is denoted as the "NR2" case.

The scenario selection was done via PCA for both cases. Moreover, the energy demand was considered to be constant at 5000 kW throughout the 24-hour period (i.e. no peak heating). The results are shown in Fig. 8

It can be seen that by dynamically extending the robust horizon (NR2 case), the optimization anticipates the upcoming rise in inlet temperatures by pre-empting the local tank heating at 7 AM (shown by a higher Q_{tn} in the NR2 case than in the NR1 case). This can also be seen from the tank temperature profile, where the temperature of the tank rises higher in the NR2 case at 7 AM than in the NR1 case. Consequently, the market purchase at 8 AM is smaller in the NR2 case, leading to lower cost. During the 4 PM phase change, the temperatures are dropping anyway so market purchase is unnecessary. This leads to the same temperature and heating profiles in the evening phase for both NR1 and NR2 cases. This is because, at this time, the tank has enough energy to satisfy the energy demand of Plant B.

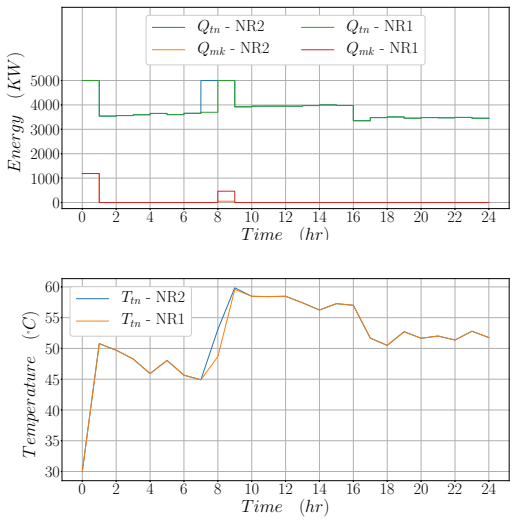


Fig. 8. The energy supplies and the tank temperature across the 24-hour period, for $N_r = 1$ and $N_r = 2$.

7. CONCLUSION AND FURTHER WORK

The case study of the thermal storage tank demonstrates that the same energy demand profile can be satisfied by heating the tank *less* if the scenario selection is data-driven and dynamic. Not only does the tank operate at a lower temperature, but the cost of operation is also significantly lower.

In addition, extending the robust horizon dynamically leads to the consideration of future changes in inlet temperatures by the MPC algorithm. This prompts preemptive control action so that the tank is heated up in anticipation even before the uncertainty data changes. The result is that the market purchase is reduced when the energy is demanded at a higher inlet temperature.

To conclude, we have demonstrated that an online PCA-based, dynamic scenario-tree adaptation approach leads to solutions that are less conservative while still hedging against the uncertainty. Moreover, the approach involves solving an optimization problem of a smaller size since less scenarios, chosen only along the dominant principal component are needed to describe the uncertainty.

With respect to further work in this domain, multistage MPC could be combined with tube-based MPC in a similar fashion as described in Subramanian et al. (2018) to seek robustness against the uncertainty in the direction of the "insignificant" principal components, which were discarded in scenario selection procedure in this work. In terms of modeling, a thermal storage system with multiple suppliers and consumers of energy is more realistic. Further, the effect of *uncertain* time-varying peak loads on the optimal operation (i.e. using scenarios to describe varying magnitudes of peak loads) can be studied.

REFERENCES

AlGhazzawi, A. and Lennox, B. (2009). Model predictive control monitoring using multivariate statistics. *Journal of Process Control*, 19(2), 314–327.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1), 65–98.

Birge, J.R. and Louveaux, F. (2011). *Introduction to Stochastic Programming*. Springer Publishing Company, Incorporated, 2nd edition.

Campo, P.J. and Morari, M. (1987). Robust model predictive control. In *1987 American Control Conference*, 1021–1026.

de Oliveira, V., Jäschke, J., and Skogestad, S. (2016). Optimal operation of energy storage in buildings: Use of the hot water system. *Journal of Energy Storage*, 5, 102 – 112.

Dunning, I., Huchette, J., and Lubin, M. (2017). Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2), 295–320.

Jolliffe, I.T. and Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical transactions Series A, Mathematical, physical, and engineering sciences*.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2018a). A distributed algorithm for scenario-base model predictive control using primal decomposition. *IFAC ADCHEM 2018 (In-Press)*.

Krishnamoorthy, D., Thombre, M., Skogestad, S., and Jäschke, J. (2018b). Data-driven scenario selection for multistage robust model predictive control. In *6th IFAC Conference on Nonlinear Model Predictive Control (In-Press)*. IFAC.

Loquasto, F. and Seborg, D.E. (2003). Model predictive controller monitoring based on pattern classification and pca. In *American Control Conference, 2003. Proceedings of the 2003*, volume 3, 1968–1973. IEEE.

Lucia, S. and Engell, S. (2013). Robust nonlinear model predictive control of a batch bioreactor using multi-stage stochastic programming. In *Control Conference (ECC), 2013 European*, 4124–4129. IEEE.

Lucia, S., Finkler, T., and Engell, S. (2013). Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9), 1306–1319.

Maurath, P.R., Laub, A.J., Seborg, D.E., and Mellichamp, D.A. (1988). Predictive controller design by principal components analysis. *Industrial & engineering chemistry research*, 27(7), 1204–1212.

Scokaert, P. and Mayne, D. (1998). Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43(8), 1136–1142.

Subramanian, S., Lucia, S., and Engell, S. (2018). A synergistic approach to robust output feedback control: Tube-based multi-stage nmpc. In *10th IFAC International Symposium on Advanced Control of Chemical Processes*, 494–499. IFAC.

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 L

Multistage model predictive control with online scenario tree update using recursive Bayesian weighting

Once the scenarios in the multistage formulation are chosen, the scenario tree is often kept fixed. This appendix contains the paper, where we propose an "adaptive-robust" idea to update the scenarios using a recursive Bayesian weighting approach, thereby gradually shrinking the uncertainty set.

- Paper published in 2019 European Control Conference, Naples, Italy.

Multistage Model Predictive Control with Online Scenario Tree Update using Recursive Bayesian Weighting

Dinesh Krishnamoorthy, Sigurd Skogestad and Johannes Jäschke

***Abstract*— This work deals with a nonlinear multistage model predictive control (MPC) formulation, where the future propagation of the uncertainty in the prediction horizon is represented via a discrete scenario tree. The scenario tree is often generated using finite realizations of the uncertainty sampled from an uncertainty set or a probability distribution function. Once the scenarios are chosen, the scenario tree is often kept fixed for all the iterations. In this paper, we propose to update the different discrete realizations of the uncertainty in the scenario tree using a recursive Bayesian weighting approach. We show that by gradually shrinking the uncertainty set, we can further reduce the conservativeness of the closed-loop solution. The effectiveness of the proposed method is demonstrated using an oil and gas production optimization case study.**

I. INTRODUCTION

Model predictive control is a popular optimal control method in the process industry due to its ability to handle multivariable constrained systems. However the performance of MPC is strongly affected by the quality of the prediction model used by the MPC. Models are almost always subject to uncertainties due to imperfect knowledge of the system or model simplification. In the presence of constraints, additional robustification must be introduced in order to ensure robust constraint satisfaction despite the uncertainty.

To this end, there has been several developments in min-max approaches [1], where the optimal input trajectory is computed by minimizing the cost of the worst case realization of the uncertainty. Although this ensures robust constraint satisfaction, this often leads to overly conservative and hence suboptimal solutions. This is because the optimization is performed for the worst-case scenario in an open-loop fashion without any notion of feedback.

In a recent review paper [2], the author argues that effective handling of uncertainty requires feedback models and hence the control trajectory computed by solving an open loop optimization problem is not optimal. Multistage scenario-based MPC, also known as feedback min-max MPC, is one such closed-loop optimization approach introduced in [3] and later extended to nonlinear systems in [4]. Here, the future evolution of the uncertainty in the prediction horizion is represented by a discrete scenario tree. Different control trajectories are then computed for the different scenarios. In other words, the multistage MPC formulation explicitly takes into account the fact that new information will become available in the future and new optimal control input will be re-computed. This can be envisaged as a player's decision making process in an evolutionary strategic game. Instead of preparing a single sequence of optimal moves, we compute several backup moves depending on the future evolution of the uncertainty throughout the game. This concept is commonly known as *recourse* in the stochastic programming literature and is an important property in optimal decision making under uncertainty.

The multistage scenario-based MPC formulation has been shown to be less conservative than traditional min-max approaches for various applications, see for e.g. [4],[5], [6], [7] to name a few. Nevertheless, the solution provided by the multistage MPC will be conservative in order to ensure robust constraint satisfaction for all the scenarios considered in the scenario tree. A common approach to choosing the discrete realizations of the uncertainty in the scenario tree is to use a combination of the maximum, minimum and nominal values of the uncertain parameters. If the assumed range of the uncertain parameter is large, the resulting span of the scenario tree is also large, and consequently the solution provided will be conservative, albeit less conservative than traditional min-max MPC.

For problems with constant but unknown parameters, i.e. time-invariant uncertain parameters, it may be desirable to approach the problem from an adaptive framework rather than a robust framework[8]. Typical adaptive control frameworks involve the use of parameter estimators that adapts the uncertain parameters online such that it converges asymptotically to the true system. However, developments in adaptive MPC have been rather limited. Parameter estimation algorithms also often requires the uncertain parameters to be observable from the measurements, which may not always be the case.

In this paper, we propose to adapt the uncertainty characteristics (i.e. the span of the uncertain parameters) instead of adapting the parameters directly. The idea of updating the uncertainty characteristics instead of using a parameter estimator itself is not entirely new. Similar ideas of adaptive robust approaches were also explored in [8], [9], [10], where the uncertainty set containing all possible values of the uncertain parameters are estimated instead of adapting the parameters directly. The different works used different approches to update the uncertinty sets, such as ensemble Kalman filter (EnKF) or set-based guranteed parameter estimation etc. In this paper, we propose an alternative appraoch

The authors gratefully acknowledge the financial support from SUBPRO, which is financed by the Research Council of Norway, major industry partners and NTNU. JJ also acknowledges support from DNV GL.

The authors are with the Department of Chemical Engineering, Norwegian University of Science and Technology (NTNU), 7491, Trondheim, Norway. dinesh.krishnamoorthy@ntnu.no, skoge@ntnu.no, johannes.jaschke@ntnu.no

to updating the uncertainty set that does not requre the use of a parameter estimation algorithm. Instead, we propose to use the recursive Baysian probability to update the scenario tree.

Using the process measurements and the model predictions from the different scenarios, we compute and assign a Bayesian weight for the different scenarios. The Bayesian weights are then used to shrink the span of the scenario tree by updating the scenarios with a very low weight. In other words, by computing Bayesian weights for the different discrete uncertainty realizations, we gradually eliminate values from the uncertainty set that do not explain the observed measurements with sufficient likelihood. This results in shrinking the span of the scenario tree over time and hence further reduce the conservativeness from the standard multistage scenario-based MPC.

Recursive Bayseian weighting approach such as the one used in this work were also used in multiple model predictive control (MMPC) formulations in works such as [11] and [12] to interpolate between different models from a model bank. We apply a similar Bayesian weighting scheme and instead of interpolating between the different scenarios, we update the scenarios with very low weights. This is an intuitive approach to updating the scenario tree online based on the available measurements and model predictions.

The reminder of the paper is organized as follows. The multistage scenario-based robust MPC framework is introduced in Section 2. The proposed online scenario tree adaptation based on recursive Bayesian weighting approach is described in Section 3. The proposed method is demonstrated using an oil and gas production optimization problem in section 4 before concluding the paper in Section 5.

II. Background

Consider a discrete-time nonlinear system parameterized by a vector of *time-invariant* uncertain parameters $\mathbf{p} \in \mathbb{R}^{n_p}$,

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

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \tag{2}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$ and $\mathbf{u}_k \in \mathbb{R}^{n_u}$ denotes the state and input vectors, respectively. The system model is represented by $\mathbf{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$. The vector of available measurements is denoted by $\mathbf{y} \in \mathbb{R}^{n_y}$ given by the measurement model $\mathbf{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$.

The objective is to minimize a cost function $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ subject to the nonlinear inequality constraints $\mathbf{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$ over a prediction horizon of length N. The optimal control problem can then be written as,

$$\min_{\mathbf{x}_k, \mathbf{u}_k} \quad \sum_{k=0}^{N-1} J(\mathbf{x}_k, \mathbf{u}_k) \tag{3a}$$

s.t.

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}) \tag{3b}$$

$$\mathbf{g}(\mathbf{x}_k, \mathbf{u_k}) \leq 0 \tag{3c}$$

$$\mathbf{x}_0 = \hat{\boldsymbol{x}} \tag{3d}$$

$$\mathbf{p} \in \mathcal{U}, \quad \forall k \in \{0, \ldots, N-1\}$$

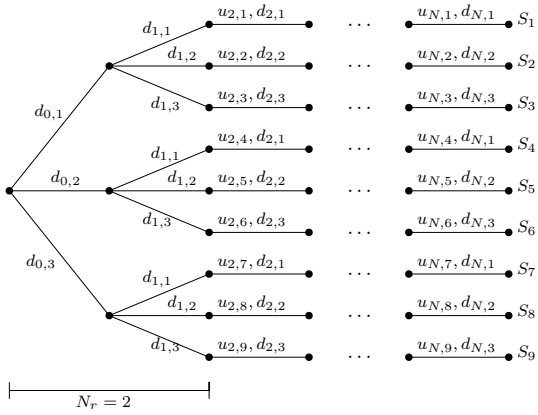


Fig. 1: Schematic representation of the scenario tree with $M = 3$ discrete realizations of the uncertainty and a robust horizon of $N_r = 2$, leading to $S = 9$ scenarios.

where $\mathcal{U} \subset \mathbb{R}^{n_p}$ denotes the bounded uncertainty characteristics, either in the form of uncertainty set or probability distribution function. Initial conditions are enforced in (3d), where $\hat{\boldsymbol{x}}$ denotes the current state estimates.

If the model was *perfect* and $\mathbf{p}$ was known accurately, then for an optimal input trajectory $\mathbf{u}^p_{[t,t+N]}$, the predicted state trajectory is given by $\mathbf{x}^p_{[t,t+N]}$. However, in the presence of uncertainty, an optimal input trajectory $\mathbf{u}^p_{[t,t+N]}$ would give rise to a cone of state trajectories $\{\mathbf{x}^p_{[t,t+N]}\}_{\mathcal{U}}$ depending on the value of the uncertain parameter $\mathbf{p} \in \mathcal{U}$. Optimizing over a single control trajectory $\mathbf{u}^p_{[t,t+N]}$ ignores the fact that new information will be made available at the next time step and a new optimal input trajectory will be re-computed. In other words, the optimization is performed in an open-loop fashion (although the implementation may be in closed-loop, if the optimal control problem is re-solved at each sampling time with only the first control input move implemented on the process). Closed-loop optimization, on the other hand, involves computing a cone of possible control trajectories $\{\mathbf{u}^p_{[t,t+N]}\}_{\mathcal{U}}$ instead of a single control trajectory $\mathbf{u}^p_{[t,t+N]}$, thereby introducing *recourse* action.

A simple approach to closed-loop optimization is by discretizing the uncertainty characteristics and solving the sampled average approximate problem as explained in [3] and [4]. Therefore the first step to designing a multistage scenario-based MPC is to select the discrete realizations of the uncertainty from the uncertainty set. In order to ensure robust constraint satisfaction for any realization of the uncertainty from the uncertainty set, a combination of maximum, minimum and nominal values of the different uncertain parameters are often chosen as the scenarios.

To this end, M discrete realizations of the uncertain parameters are sampled from the uncertainty set $\mathcal{U}$ to generate a scenario tree as shown in Fig.1. In order to prevent the exponential growth of the problem, the scenario branching is often terminated after a certain number of samples N_r in

the prediction known as robust horizon, as justified in [4]. This results in $S = M^{N_r}$ number of scenarios. The resulting scenario optimization problem can be written as,

$$\min_{\mathbf{x}_{k,j},\mathbf{u}_{k,j}} \quad \sum_{j=1}^{S} \omega_j \sum_{k=1}^{N} J(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) \tag{4a}$$

s.t.

$$\mathbf{x}_{k+1,j} = \mathbf{f}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}, \mathbf{p}_j) \tag{4b}$$

$$\mathbf{g}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) \leq 0 \tag{4c}$$

$$\mathbf{x}_0 = \hat{\boldsymbol{x}} \tag{4d}$$

$$\sum_{j=1}^{S} \bar{\mathbf{E}}_j \mathbf{u}_j = 0 \tag{4e}$$

$$\forall j \in \{1, \cdots, S\}\ \forall k \in \{0, \ldots, N-1\}$$

where ω_j represents the weight for the different scenarios. The constraints in (4e) represents the non-anticipativity or causality constraints which enforce the fact that the control inputs cannot anticipate the future realization of the uncertainty. In other words, the states that branch from the same parent node must have the same control inputs. Here $\mathbf{u}_j$ represents the sequence of optimal control input for the j^{th} scenario, i.e. $\mathbf{u}_j = \left[\mathbf{u}_{0,j}^T \cdots \mathbf{u}_{N-1,j}^T\right]^T \in \mathbb{R}^{n_u N}$. The reader is referred to [13] and [14] for detailed description of the non-anticipativity constraints and the structure of $\bar{\mathbf{E}}_j$ used in (4e).

III. Multistage MPC with Online Scenario Update

As mentioned earlier, often a combination of maximum, minimum and nominal values of the different parameters are chosen as the different uncertainty realizations in the scenario tree. If the uncertainty parameter range is rather large, then the scenario tree has a large span resulting in a conservative solution. A common assumption in most works considering multistage scenario MPC is that, once the scenarios are selected, the scenario tree remains fixed. In the case of *time-invariant* uncertain parameters, an adaptive framework may be preferable that enables the controller to improve its performance over time by employing some adaptive mechanism to eliminate/update scenarios that do not explain the observations with sufficient likelihood.

The proposed multistage model predictive control method with online scenario tree update is based on adapting the scenario tree online using the available measurements $\mathbf{y}^m \in \mathbb{R}^{n_y}$. A recursive Bayesian weighting scheme is used to assign weights to the different scenarios. The scenarios that do not explain the observations with sufficient likelihood (represented by very low weights) are then updated online to reduce the span of the scenario tree.

