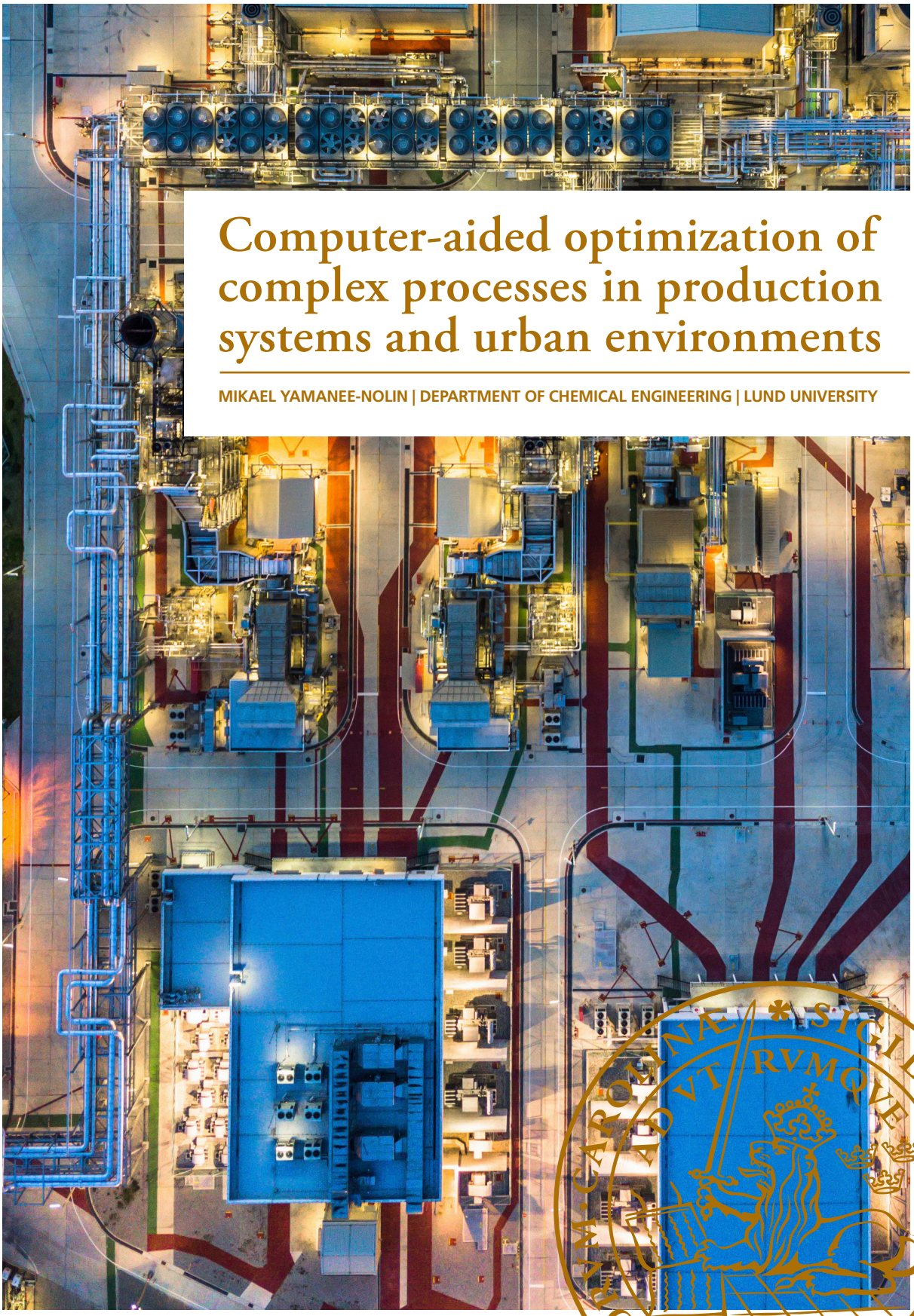
LUND UNIVERSITY

# Computer-aided optimization of complex processes in production systems and urban environments

Yamanee-Nolin, Mikael

2020

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



# Computer-aided optimization of complex processes in production systems and urban environments

MIKAEL YAMANEE-NOLIN | DEPARTMENT OF CHEMICAL ENGINEERING | LUND UNIVERSITY

# Computer-aided optimization of complex processes in production systems and urban environments

Mikael Per Daniel Yamanee-Nolin

Department of Chemical Engineering
Lund University, Sweden
2020

ACADEMIC THESIS

which, by due permission of the Faculty of Engineering of Lund University, will be publicly defended on the 8th of May 2020 at 13:15 in lecture hall KC:A at the Center for Chemistry and Chemical Engineering, Naturvetarvägen 14, Lund, for the degree of Doctor of Philosophy in Engineering.

The faculty opponent is Professor David Bogle from the Department of Chemical Engineering at University College London, UK.


LUND
UNIVERSITY

| | |
|---|---|
| **Organization**<br>LUND UNIVERSITY<br><br>Department of Chemical Engineering<br>P.O. Box 124<br>SE-221 00 LUND, SWEDEN | **Document name**<br>DOCTORAL DISSERTATION<br>**Date of issue**<br>14th of April 2020<br>**Sponsoring organization**<br>Swedish Governmental Agency for Innovation Systems (VINNOVA) |



**Author(s)**
Mikael Per Daniel Yamanee-Nolin


**Title and subtitle**
Computer-aided optimization of complex processes in production systems and urban environments


**Abstract**

Complex processes can be seen as a series of actions, changes, and/or events that are related in different temporospatial manners, and thus are potentially difficult to understand. Optimizing such a process is therefore not necessarily a simple feat, and doing it experimentally may become both time-consuming and expensive. Computer-aided optimization therefore lends itself as a viable alternative, since it can be performed in a relatively short time at a low cost, whilst providing insights into how to improve a process.

The work collected and presented in this thesis concerns three different complex processes in the two categories of production systems, and urban environments, all in need of improvements. These three cases have thus been the subjects of a series of studies collected and presented in this thesis, appended as Papers I-IX. For production systems, two cases are presented. The first case is the purification of a polyalcohol at Perstorp AB, the process of which was subject to process-disturbing oscillations that were analyzed and their effects quantified through an *in-silico* sensitivity analysis, and subsequently reduced by 99.7% via a computer-aided dynamic optimization approach. Secondly, the chromatographic purification of pharmaceuticals has been studied, where optimization techniques were used for the dynamic optimization of the loading phase of a capture step as well as the optimal design of integrated column sequences. In the category of urban environments, blue-green systems for urban flooding mitigation in the city of Malmö have been studied, where a model was developed and then applied for optimal economic siting and sizing of retrofits taking into account implementation and flooding-induced costs. With a 3% decrease in costs, whilst modest, this result shows that the blue-green systems are economical alternatives to consider for mitigating urban flooding.

The purpose of this thesis – in addition to improving the three aforementioned processes – is to provide tools, in the form of a procedure and a framework, that can be used to increase the accessibility of computer-aided optimization. The aim is that these tools can be useful to practitioners and facilitate studies, which in turn can be used as a foundation for improving any given complex situation or process. In addition, a summary is presented of the most important parts of an optimization problem and its formulation, and the most relevant optimization approaches to the work presented in this thesis are also summarized and explained. Furthermore, the presented tools are used to describe and analyze the work that has been performed and is presented in this thesis. They are also used to explain how the choices that are made regarding tools for modeling, simulation, and optimization may impact each other, and thus have an interplay. With these general and generalizable tools, the aim is to contribute towards an increased understanding and thereby accessibility of computer-aided optimization.

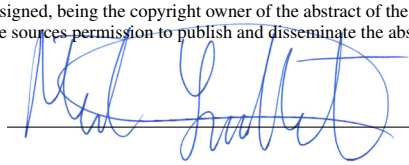


**Keywords**
Optimization, modeling, simulation, complex processes, process systems engineering, preparative chromatography, blue-green systems

**Classification system and/or index terms (if any)**

| | | |
|---|---|---|
| **Supplementary bibliography information** | | **Language**<br>English |
| **ISBN (print)**<br>978-91-7422-732-1 | | **ISBN (pdf)**<br>978-91-7422-733-8 |
| **Recipient's notes** | **Pages**<br>209 | **Price** |
| | **Security classification** | |




**Signature**  **Date:** March 20, 2020

# Computer-aided optimization of complex processes in production systems and urban environments

Mikael Per Daniel Yamanee-Nolin

Department of Chemical Engineering
Lund University, Sweden
2020


LUND
UNIVERSITY

Computer-aided optimization of complex processes in
production systems and urban environments



Cover photo


Department of Chemical Engineering
Lund University
P.O. Box 124
SE-221 00 LUND, SWEDEN

ISBN (print) 978-91-7422-732-1
ISBN (pdf) 978-91-7422-733-8

Printed by Media Tryck, Lund University, Lund, 2020


Media-Tryck is a Nordic Swan Ecolabel
certified provider of printed material.
Read more about our environmental
work at www.mediatryck.lu.se

MADE IN SWEDEN

*I hear babies cry and I watch them grow*
*They'll learn much more than we'll know*
*And I think to myself:*
*What a wonderful world!*

— Israel Kamakawiwo'ole

# Abstract

Complex processes can be seen as a series of actions, changes, and/or events that are related in different temporospatial manners, and thus are potentially difficult to understand. Optimizing such a process is therefore not necessarily a simple feat, and doing it experimentally may become both time-consuming and expensive. Computer-aided optimization therefore lends itself as a viable alternative, since it can be performed in a relatively short time at a low cost, whilst providing insights into how to improve a process.

The work collected and presented in this thesis concerns three different complex processes in the two categories of production systems, and urban environments, all in need of improvements. These three cases have thus been the subjects of a series of studies collected and presented in this thesis, appended as Papers I-IX. For production systems, two cases are presented. The first case is the purification of a polyalcohol at Perstorp AB, the process of which was subject to process-disturbing oscillations that were analyzed and their effects quantified through an *in-silico* sensitivity analysis, and subsequently reduced by 99.7% via a computer-aided dynamic optimization approach. Secondly, the chromatographic purification of pharmaceuticals has been studied, where optimization techniques were used for the dynamic optimization of the loading phase of a capture step as well as the optimal design of integrated column sequences. In the category of urban environments, blue-green systems for urban flooding mitigation in the city of Malmö have been studied, where a model was developed and then applied for optimal economic siting and sizing of retrofits taking into account implementation and flooding-induced costs. With a 3% decrease in costs, whilst modest, this result shows that the blue-green systems are economical alternatives to consider for mitigating urban flooding.

The purpose of this thesis – in addition to improving the three aforementioned processes – is to provide tools, in the form of a procedure and a framework, that can be used to increase the accessibility of computer-aided optimization. The aim is that these tools can be useful to practitioners and facilitate studies, which in turn can be used as a foundation for improving any given complex situation or process. In addition, a summary is presented of the most important parts of an optimization problem and its formulation, and the most relevant optimization approaches to the work presented in this thesis are also summarized and explained. Furthermore, the presented tools are used to describe and analyze the work that has been performed and is presented in this thesis. They are

also used to explain how the choices that are made regarding tools for modeling, simulation, and optimization may impact each other, and thus have an interplay. With these general and generalizable tools, the aim is to contribute towards an increased understanding and thereby accessibility of computer-aided optimization.

# Populärvetenskaplig sammanfattning

## Stabilare kemikalieproduktion, billigare läkemedel, och välplacerade översvämningsskydd – med hjälp av samma verktyg!

Visst vore det fint om vi kunde producera nödvändiga kemikalier mer effektivt, förse alla människor med de läkemedel de behöver till en lägre kostnad än vad som är möjligt idag, och förhindra översvämningar i stadsmiljö? Men vad har dessa tre ting gemensamt, egentligen? Jo, att man kan *optimera* de olika *processer* som ligger bakom alla tre! Processer finns i princip överallt i världen, komplexa som enkla, konkreta som abstrakta. Att optimera en process innebär att vi försöker förbättra den och göra den så bra som bara är möjligt. Om detta ska göras via experiment kan det dock ta väldigt lång tid och även bli väldigt dyrt – och ibland är det till och med praktiskt omöjligt. Ett gångbart alternativ är då att ta hjälp av modern beräkningskraft och istället utföra modellbaserade optimeringsstudier med hjälp av datorn, d.v.s. datorstödda optimeringsstudier. För detta ändamål krävs en matematisk *modell* som behöver *simuleras* för att vi på så sätt ska kunna få ut användbar information om vår verkliga process som vi kan basera vår optimering på.

I denna avhandling presenteras en övergripande, allmängiltig procedur och ett slags ramverk tillsammans med ett antal optimeringsmetoder som kombinerat kan användas som verktyg för att effektivt utföra datorstödda optimeringsstudier. Dessa verktyg baseras dels på sedan tidigare publicerad litteratur, men också på arbetet som samlas och presenteras i denna avhandling, och målet är att de ska kunna användas som hjälp för att förbättra i princip vilken verklig situation som helst.

Det samlade arbetet ger också en fingervisning om vilka resultat man kan nå via datorstödda optimeringsstudier. Till exempel så används kommersiellt tillgängliga datorprogram för att skapa modeller. Sedan används dessa modeller för att undersöka och stabilisera en process för kemikalieproduktion. Denna process led tidigare av störningar som orsakade lägre produktivitet och större energiåtgång, vilka i princip eliminerades. Vidare har även en typ av läkemedelsproduktion förbättrats genom att matematiska modeller använts för att visa hur man på bästa sätt bör skapa och köra en särskild typ av process för läkemedelsupprening, vilket i slutändan ger billigare mediciner. Slutligen har även så kallade blå-gröna system undersökts, vilket exempelvis är dammar och gröna tak som används för att minska risken för översvämningar. Dessa har modellerats för att ta reda på var man bör bygga sådana system i Malmö stad, och hur stora de bör vara, för att förhindra översvämningar utan att det blir onödigt dyrt.