A. The recursive Bayesian weighting scheme

In this paper, we use a probabilistic weighting scheme which assigns weights to each scenario, which is a value ranging from 0 to 1. This is based on the conditional probability of the j^{th} scenario being the true realization of the uncertainty given the past history of residuals and probabilities of all the scenarios. The recursive Bayes theorem for the j^{th} scenario at time step k is then given by,

$$P_{k,j} = \frac{e^{-0.5\boldsymbol{\epsilon}_{k,j}^T K \boldsymbol{\epsilon}_{k,j}} P_{k-1,j}}{\sum_{m=1}^{S} e^{-0.5\boldsymbol{\epsilon}_{k,m}^T K \boldsymbol{\epsilon}_{k,m}} P_{k-1,m}} \tag{5}$$

where the residual $\boldsymbol{\epsilon}_{k,j} \in \mathbb{R}^{n_y}$ for the j^{th} scenario at the current time step k is computed using the observations $\mathbf{y}^m$ and the model predictions (2) for the j^{th} scenario,

$$\boldsymbol{\epsilon}_{k,j} = \mathbf{y}_k^m - \mathbf{h}(\mathbf{x}_{k,j}, \mathbf{u}_{k,j}) \tag{6}$$

Here, K is a weighting matrix, which is typically chosen to be diagonal and can be seen as the inverse of the residual covariance. Hence, large values of K often leads to faster convergence towards a scenario and smaller values of K lead to a more averaging approach to the scenarios. Using the recursive probabilities (5), the weight for the j^{th} scenario at time step k are then computed as shown below,

$$W_{k,j} = \frac{P_{k,j}}{\sum_{m=1}^{S} P_{k,m}} \tag{7}$$

B. Online scenario tree update

Using the Bayesian weights, we can update the scenario tree online. We find the scenario corresponding to the smallest weight represented by $\underline{j}$ and find the scenario corresponding to the largest Bayesian weight represented by $\bar{j}$. If the Bayesian weight for the $\underline{j}^{\text{th}}$ scenario becomes lower than a user defined threshold δ, then the least likely $\underline{j}^{\text{th}}$ scenario is updated by moving it towards the most likely $\bar{j}^{\text{th}}$ scenario. For example, if the $\underline{j}^{\text{th}}$ scenario has a very low weight $W_{k,\underline{j}} < \delta$ relative to the $\bar{j}^{\text{th}}$ scenario, then the $\underline{j}^{\text{th}}$ scenario is updated in the direction of the most likely scenario using a user defined step length $\alpha < 1$ as shown below,

$$\mathbf{p}_{\underline{j}} \leftarrow \mathbf{p}_{\underline{j}} + \alpha(\mathbf{p}_{\bar{j}} - \mathbf{p}_{\underline{j}}) \tag{8}$$

The step length α must be sufficiently small (typically in the range of 0.1) to retain the originally envisioned robustness properties. Hence, at the next time step $k+1$, the $\underline{j}^{\text{th}}$ scenario in the scenario tree is updated according to (8) and the scenario MPC problem (4) is solved using the updated scenario tree. Note that, the threshold δ must be chosen such that the likelihood of the $\underline{j}^{\text{th}}$ scenario explaining the observations must be sufficiently low. This is to avoid updating a scenario which only has a marginally lower Bayesian weight relative to the other scenarios. Since we have used a recursive weighting scheme, it is also crucial that the weights are reset after each time the scenarios are updated. This is because the new weight of the updated scenario must not be based on the probability of the old scenario realization. A sketch of the proposed multistage MPC with online scenario tree update scheme is described in Algorithm 1. The algorithm presented here to compute the recursive Baysiean weights is computationally inexpensive and is known to reject poor models exponentially fast [11].

Algorithm 1 Multistage MPC framework with online scenario tree update

Define tolerance $\delta > 0$, $\alpha < 1$, initial probability for each scenario $P_{0,j} = \frac{1}{S}, \quad \forall j \in \{1, \dots, S\}$.

Input: At each time step, observations $\mathbf{y}_k^m$ and model predictions $\hat{\mathbf{y}}_{k,j}$

for $j = 1, 2, \dots, S$ **do**

$\boldsymbol{\epsilon}_{k,j} \leftarrow \mathbf{y}_k^m - \hat{\mathbf{y}}_{k,j}$

$P_{k,j} \leftarrow \frac{e^{-0.5\boldsymbol{\epsilon}_{k,j}^T K \boldsymbol{\epsilon}_{k,j}} P_{k-1,j}}{\sum_{m=1}^{S} e^{-0.5\boldsymbol{\epsilon}_{k,m}^T K \boldsymbol{\epsilon}_{k,m}} P_{k-1,m}}$

$W_{k,j} \leftarrow \frac{P_{k,j}}{\sum_{m=1}^{S} P_{k,m}}$

end for

Find scenario corresponding to smallest weight $\underline{j} = \arg\min_j \quad W_{k,j}$

Find scenario corresponding to largest weight $\bar{j} = \arg\max_j \quad W_{k,j}$

if $W_{k,\underline{j}} < \delta$ **then**

Update $\underline{j}^{th}$ scenario $\mathbf{p}_{\underline{j}} \leftarrow \mathbf{p}_{\underline{j}} + \alpha(\mathbf{p}_{\bar{j}} - \mathbf{p}_{\underline{j}})$

Reset probability $P_{k,j} \leftarrow \frac{1}{S}, \quad \forall j \in \{1, \dots, S\}$

end if

$[\mathbf{x}_{k,j}^*, \mathbf{u}_{k,j}^*] \leftarrow$ Solution of Scenario MPC problem (4)

Output: $\mathbf{u}_{k,j}^*$

The computed Bayesian weights, in addition to scenario pruning, can also be used to weight the different scenarios in the cost function by setting $\omega_j = W_{k,j}$ in (4). By doing so, we give more weight on the scenarios that explains the observations with a higher likelihood.

IV. Case Study

In this section, we demonstrate the proposed scenario tree update mechanism using a gas lift optimization case study.

A. Process description

We consider an oil and gas production network consisting $n_w = 2$ gas lifted wells as shown in Fig.2. The objective is to maximize the total oil production from the network while maintaining the total gas production within the processing capacity constraints. This is expressed as,

$$\min_{w_{gl,i}} \; -c_o \sum_{i=1}^{n_w} w_{po,i} + c_{gl} \sum_{i=1}^{n_w} w_{gl,i} \tag{9a}$$

s.t.

$$\sum_{i=1}^{n_w} w_{pg,i} \leq w_{pg}^{max} \tag{9b}$$

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

$$\mathbf{p} \in \mathcal{U} \tag{9d}$$

$$\forall i \in \{1, \dots, n_w\}$$

where $w_{gl,i}$ is the gas lift injection rate for each well and is the manipulated variable ($n_u = 2$), $w_{po,i}$ and $w_{pg,i}$ are the oil and gas production rates from each well respectively, w_{pg}^{max} is the total gas processing capacity. c_o and c_{gl} are economic terms that represents the value of oil and cost of gas compression respectively. (9c) represents the nonlinear dynamic model of the gas lifted wells. The reader is referred to [5] for more detailed description of the gas lifted well models.

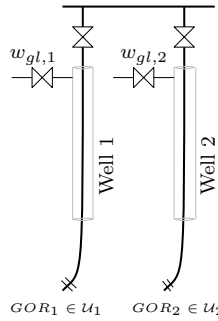


Fig. 2: Schematic representation of two gas lifted wells.

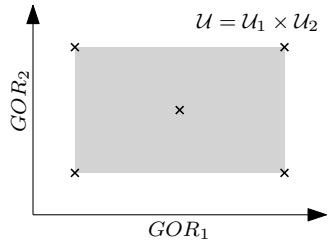


Fig. 3: Uncertainty subspace showing the 5 discrete scenarios used to generate the scenario tree.

The gas-oil-ratio GOR is a reservoir property that denotes the ratio of oil and gas entering each well from the reservoir, which is an uncertain parameter (i.e. $\mathbf{p} = GOR$). $\mathcal{U}$ denotes the uncertainty characteristics to which the uncertain parameter is known to belong. Since we have two wells, we have two uncertain parameters ($n_p = 2$), namely, the gas-oil ratio for each well. In this simulation example, we consider the uncertainty in $GOR_i \in \mathcal{U}_i$ to be equally distributed with $GOR_1 \in [0.05, 0.15]$kg/kg and $GOR_2 \in [0.11, 0.13]$kg/kg. Hence the uncertainty set $\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2$ is given as a box uncertainty set. For the multistage scenario MPC, $M = 5$ discrete realizations of the uncertainty are considered that corresponds to the combination of minimum, maximum and nominal values of the gas-oil-ratio of the two wells. This is schematically represented in Fig.3, where the discrete scenarios are marked by an '$\times$'. Multistage scenario MPC was then applied with a robust horizon of $N_r = 1$ leading to $S = 5$ discrete scenarios, as described in [15].

The multistage scenario MPC is setup with a sampling time of 5min and with a prediction horizon of 2hours using `CasADi v3.1.0` [16] with `MATLAB` programming environment. The resulting nonlinear programming (NLP)

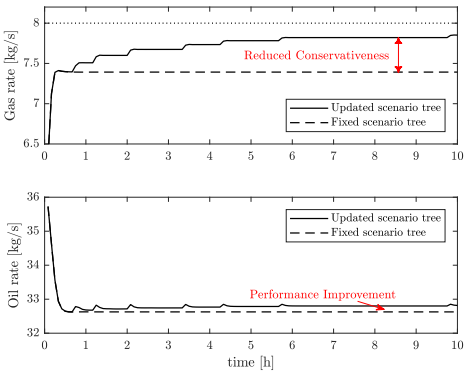


Fig. 4: Simulation results comparing the proposed method with online update of scenario tree with fixed scenario tree multistage MPC

problem is solved using `IPOPT v.3.12.2` running with a `MUMPS` linear solver. The simulations were performed on a 2.6GHz workstation with 16GB memory. The plant model was implemented using the `IDAS` integrator. It is worth noting that, given the timescale of the optimization problem, the changes in the reservoir properties are very slow and can be considered constant for the optimization problem. Hence, the uncertain parameter GOR, which is a reservoir property, can be assumed to be a time invariant parameter for the production optimization problem, with a given initial uncertainty characteristics $\mathcal{U}$.

B. Simulation results

In this section, we apply the proposed multistage MPC formulation on the gas lift optimization case study, where the scenarios are updated online using the recursive Bayesian weighting scheme, with a suitable step length $\alpha = 0.2$. The results from the standard multistage MPC with a fixed scenario tree was used to benchmark the performance of the proposed scheme. To update the scenario tree, pressure measurements and flow measurements through the wellhead choke are used to compute the Bayesian weight for the different scenarios.

In the first simulation, we assume the true realization of the gas-oil-ratio in the plant is $GOR = [0.0975, 0.1201]^\mathsf{T}$ and the maximum gas capacity limit is considered to be $w_{pg}^{max} = 8$kg/s. Fig.4 shows the closed-loop simulation results for the multistage MPC with updated scenario tree (solid lines) compared to the standard multistage MPC with fixed scenario tree (dashed lines). The first subplot shows the total gas production rate and the second plot shows the total oil production rate.

In the proposed multistage MPC scheme, the scenarios with very low weights are updated as described in Algorithm 1. By doing so, the span of the uncertainty subspace covered by the scenarios gradually reduces. Consequently,

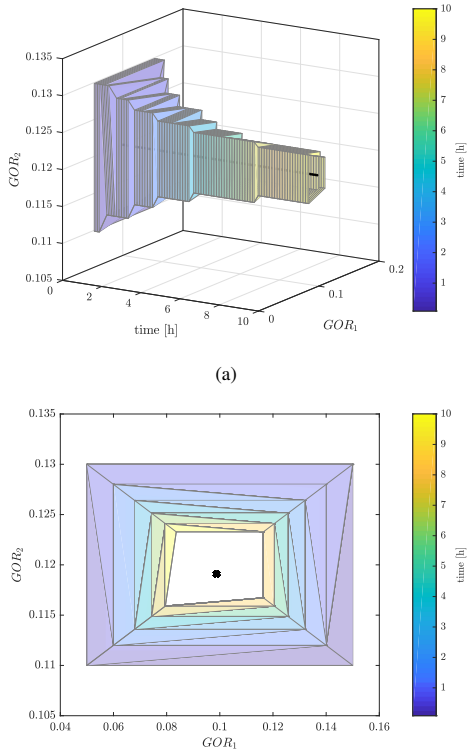
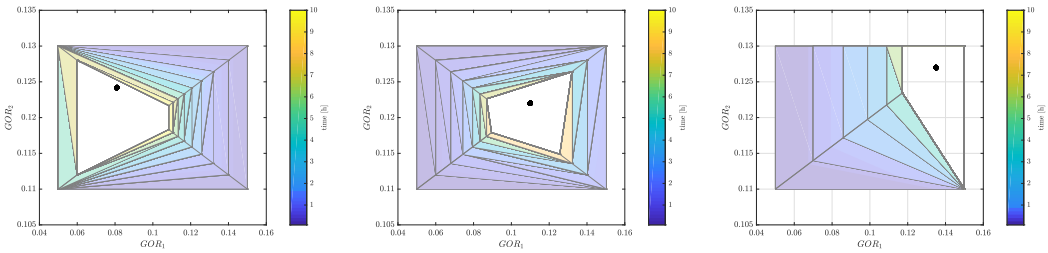


Fig. 5: Uncertainty space spanned by the different scenarios updated online, initiated with a box comprised of a combination of minimum and maximum GOR values. (a) 3-D view of the uncertainty subspace over time. (b) corresponding 2-D view. The true realization is marked by '$\times$'

the conservativeness also reduces gradually. This can be clearly seen in the plot in Fig.4, where the total gas produced is utilizing more of the available capacity to increase the total oil production compared to the standard multistage MPC with a fixed scenario tree. Fig.5 shows the evolution of the discrete scenarios in the scenario subspace. It can be seen that the span of the uncertainty subspace is gradually reduced compared to the initial box uncertainty.

We then test the proposed approach for 30 different realizations of the uncertainty randomly selected from $\mathcal{U}$ to be the true realization in the plant, as shown in Fig.7 (bottom subplot). To evaluate the performance, we plot the integrated objective, which is the oil production rate integrated over a period of 10 hours for each simulation run and compare it with the multistage MPC with fixed scenario tree, see Fig.7 (top subplot).

It can be clearly seen that by updating the scenario tree and gradually shrinking the uncertainty space covered by the



(a) (b) (c)

Fig. 6: 2-D view of the uncertainty subspace for three different uncertainty realizations for (a) run number 8, (b) run number 18, (c) run number 12. The true realization of GOR is shown as a solid black '$\times$'

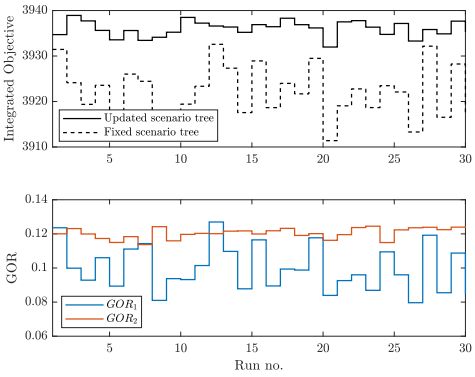


Fig. 7: Monte Carlo Simulations results with different realizations of the uncertain parameters.

scenario tree, the conservativeness can be further reduced, leading to less conservative operation. The uncertainty space covered by the updated scenario tree for run numbers 8, 12 and 18 are shown in Fig.6, to portray how the initial box uncertainty set is updated for different true realizations of the uncertain parameter.

V. Conclusion

When the time-invariant uncertain parameters cannot be estimated directly, methods that update the uncertainty characteristics may be useful in reducing the conservativeness. In this paper, we presented one such adaptive robust multistage MPC framework, where the scenario tree was updated online by computing recursive Bayesian weights for the different scenarios as summarised in Algorithm 1. It is important to note that this paper considers only the classs of systems with time-invariant parameters, where the idea is to narrow down the uncertainty using available measurements.

This is a simple, yet practical approach to updating the scenario trees based on the observed closed-loop performance. Using the gas lift optimization case study, we showed that by updating the scenario tree online, the conservativeness can be further reduced, leading to increased profits.

References

[1] P. J. Campo and M. Morari, "Robust model predictive control," in *American Control Conference, 1987*. IEEE, 1987, pp. 1021–1026.

[2] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[3] P. Scokaert and D. Mayne, "Min-max feedback model predictive control for constrained linear systems," *IEEE Transactions on Automatic control*, vol. 43, no. 8, pp. 1136–1142, 1998.

[4] S. Lucia, T. Finkler, and S. Engell, "Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty," *Journal of Process Control*, vol. 23, no. 9, pp. 1306–1319, 2013.

[5] D. Krishnamoorthy, B. Foss, and S. Skogestad, "Real time optimization under uncertainty - applied to gas lifted wells," *Processes*, vol. 4, no. 4, 2016.

[6] A. Verheyleweghen and J. Jäschke, "Framework for combined diagnostics, prognostics and optimal operation of a subsea gas compression system," *IFAC-PapersOnLine (IFAC World Congress)*, vol. 50, no. 1, pp. 15 916–15 921, 2017.

[7] M. Maiworm, T. Bäthge, and R. Findeisen, "Scenario-based model predictive control: Recursive feasibility and stability," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 50–56, 2015.

[8] M. Guay, V. Adetola, and D. DeHaan, *Robust and Adaptive Model Predictive Control of Nonlinear Systems*. Institution of Engineering and Technology, 2015.

[9] S. Thangavel, S. Lucia, R. Paulen, and S. Engell, "Dual robust nonlinear model predictive control: A multi-stage approach," *Journal of Process Control*, vol. 72, pp. 39–51, 2018.

[10] S. Lucia and R. Paulen, "Robust nonlinear model predictive control with reduction of uncertainty via robust optimal experiment design," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1904–1909, 2014.

[11] B. Aufderheide and B. W. Bequette, "Extension of dynamic matrix control to multiple models," *Computers & chemical engineering*, vol. 27, no. 8-9, pp. 1079–1096, 2003.

[12] N. N. Nandola and S. Bhartiya, "A multiple model approach for predictive control of nonlinear hybrid systems," *Journal of process control*, vol. 18, no. 2, pp. 131–148, 2008.

[13] D. Krishnamoorthy, B. Foss, and S. Skogestad, "A distributed algorithm for scenario-based model predictive control using primal decomposition." *IFAC-PapersOnLine (IFAC ADCHEM)*, vol. 51, no. 18, pp. 351–356, 2018.

[14] ——, "A primal decomposition algorithm for distributed multistage scenario model predictive control." *Journal of Process Control*, vol. I(n-Press), 2019.

[15] ——, "Gas lift optimization under uncertainty," *Computer Aided Chemical Engineering*, vol. 40, pp. 1753–1758, 2017.

[16] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, In Press, 2018.

Appendix M

Real-time optimization using surrogate optimizers

This chapter is based on the extended abstract presented in Foundations on Process Analytics and Machine Learning (FOPAM), Raleigh, NC.[1]

The process industry is one of the biggest producers of data, where a typical process generates hundreds of thousands of data in real-time. The process industry has used real-time data for decision making for more than 5 decades and has up to now been the leading industry in terms of intelligent use of data. However, in spite of many successes, there is still a large potential for further improvements in developing online process optimization tools to address future challenges with industry 4.0. With the recent surge of interest in machine learning, we discuss some future research directions on how machine learning tools can be used to address some of the challenges listed in Section 1.1.

M.1 Surrogate optimizers

Another challenge with online process optimization is the computational issues and numerical robustness (Challenge 3). For many large-scale problems, solvers may take a long time to converge to the optimal solution or, in some cases, may even fail to converge to an optimal solution. For example, with today's computational power, numerical optimization solvers for very simple chemical processes running on standard workstations (for example, 2.6GHz processor and 16GB RAM or similar configuration) typically converge in the time scale of seconds to several minutes. This makes it very difficult to run such numerical computations for more complex processes on embedded platforms and cloud computing, due to their limited computation capacity. This challenge is escalated further with the inclusion of

[1] Recipient of the Young Researcher NSF Travel Award.

integer decision variables, leading to mixed-integer and combinatorial optimization problems, which are very common in the field of process systems engineering. Currently, the computation cost for solving such problems are prohibitively large for online decision-making. Furthermore, there is an increasing interest and need to deploy such decision-support tools on mobile platforms and web-solutions in many industrial applications. Engineers and operators can clearly benefit if autonomous decision-support tools such as optimization, process control, soft sensors etc. can also be made available on cloud platforms.

In order to avoid solving numerical optimization problem online, we aim to approximate computationally expensive optimization problems using machine-learning algorithms. Instead of developing surrogate models that will be used in the optimizer, we propose to build "surrogate optimizers" that approximate the numerical optimization solvers.

In this chapter, we briefly discuss these ideas and present some surrogate optimizer structures along with some preliminary results in this direction.

M.1.1 Open-loop surrogate network based on measured disturbances

In this approach, the main idea is to solve the numerical optimization problems offline and use the trained network online to drive the process to its optimal operation. By doing so, we effectively move the computationally expensive optimization problems offline and the trained network approximates the numerical optimization solver and acts as the surrogate for online optimization. In other words, we train a neural network to map the relation between disturbances d to the optimal setpoints, such that given the current disturbance, the trained network provides the corresponding optimal solution as output as shown in Fig. M.1a. This structure approximates the optimization block from Fig. 2.1. This is analogous to multi-parametric programming, where the models are used offline to generate an optimal solution space as a function of the disturbances/parameters [138]. Here, instead of using a pre-computed solution space using first-principle models, we use a neural network to map the relation between the various disturbances to the optimal setpoints.

M.1.2 Closed-loop surrogate network based on available measurements

Alternatively, models can also be trained to directly map the relation between the available measurements to the optimal setpoints in the same fashion as training a model from the disturbances to the optimal setpoints as shown in Fig. M.1b. In this structure, the neural network approximates both the model update and the optimization block from Fig. 2.1. This approach would be preferred over the open-loop approach in Fig. M.1a, since this structure does not require the disturbances to be measured/estimated, which can be advantageous in many systems where disturbance measurements are not available.

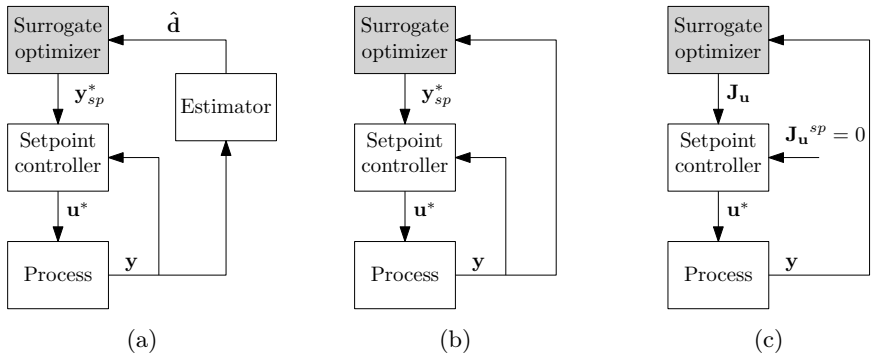


Figure M.1: Surrogate optimizers: (a) Open-loop surrogate optimization structure using neural network that maps measured disturbances to optimal setpoint. (b) Closed-loop surrogate optimization structure using neural network that maps measurements to optimal setpoint. (c) Gradient-based surrogate optimization structure using a neural network to estimate the steady-state cost gradient.

M.1.3 Gradient-based approach

In this thesis, we had proposed different algorithms to estimate the steady-state gradient directly from the measurements to optimize the process (cf. Chapters 3 and 5). In the same spirit, another approach is to train a surrogate network that directly maps the measurements to the steady-state cost gradient. This approach does not require solving numerical optimization problems either offline or online. The trained model is then used online to estimate the steady-state cost gradient using real-time measurements. Optimal operation can then be achieved by simply controlling the estimated gradients to a constant setpoint of zero using simple feedback controllers, thereby achieving the necessary condition of optimality. A schematic representation of the proposed surrogate optimization approach is shown in Fig. M.1c.

M.1.4 Simulation results

The three different surrogate optimizer structures are applied to the same CSTR model described by (3.9). The optimization problem was solved offline for various realizations of the disturbances to generate the training data. As a proof of concept, a shallow neural network with 10 neurons was trained and validated using this data to approximate the optimization problem from the disturbance to the optimal setpoints. The trained neural network was then used online to optimize the process using the structure in Fig. M.1a. The process was simulated with disturbances in the feed concentrations as shown in Fig. M.2. Note that in this simulation, we assume that the disturbances are measured. The cost function using the proposed approach was compared with the ideal steady-state optimal cost (solid black lines) and is shown in Fig. M.2 (solid red lines).

We then use the training data to train and validate a neural network from the available measurements to the optimal setpoints to build a closed-loop surrogate

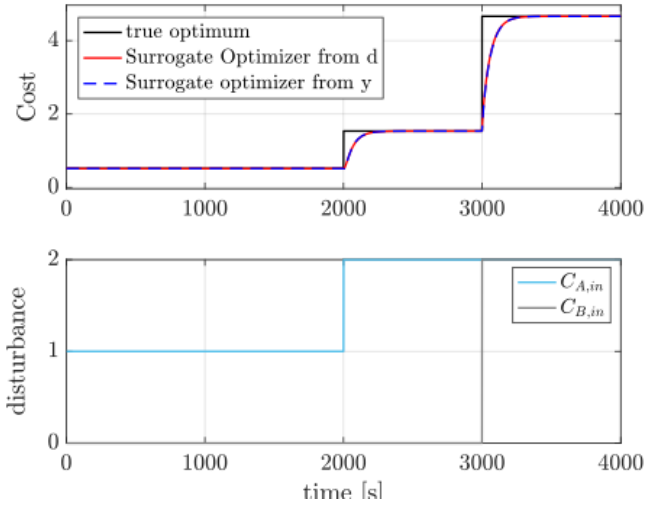


Figure M.2: Cost obtained using the surrogate optimizer structure from Fig. M.1a (solid red) and Fig. M.1b (dashed blue) compared with the ideal optimal cost (solid black)

network using available measurements as shown in Fig. M.1b. The process was simulated for the same disturbances and in this case, we assume that the disturbance measurements are not available. The simulation results using this approach is also shown in Fig. M.2 (dashed blue lines) to compare the open-loop and closed-loop surrogate approaches and we see that the system is able to reach the optimum despite the unmeasured disturbances.

The gradient-based surrogate optimization approach was also tested on the same CSTR case example. The process was simulated to generate measurement data for different disturbance realizations. The measurement data along with the corresponding steady-state cost gradient evaluated offline was used to train and validate a neural network with 10 neurons. The trained model was then used to estimate the steady-state gradient, which was controlled to a constant setpoint of zero using a PI controller. The cost function along with the estimated gradient for the same disturbance as in the previous case is shown in Fig. M.3(solid blue lines) along with the ideal steady-state cost (solid black lines).

M.2 Chapter summary

With the vast amount of data that is generated in the process industry, there is a clear need for new technologies and approaches to integrate and interpret this data more efficiently in order to drive faster and more accurate decisions. Process optimization using new paradigms emerging from the computer science and cybernetics domain such as data machine learning, artificial intelligence and data analytics will be one of the key factors affecting the success of the modern

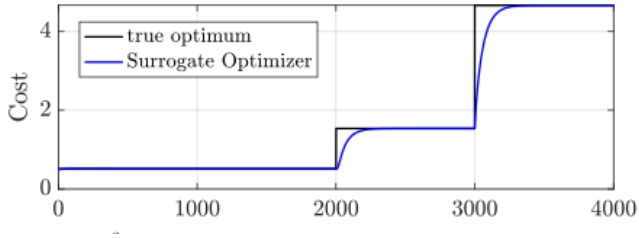


Figure M.3: Cost obtained using the gradient-based surrogate optimizer structure from Fig.4 (solid blue) compared with the ideal optimal cost (solid black)

process industry.

Appendix N

Fast economic model predictive control of gas-lifted wells

As discussed in this thesis, solving dynamic optimization problems in the context of economic NMPC may be computationally intensive. This leads to non-negligible computation delay that can degrade the performance. To address the problem of computation delay, we can use the advanced-step path following economic NMPC framework. Here, the computationally expensive NLP problem is solved offline, and the NLP sensitivities are used to solve a path-following QP problem online to update the solution. This is similar to the path-following sensitivity approach used in Chapter 10, where the idea is to parameterize the NLP w.r.t. the initial condition constraint (i.e. $\boldsymbol{p} = \mathbf{x}_0$).

This appendix contains the paper showing the application of the fast advanced-step economic NMPC [178] to an oil production optimization problem.

- Paper published in 2018 IFAC International Workshop on Automatic Control in Offshore Oil and Gas Production (OOGP), Esbjerg, Denmark

Fast Economic Model Predictive Control for a Gas Lifted Well Network ⋆

Eka Suwartadi * **Dinesh Krishnamoorthy** * **Johannes Jäschke** *

* *Dept. of Chemical Engineering, Norwegian Univ. of Science & Technology , NO-7491 Trondheim , (e-mail: eka.suwartadi@ntnu.no, dinesh.krishnamoorthy@ntnu.no, and jaschke@ntnu.no).*

Abstract: This paper considers the optimal operation of an oil and gas production network by formulating it as an economic nonlinear model predictive control (NMPC) problem. Solving the associated nonlinear program (NLP) can be computationally expensive and time consuming. To avoid a long delay between obtaining updated measurement information and injecting the new inputs in the plant, we apply a sensitivity-based predictor-corrector path-following algorithm in an advanced-step NMPC framework. We demonstrate the proposed method on a gas-lift optimization case study, and compare the performance of the path-following economic NMPC to a standard economic NMPC formulation.

Keywords: Sensitivity-based NMPC, Path-following algorithm, Dynamic optimization, Production optimization, Gas-lift optimization

1. INTRODUCTION

Operation of an oil and gas production network involves making daily control decisions in order to maximize the revenue while satisfying process and operating constraints. This is known as short-term production optimization or daily production optimization (DPO). Mathematical tools are increasingly used in production optimization to compute the optimal decision variables based on a digital representation of the system. The use of mathematical optimization for daily production optimization has enabled production increases in the range of 1-4% (Stenhouse et al., 2010). A comprehensive survey of optimization tools used for production optimization can be found in Bieker et al. (2007). DPO problems may be formulated as either static optimization problems or dynamic optimization problems. Useful discussions on the static and dynamic formulations for DPO are provided in Foss et al. (2017), where the authors note that DPO applications may benefit from dynamic formulations in some cases, particularly when the profit is strongly affected by short-term control decisions.

In this paper, we consider such a dynamic production optimization problem, where the control and optimization layers are tightly integrated. In this case, the production optimization problem is formulated as an economic nonlinear model predictive control (ENMPC) problem. The key idea in ENMPC is to use a single dynamic optimization problem to control the process, and to optimize the economic performance simultaneously. By doing so, the economic cost is optimized also during transient operation of the system. In the face of volatile oil prices and competitive markets, optimizing the transients to maximize profits has become more and more desirable.

⋆ The authors gratefully acknowledge the financial support from Research Council of Norway through IKTPLUSS Young Researcher Grant and SFI SUBPRO programs.

For large-scale production networks, the economic NMPC problem formulation may become rather large with several hundred decision variables. For example, the production optimization of the Troll oil field in the Norwegian continental shelf includes more than one hundred subsea wells (Hauge et al., 2005). This leads to optimization problems that are computationally very intensive. Moreover, nonlinear models are typically used for economic optimization, which further adds to the computational complexity. Computational cost has been a prohibitive factor for the widespread implementation of dynamic optimization in the oil and gas industry, despite being a promising approach, as noted in Forbes et al. (2015). Solving the large-scale nonlinear programming (NLP) problem may take a significant amount of time and this computational delay can lead to performance degradation or even to closed-loop instabilities (Findeisen and Allgöwer, 2004). Hence there is a clear need for numerical methods that make it possible to obtain updated solutions to the large-scale NLP in very short time.

In this paper we address this issue by applying a sensitivity-based path-following algorithm in an advanced-step NMPC framework (Suwartadi et al., 2017) to a gas-lift optimization problem. The algorithm has the advantage that it can handle large parameter changes, and still give an accurate approximation of the solution. Several sensitivity-based approaches have been proposed to address the issue of computation time. See e.g. Diehl et al. (2005), Zavala and Biegler (2009), Jäschke et al. (2014). A review article on fast NMPC schemes is given by Wolf and Marquardt (2016).

At each sample time, the NMPC optimization problems are identical, except for one time varying parameter, namely, the initial state. All the fast sensitivity approaches capitalize on this property. When new measurements of the states become available, these approaches use the

sensitivity of the optimal solution that was computed at a previous time step to obtain fast approximate solutions to the new resulting nonlinear optimization problem. Such approximations enable fast implementation of the optimal input in the plant in a minimal delay.

The remainder of the paper is organized as follows. The daily production optimization problem for a gas lifted well network is introduced in Section 2. The sensitivity-based economic NMPC with the path-following approach is presented in Section 3. Simulation results from a gas-lift optimization case study are presented in Section 4, before concluding the paper in Section 5.

2. PROBLEM FORMULATION

An offshore oil and gas production network typically consists of several wells that produce to a common processing facility. The reservoir fluid enters through the well bore of each well and is produced to a topside processing facility via a common production manifold as shown in Fig.1. In some wells, the reservoir pressure may not be sufficient to lift the fluids to the surface. In such cases, artificial lift methods are employed to boost the production from the wells. In this paper, we consider the gas lift method, which is a commonly used artificial lift technology. In gas lifted wells, compressed gas is injected into the well tubing via the annulus to reduce the mixture density. This results in reduced hydrostatic pressure, and hence boosted production. However, higher gas injection rates also increase the frictional pressure drop. The oil production starts to decline if the effect of the frictional pressure drop is dominant over the effect of the hydrostatic pressure drop.

Production from a cluster of $\mathcal{N} = \{1, \ldots, n_w\}$ gas lifted wells can be modelled as a semi-explicit index-1 DAE system of the form,

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{z}, \mathbf{u}), \tag{1a}$$
$$0 = g(\mathbf{x}, \mathbf{z}, \mathbf{u}), \tag{1b}$$

where $f(\mathbf{x}, \mathbf{z}, \mathbf{u})$ is the set of differential equations and $g(\mathbf{x}, \mathbf{z}, \mathbf{u})$ is the set of algebraic equations. The dynamics arise in the model due to the mass balances for oil and gas phases in each well and the riser. Algebraic equations are used to describe the densities, pressures and flow rates for each well and the riser, as described in detail by Krishnamoorthy et al. (2016).

The inlet separator in the topside processing facility sets the downstream boundary conditions which are typically kept at a constant pressure by tight regulatory control. The upstream boundary conditions are set by the reservoir inflow conditions. The DPO problem is concerned with the production network exposed to these upstream and downstream conditions, and hence the reservoir model and topside processing facilities are not included in the production optimization problem.

Gas lifted wells are often controlled by adjusting the gas lift injection rate of each well. The production network is also subject to process and operating constraints. For example, the total gas processing capacity in the topside facilities may be constrained, or the rate of compressed gas injection may be limited. The optimization problem then involves computing the optimal gas lift injection rates

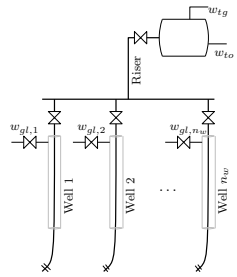


Fig. 1. Schematic representation of a gas lifted production network with n_w wells producing to a common topside processing facility.

for each well such that the operating profits are maximized subject to the network processing and operating constraints.

Before this can be formulated as an economic NMPC problem, the infinite dimensional problem is discretized into finite horizon optimal control problem using direct collocation method. The discretized system dynamics at any time instant l can be expressed as,

$$\mathbf{F}\left(\boldsymbol{\chi}_{l+1}, \boldsymbol{\chi}_l, \boldsymbol{\zeta}_l, \boldsymbol{\nu}_l\right) = 0. \tag{2}$$

A detailed explanation on how the system is discretized into a nonlinear programming problem using direct collocation can be found in Krishnamoorthy et al. (2016).