# Acknowledgments

I firmly believe that one of the most important parts of life is gratitude, and I also believe it helps set you in a positive mindset. Therefore, there is no procrastination in this endeavor, and I would like to take the opportunity here to express my gratefulness to those who have been with me on this path. To begin, I would like to thank my main supervisor, Professor Bernt Nilsson. I am very grateful for your exemplary capacity for building grand visions, creating opportunities, and clearing obstacles whilst keeping individual well-being in focus, and you have certainly made my journey far more enjoyable than I could have hoped for. I am also grateful towards Niklas Andersson, my co-supervisor, who has always been a great support in all kinds of questions, ranging from technical problems, to issues with writing and the publishing process, to stepping in for Bernt when most needed. You have never failed to set a good precedent with your splendid priorities, ingenuity, and uplifting spirits. I would also like to thank my former co-supervisor, Anders Bojsen, without whom I probably never would have even considered doing a PhD. It was a pleasure and privilege to work with you, and I learned a lot about work capacity (in general, as well as my own specifically) and work-life balance during the time(s) I worked with you.

There are a few colleagues in particular that I would like to extend my gratitude towards. Someone who was almost like a mentor for me, but probably closer to an academic older brother, is my good friend and former research colleague, Anton Sellberg. I am very grateful for the laughs and experiences we shared (and all of us continue to share!), for your advice and wholesome pieces of wisdom, and for the directness with which you conduct yourself. I would also like to thank Salar Haghighatafshar, whose passion for his work is inspiring to the degree that it is contagious beyond belief, lending me energy when I needed it most. Your eye for artistry and aesthetics is fantastic, be it within photography or regarding how to present our research. I am also grateful to have had the opportunity to work with three sharp minds and hard workers at Perstorp AB, namely Oleg Pajalic, Mark Max-Hansen, and John Berggren. The deep, entertaining and wavy discussions we have had could probably power a medium-sized European nation if someone figured out how to extract usable energy from spoken words.

I have had the opportunity to share an office, share cups of coffee, and work with a row of brilliant people, and I would like to thank a few people in particular. Starting with those who have come and gone in Professor Bernt Nilsson's team of PhD students, I would like to begin with Karolina Arkell. Thank you for welcoming me on my first

day – I probably would have frozen to pieces in a pile of snow if not for you – and for being the structured kind of role model I needed when starting out. Hans-Kristian Knutson, thank you for the good lunches, the good atmosphere, and for the good advice on how to combine the academic and industrial sides of our work. Anton Löfgren, thank you for being a solid rock of knowledge and a flowing river of curiosity at all times, helping others (including myself, of course) to expand their views and to learn new things. Simon Tallvod, thank you for your almost mysterious versatility and creativity, which contribute to your ability to lighten up any day with your quirkiness, and thank you for the great times we spent teaching together. Ximo Gomis Fons, thank you for your patience with my Scanian accent, and for your genuine earnestness, and capacity for making research concrete, for making it – simply – come to life. Madelène Isaksson, thank you for being a fantastic office mate, with fresh ideas and perspectives on everything from this very thesis to how to organize the Christmas decorations, with clarity in communication, and with a good laugh always at the ready. I would also like to take the opportunity to thank my friend Maria Messer, administrator at the Department of Chemical Engineering. Thank you for being the department's own unstoppable force, always doing more than I feel we could ever notice and thank you for. Special thanks go to my special teammates Anneleen De Cuyper, Pim van den Oetelaar, Alexa van Heel, and Katie Cavendish – remember: "Things don't, people do." – as well as to my unofficial mentor Anna Blomborg for supporting me through what would have been an otherwise impossibly hectic period, helping in filling it with valuable experiences, learning, and development. I would also like to say 'thank you' to all of my colleagues on Floors -1 and +1 for making me feel at home during my years as a PhD student – it has been a great place to be, thanks to you!

I would also like to take a moment to thank my friends and family, of course. All my dear friends, my siblings, my parents, my in-laws, nieces and nephews – you are too many to be named, except maybe for Matte, my dear grandmother – thank you for being wonderful people, learning and teaching, caring and challenging, throughout life, through everything we get up to. Finally, my most heartfelt gratitude is reserved for my dear wife, and my dear son. Arina and Allan, my best friends and most beloved people on this Earth, I love you both with all my heart, beyond words. Thank you both for our shared lives, thank you for bringing color and focus to my life in a way that will always amaze me, and thank you for continuing to give me new things to learn – and especially laugh – about!

The Leap Day of 2020, Lund
*Mikael Per Daniel Yamanee-Nolin*

# Contents

# Preface

## Contents and Contributions of the Thesis

This thesis consists of five introductory chapters, and nine papers. This section describes the five chapters, the contributions of each paper, and the contributions made by the author. The papers, which will be referred to in the text by their roman numerals, are appended at the end of the thesis.

### Chapter 1 — Introduction

The first chapter describes the aim and purpose of this thesis. A brief introduction to optimization is offered, and the three cases of complex processes – two in production systems, and one in urban environments – that have been the primary subjects of the work collected and presented in this thesis are described.

### Chapter 2 — Optimization in engineering applications

In this chapter, the most important parts of an optimization problem – namely the objective function, decision variables, and constraints – are presented. A general procedure for solving an optimization problem is also presented, along with the optimization approaches (and abstractions thereof) that are most pertinent to the work presented in this thesis.

### Chapter 3 — The interplay between optimization, modeling, and simulation

How the choices of optimization, modeling, and simulation tools impact each other is covered in this chapter based on a reflection of the work presented in this thesis, combined with the aforementioned optimization procedures as well as procedures for creating mathematical process models and appropriate simulation strategies. An integrated procedure for solving an optimization problem is presented along with a purpose-centric framework, and the aim is that these may be used to facilitate optimization studies in order to improve decisions and processes.

### Chapter 4 — Computer-aided optimization of complex processes

The integrated procedure and purpose-centric framework from the previous chapter are used in this chapter to describe and contextualize the work presented in this thesis. This analysis highlights some of the most important results from the publications, and serves as an example to demonstrate the usefulness of utilizing the aforementioned procedure and framework for computer-aided optimization of complex processes.

### Chapter 5 — Concluding Remarks

This chapter concludes the thesis with conclusions drawn from the presented work and ideas for future work.

### Paper I — Unbiased selection of decision variables for optimization

Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2017). Unbiased Selection of Decision Variables for Optimization. In Espuña, A., Graells, M., and Puigjaner, L., editors, *27th European Symposium on Computer Aided Process Engineering*, volume 40 of *Computer Aided Chemical Engineering*, pages 253–258. Elsevier

A subset selection algorithm previously applied for parameter estimation was in this paper applied towards the selection of decision variables for optimization. The algorithm was shown to be capable of reducing the number of potential options, which may be numerous for a complex process, by ranking the available decision variables according to a key quantity. This methodology for ranking and recommendation can help a practitioner make a more informed selection.

I implemented the subset selection algorithm for decision variable selection, took the lead in model creation, set up the optimization problem, analyzed the acquired data, and wrote the paper.

### Paper II — Analysis of an Oscillating Two-Stage Evaporator System through Modelling and Simulation: an Industrial Case Study

Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2018). Analysis of an Oscillating Two-Stage Evaporator System through Modelling and Simulation: an Industrial Case Study. *Chemical Engineering Transactions*, 69:481–486

An *in-silico* sensitivity analysis of an oscillating two-stage evaporator was performed in order to analyze and quantify the impact of the oscillations on process performance. This analysis was performed by automating the perturbations and data collection necessary for the sensitivity analysis, and analyzing the correlations between different parts of the process using phase planes.

I further developed the model from Paper I, implemented the automation of the sensitivity analysis, analyzed the acquired data, and wrote the paper.

### Paper III — Single-shooting optimization of an industrial process through co-simulation of a modularized Aspen Plus Dynamics model

Yamanee-Nolin, M., Löfgren, A., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2019). Single-shooting optimization of an industrial process through co-simulation of a modularized Aspen Plus Dynamics model. In Kiss, A. A., Zondervan, E., Lakerveld, R., and Özkan, L., editors, *29th European Symposium on Computer Aided Process Engineering*, volume 46 of *Computer Aided Chemical Engineering*, pages 721–726. Elsevier

The method presented in this paper consisted of modeling a two-stage evaporator and a methanol column in a modularized fashion, utilizing Aspen Plus Dynamics. The objective of this study was to minimize the total cost induced by the steam consumption. The optimization was done via cosimulation of the two modules using a toolchain named the Python Module Coupler (PyMoC).

I edited the model from Paper II for coupling with the second model, which I also developed in conjunction with the work presented in this paper, created and implemented the PyMoC toolchain in Python, partook in the formulation of the optimization problem, performed the optimization, analyzed the results, and wrote the paper.

### Paper IV — Trajectory optimization of an oscillating industrial two-stage evaporator utilizing a Python-Aspen Plus Dynamics toolchain

Yamanee-Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2020). Trajectory optimization of an oscillating industrial two-stage evaporator utilizing a Python-Aspen Plus Dynamics toolchain. *Chemical Engineering Research and Design*, 155:12–17

A dynamic optimization study was performed in this paper. The oscillations that were analyzed and quantified in Paper II were minimized – nearly eliminated – in Paper IV using a modified version of PyMoC, which was first introduced in Paper III. Modifying the toolchain for cosimulation to handle a direct sequential approach for trajectory optimization was the key to realizing this study, and shows the flexibility and generalizability of the toolchain itself.

I modified the PyMoC toolchain, formulated the optimization problem, performed the optimization, analyzed the results, and wrote the paper.

### Paper V — Modularization, co-simulation, and optimization of complex dynamic production processes in the ProOpt project

Yamanee-Nolin, M. (2019). Modularization, co-simulation, and optimization of complex dynamic production processes in the ProOpt project. Technical report, Department of Chemical Engineering, Faculty of Engineering, Lund University, P.O. Box 124 SE-22100 LUND

This technical report offers some insight into the background and results of the ProOpt project. It also provides the reasoning for modularization as well as details on the workings of PyMoC toolchain, how it handles cosimulation, and explains and motivates the

practical aspects for the user.

I wrote this internal report, created the illustrative examples as well as the described toolchain, and played a significant part in the creation of the included models.

### Paper VI — Multi-flowrate Optimization of the Loading Phase of a Preparative Chromatographic Separation

Sellberg, A., Nolin, M., Löfgren, A., Andersson, N., and Nilsson, B. (2018). *Multi-flowrate Optimization of the Loading Phase of a Preparative Chromatographic Separation*, volume 43 of *Computer Aided Chemical Engineering*, pages 1619–1624. Elsevier B.V

This paper presents a study of the optimal loading of the capture step of preparative chromatographic purification using a time-variant flow rate. A preexisting modeling, simulation, and optimization toolchain was applied to minimize an objective function consisting of a weighted sum of productivity, resin utilization, and yield.

I contributed towards planning, the optimization problem formulation, as well as the writing of the paper.

### Paper VII — Optimization of integrated chromatography sequences for purification of biopharmaceuticals

Löfgren, A., Yamanee-Nolin, M., Tallvod, S., Fons, J. G., Andersson, N., and Nilsson, B. (2019). Optimization of integrated chromatography sequences for purification of biopharmaceuticals. *Biotechnology Progress*, 35(6)

A new method for nonlinear optimization of integrated sequences was presented. The method takes individual column sizes, flow rates, and scheduling into consideration, and can be used for a wide range of different downstream design cases, which was showcased in the paper.

I contributed in particular to method formulation and creation, as well as the later phases of writing the paper.

### Paper VIII — A physically based model for mesoscale SuDS – an alternative to large-scale urban drainage simulations

Haghighatafshar, S., Yamanee-Nolin, M., and Larson, M. (2019b). A physically based model for mesoscale SuDS – an alternative to large-scale urban drainage simulations. *Journal of Environmental Management*, 240:527–536

A novel model for describing the dynamic behavior of sustainable drainage systems known as blue-green systems was presented. Based on easily quantifiable parameters of the physical characteristics of the system components, the model was designed to accept a time series of a rainfall event and use it to simulate the flow of rainwater from the modeled blue-green system to the urban drainage piping network.

I implemented the model and simulation strategy in Python, calibrated the model, and also contributed towards the analysis and evaluation of model output as well as the writing of the paper.

### Paper IX — Hydroeconomic optimization of mesoscale blue-green stormwater systems at the city level

Haghighatafshar, S., Yamanee-Nolin, M., Klinting, A., Roldin, M., Gustafsson, L.-G., Aspegren, H., and Jönsson, K. (2019a). Hydroeconomic optimization of mesoscale blue-green stormwater systems at the city level. *Journal of Hydrology*, 578

An optimization study where the hydroeconomically optimal sizing and siting of blue-green systems across the city of Malmö was presented. For this purpose, the model created in Paper VIII was used in conjunction with models describing impervious-area rainfall runoff and the piping network behavior. An optimization framework wrapping around these models (and the cosimulation thereof) was implemented in Python and successfully used to perform the optimization.

I implemented the optimization framework in Python, contributed towards the results analysis, and wrote parts of the paper.

### Related Oral Presentations (Lead Author/Presenter)

Below is a list of conference presentations by the author related to the work described in the above papers.

Oral presentation at the conference Distillation & Absorption 2018, Florence, Italy.
Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2018). Analysis of an Oscillating Two-Stage Evaporator System through Modelling and Simulation: an Industrial Case Study. *Chemical Engineering Transactions*, 69:481–486

# 1

# Introduction

Do you agree that it would be nice if things were just *better*? Good! Then you agree that optimization is both important and interesting! Optimization in its purest form is concerned with the minimization of a mathematical function. Applied in the right way, optimization becomes a generalizable methodology that is very useful for improving just about any conceivable situation or decision-making process in an effective and efficient way. Computer-aided (or model-based) optimization in particular becomes a fantastic tool to find out how to improve, for instance, a production process in a way that reduces the time and costs incurred by extensive and expensive experiments. Sometimes, computer-aided optimization may facilitate studies that would be practically impossible to perform in real life due to spatial and financial obstacles.

Computer-aided optimization draws upon the usage of one or several mathematical *models*, which need to be *simulated* in order to retrieve information regarding how to *optimize* a problem. The models typically consist of a set of equations that can be established either through first principles to create mechanistic models, by using a plethora of data to fit statistical models, or through a hybrid approach that makes use of the advantages of both aforementioned approaches. The models need to be able to accept input that affect the output relevant to the optimization. The models must then be simulated, i.e. the equation set needs to be solved. This can be done statically or dynamically (or both) depending on the purpose of the study and the physical problem that is to be described. In the end, retrieving information from the simulation is essential in order to calculate the metric that is intended to be improved in the optimization problem.