The economic NMPC problem can then be formulated as

$$\begin{aligned}
\mathcal{P}_N\left(\mathbf{x}_k\right): \min_{\boldsymbol{\chi}_l, \boldsymbol{\zeta}_l, \boldsymbol{\nu}_l} \; & \Psi\left(\boldsymbol{\chi}_{k+N}, \boldsymbol{\zeta}_{k+N}\right) + \sum_{l=k}^{k+N-1} \psi\left(\boldsymbol{\chi}_l, \boldsymbol{\zeta}_l, \boldsymbol{\nu}_l\right) \\
\text{s.t.} \; & \mathbf{F}\left(\boldsymbol{\chi}_{l+1}, \boldsymbol{\chi}_l, \boldsymbol{\zeta}_l, \boldsymbol{\nu}_l\right) = 0, \quad \forall l \in \mathcal{N} \\
& \mathbf{G}\left(\boldsymbol{\chi}_l, \boldsymbol{\zeta}_l, \boldsymbol{\nu}_l\right) \leq 0, \quad \forall l \in \mathcal{N} \\
& \left(\boldsymbol{\chi}_l, \boldsymbol{\zeta}_l, \boldsymbol{\nu}_l\right) \in \mathcal{Z}, \quad \forall l \in \mathcal{N} \\
& \left(\boldsymbol{\chi}_{k+N}, \boldsymbol{\zeta}_{k+N}\right) \in \mathcal{X}_f \\
& \boldsymbol{\chi}_0 = \mathbf{x}_k.
\end{aligned} \tag{3}$$

Here $\boldsymbol{\chi}_l \in \mathbb{R}^{n_\chi}$, $\boldsymbol{\nu}_l \in \mathbb{R}^{n_\nu}$, and $\boldsymbol{\zeta}_l \in \mathbb{R}^{n_\zeta}$ represent the predicted state, control input, and algebraic at time instance l, respectively for all l belonging to $\mathcal{N} = \{k, \ldots, k+N\}$. The constraints include the system dynamics as equality constraint, nonlinear inequality constraints $\mathbf{G}$, and the equality constraint of the initial predicted state $\boldsymbol{\chi}_0$ equal to the actual state $\mathbf{x}_k \in \mathbb{R}^{n_\chi}$ obtained from measurement data. The path constraint confines the predicted state, algebraic, and control inside the set $\mathcal{Z}$, and the final state and algebraic variables $\left(\boldsymbol{\chi}_{k+N}, \boldsymbol{\zeta}_{k+N}\right)$ are constrained to lie in the set $\mathcal{X}_f$. The objective function comprises the final cost $\Psi\left(\boldsymbol{\chi}_{k+N}, \boldsymbol{\zeta}_{k+N}\right) \in \mathcal{C}^2 : \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_\zeta} \to \mathbb{R}$ and the stage cost $\psi\left(\boldsymbol{\chi}_l, \boldsymbol{\zeta}_l, \boldsymbol{\nu}_l\right) \in \mathcal{C}^2 : \mathbb{R}^{n_\chi} \times \mathbb{R}^{n_\zeta} \times \mathbb{R}^{n_\nu} \to \mathbb{R}$.

In the gas lifted well problem, the stage cost is defined as

$$\psi\left(\boldsymbol{\chi}_l, \boldsymbol{\zeta}_l, \boldsymbol{\nu}_l\right) := \sum_{i=1}^{n_w} \left(-r_o w_{po,i} + r_{gl} w_{gl,i}\right), \tag{4}$$

where r_o is the oil price, r_{gl} is the cost for gas lift injection, and n_w denotes the number of production wells. The nonlinear inequality constraints enforce the total gas capacity constraints,

$$\mathbf{G}\left(\boldsymbol{\chi}_l, \boldsymbol{\zeta}_l, \boldsymbol{\nu}_l\right) := \sum_{i=1}^{n_w} \left(w_{pg,i}\right) - w_g^{max}. \tag{5}$$

The path constraints are in the form of bound constraints for the state, control inputs as well as the algebraic variables

$$\begin{pmatrix} \underline{\boldsymbol{\chi}} \\ \underline{\boldsymbol{\zeta}} \\ \underline{\boldsymbol{\nu}} \end{pmatrix} \leq \begin{pmatrix} \boldsymbol{\chi}_l \\ \boldsymbol{\zeta}_l \\ \boldsymbol{\nu}_l \end{pmatrix} \leq \begin{pmatrix} \overline{\boldsymbol{\chi}} \\ \overline{\boldsymbol{\zeta}} \\ \overline{\boldsymbol{\nu}} \end{pmatrix}. \tag{6}$$

The notations $\bar{\cdot}$ and $\underline{\cdot}$ represent the upper and lower bound for the corresponding variable. It is easily possible to limit the state variables with the bound constraints since the system dynamics are transcribed using direct collocation, in which the state, algebraic, and control are treated as optimization variables.

Once the economic NMPC problem is formulated, it can be solved in a receding horizon fashion. At each sample time k, the state measurement or estimate $\mathbf{x}_k$ is assigned as the initial state for the optimization problem (3). The optimization problem is solved to compute the optimal input trajectory $\boldsymbol{\nu}^*_{[l,l+N]}$ over the prediction horizon. The first step of the optimal control sequence is implemented in the plant, i.e. $\mathbf{u}^*_k = \boldsymbol{\nu}^*_1$. At the next time step $k+1$, new measurements of the state $\mathbf{x}_{k+1}$ are obtained and the optimization procedure is repeated, hence enabling closed-loop implementation.

We consider a full-state feedback control structure. The measurements from the plant are used for estimating the states. The estimated states are then used for full state feedback for the economic NMPC, which computes the optimal inputs for the plant. The state estimation is performed online by an extended Kalman filter (EKF). The EKF is implemented in discrete time as described in Krishnamoorthy et al. (2017). Commonly available measurements such as annulus pressure, wellhead pressure and down hole pressure for each well along with the riser head pressure, manifold pressure and total oil and gas production rates are used as measurements for the EKF.

In an ideal case, the optimization problem (3) would be solved instantly and the optimal input would be implemented in the plant without time delay. In practice, however, there is always some delay between the state measurement/estimation and the implementation of the optimal control input. This is mainly because of the computational time required to solve the optimization problem (3). For many linear MPC applications, the computational delay is rather small and can be neglected. However, for large-scale nonlinear systems, as the number of optimization variable increases, solving the optimization problem requires more time, and the computational delay may no longer be neglected.

3. FAST ECONOMIC NMPC

To address the issue of computational delay, fast sensitivity-based NMPC approaches have been developed. One such approached is the advanced-step NMPC (asNMPC) introduced by Zavala and Biegler (2009), where at time k, the NMPC problem is solved with the predicted state value of time $k+1$ instead of using the measured/estimated state $\mathbf{x}_k$. An approximation of the NLP solution is then computed using a single sensitivity step (Zavala and Biegler, 2009), or a path-following approach (Jäschke et al., 2014; Suwartadi et al., 2017) as soon as the measurement $\mathbf{x}_{k+1}$ is available at time $k+1$. In this work we use the predictor-corrector path-following algorithm based on Suwartadi et al. (2017), described below.

Since the optimization problem (3) differs only in the initial state variable, which is denoted as the equality constraint $\boldsymbol{\chi}_0 = \mathbf{x}_k$, from one NMPC iteration to another, the problem can be cast as the following parametric NLP problem

$$\begin{aligned} \min_{\boldsymbol{X}} \quad & \mathcal{J}\left(\boldsymbol{X}, \boldsymbol{p}\right) \\ \text{s.t. } & c_i\left(\boldsymbol{X}, \boldsymbol{p}\right) = 0, i \in \mathcal{E}, \\ & c_i\left(\boldsymbol{X}, \boldsymbol{p}\right) \leq 0, i \in \mathcal{I}, \end{aligned} \tag{7}$$

where $\boldsymbol{X} \in \mathbb{R}^{n_{\boldsymbol{X}}}$ is the primal variable, $\boldsymbol{p} \in \mathbb{R}^{n_{\boldsymbol{p}}}$ is the parameter, and the objective function $\mathcal{J} : \mathbb{R}^{n_{\boldsymbol{X}}} \times \mathbb{R}^{n_{\boldsymbol{p}}} \longrightarrow \mathbb{R}$. The equality and inequality constraints $c : \mathbb{R}^{n_{\boldsymbol{X}}} \times \mathbb{R}^{n_{\boldsymbol{p}}} \longrightarrow \mathbb{R}^{n_c}$ are represented by the sets $\mathcal{E} = \{1, \ldots, m\}$ and $\mathcal{I} = \{m+1, \ldots, n_c\}$, respectively.

We define Lagrangian of the problem (7) as

$$\mathcal{L}\left(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}\right) := \mathcal{J}\left(\boldsymbol{X}, \boldsymbol{p}\right) + \boldsymbol{\lambda}^T c\left(\boldsymbol{X}, \boldsymbol{p}\right), \tag{8}$$

where $\boldsymbol{\lambda}$ is the dual variable or Lagrange multiplier. The first-order optimality (*Karush-Kuhn-Tucker (KKT)*) conditions are

$$\begin{aligned} \nabla_{\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}) &= 0, \\ c_i\left(\boldsymbol{X}, \boldsymbol{p}\right) &= 0, \, i \in \mathcal{E}, \\ c_i\left(\boldsymbol{X}, \boldsymbol{p}\right) &\leq 0, \, i \in \mathcal{I}, \\ \boldsymbol{\lambda}_i^T c_i\left(\boldsymbol{X}, \boldsymbol{p}\right) &= 0, \, i \in \mathcal{I}, \\ \boldsymbol{\lambda}_i &\geq 0, \, i \in \mathcal{I}. \end{aligned} \tag{9}$$

Active inequality constraints are denoted by the set $\mathcal{A}\left(\boldsymbol{X}, \mathbf{p}\right) = \{c_i\left(\boldsymbol{X}, \mathbf{p}\right) = 0, i \in \mathcal{I}\}$. For a given multiplier $\boldsymbol{\lambda}$ and $\boldsymbol{X}$ that satisfies the KKT conditions (9), the active inequality set $\mathcal{A}\left(\boldsymbol{X}, \mathbf{p}\right)$ has two subsets, which are a weakly active set $\mathcal{A}_0\left(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}\right) = \{i \in \mathcal{A}\left(\boldsymbol{X}, \boldsymbol{p}\right) \mid \boldsymbol{\lambda}_i = 0\}$ and a strong active set $\mathcal{A}_+\left(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}\right) = \{i \in \mathcal{A}\left(\boldsymbol{X}, \boldsymbol{p}\right) \mid \boldsymbol{\lambda}_i > 0\}$.

Furthermore, we define the optimality residual as

$$\eta\left(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}\right) = \left\| \begin{pmatrix} \nabla_{\boldsymbol{X}} \mathcal{J}\left(\boldsymbol{X}, \boldsymbol{p}\right) + \nabla_{\boldsymbol{X}} c\left(\boldsymbol{X}, \boldsymbol{p}\right) \boldsymbol{\lambda} \\ c\left(\boldsymbol{X}, \boldsymbol{p}\right)_{\mathcal{E}} \\ \left[\min\left(c\left(\boldsymbol{X}, \boldsymbol{p}\right), \boldsymbol{\lambda}\right)\right]_{\mathcal{I}} \end{pmatrix} \right\|_{\infty}. \tag{10}$$

Here, we assume that the linear independent constraint qualification (LICQ) is satisfied at any point $\boldsymbol{X}$ that satisfies the KKT conditions (9).

Definition 1. (LICQ). Given a vector $\boldsymbol{p}$ and a point $\boldsymbol{X}$, the *linear independence constraint qualification* (LICQ) holds at $\boldsymbol{X}$ if the set of vectors $\{\nabla_{\boldsymbol{X}} c_i\left(\boldsymbol{X}, \boldsymbol{p}\right)\}_{i \in \mathcal{E} \cup \mathcal{A}(\boldsymbol{X}, \boldsymbol{p})}$ are linearly independent.

We also assume that the strong second order sufficient condition is also satisfied.

Definition 2. (SSOSC). The *strong second order sufficient condition* (SSOSC) holds at $(\boldsymbol{X}, \boldsymbol{\lambda})$ that satisfies the KKT conditions, if $\boldsymbol{d}^T \nabla^2_{\boldsymbol{X}\boldsymbol{X}} \mathcal{L}\left(\boldsymbol{X}, \boldsymbol{p}, \boldsymbol{\lambda}\right) \boldsymbol{d} > 0$ for all $\boldsymbol{d} \neq 0$ such that $\nabla_{\boldsymbol{X}} c_i\left(\boldsymbol{X}, \boldsymbol{p}\right)^T \boldsymbol{d} = 0$ for $i \in \mathcal{E} \cup \mathcal{A}_+\left(\boldsymbol{X}, \boldsymbol{p}, \boldsymbol{\lambda}\right)$.

We are now ready to state the result for sensitivity of the NLP, where $\boldsymbol{X}^*(\boldsymbol{p})$ and $\boldsymbol{\lambda}^*(\boldsymbol{p})$ are the primal and dual solutions of (7), respectively.

Theorem 3. Let $\mathcal{J}$, c be twice continuously differentiable in $\boldsymbol{p}$ and $\boldsymbol{X}$ near a solution of (7) $(\boldsymbol{X}^*, \boldsymbol{p}_0)$, and let LICQ and SSOSC hold at $(\boldsymbol{X}^*, \boldsymbol{p}_0)$. Then the solution $(\boldsymbol{X}^*(\boldsymbol{p}), \boldsymbol{\lambda}^*(\boldsymbol{p}))$ is Lipschitz continuous in a neighborhood of $(\boldsymbol{X}^*, \boldsymbol{\lambda}^*, \boldsymbol{p}_0)$, and the solution function $(\boldsymbol{X}^*(\boldsymbol{p}), \boldsymbol{\lambda}^*(\boldsymbol{p}))$ is directionally differentiable. Moreover, the directional derivative uniquely solves the following quadratic problem:

$$\begin{aligned}
\min_{\Delta \boldsymbol{X}} \quad & \frac{1}{2} \Delta \boldsymbol{X}^T \nabla^2_{\boldsymbol{X}\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}^*, \boldsymbol{p_0}, \boldsymbol{\lambda}^*) \Delta \boldsymbol{X} \\
& + \Delta \boldsymbol{X}^T \nabla_{\boldsymbol{p}\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}^*, \boldsymbol{p_0}, \boldsymbol{\lambda}^*) \Delta \boldsymbol{p} \qquad (11) \\
\text{s.t.} \quad & \\
& \nabla_{\boldsymbol{X}} c_i(\boldsymbol{X}^*, \boldsymbol{p_0})^T \Delta \boldsymbol{X} \\
& + \nabla_{\boldsymbol{p}} c_i(\boldsymbol{X}^*, \boldsymbol{p_0})^T \Delta \boldsymbol{p} = 0 \qquad i \in \mathcal{A}_+ \cup \mathcal{E}, \\
& \nabla_{\boldsymbol{X}} c_j(\boldsymbol{X}^*, \boldsymbol{p_0})^T \Delta \boldsymbol{X} \\
& + \nabla_{\boldsymbol{p}} c_j(\boldsymbol{X}^*, \boldsymbol{p_0})^T \Delta \boldsymbol{p} \leq 0 \qquad j \in \mathcal{A}_0.
\end{aligned}$$

Proof. See Robinson (1980) and (Bonnans and Shapiro, 1998, Section 5.2). □

Instead of solving a full NLP problem, one can solve a quadratic programming (QP) problem (11) to compute a first-order approximation of the solution to the optimization problem (3) in the vicinity of perturbation $\boldsymbol{p}_0$. We refer the QP (11) to as *pure-predictor QP*. Note that there is no requirement of strict complimentary in the theorem, allowing for active-set changes.

To improve the approximation of the solution, we introduce a corrector term in the objective function, and taking into account that the parameter $\boldsymbol{p}$ enters linearly in the problem, we can formulate the following QP (Suwartadi et al., 2017; Kungurtsev and Diehl, 2014)

$$\begin{aligned}
\min_{\Delta \boldsymbol{X}} \quad & \frac{1}{2} \Delta \boldsymbol{X}^T \nabla^2_{\boldsymbol{X}\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p}, \boldsymbol{\lambda}^*) \Delta \boldsymbol{X} \\
& + \nabla_{\boldsymbol{X}} \mathcal{J}^T \Delta \boldsymbol{X} \qquad (12) \\
\text{s.t.} \quad & \\
& c_i(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p})^T \Delta \boldsymbol{p} \\
& + \nabla_{\boldsymbol{X}} c_i(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p})^T \Delta \boldsymbol{X} = 0 \ , i \in \mathcal{A}_+ \cup \mathcal{E}, \\
& c_j(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p})^T \Delta \boldsymbol{p} \\
& + \nabla_{\boldsymbol{X}} c_j(\boldsymbol{X}^*, \boldsymbol{p}_0 + \Delta \boldsymbol{p})^T \Delta \boldsymbol{X} \leq 0 \ , j \in \mathcal{I} \setminus \mathcal{A}_+.
\end{aligned}$$

We denote the QP above as *predictor-corrector QP*, see Algorithm 1. This QP formulation provides a reasonably good approximation of the NLP solution in a small neighborhood of $\boldsymbol{p}_0$. To allow large perturbation (large $\Delta \boldsymbol{p}$), we employ a path-following approach, that is, to solve a series of QP problems. This is analogous to Euler integration scheme for ordinary differential equations. The parameter $\boldsymbol{p}$ is updated according to $\boldsymbol{p}(t_j) = (1 - t_j)\boldsymbol{p}_0 + t_j \boldsymbol{p}_f$, where $t_0 = 0$ until it reaches $t = 1$, that is $t_0 = 0, t_1 < t_2 \ldots < t_j < t_{j+1} \ldots < t_{end} = 1$. The parameter $\boldsymbol{p}_f$ corresponds to new measurement data. During the course of path-following iteration, the primal variable $\Delta \boldsymbol{X}$ and dual variable $\Delta \boldsymbol{\lambda}$ are updated for each t_j. The optimality residual condition η (10) is computed and compared against its maximum tolerance η_{max}. If necessary, the stepsize Δt is reduced to satisfy $\eta < \eta_{max}$. The method is implemented as a subroutine in Algorithm 1, denoted as `QP_PC_PF`.

As described in Suwartadi et al. (2017) and Jäschke et al. (2014), we apply the path-following QP within the advanced-step NMPC (*asNMPC*) framework (see Zavala and Biegler (2009)) and refer the method to as *pf-NMPC*. The *pf-NMPC* procedure includes the following three steps.

(1) Solve the NLP problem (3) at time k constraining the initial state value to the predicted state at $k+1$.
(2) When the measurement $\mathbf{x}_{k+1}$ becomes available at time $k+1$, compute an approximation of the NLP solution (3) using the QP (12) in a path-following manner.
(3) Implement the optimal control input and update $k \leftarrow k+1$ and repeat from Step 1.

A sketch of the pf-NMPC procedure is described in Algorithm 1.

4. SIMULATION RESULTS

In this section we use a gas lifted well network with $n_w = 2$ wells to demonstrate the pf-NMPC controller and compare its results against the ideal NMPC (iNMPC) controller which solves the problem (3) using an NLP solver. All simulations are done in `MATLAB` using `CasADi` version 3.2.0 as algorithmic differentiation tool (Andersson, 2013) where `IPOPT` (Wächter and Biegler, 2006) is included as an NLP solver. We use the QP solver from `TOMLAB MINOS` (Murtagh and Saunders, 1982).