In this general methodology of modeling, simulation, and optimization, there are many questions to answer and factors to consider. The purpose of a study is at least as important to keep in mind as the methods and tools that are used. However, the choices in terms of problem formulation, modeling, and simulation will all interact with each other and make or break opportunities in a given study. These choices can, for instance, be choices regarding what tools to use, which methods to apply, or regarding specific equations in the problem formulation. In other words, *why* a study is performed, *how* it is performed, and *what* the results are and what physical situation is being studied, all become interconnected in ways that need to be considered for a study to be successful.

The purpose of this thesis is to provide general and generalizable tools – in the form of a procedure and a framework – that can be combined and used for abstraction in order to increase the understanding and the accessibility of computer-aided optimization.

The aim is that these tools can support and be useful for practitioners, thus facilitating studies that in turn can be used as a foundation for improving any given situation or process. This purpose is achieved by reflecting on the impact and the interplay of the choices that are made regarding modeling, simulation, and optimization during studies of different physical problems. This reflection is based on available literature in combination with the work presented and contextualized in this thesis, appended as Papers I through IX, which are studies focused on two different cases of complex processes in production systems, as well as one case of a complex process in the urban environment. In each of these three cases, there was a need to improve the particular process. Thus, an additional, concrete goal of this thesis is to contribute towards the stabilization and improvement of a polyalcohol purification process, cheaper production of pharmaceuticals through efficient chromatographic separation, and more effective urban flooding countermeasures using blue-green systems – all through computer-aided optimization.

## 1.1 A definition and three cases of complex processes

Since the work presented in the current thesis primarily revolves around computer-aided optimization of complex processes in production systems and the urban environment, we must first agree upon what we mean by 'complex process'. As a starting point, the Cambridge Dictionary[1] provides two definitions for the adjective 'complex', stating that something is complex if it:

i. involves a lot of different but related parts, and/or

ii. is difficult to understand or find an answer to, due to having many different parts

With 'process' defined as 'a series of actions/changes'[2], these definitions combined capture the essence of what is meant by complex processes in this thesis: a series of actions, changes and/or events that are related in different temporospatial manners and therefore possibly difficult to understand. In the context of production systems, a process can be considered complex if there are many interconnected unit operations and/or dominating physical phenomena that are pertinent to the process. For instance, a purification process that has been retrofitted for the purpose of resource recovery becomes more complex as matter and energy are recycled. Whilst there are many interconnected parts of urban drainage processes as well, these are considered complex also due to the difficulty of finding answers directly from the process itself, which in turn is partly caused by the unpredictability and lack of control of important factors, such as load and position of real-life rainfall.

Continuing, the processes that are relevant to this thesis need to be introduced. In total three different cases are presented in the two categories of production system processes and urban environment processes. The first two cases are complex processes in production systems. The first process is the industrial polyalcohol purification process at Perstorp AB, for which a Python-Aspen Plus Dynamics toolchain (described in Paper

[1] https://dictionary.cambridge.org/dictionary/english/complex

[2] https://dictionary.cambridge.org/dictionary/english/process

V) was created and used for performing a case study for an algorithm for selecting decision variables (Paper I), another case study to analyze and quantify process-disturbing oscillations (Paper II), and two separate optimization studies (Papers III and IV). The second process is in the field of chromatographic purification of pharmaceuticals, where two optimization studies have been performed (Papers VI and VII). The third and final case is in the urban environment. This case is a type of sustainable urban drainage system known as blue-green systems, aptly named from their use of urban foliage and ponds to combat flooding caused by heavy rainfall. The blue-green systems have been modeled and analyzed (Paper VIII) as well as optimized with respect to implementation extent and location (Paper IX). The appended papers will be referred to by their roman numerals – as in the above – and used as examples to explain relevant concepts throughout this thesis. The remainder of this section provides descriptions of the physical processes of the three different cases.

### 1.1.1 Production systems

#### *1.1.1.1 Industrial purification of a polyalcohol*

The purification of a polyalcohol at Perstorp AB entails a large, industrial process that makes use of a range of heat intense unit operations, from evaporators, to crystallizers and distillation columns in series. Of particular interest to the work presented in this thesis has been the evaporator system directly following an upstreams batch system, presented schematically in the dashed box in Figure 1.1. This evaporator system is a complex process partly due to the recycling within the evaporator system itself; partly due to recycling of matter and energy from other parts of the plant, both of which are intended to recover or reuse resources; and partly due to the physical phenomena, e.g. vapor-liquid equilibrium (VLE), that are important to this separation process.

The system begins with a balance tank acting as a buffer between the upstream batch system and the continuous evaporator system. The upstream system intermittently feeds the balance tank with a solution that is highly concentrated with respect to the product, i.e. the polyalcohol (The ‘Feed’ stream in Figure 1.1). Furthermore, the balance tank is fed by a dilute recycling stream from a part of the downstream process. The balance tank mixture is then fed to a stripping column to remove water. The bottom stream of the stripping column is heated by a stream from a mechanical vapor recompression (MVR) circuit, containing added steam and water (the former controlled via the boiling-point elevation (BPE) in the second evaporator stage), before being fed to the first evaporator stage. The top stream of the first evaporator stage drives the stripping column, whereas the bottom stream is fed to the second stage of the evaporator system. The bottom of the second stage is the product stream, named ‘Prod’ in Figure 1.1, which is sent downstream for further purification. It should be noted that not all vapor is recycled in the MVR circuit; parts of the streams are removed from the system and primarily sent to a methanol column (which is included in the study presented in Paper III), visualized as the streams exiting the dashed box at the bottom right of Figure 1.1.

The feed from the batch system contains water, salt, methanol, formaldehyde and a range of by-products, in addition to the polyalcohol product. The presence of formaldehyde makes the VLE more complicated because we must take the oligomerization into account, which was done by utilizing the findings of Maurer (1986). Without the Mau-

rer reactions for formaldehyde oligomerization, computational accuracy with regards to the VLE would suffer greatly, which in turn would affect e.g. boiling points, and thus the mass and energy balances. However, if all of the Maurer reactions are implemented, computational performance would suffer. Taking this balance between accuracy and performance into consideration, only the methylene glycol reaction as well as the polyoxymethylene reactions for $n \leq 4$ from Maurer (1986) were implemented.

One of the main problems in this particular industrial process was the product concentration oscillations that arose in the balance tank due to the mixing of the concentrated, intermittent feed stream and the dilute, continuous recycling stream. This was of special focus in Paper II where the oscillations were first analyzed and quantified via sensitivity and phase-plane analysis, and then minimized using a dynamic optimization approach in Paper IV. Whilst not the main focus, the oscillations were taken into consideration in the co-simulation/optimization study presented in Paper III as well, since they were expected to affect the solution when attempting to minimize the overall steam consumption in the evaporator system and methanol column. The process was also used as a case for the application of a subset selection algorithm to select decision variables for optimization in Paper I. The mathematical process flowsheet models utilized for these studies were created using tools available in the AspenTech suite, which – similarly to the applied methods and approaches for optimization – are common within the field of process systems engineering (Grossmann and Harjunkoski, 2019). These models are described in greater detail along with the Python-based toolchain in the supplementary technical report appended as Paper V.