First, we run a steady-state optimization with total gas production capacity constraint equals to 9.5 kg/s. The optimized steady-state control inputs, algebraic and state variables are incorporated in the stage cost regularization terms, i.e.,

$$\begin{aligned}
\psi_m(\boldsymbol{\chi}_l, \boldsymbol{\nu}_l, \boldsymbol{\zeta}_l) = \psi(\boldsymbol{\chi}_l, \boldsymbol{\nu}_l, \boldsymbol{\zeta}_l) \qquad (13) \\
+ \alpha(\|\boldsymbol{\chi}_l - \mathbf{x}_s\|, \|\boldsymbol{\nu}_l - \mathbf{u}_s\|, \|\boldsymbol{\zeta}_l - \mathbf{z}_s\|),
\end{aligned}$$

where $\mathbf{x}_s$, $\mathbf{u}_s$, and $\mathbf{z}_s$ are the steady-state state variable, control input, algebraic variable respectively.

Next, we run the NMPC controllers and initiate the system with an initial condition far away from the optimal point. The NMPC controllers are implemented with sampling time of 5 minutes with a prediction horizon of 2 hours yielding 3014 optimization variables and 2966 nonlinear constraints in the NLP.

We set $\eta_{max} = 10^{-5}$ and initial $\Delta t = 1.0$ for the pf-NMPC controller. Note that we use an adaptive steplength strategy for Δt in the Algorithm 1 meaning that the steplength Δt may be reduced in case $\eta_{max} > 10^{-5}$.

4.1 Comparison of Open-loop Optimization Results

We compare the open loop solutions from the ideal NMPC and pf-NMPC controllers at time t = 10 minute (at second NMPC iteration). The results are shown in Figure 2 in which the total oil and gas production are depicted. The solutions from pf-NMPC accurately track those of the ideal

Algorithm 1 Economic pf-NMPC algorithm

Input: initial state $\mathbf{x}_0$, initial Δt, and η_{max}.

for $k = 0, 1, 2, \ldots$ **do**
  $[\boldsymbol{X}^*, \boldsymbol{\lambda}^*] \leftarrow$ solution NLP $\mathcal{P}_N\left(\boldsymbol{\chi}_{k+1}\right)$ for $k+1$.
  if a measurement of $\mathbf{x}_{k+1}$ is available **then**
    Set $\boldsymbol{p}_0 \leftarrow \boldsymbol{\chi}_{k+1}$.
    Set $\boldsymbol{p}_f \leftarrow \mathbf{x}_{k+1}$.
    $\boldsymbol{X} \leftarrow$ QP_PC_PF$(\boldsymbol{X}^*, \boldsymbol{\lambda}^*, \boldsymbol{p}_0, \boldsymbol{p}_f, \Delta t)$
    Extract first input value from $\boldsymbol{X}$ and inject to the plant as $\mathbf{u}_k$.
    Update initial state $\mathbf{x}_0 \leftarrow \mathbf{x}_{k+1}$.
    Set $k+1 \leftarrow k$.
  end if
end for

function QP_PC_PF$(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{p}_0, \boldsymbol{p}_f, \Delta t)$
  Define parameter γ satisfying $0 < \gamma < 1$.
  Determine $\mathcal{A}_+$.
  Set parameter η_{max}.
  Set $j \leftarrow 0$.
  Set $t_j \leftarrow 0$.
  while $t_j < 1$ **do**
    Compute $\eta_j := \eta\left(\boldsymbol{X}_j, \boldsymbol{\lambda}_j, \boldsymbol{p}\left(t_j\right)\right)$.
    if QP is feasible **then** ▷ solve QP
      Compute $\eta_{j+\Delta} := \eta\left(\boldsymbol{X}_j + \Delta\boldsymbol{X},\ \Delta\boldsymbol{\lambda}, \boldsymbol{p}\left(t_{j+\Delta t}\right)\right)$.
      if $\eta_{j+\Delta} > \eta_{max}$ **then**.
        Decrease Δt. ▷ reduce QP stepsize
        $j \leftarrow j+1$.
      else
        $\boldsymbol{X} \leftarrow \boldsymbol{X} + \Delta\boldsymbol{X}$
        $\boldsymbol{\lambda} \leftarrow \Delta\boldsymbol{\lambda}$
        $t_{j+1} \leftarrow t_j + \Delta t$
        $\mathbf{p}\left(t_j\right) = \left(1 - t_j\right)\mathbf{p}_0 + t_j\mathbf{p}_f$
        if $\eta_{j+\Delta} < \eta_j^{1+\gamma}$ **then** ▷ very good step
          Increase Δt.
        end if
        Update $\mathcal{A}_+$. ▷ from QP's dual solution.
        $j \leftarrow j+1$.
      end if
    else
      Decrease Δt. ▷ reduce QP stepsize
      $j \leftarrow j+1$.
    end if
  end while
  return $\boldsymbol{X}$
end function

Output: $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \ldots$

NMPC controller. The errors between the ideal NMPC and pf-NMPC are plotted in Figure 3.

4.2 Closed-loop Results

Next, we compare the closed-loop solution for both NMPC controllers. The solutions, control input profiles as well as total gas and oil productions, are displayed in Figure 4 and Figure 5. The total gas production capacity constraint, shown in Figure 5, is active. The solutions of pf-NMPC approximate the solutions of the ideal NMPC controllers very well. Moreover, we compare the online optimization

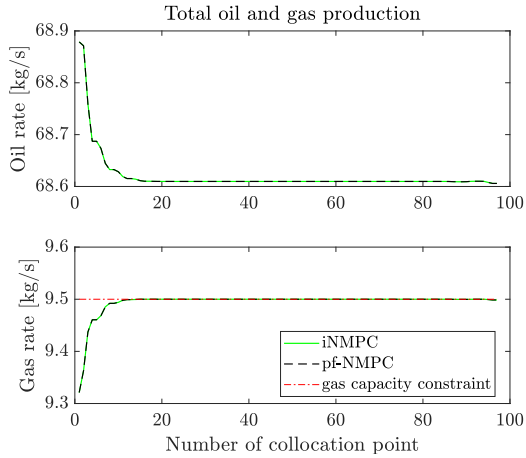


Fig. 2. Comparison of total oil and gas production for iNMPC and pf-NMPC controllers from open loop solutions at iteration number two. The solution of iNMPC is depicted in green color, which overlaps the solution of pf-NMPC denoted in black color.

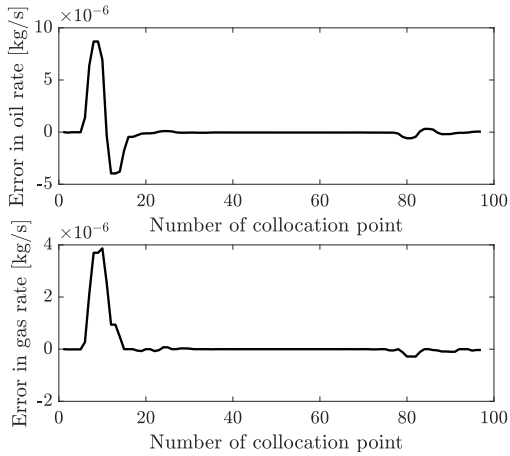


Fig. 3. Total production errors for open-loop solution at iteration number two.

online optimization runtime (in sec.)			
	min	max	average
iNmpc	0.85	0.94	0.88
pf-Nmpc	0.34	0.39	0.36

runtime for 60 NMPC iterations. It is shown that the pf-NMPC controller is able to speed up the optimization more than two times faster than those of the ideal NMPC controller. However, a rigorous comparison of runtime is very implementation dependent, and outside the scope of this paper.

5. CONCLUSION

In this paper we presented an economic nonlinear model predictive control for a gas lifted well network. To address the issue of computational delay associated with economic

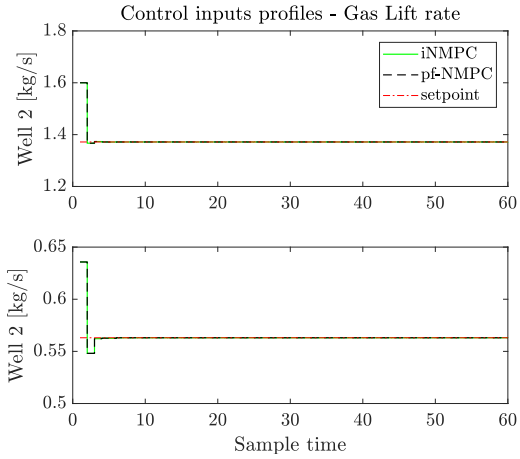


Fig. 4. Control inputs comparison of iNMPC and pf-NMPC controllers from closed-loop solutions.

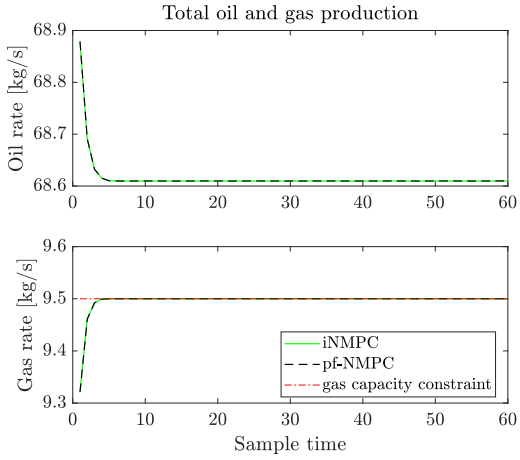


Fig. 5. Total production of oil and gas of iNMPC and pf-NMPC controllers from closed-loop solutions.

NMPC, we presented a path-following predictor-corrector approach. Using simulation results, we showed that the pf-NMPC is able to provide a fast solution, while honoring the active constraints and reasonably approximating the solutions.

REFERENCES

Andersson, J. (2013). *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium.

Bieker, H.P., Slupphaug, O., Johansen, T.A., et al. (2007). Real-time production optimization of oil and gas production systems: A technology survey. *SPE Production & Operations*, 22(04), 382–391.

Bonnans, J.F. and Shapiro, A. (1998). Optimization problems with perturbations: A guided tour. *SIAM review*, 40(2), 228–264.

Diehl, M., Bock, H.G., and Schlöder, J.P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on control and optimization*, 43(5), 1714–1736.

Findeisen, R. and Allgöwer, F. (2004). Computational delay in nonlinear model predictive control. *IFAC Proceedings Volumes*, 37(1), 427–432.

Forbes, M.G., Patwardhan, R.S., Hamadah, H., and Gopaluni, R.B. (2015). Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine*, 48(8), 531–538.

Foss, B., Knudsen, B.R., and Grimstad, B. (2017). Petroleum production optimization–a static or dynamic problem? *Computers & Chemical Engineering*.

Hauge, J., Horn, T., et al. (2005). The challenge of operating and maintaining 115 subsea wells on the troll field. In *Offshore technology conference*. Offshore Technology Conference.

Jäschke, J., Yang, X., and Biegler, L.T. (2014). Fast economic model predictive control based on nlp-sensitivities. *Journal of Process Control*, 24(8), 1260–1272.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2017). Gas lift optimization under uncertainty. *Computer Aided Chemical Engineering*, 40, 1753–1758.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2016). Real time optimization under uncertainty - applied to gas lifted wells. *Processes*, 4(4). doi:10.3390/pr4040052.

Kungurtsev, V. and Diehl, M. (2014). Sequential quadratic programming methods for parametric nonlinear optimization. *Computational Optimization and Applications*, 59(3), 475–509.

Murtagh, B.A. and Saunders, M.A. (1982). A projected lagrangian algorithm and its implementation for sparse nonlinear constraints. In *Algorithms for Constrained Minimization of Smooth Nonlinear Functions*, 84–117. Springer.

Robinson, S.M. (1980). Strongly regular generalized equations. *Mathematics of Operations Research*, 5(1), 43–62.

Stenhouse, B.J., Woodman, M., Griffiths, P., et al. (2010). Model based operational support-adding assurance to operational decision making. In *SPE Intelligent Energy Conference and Exhibition*. Society of Petroleum Engineers.

Suwartadi, E., Kungurtsev, V., and Jäschke, J. (2017). Sensitivity-based economic nmpc with a path-following approach. *Processes*, 5(1), 8.

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.

Wolf, I.J. and Marquardt, W. (2016). Fast nmpc schemes for regulatory and economic nmpc–a review. *Journal of Process Control*, 44, 162–183.

Zavala, V.M. and Biegler, L.T. (2009). The advanced-step nmpc controller: Optimality, stability and robustness. *Automatica*, 45(1), 86–93.

Appendix O

Dynamic extremum seeking scheme - applied to gas-lift optimization

This appendix contains the paper showing the application of dynamic extremum seeking control proposed by Bamberger and Isermann [7] to a network of gas lifted wells.

- Paper published in 2019 IFAC International Symposium on Dynamic Control of Process Systems (DYCOPS), Florianopolis, Brazil.

A Dynamic Extremum Seeking Scheme Applied to Gas Lift Optimization ⋆

Dinesh Krishnamoorthy, Jeongrim Ryu, Sigurd Skogestad

Dept. of Chemical Engineering, Norwegian Univ. of Science & Technology , NO-7491 Trondheim , (e-mail: dinesh.krishnamoorthy@ntnu.no, jeongrir@stud.ntnu.no, skoge@ntnu.no).

Abstract: This paper presents the application of a data-driven optimization scheme using transient measurements to a gas-lift optimization problem. Optimal operation of a gas-lifted field involves controlling the marginal gas-oil ratio (mGOR), which is the steady-state gradient of the oil rate from the gas lift injection rate. In this paper we apply a dynamic extremum seeking scheme to estimate the marginal GOR online using transient measurements, which is based on identifying a local linear *dynamic* model around the current operating point instead of a local linear static model. By doing so, we can use the transient measurements and effectively remove the time-scale separation between the plant dynamics and the perturbation signal, that is typically required in the classical extremum seeking scheme. This results in significantly faster convergence to the optimum compared to classical extremum seeking scheme. The effectiveness of the proposed method is demonstrated using simulation results for a single gas lifted well, as well as a network of gas lifted wells.

Keywords: Production optimization, Measurement-based optimization, real-time optimization, extremum seeking control

1. INTRODUCTION

In oil production wells, when the reservoir pressure is sufficiently high, then the fluids from the reservoir flows naturally to the surface. Over time, the reservoir pressure drops and may no longer be sufficient to lift the fluids economically to the surface. In such cases, artificial lift methods are used to boost the production from the wells. One such commonly used artificial lift method is the gas-lift method, where compressed gases are injected into the well tubing via the well annulus. This reduces the fluid mixture density, hence reducing the hydrostatic pressure drop across the well tubing, leading to an increased oil production. However, injecting too much gas increases the frictional pressure drop, which has a detrimental effect on the oil production. The oil production rate starts to decrease if the effect of the frictional pressure drop becomes dominant over the effect of the hydrostatic pressure drop. Each gas-lifted well then has an optimal gas lift injection rate that maximizes the oil production. In addition, the amount of gas available for gas lift may be limited. The production optimization problem then deals with the problem of finding the optimal gas lift allocation for the gas lifted wells, in order to maximize the total oil production.

Daily production optimization is an important task for maximizing the daily operating revenue from a production network. Traditionally, production engineers use so-called *gas-lift performance curves* for daily production optimization, which maps the static relationship between the oil production and the gas lift injection rate for each well (Rashid, 2010). The gas lift performance curves are typically obtained using commercially available steady-state multiphase flow simulators. Steady-state nonlinear optimization tools may then be used to compute the optimal gas lift allocation among the different wells. Production engineers may also often use the gas-lift performance curves directly for production optimization by using a quantity known as *marginal gas-oil ratio* ($mGOR$). Marginal gas-oil ratio or simply marginal GOR, is a quantity that describes the increase in oil rate per unit change in the gas-lift injection rate. In other words, marginal GOR is given by the gradient of the gas-lift performance curves (Bieker et al., 2007).

The optimal allocation of the gas lift among the different wells is achieved when the marginal GOR is the same for all the wells (Urbanczyk et al., 1994). The principle of equal marginal cost has been proven to be the optimal solution for any parallel unit, e.g. by Downs and Skogestad (2011). Therefore, optimal operation of a gas lifted well network can be achieved by simply controlling the marginal GOR to be equal for all the wells. This is schematically represented for two wells in Fig.1.

The use of centralized dynamic optimization tools such as economic NMPC for production optimization has recently been gaining popularity. Codas et al. (2016) and Krishnamoorthy et al. (2016a) used economic MPC formulations to optimize production from a gas lifted well network. However, solving a numerical optimization prob-

⋆ The authors gratefully acknowledge the financial support from SFI SUBPRO, which is financed by the Research Council of Norway, major industry partners and NTNU.

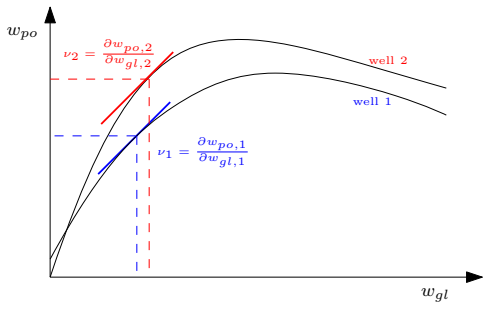


Fig. 1. Schematic representation of gas lift performance curves and the marginal GOR ν.

lem may be computationally intensive and can potentially lead to computational delays. Campos et al. (2009) point that many numerical issues need to be addressed before dynamic optimization can be used in practice for offshore oil and gas applications.

On the other hand there have been developments in optimization approaches that do not require solving numerical optimization problems. Instead, optimal operation is achieved via feedback control. Self-optimizing control is one such method (Skogestad, 2000), where the objective is to find the right controlled variable, which when kept constant, leads to near optimal operation (i.e. minimum loss). The use of self-optimizing control using nullspace method for gas lift optimization was demonstrated by Alstad (2005). Since self-optimizing control is based on local linearization around a nominal optimal point, it may lead to steady-state losses if the disturbances moves the operation of the process far away from this nominal operating point. The ideal self-optimizing variable for the gas-lift problem would indeed be the marginal GOR, which is the slope of the gas-lift performance curve, see Fig.1. However, the major challenge is that the marginal GOR is not a readily available measurement for control.

The optimization approaches mentioned above rely on the use of complex physical models either online or offline. However, models are often uncertain due to lack of knowledge or simplification, which can affect the optimal operation point computed by these methods. To address the issues related to model uncertainty, purely data-driven optimization tools become an attractive alternative to optimize the process under plant-model mismatch. To this end, we will focus on data-driven optimization tools that do not require complex physical models in the reminder of the paper.

The use of data-driven methods such as extremum seeking control for oil and gas production optimization has recently been gaining steady interest. Peixoto et al. (2015) and Krishnamoorthy et al. (2016b) applied the classical extremum seeking scheme for gas lift optimization for a single gas lifted well. Extremum seeking control involves estimating the steady-state gradient (i.e. marginal GOR) directly using the gas lift rate and oil production rate measurements. The estimated marginal GOR is then controlled to a constant setpoint using simple integral action to drive the system to its optimum.