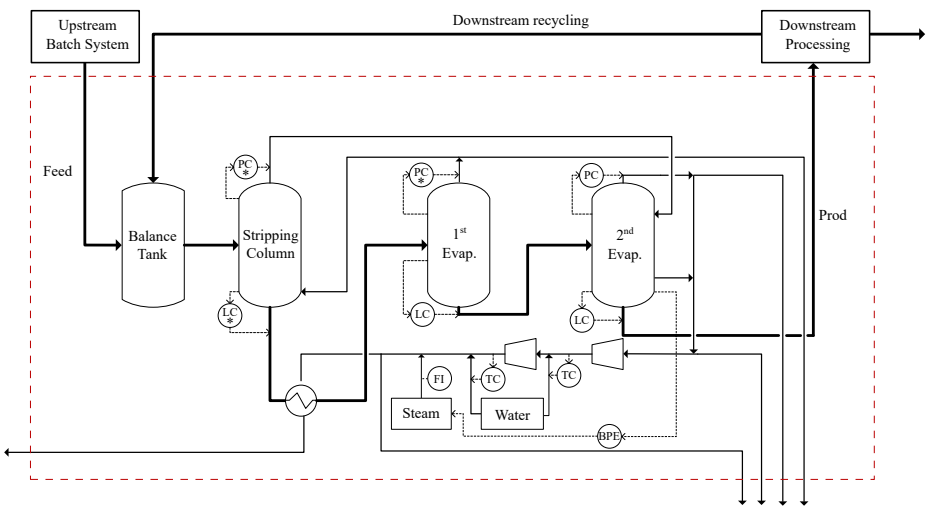



**Figure 1.1** The two-stage evaporator system in the polyalcohol purification process at Perstorp AB. The evaporator system is depicted together with the upstream batch system and downstream processing for context.

#### 1.1.1.2 Chromatographic purification of pharmaceuticals

Preparative liquid chromatography is a widely used separation technique across several industries, where the goal is to separate a target component from a liquid mixture that contains at least two components, to purify the target component (Sellberg, 2018). An illustrative example of a chromatographic process is presented in Figure 1.2, which works by first having the sample loaded into the column. The two buffers, named A and B, are then used to control the separation either under isocratic conditions or by applying a buffer concentration gradient (Schmidt-Traub et al., 2012). The separation occurs in the column through adsorption of solutes in the mobile (liquid) phase onto the stationary (solid) phase of the column, the choice of which depends on the solutes to be separated. The different types of solutes in the sample will elute at different rates depending on their physical properties and the resulting interaction with the stationary phase. Those solutes that interact more strongly with the stationary phase are hindered and will elute at a later time than those solutes that interact with the stationary phase to a lesser extent. Following the column, measurements can be performed continuously in order to analyze the separation. Common techniques include measuring the UV absorbance, the conductivity, and the pH of the solution leaving the column, and using this information to separate the flow into product and waste containers.

Ideally, the complete separation of the components is desired, but this may be very difficult to achieve in reality. Complexities arise from the different ways to set up a chromatographic separation process, which may range from the rather simple batch setup described above, to the more complicated Multicolumn Countercurrent Solvent Gradient Purification (MCSGP) (Aumann and Morbidelli, 2007) or Simulated Moving Bed (SMB) (Rodrigues et al., 2015) techniques. Gomis-Fons et al. (2019) also provides a good example of a more complex configuration than the illustration provided in Figure 1.2. Additionally, the nonlinear interaction between mobile phase, stationary phase and solutes results in a complex separation process even for the batch setup. These physiochemical phenomena are shown for a typical adsorption column in Figure 1.3, where four different scales can be discerned (Sellberg, 2018). At the macro scale, the stationary phase is in the form of a packed bed of resin that the mobile phase needs to be pumped through, with issues such as radial distribution of mobile phase veloc-

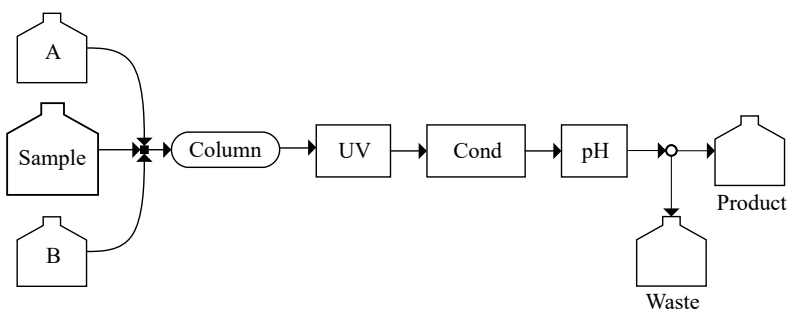



**Figure 1.2** A simple schematic of a chromatographic separation process with a sample, two buffers (A and B), and the column itself followed by analytical sensors. In the end, the product is separated from the waste.

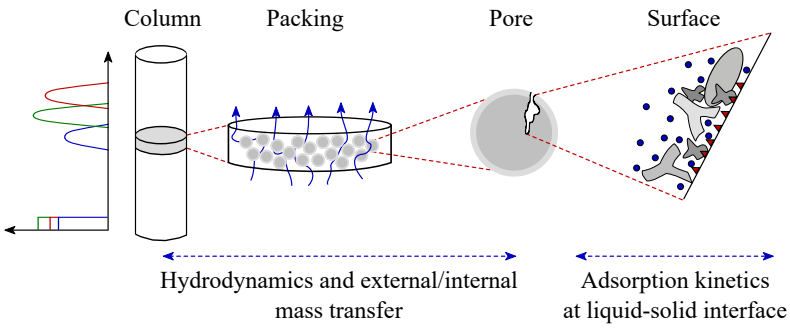



**Figure 1.3** An overview of the different scales in chromatographic separation and how they relate to each other. Slightly modified and printed with due permission of the original creator, Anders Bojsen, previously at the Department of Chemical Engineering at Lund University.

ities (Guiochon, 2006) and hydrodynamic memory of the packed bad (Hekmat et al., 2011) that may need consideration. The mobile phase is in contact with the particles in the packed bed throughout the column, to the surface and into the pores of which the solutes are transported via mass diffusion (Schmidt-Traub et al., 2012). Finally, at the atomic scale in the pores, there are surfaces on which there are sites binding the solutes, the action of which will affect the solutes differently since they will be bound to the sites to different extents. The mass transfer phenomena at and between these different levels are all coupled and all affect the degree of separation. However, the stationary phase dynamics at the atomic scale can be considered most important since this generally determines the separation (Sellberg, 2018).