Since extremum seeking control involves estimating the steady-state gradient directly from the measurements, the use of transient measurements leads to erroneous gradient estimation. Therefore, such methods often require clear time scale separation between the plant dynamics and the perturbation and the convergence to the optimum, such that the plant can be approximated as a static map (Krstić and Wang, 2000). This results in very slow convergence to the optimum.

For processes such as gas lifted oil wells that have long settling times (typically in the range of minutes to hours), the convergence to the optimum can be prohibitively slow due to the time scale separation requirements. This impedes the direct applicability of the classical extremum seeking control scheme for oil and gas applications. Although Extremum seeking control was used for optimizing a gas lifted well by Peixoto et al. (2015), Krishnamoorthy et al. (2016b) and Pavlov et al. (2017) to name a few, the convergence time to the optimum was not the main focus of these works.

There have been several improvements in extremum seeking control to address the issue of slow convergence. For example, Hunnekens et al. (2014) proposed to use a least-square based method to improve the convergence. However, this method still assumes the plant as a static map, restricting the use of transient measurements. Trollberg and Jacobsen (2016) proposed the so-called greedy extremum seeking control to optimize during the transients for chemical and bio-processes with long settling times. However, the greedy extremum seeking control can be implemented only for a class of systems with a specific timescale structure. Peixoto et al. (2017), recently proposed a phase-lock-loop based extremum seeking control to account for the phase shift due to the plant dynamics and speed up the convergence to the optimum point.

In this paper, we investigate a different approach, where we directly use the transient measurements to identify a local linear *dynamic* model around the current operating point, instead of a local linear static model. For example, we can identify linear ARX models from the process measurements that are locally valid around the current operating point. The steady-state gradient can then be estimated from the identified local linear dynamic model.

In fact, the use of measurements to identify a dynamic model around the current operating point for optimization dates back to 1977 in the work by Bamberger and Isermann (1978). This was later extended by McFarlane and Bacon (1989), where the authors presented an empirical strategy for open-loop online optimization using ARX models. The main motivation for these works were indeed to optimize the steady-state behaviour of slow dynamic processes in a relatively short period of time. With the recent surge of interest in extremum seeking control for oil and gas applications, we reframe this old idea in the context of extremum seeking control in this paper and show that the proposed *dynamic* extremum seeking control addresses the convergence issues of classical extremum seeking control, especially for processes with long settling times. In addition, we also propose a simple control structure for distributed dynamic extremum seeking control scheme and apply to an oil and gas production network.

The reminder of the paper is organized as follows. Section 2 introduces the proposed dynamic extremum seeking scheme. Section 3 demonstrates the effectiveness of the proposed dynamic ESC compared to the classical ESC scheme for a single gas lifted well. A distributed dynamic ESC scheme is also proposed and applied in Section 3 to a production network with 6 gas lifted wells before concluding the paper in Section 4.

2. DYNAMIC EXTREMUM SEEKING SCHEME

Consider a nonlinear process, where the objective is to drive the cost J to its minimum by using the input u.

Assumption 1. The plant cost J can be measured.

Assumption 2. The plant cost J can be represented as Hammerstein model with a combination of a nonlinear time invariant mapping $f(\cdot) : \mathbb{R} \to \mathbb{R}$, with proper, stable, finite-dimensional, linear, time-invariant (FDLTI) dynamics $G(s)$, at its output, see Fig.2.

Assumption 3. $f(u)$ is sufficiently smooth and continuously differentiable such that

$$\frac{\partial f}{\partial u}(u^*) = 0 \tag{1}$$

$$\frac{\partial^2 f}{\partial u^2}(u^*) > 0 \tag{2}$$

Assumption 3 ensures that $f(u)$ has a unique minimizer at $u = u^*$ and the goal is to drive u to the neighborhood of u^*.

In this section, we propose a dynamic extremum seeking scheme which is based on identifying a local linear *dynamic* model around the current operating point using transient measurements. The cost and input measurements from a fixed moving window containing the last N data samples are used to continuously fit an ARX model of the form,

$$\begin{aligned} J(t) = &- a_1 J(t-1) - \cdots - a_{n_a} J(t-n_a) \\ &+ bu(t-1) + \cdots + b_{n_b} u(t-n_b) + e(t) \end{aligned} \tag{3}$$

Remark 1. The input and cost measurements in (3) are pre-processed such that they are mean-centered (Ljung, 1999).

We estimate the ARX polynomials,

$$\theta = [a_1 \ \cdots \ a_{n_a} \ b_1 \ \cdots \ b_{n_b}] \tag{4}$$

using linear least squares estimation

$$\hat{\theta} = arg \ \min_{\theta} \|J - \Phi^{\mathsf{T}}\theta\|_2^2 \tag{5}$$

where Φ is given by the expression

$$\Phi = [-J(t-1) \ \ldots \ -J(t-n_a) \ u(t-1) \ \ldots \ u(t-n_b)]^{\mathsf{T}} \tag{6}$$

Introducing the notation

$$A_{poly}(q) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}$$

and

$$B_{poly}(q) = b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b}$$

yields a local linear dynamic system of the form,

$$J(t) = \frac{B_{poly}(q)}{A_{poly}(q)} u(t) + \frac{1}{A_{poly}(q)} e(t) \tag{7}$$

The steady-state gradient around the current operating point can then be estimated by,

$$\hat{\mathbf{J}}_{\mathbf{u}} = A_{poly}(q)^{-1} B_{poly}(q) \tag{8}$$

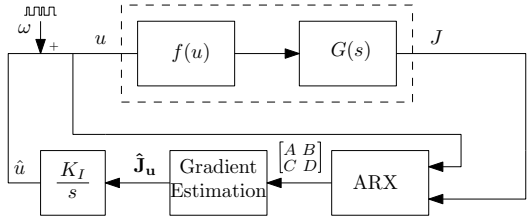


Fig. 2. A schematic representation of the proposed dynamic extremum seeking control scheme for a class of systems that can be modelled as Hammerstein models.

Alternatively, the identified ARX polynomials $A_{poly}(q)$ and $B_{poly}(q)$ can be converted to continuous time state-space system [1] as shown below,

$$\begin{aligned} \dot{x} &= Ax + Bu \\ J &= Cx + Du \end{aligned} \tag{9}$$

The steady state gain is then given by setting $\dot{x} = 0$ and eliminating the states x in (9),

$$J = \underbrace{\left(-CA^{-1}B + D\right)}_{\hat{\mathbf{J}}_{\mathbf{u}}} u \tag{10}$$

Once the steady-state gradient $\hat{\mathbf{J}}_{\mathbf{u}}$ is estimated, a simple integral action can be used to drive the system to its extremum. In discrete time, this can be expressed as,

$$\hat{u}(t+1) = \hat{u}(t) + \frac{K_I}{T_s} \hat{\mathbf{J}}_{\mathbf{u}} \tag{11}$$

where K_I is the integral gain and T_s is the sample time. Additional perturbation ω such as a pseudo random binary sequence (PRBS) signal is added to the input signal to provide sufficient excitation, $u(t+1) = \hat{u}(t+1) + \omega$. A schematic representation of the proposed dynamic extremum seeking scheme using ARX model identification is shown in Fig.2.

3. ILLUSTRATIVE EXAMPLE

3.1 Process description

We consider a production network with n_w gas lifted wells. The steady-state oil production rate for the i^{th} well $w_{po,i}$ is a function of the corresponding gas lift injection rate $u_i = w_{gl,i}$ and is given by the gas lift performance curve $w_{po,i} = f(w_{gl,i})$. Each gas lifted well is then modelled as a Hammerstein model with a proper stable first order linear dynamics $G_i(s)$

$$w_{po,i} = f(w_{gl,i}) G_i(s) w_{gl,i} \tag{12}$$

The use of such simplified Hammerstein models for gas-lifted well is justified in Peixoto et al. (2015) and Peixoto et al. (2017), where the authors show that the main features of the mechanistic model from Eikrem et al. (2006) can be sufficiently captured by using such a Hammerstein model. Plucenio et al. (2009) also use a Hammerstein model for gas lifted wells. Empirical models are also often used in practice (Hamedi et al., 2011).

[1] for example using `idss` and `d2c` command in `MATLAB`

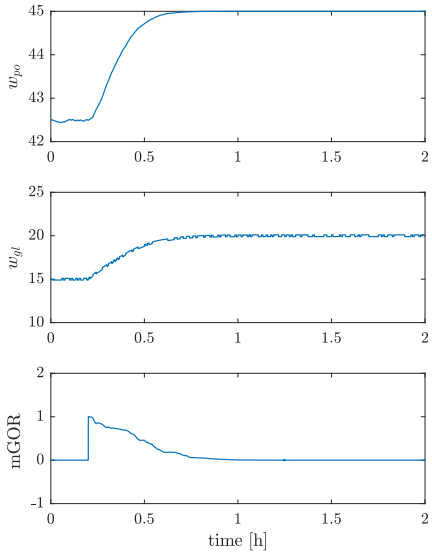


Fig. 3. Simulation results for the dynamic ESC for a single gas lifted well.

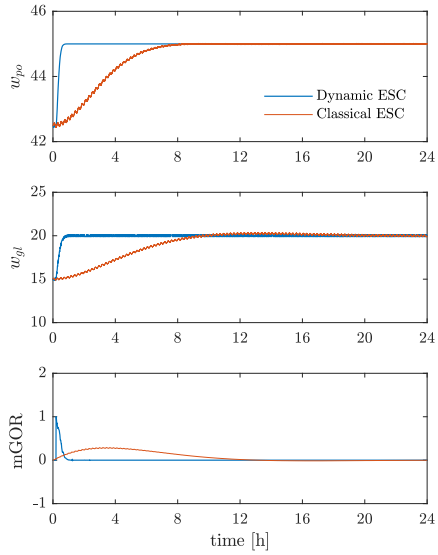


Fig. 4. Simulation results for the classical ESC for a single gas lifted well (red) compared to the dynamic ESC (blue).

We assume that the measurements for the oil production rate for each well $J_i = w_{po,i}$ is available. The objective is to maximize the total oil production rate w_{to} which is given by

$$\max J = w_{to} = \sum_{i=1}^{n_w} w_{po,i} \tag{13}$$

In many fields, the amount of gas available for gas lift injection is limited to w_{gl}^{max} and the total amount of the lift gas must be optimally allocated among the different wells. This is represented by the following constraint,

$$\sum_{i=1}^{n_w} w_{gl,i} \leq w_{gl}^{max} \tag{14}$$

Marginal gas oil ratio, which is defined as the change in the oil rate per unit change in the gas lift injection rate, is represented by the symbol ν

$$\text{mGOR}_i = \nu_i = \frac{\partial w_{po,i}}{\partial w_{gl,i}} \qquad \forall i \in \{1, \ldots, n_w\} \tag{15}$$

which is equivalent to the steady state gradient of (12) with respect to the gas lift injection rate.

3.2 Single gas lifted well

In the first simulation case, we demonstrate the effectiveness of the proposed dynamic extremum seeking scheme compared to the classical extremum seeking scheme using a single gas lifted well (i.e. $n_w = 1$), with special focus on the convergence time to the optimum. The gas lifted well model from Krishnamoorthy et al. (2016a) is captured by a Hammerstein system with the gas lift performance curve $w_{po} = -0.1(w_{gl} - 20)^2 + 45$ and first order linear dynamics

$$G(s) = \frac{1}{(1 + \tau s)} \tag{16}$$

where the time constant was set to $\tau = 174$s. For the dynamic extremum seeking scheme, we identify an ARX model with orders $n_a = 1$ and $n_b = 1$, using the past data points from a fixed moving window size of $N = 720$ samples. This implies that $A_{poly}(q) = 1 + a_1 q^{-1}$ and $B_{poly}(q) = b_1 q^{-1}$. Therefore two parameters, namely, a_1 and b_1 are fitted. The measurements are assumed to be available with a sample time of 1s. An additional PRBS perturbation with an amplitude of $0.1kg/s$ was added to the gas lift injection rate. An adaptation gain of $K_I = 0.005$ was used to drive the estimated steady state gradient (marginal GOR) to zero, since this is an unconstrained problem.

Fig.3 shows the simulation results for the proposed dynamic extremum seeking scheme for a single gas lifted well. Note that the extremum seeking controller was turned on after the first 720s (0.2h). It can be seen that the proposed scheme successfully drives the system to its optimal operating point within 1 hour.

We then compare the performance of the proposed method with the classical high-pass filter and low-pass filter based extremum seeking control (Krstić and Wang, 2000), where the system was perturbed with a sinusoidal signal with a time period of 800s and an amplitude of $0.1kg/s$. An adaptation gain of $K_I = 0.00075$ was used to drive the estimated steady state gradient to zero.

Fig.4 shows the simulation results of the classical extremum seeking scheme compared to the proposed dynamic extremum seeking scheme. It can be clearly seen that the classical extremum seeking scheme has a significantly slower convergence compared to the proposed dynamic extremum seeking scheme due to the static map

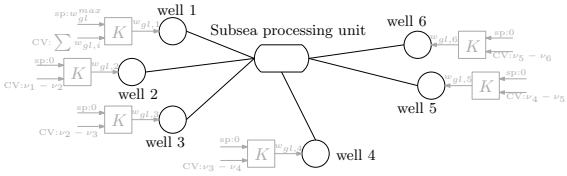


Fig. 5. Schematic representation of 6 gas lifted wells producing to a subsea processing unit. The proposed control structure is shown in grey blocks.

assumption. The classical ESC takes more than 15 hours to converge to the optimum, whereas the proposed dynamic ESC scheme converges only within 1 hour, making it more relevant for practical implementation. This example clearly demonstrates the effectiveness of the proposed dynamic extremum seeking scheme.

3.3 Gas lifted well network

In this simulation case, we now apply the proposed dynamic extremum seeking scheme to a production network consisting of $n_w = 6$ gas lifted wells producing to a common subsea processing unit. The total available gas for gas lift is limited to $w_{gl}^{max} = 56kg/s$ during normal operation, which must be optimally allocated among the six wells. All the six wells are modelled as Hammerstein models with a polynomial function for the gas lift performance curve and linear first order dynamics with the time constants varying between 170 - 180s. See Ryu (2018) for detailed description of the models used for the six wells.

In the case of limited gas lift, the optimum operation happens when all the available gas is used for lifting (i.e. the constraint (14) is active at the optimum), which is typically the case in most gas lifted fields. According to good plantwide control practice (Skogestad and Postlethwaite, 2007), we then control the active constraint tightly using one of the wells. We use the remaining $(n_w - 1)$ unconstrained degrees of freedom to optimize the production from the well network. This is achieved by maintaining the marginal GOR for all the wells to be equal, according to the principle of equal slopes as described by Downs and Skogestad (2011).

We propose a simple decentralized control structure such that we have $(n_w - 1)$ feedback controllers to control the difference in the marginal GOR between two wells to a constant setpoint of zero and 1 feedback controller to control the active constraint tightly. In other words, the controlled variables for the $(n_w - 1)$ feedback controllers would be $(\nu_i - \nu_{i+1})$ for all $i \in \{1, \ldots, n_w - 1\}$ which is controlled to a constant setpoint of zero, thereby fulfilling the principle of equal slopes for optimal operation, and one feedback controller to control the total input usage $\sum_i^{n_w} w_{gl,i}$ to a constant setpoint of w_{gl}^{max}, as described by Krishnamoorthy et al. (2018). The marginal GOR ν_i for each well is estimated using the proposed dynamic extremum seeking scheme (10).

The simulation starts with normal operation with the total gas capacity constrained at $w_{gl}^{max} = 56kg/s$. At time $t = 12$h, due to some unexpected topside disturbance, the processing capacity is reduced to $w_{gl}^{max} = 52kg/s$. Fig.6

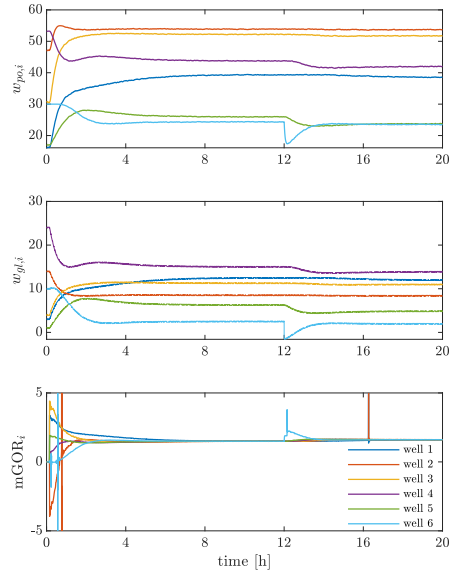


Fig. 6. Simulation results for the dynamic extremum seeking scheme applied to a network of 6 gas lifted wells.

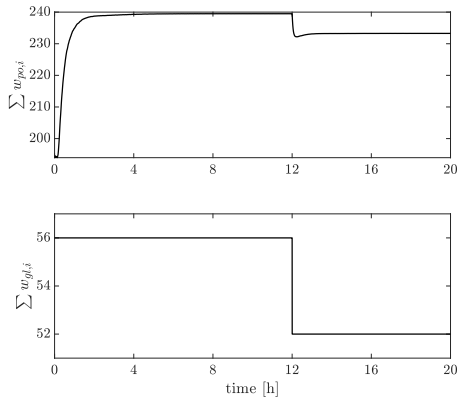


Fig. 7. Simulation results showing the total oil production rate from the well network (top subplot) and the total gas lift injection rate (bottom subplot).

shows the simulations results for the 6 well case. The top subplot shows the oil production from the 6 wells and the second subplot shows the gas lift injection rates. From the principle of equal slopes, it is known that the optimal operation happens when then marginal GOR for all the wells are equal. It can be clearly seen that the marginal GOR for all the wells converge to a value of $1.5kg/kg$ during normal operation and the marginal GOR of all the wells change to a value of $1.61kg/kg$ when the processing capacity is reduced. Fig.7 shows the total oil production and the total gas lift injection rate.

Table 1 shows the oil production rate converged to steady-state using the proposed dynamic extremum seeking scheme compared to the true optimum which is computed by solving a nonlinear optimization problem (used as benchmark). This shows that the proposed scheme is able to drive the system to its true optimum.

Table 1. Oil production rates converged to the steady state using the proposed method compared to the true optimum .