Due to the many different applications of chromatographic separation processes, there are many interesting problems to study. In the work presented in this thesis, two quite different situations have been studied. In Paper VI, the loading step of a capture column was optimized using a multi-flow rate approach. This entailed a single column in a batch setup, and used a detailed, mechanistic model that was solved as a constraint in the optimization problem. A toolchain consisting of several different software was used to model the process, simulate the model, and to extract analytical Jacobian information for use by the optimizer. In all of these aspects, Paper VII is almost the opposite, as a heuristic was developed for optimizing integrated chromatography sequences, i.e. several columns in a series, with regards to the size and thus the resulting investment cost of the columns and the productivity. No separation model was necessary to accomplish this optimization study. The gradient-based optimizer using finite differences for the Jacobian known as SLSQP (Kraft, 1988) was then used to perform the actual optimization study. In the context of this thesis, the case of chromatographic separation may be considered a secondary case included primarily for the optimization approaches that are of particular interest to the second chapter. However, this case will not be revisited in the fourth chapter, in which the procedure and framework provided for the purposes of this thesis are combined and applied for description/analysis of the other two cases.

## 1.1.2 Urban environments

### *1.1.2.1 Blue-green systems for sustainable urban drainage*

When it rains, it pours – and when it pours in densely populated areas, drainage networks that rely exclusively on piping systems may become overloaded, causing urban flooding. One solution to this problem is to build larger systems that can handle more stormwater at any given time, but this would be expensive and presumably unsustainable in an era of intensified urbanization during which the weather is growing more extreme (Haghighatafshar, 2019). A more flexible alternative to adding more concrete-based structures is to implement blue-green systems (Eckart et al., 2012). The general idea behind these open systems is that we should modify the available urban surfaces so that they absorb and accommodate more water on the surface via infiltration, storage, and retention. This modification can be done by changing impervious surfaces (e.g. asphalt and concrete) into permeable surfaces (e.g. open grass areas or ponds), as visualized in Figure 1.4.

Collecting stormwater using blue-green systems slows the release of water into to sewer networks, decreasing the risk of flow rate spikes in urban piping systems, and thereby decreases the risk of urban flooding (Sörensen and Emilsson, 2019). In addition to slowing down the release to the sewer network, the water may also evaporate when collected in larger areas, which removes the load from the piping system completely. In an analysis presented by Haghighatafshar et al. (2018b), the resulting volumetric load on the piping network from a 6.5-year (indicating the average occurrence and therefore intensity of the rain) rainfall on an area mostly consisting of impervious surfaces was quantified. The authors wanted to define a rain that gave the same resulting volumetric load if said area had blue-green systems implemented. Through simulations, it was

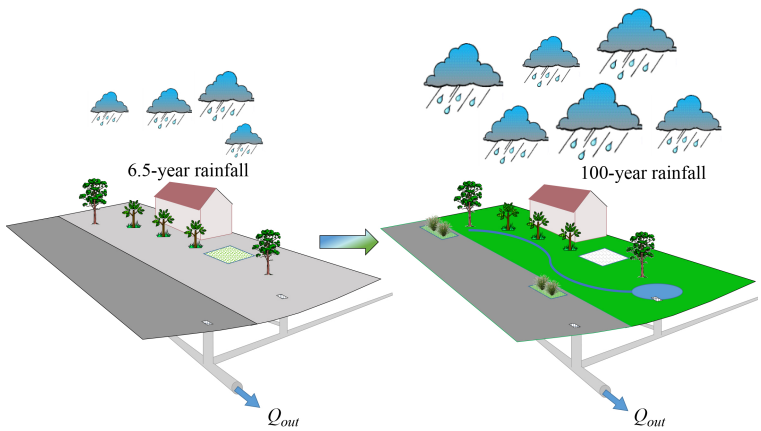



**Figure 1.4** Blue-green systems and their effect on mitigating floods due to overload in urban drainage systems: the same area with implemented blue-green systems delivers the same load to the piping network ($Q_{out}$) during much heavier rainfall compared to the same area with mostly impervious surfaces. The blue-green systems also carry aesthetic advantages that urban citizens generally enjoy! Modified and printed with due permission of the original creator, Lars-Göran Gustafsson, DHI Sverige AB.

found that a 100-year rainfall would give the same load to the piping network, denoted $Q_{out}$ in Figure 1.4. This example gives a good idea of the potential blue-green systems have in mitigating urban flooding, which can save lives and reduce damage-induced costs.

This process is made complex not only by the fact that the input, i.e. the rainfall, can be hard to predict and is in natural situations beyond human control, but also due to the way the blue-green systems are coupled amongst themselves and together with existing piping networks (Hoang and Fenner, 2016). Furthermore, these piping networks extend throughout cities, and may also contain local loops. The complexity thus spans from the micro scale (e.g. the function of one single catchment) across the meso scale (e.g. the relation of several coupled catchments) on towards the macro scale (e.g. the connections across an entire city) (Haghighatafshar, 2019).

In order to perform a computer-aided optimization study of the blue-green systems placement and sizes in Malmö, a conceptual model for the blue-green systems was developed and calibrated in Paper VIII. This model performed well with great accuracy and computational performance compared to available measurements for a set of different rainfalls, and was subsequently used in a co-simulation based optimization study presented in Paper IX. In the optimization study, the conceptual model was used together with a city-wide piping network model to find the optimal siting and sizing of blue-green systems in the city of Malmö in terms of minimizing the combined costs incurred by the floodings as well as the blue-green system implementation.

# 2

# Optimization in engineering applications

Nothing, or at least very little, is impossible to improve. Therefore, engineers optimize, the task of which can be defined as finding the best solution within constraints given a specific process or system (Biegler, 2010). To do this, we need to formulate mathematical versions of the physical problem(s), and then we need to choose how to approach the mathematical problem. Given a mathematical model representing the real-world process, we can then solve the optimization problem. By convention, *to optimize* is the same as *to minimize*, and thus the two terms are used interchangeably.

In this chapter, general optimization problem formulation is presented along with its core components, followed by a summary of the most relevant optimization approaches with regards to the work presented in this thesis.

## 2.1 Optimization problem formulation

In engineering practice, there is a need for a systematic approach towards formulating and communicating the mathematical optimization problem, and problem formulation is probably the most important part of solving an optimization problem (Edgar and Himmelblau, 1989). In general, a multi-objective, multi-variate, constrained, and non-linear optimization problem can be formulated as shown in Problem 2.1 (Deb, 2009)[1]:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \boldsymbol{F}(\boldsymbol{u}) = [F_1(\boldsymbol{u}),\ F_2(\boldsymbol{u}),\ ...,\ F_m(\boldsymbol{u})] \\ \text{s.t.} \quad & \boldsymbol{C}_{eq}(\boldsymbol{u}) = 0 \\ & \boldsymbol{C}_{ineq}(\boldsymbol{u}) \leq 0 \\ & \boldsymbol{u}_{lb} \leq \boldsymbol{u} \leq \boldsymbol{u}_{ub},\ \boldsymbol{u} \in \mathbb{R}^{N_u} \end{aligned} \tag{2.1}$$

where $\boldsymbol{F}$ is the multi-objective function to be minimized, comprised of $m$ single objectives represented by $F_k$ $(1 \leq k \leq m)$ evaluated using a predictive model as schematically

[1] It should be noted that there are more ways in which to give a general representation of the mathematical formulation, e.g. with all the state and algebraic variables included as presented by Holmqvist (2013) and Knutson (2016), but Problem 2.1 is succinct and general enough for the purposes of the current thesis.

presented in Figure 2.1; $\boldsymbol{u}$ is the set of decision variables with lower bounds $\boldsymbol{u}_{lb}$ and upper bounds $\boldsymbol{u}_{ub}$, with $N_u$ decision variables and $\boldsymbol{u} \in \mathbb{R}^{N_u}$; this problem is furthermore *subject to* (s.t.) the sets of equality constraints, $\boldsymbol{C}_{eq}$, and inequality constraints, $\boldsymbol{C}_{ineq}$, that need to be satisfied. If no constraints at all have been formulated (beyond the predictive model), we refer to the problem as unconstrained – otherwise, the problem is constrained. Please note that the bold notation is used to denote vectors.