	True optimum		Converged solution	
w_{gl}^{max}	$56kg/s$	$52kg/s$	$56kg/s$	$52kg/s$
well 1	39.375	38.51	39.37	38.57
well 2	53.875	53.70	53.88	53.63
well 3	52.187	51.75	52.19	51.77
well 4	43.75	42.02	43.75	42.01
well 5	25.9375	23.78	25.93	23.78
well 6	24.375	23.51	24.4	23.49
Total	239.5	233.28	239.5	233.25

4. CONCLUSION

In this paper we proposed a dynamic extremum seeking scheme for a class of systems that can be modeled as Hammerstein models, which is based on identifying a local linear dynamic model around the current operating point. The steady-state gradient is estimated from the identified ARX model using (10). By using transient measurements for the gradient estimation, we have effectively eliminated the time scale separation required between the plant dynamics and the dither signal. This leads to a significantly faster convergence to the optimum, especially for systems with very long settling times, as demonstrated in Fig.3 and Fig.4. Additionally, for a network of gas lifted wells, we presented a simple decentralized framework for optimal allocation of lift gas in a network of gas lifted wells using the proposed dynamic extremum seeking scheme.

REFERENCES

Alstad, V. (2005). Studies on selection of controlled variables.

Bamberger, W. and Isermann, R. (1978). Adaptive on-line steady-state optimization of slow dynamic processes. *Automatica*, 14(3), 223–230.

Bieker, H.P., Slupphaug, O., Johansen, T.A., et al. (2007). Real-time production optimization of oil and gas production systems: A technology survey. *SPE Production & Operations*, 22(04), 382–391.

Campos, M., Teixeira, H., Liporace, F., and Gomes, M. (2009). Challenges and problems with advanced control and optimization technologies. *IFAC Proceedings Volumes*, 42(11), 1–8.

Codas, A., Jahanshahi, E., and Foss, B. (2016). A two-layer structure for stabilization and optimization of an oil gathering network. *IFAC-PapersOnLine*, 49(7), 931–936.

Downs, J.J. and Skogestad, S. (2011). An industrial and academic perspective on plantwide control. *Annual Reviews in Control*, 35(1), 99–110.

Eikrem, G.O., Aamo, O.M., Foss, B.A., et al. (2006). Stabilization of gas-distribution instability in single-point dual gas lift wells. *SPE Production & Operations*, 21(02), 252–259.

Hamedi, H., Rashidi, F., and Khamehchi, E. (2011). A novel approach to the gas-lift allocation optimization problem. *Petroleum Science and Technology*, 29(4), 418–427.

Hunnekens, B., Haring, M., van de Wouw, N., and Nijmeijer, H. (2014). A dither-free extremum-seeking control approach using 1st-order least-squares fits for gradient estimation. In *53rd IEEE Conference on Decision and Control*, 2679–2684. IEEE.

Krishnamoorthy, D., Foss, B., and Skogestad, S. (2016a). Real time optimization under uncertainty - applied to gas lifted wells. *Processes*, 4(4). doi:10.3390/pr4040052.

Krishnamoorthy, D., Jahanshahi, E., and Skogestad, S. (2018). Gas-lift optimization by controlling marginal gas-oil ratio using transient measurements. *IFAC-PapersOnLine*, 51(8), 19–24.

Krishnamoorthy, D., Pavlov, A., and Li, Q. (2016b). Robust extremum seeking control with application to gas lifted oil wells. *IFAC-PapersOnLine*, 49(13), 205–210.

Krstić, M. and Wang, H.H. (2000). Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36(4), 595–601.

Ljung, L. (1999). *System identification: theory for the user*. PTR Prentice Hall, Upper Saddle River, NJ, 2 edition.

McFarlane, R. and Bacon, D. (1989). Empirical strategies for open-loop on-line optimization. *The Canadian Journal of Chemical Engineering*, 67(4), 665–677.

Pavlov, A., Haring, M., and Fjalestad, K. (2017). Practical extremum-seeking control for gas-lifted oil production. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, 2102–2107. IEEE.

Peixoto, A.J., Pereira-Dias, D., Tocaimasa, G.J., and Oliveira, T.R. (2017). Fast extremum seeking for a class of non hammerstein-wienner systems. In *American Control Conference (ACC), 2017*, 5666–5671. IEEE.

Peixoto, A.J., Pereira-Dias, D., Xaud, A.F., and Secchi, A.R. (2015). Modelling and extremum seeking control of gas lifted oil wells. *IFAC-PapersOnLine*, 48(6), 21–26.

Plucenio, A., Pagano, D.J., Camponogara, E., Traple, A., and Teixeira, A. (2009). Gas-lift optimization and control with nonlinear mpc. *IFAC Proceedings Volumes*, 42(11), 904–909.

Rashid, K. (2010). Optimal allocation procedure for gas-lift optimization. *Industrial & Engineering Chemistry Research*, 49(5), 2286–2294.

Ryu, J. (2018). *Dynamic Extremum Seeking Control Using ARX models*. Master's thesis, NTNU, Trondheim, Norway.

Skogestad, S. (2000). Plantwide control: the search for the self-optimizing control structure. *Journal of process control*, 10(5), 487–507.

Skogestad, S. and Postlethwaite, I. (2007). *Multivariable feedback control: analysis and design*. Wiley New York, 2 edition.

Trollberg, O. and Jacobsen, E.W. (2016). Greedy extremum seeking control with applications to biochemical processes. *IFAC-PapersOnLine*, 49(7), 109–114.

Urbanczyk, C.H., Wattenbarger, R.A., et al. (1994). Optimization of well rates under gas coning conditions. *SPE Advanced Technology Series*, 2(02), 61–68.

References

[1] V. Alstad. *Studies on selection of controlled variables*. PhD thesis, Norwegian University of Science and Technology, 2005.

[2] V. Alstad and S. Skogestad. Null space method for selecting optimal measurement combinations as controlled variables. *Industrial & engineering chemistry research*, 46(3):846–853, 2007.

[3] V. Alstad, S. Skogestad, and E. S. Hori. Optimal measurement combinations as controlled variables. *Journal of Process Control*, 19(1):138–148, 2009.

[4] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.

[5] K. B. Ariyur and M. Krstic. *Real-time optimization by extremum-seeking control*. John Wiley & Sons, 2003.

[6] Y. Arkun and G. Stephanopoulos. Studies in the synthesis of control structures for chemical processes: Part IV. design of steady-state optimizing control structures for chemical process units. *AIChE Journal*, 26(6):975–991, 1980.

[7] W. Bamberger and R. Isermann. Adaptive on-line steady-state optimization of slow dynamic processes. *Automatica*, 14(3):223–230, 1978.

[8] A. Banaszuk, K. Ariyur, M. Krstić, and C. Jacobson. An adaptive algorithm for control of combustion instability. *Automatica*, 40(11):1965–1972, 2004.

[9] R. D. Bartusiak. Nlmpc: A platform for optimal control of feed-or product-flexible manufacturing. In *Assessment and Future Directions of Nonlinear Model Predictive Control*, pages 367–381. Springer, 2007.

[10] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[11] D. Bernardini and A. Bemporad. Scenario-based model predictive control of stochastic constrained linear systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 6333–6338. IEEE, 2009.

[12] D. P. Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

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

[14] L. T. Biegler. Efficient solution of dynamic optimization and NMPC problems. In *Nonlinear Model Predictive Control*, pages 219–243. Springer, 2000.

[15] L. T. Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, volume 10. SIAM, 2010.

[16] J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations research*, 33(5):989–1007, 1985.

[17] J. F. Bonnans and A. Shapiro. Optimization problems with perturbations: A guided tour. *SIAM review*, 40(2):228–264, 1998.

[18] H. Bonnowitz, J. Straus, D. Krishnamoorthy, E. Jahanshahi, and S. Skogestad. Control of the steady-state gradient of an ammonia reactor using transient measurements. *Computer-Aided Chemical Engineering*, 43:1111–1116, 2018.

[19] S. Boyd. Notes on subgradient methods. *Notes for EE364B, Stanford University*, pages 1–39, 2014.

[20] S. Boyd, L. Xiao, A. Mutapcic, and J. Mattingley. Notes on decomposition methods. *Notes for EE364B, Stanford University*, pages 1–36, 2007.

[21] G. C. Calafiore and L. Fagiano. Robust model predictive control via scenario optimization. *IEEE Transactions on Automatic Control*, 58(1):219–224, 2013.

[22] M. M. Câmara, A. D. Quelhas, and J. C. Pinto. Performance evaluation of real industrial RTO systems. *Processes*, 4(4):44, 2016.

[23] M. C. Campi, S. Garatti, and M. Prandini. The scenario approach for systems and control design. *Annual Reviews in Control*, 33(2):149–157, 2009.

[24] P. J. Campo and M. Morari. Robust Model Predictive Control. In *American Control Conference, 1987*, pages 1021–1026. IEEE, 1987.

[25] M. Campos, H. Teixeira, F. Liporace, and M. Gomes. Challenges and problems with advanced control and optimization technologies. *IFAC Proceedings Volumes*, 42(11):1–8, 2009.

[26] S. Cao and R. R. Rhinehart. An efficient method for on-line identification of steady state. *Journal of Process Control*, 5(6):363–374, 1995.

[27] B. Chachuat, A. Marchetti, and D. Bonvin. Process optimization via constraints adaptation. *Journal of Process Control*, 18(3-4):244–257, 2008.

[28] B. Chachuat, B. Srinivasan, and D. Bonvin. Adaptation strategies for real-time optimization. *Computers & Chemical Engineering*, 33(10):1557–1567, 2009.

[29] C. Y. Chen and B. Joseph. On-line optimization using a two-phase approach: an application study. *Industrial & engineering chemistry research*, 26(9): 1924–1930, 1987.

[30] J. Chen. Logarithmic integrals, interpolation bounds and performance limitations in MIMO feedback systems. *IEEE Transactions on Automatic Control*, 45(6):1098–1115, June 2000.

[31] D. F. Chichka, J. L. Speyer, and C. Park. Peak-seeking control with application to formation flight. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 3, pages 2463–2470. IEEE, 1999.

[32] M. Chioua, B. Srinivasan, M. Guay, and M. Perrier. Performance improvement of extremum seeking control using recursive least square estimation with forgetting factor. *IFAC-PapersOnLine*, 49(7):424–429, 2016.

[33] J. Choi, M. Krstic, K. Ariyur, and J. Lee. Extremum seeking control for discrete-time systems. *IEEE Transactions on automatic control*, 47(2):318–323, 2002.

[34] M. L. Darby, M. Nikolaou, J. Jones, and D. Nicholson. RTO: An overview and assessment of current practice. *Journal of Process Control*, 21(6):874–884, 2011.

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

[36] D. Dochain, M. Perrier, and M. Guay. Extremum seeking control and its application to process and reaction systems: A survey. *Mathematics and Computers in Simulation*, 82(3):369–380, 2011.

[37] J. J. Downs and S. Skogestad. An industrial and academic perspective on plantwide control. *Annual Reviews in Control*, 35(1):99–110, 2011.

[38] J. C. Doyle. Synthesis of robust controllers and filters. In *The 22nd IEEE Conference on Decision and Control*, pages 109–114. IEEE, 1983.

[39] C. Draper and Y. Li. *Principles of optimalizing control systems and an application to the internal combustion engine*. American Society of Mechanical Engineers, 1951.

[40] H. Dürr, M. Stanković, C. Ebenbauer, and K. Johansson. Lie bracket approximation of extremum seeking systems. *Automatica*, 49(6):1538–1552, 2013.

[41] C. G. Economou, M. Morari, and B. O. Palsson. Internal model control: Extension to nonlinear system. *Industrial & Engineering Chemistry Process Design and Development*, 25(2):403–411, 1986.

[42] M. Ellis, H. Durand, and P. D. Christofides. A tutorial review of economic model predictive control methods. *Journal of Process Control*, 24(8):1156–1178, 2014.

[43] E. Eskinat, S. Johnson, and W. L. Luyben. Use of Hammerstein models in identification of nonlinear systems. *AIChE Journal*, 37(2):255–268, 1991.

[44] L. Fagiano, G. Schildbach, M. Tanaskovic, and M. Morari. Scenario and adaptive model predictive control of uncertain systems. *IFAC-PapersOnLine*, 48(23):352–359, 2015.

[45] R. Findeisen and F. Allgöwer. Computational delay in nonlinear model predictive control. *IFAC Proceedings Volumes*, 37(1):427–432, 2004.

[46] W. Findeisen, F. N. Bailey, M. Brdys, K. Malinowski, P. Tatjewski, and A. Wozniak. *Control and coordination in hierarchical systems*. John Wiley & Sons, 1980.

[47] W. R. Fisher, M. F. Doherty, and J. M. Douglas. The interface between design and control. 3. selecting a set of controlled variables. *Industrial & engineering chemistry research*, 27(4):611–615, 1988.

[48] J. Forbes, T. Marlin, and J. MacGregor. Model adequacy requirements for optimizing plant operations. *Computers & chemical engineering*, 18(6):497–510, 1994.

[49] M. G. Forbes, R. S. Patwardhan, H. Hamadah, and R. B. Gopaluni. Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine*, 48(8):531–538, 2015.

[50] B. Foss, B. R. Knudsen, and B. Grimstad. Petroleum production optimization–a static or dynamic problem? *Computers & Chemical Engineering*, 114:245–253, 2018.

[51] G. François, B. Srinivasan, and D. Bonvin. Use of measurements for enforcing the necessary conditions of optimality in the presence of constraints and uncertainty. *Journal of Process Control*, 15(6):701–712, 2005.

[52] R. G. Franks. *Mathematical modeling in chemical engineering*. Wiley, 1967.

[53] L. Fu and Ü. Özgüner. Extremum seeking with sliding mode gradient estimation and asymptotic regulation for a class of nonlinear systems. *Automatica*, 47(12):2595–2603, 2011.

[54] C. E. Garcia and M. Morari. Optimal operation of integrated processing systems. part i: Open-loop on-line optimizing control. *AIChE Journal*, 27 (6):960–968, 1981.

[55] M. Ge and E. C. Kerrigan. Noise covariance identification for time-varying and nonlinear systems. *International Journal of Control*, 90(9):1903–1915, 2017.

[56] A. Ghaffari, M. Krstić, and D. Nešić. Multivariable Newton-based extremum seeking. *Automatica*, 48(8):1759–1767, 2012.

[57] M. P. Golden and B. E. Ydstie. Adaptive extremum control using approximate process models. *AIChE journal*, 35(7):1157–1169, 1989.

[58] S. F. Graebe and A. L. Ahlen. Dynamic transfer among alternative controllers and its relation to antiwindup controller design. *IEEE Transactions on Control Systems Technology*, 4(1):92–99, 1996.

[59] I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and engineering*, 3(3):227–252, 2002.

[60] M. Guay, V. Adetola, and D. DeHaan. *Robust and Adaptive Model Predictive Control of Nonlinear Systems*. Institution of Engineering and Technology, 2015.

[61] V. Gunnerud and B. Foss. Oil production optimization—a piecewise linear model, solved with two decomposition strategies. *Computers & Chemical Engineering*, 34(11):1803–1812, 2010.

[62] I. J. Halvorsen, S. Skogestad, J. C. Morud, and V. Alstad. Optimal selection of controlled variables. *Industrial & Engineering Chemistry Research*, 42(14): 3273–3284, 2003.

[63] K. G. Hanssen and B. Foss. Production optimization under uncertainty-applied to petroleum production. *IFAC-PapersOnLine*, 48(8):217–222, 2015.

[64] K. G. Hanssen and B. Foss. Scenario based implicit dual model predictive control. *IFAC-PapersOnLine*, 48(23):416–421, 2015.

[65] R. Hanus, M. Kinnaert, and J.-L. Henrotte. Conditioning technique, a general anti-windup and bumpless transfer method. *Automatica*, 23(6):729–739, 1987.

[66] J. D. Hedengren and A. N. Eaton. Overview of estimation methods for industrial dynamic systems. *Optimization and Engineering*, 18(1):155–178, 2017.

[67] T. Helgason and S. W. Wallace. Approximate scenario solutions in the progressive hedging algorithm. *Annals of Operations Research*, 31(1):425–444, 1991.

[68] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

[69] E. O. Hülse and E. Camponogara. Robust formulations for production optimization of satellite oil wells. *Engineering Optimization*, pages 1–18, 2016.

[70] B. Hunnekens, M. Haring, N. van de Wouw, and H. Nijmeijer. A dither-free extremum-seeking control approach using 1st-order least-squares fits for gradient estimation. In *53rd IEEE Conference on Decision and Control*, pages 2679–2684. IEEE, 2014.

[71] M. G. Jacobsen and S. Skogestad. Active constraint regions for optimal operation of distillation columns. *Industrial & Engineering Chemistry Research*, 51(7):2963–2973, 2012.

[72] J. Jäschke and S. Skogestad. NCO tracking and self-optimizing control in the context of real-time optimization. *Journal of Process Control*, 21(10): 1407–1416, 2011.

[73] J. Jäschke and S. Skogestad. Optimal controlled variables for polynomial systems. *Journal of Process Control*, 22(1):167–179, 2012.

[74] J. Jäschke, X. Yang, and L. T. Biegler. Fast economic model predictive control based on NLP-sensitivities. *Journal of Process Control*, 24(8):1260–1272, 2014.

[75] J. Jäschke, Y. Cao, and V. Kariwala. Self-optimizing control–a survey. *Annual Reviews in Control*, 2017.

[76] J. Kadam, W. Marquardt, M. Schlegel, T. Backx, O. Bosgra, P. Brouwer, G. Dünnebier, D. Van Hessem, A. Tiagounov, and S. De Wolf. Towards integrated dynamic real-time optimization and control of industrial processes. *Proceedings foundations of computer-aided process operations (FOCAPO2003)*, pages 593–596, 2003.

[77] J. Kallrath. Mixed integer optimization in the chemical process industry: Experience, potential and future perspectives. *Chemical Engineering Research and Design*, 78(6):809–822, 2000.

[78] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[79] E. P. Kanu, J. Mach, K. E. Brown, et al. Economic approach to oil production and gas allocation in continuous gas lift (includes associated papers 10858 and 10865). *Journal of Petroleum Technology*, 33(10):1–887, 1981.

[80] B. Karg and S. Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *arXiv preprint arXiv:1806.10644*, 2018.

[81] B. D. Keating and A. Alleyne. Combining self-optimizing control and extremum seeking for online optimization with application to vapor compression cycles. In *American Control Conference (ACC), 2016*, pages 6085–6090. IEEE, 2016.

[82] E. C. Kerrigan and J. M. Maciejowski. Soft constraints and exact penalty functions in model predictive control. In *Proc. UKACC International Conference (Control2000)*, 2000.

[83] E. C. Kerrigan and J. M. Maciejowski. Robustly stable feedback min-max model predictive control. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 4, pages 3490–3495. IEEE, 2003.

[84] L. Keviczky and R. Haber. Adaptive dual extremum control by Hammerstein model. In *Proc. IFAC Conf. on Stochastic Control, Budapest*, 1974.

[85] J.-S. Kim. Recent advances in adaptive MPC. In *ICCAS 2010*, pages 218–222. IEEE, 2010.

[86] E. Klintberg, J. Dahl, J. Fredriksson, and S. Gros. An improved dual Newton strategy for scenario-tree MPC. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 3675–3681. IEEE, 2016.

[87] D. Krishnamoorthy and S. Skogestad. A fast robust extremum seeking scheme using transient measurements. *Automatica*, Under review, 2019.

[88] D. Krishnamoorthy and S. Skogestad. Real-time optimization strategies using surrogate optimizers. In *2019 Foundations on Process Analytics and Machine Learning (FOPAM)*. CACHE, 2019.

[89] D. Krishnamoorthy and S. Skogestad. Online process optimization with active constraint set changes using simple control structures. *Industrial & Engineering Chemistry Research*, 58(30):13555–13567, 2019.

[90] D. Krishnamoorthy, E. M. Bergheim, A. Pavlov, M. Fredriksen, and K. Fjalestad. Modelling and robustness analysis of model predictive control for electrical submersible pump lifted heavy oil wells. *IFAC-PapersOnLine*, 49(7):544–549, 2016.

[91] D. Krishnamoorthy, B. Foss, and S. Skogestad. Real time optimization under uncertainty - applied to gas lifted wells. *Processes*, 4(4), 2016. doi: 10.3390/pr4040052.

[92] D. Krishnamoorthy, A. Pavlov, and Q. Li. Robust extremum seeking control with application to gas lifted oil wells. *IFAC-PapersOnLine*, 49(13):205–210, 2016.

[93] D. Krishnamoorthy, B. Foss, and S. Skogestad. Gas lift optimization under uncertainty. *Computer Aided Chemical Engineering*, 40:1753–1758, 2017.

[94] D. Krishnamoorthy, B. Foss, and S. Skogestad. A distributed algorithm for scenario-based model predictive control using primal decomposition. *IFAC papers-online (ADCHEM)*, 51:351–356, 2018.

[95] D. Krishnamoorthy, B. Foss, and S. Skogestad. Steady-state real-time optimization using transient measurements. *Computers & Chemical Engineering*, 115:34–45, 2018.

[96] D. Krishnamoorthy, E. Jahanshahi, and S. Skogestad. Gas-lift optimization by controlling marginal gas-oil ratio using transient measurements. *IFAC-PapersOnLine*, 51(8):19–24, 2018.

[97] D. Krishnamoorthy, S. Skogestad, and J. Jäschke. Multistage model predictive control with online scenario tree update using recursive bayesian weighting. 51(20):462–468, 2018.

[98] D. Krishnamoorthy, E. Suwartadi, B. Foss, S. Skogestad, and J. Jäschke. Improving scenario decomposition for multistage MPC using a sensitivity-based path-following algorithm. *IEEE Control Systems Letters*, 4(2):581–586, 2018.

[99] D. Krishnamoorthy, M. Thombre, S. Skogestad, and J. Jäschke. Data-driven scenario selection for multistage robust model predictive control. *IFAC-PapersOnLine*, 51(20):462–468, 2018.

[100] D. Krishnamoorthy, K. Fjalestad, and S. Skogestad. Optimal operation of oil and gas production using simple feedback control structures. *Control Engineering Practice*, 91:104017, 2019.

[101] D. Krishnamoorthy, B. Foss, and S. Skogestad. A primal decomposition algorithm for distributed multistage scenario model predictive control. *Journal of Process control*, 81:162–171, 2019.

[102] D. Krishnamoorthy, E. Jahanshahi, and S. Skogestad. A feedback real time optimization strategy using a novel steady-state gradient estimate and transient measurements. *Industrial and Engineering Chemistry Research*, 58: 207–216, 2019.

[103] D. Krishnamoorthy, E. Jahanshahi, and S. Skogestad. Control of the steady-state gradient of an evaporator process using transient measurements. *PSE Asia*, 2019.

[104] D. Krishnamoorthy, J. Ryu, and S. Skogestad. Dynamic extremum seeking scheme applied to gas lift optimization. *IFAC-PapersOnLine (DYCOPS)*, 2019.

[105] D. Krishnamoorthy, C. Valli, and S. Skogestad. Real-time optimal resource allocation in an industrial symbiotic network using transient measurements. In *2020 American Control Conference (ACC)*, page submitted. IEEE, 2020.

[106] M. Krstić. Performance improvement and limitations in extremum seeking control. *Systems & Control Letters*, 39(5):313–326, 2000.

[107] M. Krstić and H.-H. Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36(4):595–601, 2000.

[108] V. Kumar and N. Kaistha. Hill-climbing for plantwide control to economic optimum. *Industrial & Engineering Chemistry Research*, 53(42):16465–16475, 2014.

[109] V. Kungurtsev and J. Jäschke. A predictor-corrector path-following algorithm for dual-degenerate parametric optimization problems. *SIAM Journal on Optimization*, 27(1):538–564, 2017.

[110] T. Larsson and S. Skogestad. Plantwide control-a review and a new design procedure. *Modeling, Identification and control*, 21(4):209, 2000.

[111] L. S. Lasdon. Optimization theory for large scale systems. *Series of Operation Research*, 1970.

[112] M. Leibman, T. Edgar, and L. Lasdon. Efficient data reconciliation and estimation for dynamic processes using nonlinear programming techniques. *Computers & chemical engineering*, 16(10-11):963–986, 1992.

[113] C. Letchindjio, J. Deschenes, L. Dewasme, and A. Wouwer. Extremum seeking based on a Hammerstein-Wiener representation. *IFAC-PapersOnLine*, 51(18):744–749, 2018.

[114] L. Ljung. *System identification: Theory for the user*. PTR Prentice Hall, Upper Saddle River, NJ, 2 edition, 1999.

[115] S. Lucia and R. Paulen. Robust nonlinear model predictive control with reduction of uncertainty via robust optimal experiment design. *IFAC Proceedings Volumes*, 47(3):1904–1909, 2014.

[116] S. Lucia, T. Finkler, and S. Engell. Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9):1306–1319, 2013.

[117] S. Lucia, S. Subramanian, and S. Engell. Non-conservative robust nonlinear model predictive control via scenario decomposition. In *Control Applications (CCA), 2013 IEEE International Conference on*, pages 586–591. IEEE, 2013.

[118] A. Maarleveld and J. Rijnsdorp. Constraint control on distillation columns. *Automatica*, 6(1):51–58, 1970.

[119] J. M. Maciejowski. *Predictive control: with constraints*. Pearson education, 2002.

[120] A. Marchetti, B. Chachuat, and D. Bonvin. Modifier-adaptation methodology for real-time optimization. *Industrial & engineering chemistry research*, 48 (13):6022–6033, 2009.

[121] S. Marinkov, B. de Jager, and M. Steinbuch. Extremum seeking control with data-based disturbance feedforward. In *2014 American Control Conference*, pages 3627–3632, June 2014. doi: 10.1109/ACC.2014.6858832.

[122] S. Marinkov, B. de Jager, and M. Steinbuch. Extremum seeking control with adaptive disturbance feedforward. *IFAC Proceedings Volumes*, 47(3): 383–388, 2014.

[123] T. E. Marlin and A. N. Hrymak. Real-time operations optimization of continuous processes. In *AIChE Symposium Series*, volume 93, pages 156–164. New York, NY: American Institute of Chemical Engineers, 1971-c2002., 1997.

[124] R. Martí, S. Lucia, D. Sarabia, R. Paulen, S. Engell, and C. de Prada. Improving scenario decomposition algorithms for robust nonlinear model predictive control. *Computers & Chemical Engineering*, 79:30–45, 2015.

[125] D. Mayne. Robust and stochastic MPC: Are we going in the right direction? *IFAC-PapersOnLine*, 48(23):1–8, 2015.

[126] D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[127] R. McFarlane and D. Bacon. Empirical strategies for open-loop on-line optimization. *The Canadian Journal of Chemical Engineering*, 67(4):665–677, 1989.

[128] A. Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[129] S. Mochizuki, L. A. Saputelli, C. S. Kabir, R. Cramer, M. Lochmann, R. Reese, L. Harms, C. Sisk, J. R. Hite, A. Escorcia, et al. Real time optimization: Classification and assessment. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2004.

[130] M. Morari, Y. Arkun, and G. Stephanopoulos. Studies in the synthesis of control structures for chemical processes: Part I: Formulation of the problem. process decomposition and the classification of the control tasks. analysis of the optimizing control structures. *AIChE Journal*, 26(2):220–232, 1980.

[131] J. C. Morud and S. Skogestad. Analysis of instability in an industrial ammonia reactor. *AIChE Journal*, 44(4):888–895, 1998. ISSN 1547-5905.

[132] B. A. Murtagh and M. A. Saunders. A projected lagrangian algorithm and its implementation for sparse nonlinear constraints. In *Algorithms for Constrained Minimization of Smooth Nonlinear Functions*, pages 84–117. Springer, 1982.

[133] Y. Pan, Ü. Özgüner, and T. Acarman. Stability and performance improvement of extremum seeking control with sliding mode. *International Journal of Control*, 76(9-10):968–985, 2003.

[134] B. P. G. V. Parys, D. Kuhn, P. J. Goulart, and M. Morari. Distributionally robust control of constrained stochastic systems. *IEEE Transactions on Automatic Control*, 61(2):430–442, Feb 2016. ISSN 0018-9286. doi: 10.1109/TAC.2015.2444134.

[135] A. Pavlov, D. Krishnamoorthy, K. Fjalestad, E. Aske, and M. Fredriksen. Modelling and model predictive control of oil wells with electric submersible pumps. In *IEEE Conference on Control Applications (CCA)*, pages 586–592. IEEE, 2014.

[136] A. Pavlov, M. Haring, and K. Fjalestad. Practical extremum-seeking control for gas-lifted oil production. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 2102–2107. IEEE, 2017.

[137] A. Peixoto, D. Pereira-Dias, G. Tocaimasa, and T. Oliveira. Fast extremum seeking for a class of non Hammerstein-wienner systems. In *American Control Conference (ACC), 2017*, pages 5666–5671. IEEE, 2017.

[138] E. Pistikopoulos. Perspectives in multiparametric programming and explicit model predictive control. *AIChE journal*, 55(8):1918–1925, 2009.

[139] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

[140] A. D. Quelhas, N. J. C. de Jesus, and J. C. Pinto. Common vulnerabilities of rto implementations in real chemical processes. *The Canadian Journal of Chemical Engineering*, 91(4):652–668, 2013.

[141] G. P. Rangaiah and V. Kariwala. *Plantwide control: Recent developments and applications*. John Wiley & Sons, 2012.

[142] K. D. Rao and J. L. Narayana. An approach for a faster GPS tracking extended Kalman filter. *Navigation*, 42(4):619–630, 1995.

[143] J. B. Rawlings and R. Amrit. Optimizing process economic performance using model predictive control. In *Nonlinear model predictive control*, pages 119–138. Springer, 2009.

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

[145] A. Reyes-Lúa, C. Zotică, T. Das, D. Krishnamoorthy, and S. Skogestad. Changing between active constraint regions for optimal operation: Classical advanced control versus model predictive control. In *Computer Aided Chemical Engineering*, volume 43, pages 1015–1020. Elsevier, 2018.

[146] A. Reyes-Lúa, C. Zotică, and S. Skogestad. Optimal operation with changing active constraint regions using classical advanced control. *IFAC-PapersOnLine*, 51(18):440–445, 2018.

[147] P. D. Roberts and T. Williams. On an algorithm for combined system optimisation and parameter estimation. *Automatica*, 17(1):199–209, 1981.

[148] S. M. Robinson. Strongly regular generalized equations. *Mathematics of Operations Research*, 5(1):43–62, 1980.

[149] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1): 119–147, 1991.

[150] A. Ruszczyński. An augmented Lagrangian decomposition method for block diagonal linear programming problems. *Operations Research Letters*, 8(5): 287–294, 1989.

[151] A. Ruszczyński. Parallel decomposition of multistage stochastic programming problems. *Mathematical programming*, 58(1-3):201–228, 1993.

[152] T. Samad. A survey on industry impact and challenges thereof. *IEEE Control Systems Magazine*, 37(1):17–18, 2017.

[153] T. Samad. Impact of Control on Industry: A Survey. *IFAC Newsletter*, April (2):1–2, 2019.

[154] L. O. Santos, P. A. Afonso, J. A. Castro, N. M. Oliveira, and L. T. Biegler. On-line implementation of nonlinear MPC: an experimental case study. *Control Engineering Practice*, 9(8):847–857, 2001.

[155] Y. Sawaragi, T. Takamatsu, K. Fukunaga, E. Nakanishi, and H. Tamura. Dynamic version of steady state optimizing control of a distillation column by trial method. *Automatica*, 7(4):509–516, 1971.

[156] G. Schildbach, L. Fagiano, C. Frei, and M. Morari. The scenario approach for stochastic model predictive control with bounds on closed-loop constraint violations. *Automatica*, 50(12):3009–3018, 2014.

[157] P. Scokaert and D. Mayne. Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic control*, 43(8): 1136–1142, 1998.

[158] D. E. Seborg, D. A. Mellichamp, T. F. Edgar, and F. J. Doyle III. *Process dynamics and control*. John Wiley & Sons, 2010.

[159] R. Serban and A. C. Hindmarsh. CVODES: An ODE solver with sensitivity analysis capabilities. Technical report, Technical Report UCRL-JP-200039, Lawrence Livermore National Laboratory, 2003.

[160] J. Sharafi, W. Moase, and C. Manzie. Fast extremum seeking on Hammerstein plants: A model-based approach. *Automatica*, 59:171–181, 2015.

[161] R. Sharma and B. Glemmestad. On generalized reduced gradient method with multi-start and self-optimizing control structure for gas lift allocation optimization. *Journal of Process Control*, 23(8):1129–1140, 2013.

[162] F. G. Shinskey. *Process Control Systems: Application, Design and Tuning*. McGraw-Hill, Inc., New York, NY, USA, 4th edition, 1996. ISBN 0070569037.

[163] D. Shook. Best practices improve control system performance. *Oil & gas journal*, 104(38):52–52, 2006.

[164] D. Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[165] S. Skogestad. Plantwide control: the search for the self-optimizing control structure. *Journal of process control*, 10(5):487–507, 2000.

[166] S. Skogestad. Simple analytic rules for model reduction and PID controller tuning. *Journal of process control*, 13(4):291–309, 2003.

[167] S. Skogestad. Control structure design for complete chemical plants. *Computers & Chemical Engineering*, 28(1):219–234, 2004.

[168] S. Skogestad and I. Postlethwaite. *Multivariable feedback control: analysis and design*. Wiley New York, 2 edition, 2007.

[169] A. L. Soyster. Technical note—convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations research*, 21(5):1154–1157, 1973.

[170] B. Srinivasan, D. Bonvin, E. Visser, and S. Palanki. Dynamic optimization of batch processes: Ii. role of measurements in handling uncertainty. *Computers & chemical engineering*, 27(1):27–44, 2003.

[171] B. Srinivasan, L. T. Biegler, and D. Bonvin. Tracking the necessary conditions of optimality with changing set of active constraints using a barrier-penalty function. *Computers & Chemical Engineering*, 32(3):572–579, 2008.

[172] B. Srinivasan, G. François, and D. Bonvin. Comparison of gradient estimation methods for real-time optimization. In *21st European Symposium on Computer Aided Process Engineering-ESCAPE 21*, number EPFL-CONF-155235, pages 607–611. Elsevier, 2011.

[173] B. Srinivasan, G. François, and D. Bonvin. Comparison of gradient estimation methods for real-time optimization. In *21st European Symposium on Computer Aided Process Engineering-ESCAPE 21*, number EPFL-CONF-155235, pages 607–611. Elsevier, 2011.

[174] J. Straus and S. Skogestad. Economic NMPC for heat-integrated chemical reactors. In *2017 21st International Conference on Process Control (PC)*, pages 309–314, June 2017. doi: 10.1109/PC.2017.7976232.

[175] J. Straus and S. Skogestad. Self-optimizing control in chemical recycle systems. *IFAC-PapersOnLine*, 51(18):530–535, July 2018.

[176] J. Straus, D. Krishnamoorthy, and S. Skogestad. Combining self-optimizing control and extremum seeking control - applied to ammonia reacotr case study. *Journal of Process control*, 78:78–87, 2019.

[177] X. Sun, L. Jin, and M. Xiong. Extended Kalman filter for estimation of parameters in nonlinear state-space models of biochemical networks. *PloS one*, 3(11):e3758, 2008.

[178] E. Suwartadi, V. Kungurtsev, and J. Jäschke. Sensitivity-based economic NMPC with a path-following approach. *Processes*, 5(1):8, 2017.

[179] G. Takacs. *Electrical submersible pumps manual: design, operations, and maintenance*. Gulf professional publishing, 2017.

[180] Y. Tan, W. Moase, C. Manzie, D. Nešić, and I. Mareels. Extremum seeking from 1922 to 2010. In *29th Chinese Control Conference (CCC)*, pages 14–26. IEEE, 2010.

[181] F. Trespalacios and I. E. Grossmann. Review of mixed-integer nonlinear and generalized disjunctive programming methods. *Chemie Ingenieur Technik*, 86(7):991–1012, 2014.

[182] O. Trollberg and E. W. Jacobsen. Greedy extremum seeking control with applications to biochemical processes. *IFAC-PapersOnLine (DYCOPS-CAB)*, 49(7):109–114, 2016.

[183] C. H. Urbanczyk, R. A. Wattenbarger, et al. Optimization of well rates under gas coning conditions. *SPE Advanced Technology Series*, 2(02):61–68, 1994.

[184] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[185] S. Wenzel, R. Paulen, S. Krämer, B. Beisheim, and S. Engell. Shared resource allocation in an integrated petrochemical site by price-based coordination using quadratic approximation. In *2016 European Control Conference (ECC)*, pages 1045–1050. IEEE, 2016.

[186] L. Würth, R. Hannemann, and W. Marquardt. Neighboring-extremal updates for nonlinear model-predictive control and dynamic real-time optimization. *Journal of Process Control*, 19(8):1277–1288, 2009.

[187] L. Ye, Y. Cao, Y. Li, and Z. Song. Approximating necessary conditions of optimality as controlled variables. *Industrial & Engineering Chemistry Research*, 52(2):798–808, 2013. doi: 10.1021/ie300654d.

[188] R. Yelchuru and S. Skogestad. Convex formulations for optimal selection of controlled variables and measurements using mixed integer quadratic programming. *Journal of Process control*, 22(6):995–1007, 2012.

[189] V. M. Zavala and L. T. Biegler. The advanced-step NMPC controller: Optimality, stability and robustness. *Automatica*, 45(1):86–93, 2009.