The general non-linear program (NLP) presented in Problem 2.1 can be classified according to whether it is differentiable, and whether it is convex (Biegler, 2010). Firstly, if the objective functions and the constraint equations have first and second order derivatives that are continuous, then the problem is twice continuously differentiable (Biegler, 2010); this means that the problem is smooth and without discontinuities, and we can therefore be sure that an optimum does exist. Secondly, if the objective functions are all convex and the feasible region is convex, then the NLP is convex (Deb, 2009). If the NLP is convex, we can be sure that every local minimum found is the global minimum; non-convex problems, on the other hand, may have multiple local minima, meaning that there may exist several different feasible solutions that minimize the objective in the immediate neighborhood around those solutions, which may not necessarily be global minima (Biegler, 2010). This multitude of feasible solutions will thus increase the difficulty of finding the global minimum, which represents the absolute best solution.

As can be seen in Problem 2.1, a general NLP has three core components: the objective function(s) ($\boldsymbol{F}$), a set of decision variables ($\boldsymbol{u}$), and a set of constraints that in turn can be equality ($\boldsymbol{C}_{eq}$) or inequality ($\boldsymbol{C}_{ineq}$) constraints (Biegler, 2010). When performing an optimization study, the objective function will be relying on a given predictive model of the process at hand, which takes some input and returns some output, and it will be systematically evaluated for a range of values of the decision variables until the optimum (minimum) has been reached. This general description is visualized in Figure 2.1, where initial values for the decision variables, $\boldsymbol{u}_{initial}$, are determined and supplied by the user and used as a starting point to move forward from during the optimization. The optimal solution is the one where the objective function has been minimized within some convergence tolerance or similar type of condition, while satisfying the constraints. These three components – the objective functions, the decision variables, and the constraints – therefore warrant special consideration during any optimization study, and we will therefore explore the concepts here.

### 2.1.1 Objective functions

In everyday life, an objective is largely any sort of goal we would like to accomplish. Thus, objectives are the measure or metric by which we as humans – individuals or groups – make decisions on how to improve a certain situation. A good example are so called Key Performance Indicators (KPIs), which are measurements that are used within different organizations to guide individual-level decision-making for the organization to accomplish a given target. When we want to perform a computer-aided optimization study, we need to translate our objective into mathematical terms, into an objective *function* that is to be minimized. In other words, we need to quantify our real-world objective in order to make comparisons and determine what inputs create the best solution. One perspective on decision-making is that it is an exercise in optimiza-

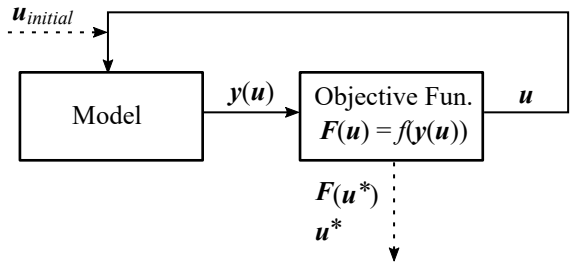



**Figure 2.1** A schematic of a general evaluation of an objective function in an unconstrained problem. The decision variable, $\boldsymbol{u}$, will be given an initial value ($\boldsymbol{u}_{initial}$) by the user and used to initially evaluate the model. The objective function in turn takes the model output for the specific value of $\boldsymbol{u}$, here called $\boldsymbol{y}(\boldsymbol{u})$, as input. The evaluation will continue to loop, and during each cycle the decision variable is adjusted/updated; this continues until an optimal (minimal) value for $\boldsymbol{F}(\boldsymbol{u})$ has been reached, with $\boldsymbol{u}^*$ representing the optimal solution.

tion, and since we as humans make plenty of decisions every single day, we are quite used to formulating objective functions - although perhaps we do it implicitly, in our subconsciousness.

In process engineering practice and research, there is a need to closely consider the objective since it essentially determines the target for the optimization (Knutson, 2016). Common approaches for formulating the objective function include utilizing economic criteria or performance criteria. Economic criteria classically consists of considerations of profits and costs concerning the decisions that are made (Edgar and Himmelblau, 1989), and can be formulated by using for example different methods such as the Net-Present Value (NPV). However, it is important to consider what method to use, as different methods will impact the optimal solution and make the optimization lean towards either short-term or long-term optima (or a compromise) (Novak Pintarič and Kravanja, 2015). Irrespectively, all these methods require the use of basic concepts such as prices, capital costs, and operating costs that Dimian et al. (2014) covered thoroughly. For instance, Díaz-Trujillo and Nápoles-Rivera (2019) formulated the economic part of their objective function so as to maximize total profit. The total profit they proposed to include revenues generated by sales of biogas and bio-based fertilizer whilst taking into account the cost of raw material, transportation, and processing. On the other hand, studies may also focus on costs only. A straightforward example of an economic approach focusing on costs only is presented by Diab and Gerogiorgis (2018), who minimized the total cost of a continuous manufacturing process of an anticoagulant pharmaceutical ingredient. The authors formulated the total cost as the sum of capital expenditure and operating expenditure, using the NPV method for discounting the latter. Another study focusing on costs is presented by Leonzio et al. (2020), where the authors aimed to minimize the total costs of carbon capture utilization and storage in the UK, including costs for capture and compression, transportation, storage, as well as for production of different products. In a similar vein from a cost perspective, in Paper IX, we aimed to minimize the total cost of retrofitting blue-green systems to different areas of Malmö. We formulated the total cost as the cost of implementation and the cost for damages due to flooded manholes across the city over the lifespan of

the blue-green systems (therefore discounted by NPV); the latter was estimated using a real case of extreme flooding from 2014. Assuming that investing in blue-green systems would decrease the risk of flooding, the idea was to minimize the combination of these conflicting goals by providing an economic perspective on – essentially – the costs of investing and the costs of not investing.

When formulating performance-focused objective functions, no extra step is taken to convert the performance into economic value. Instead, we are primarily concerned with some measure of performance for the system, which could be a wide range of different considerations, all depending on the system at hand. For instance, in addition to formulating an economic objective, Díaz-Trujillo and Nápoles-Rivera (2019) also formulated an objective concerned with the environmental performance of their system, as they aimed to maximize the total saving of greenhouse gas emissions. They formulated this objective as the avoided emissions by utilizing biomass but also took into account the emissions associated with transport, processing/purification as well as end use of products. In Paper VI, we formulated a scalarized multi-objective consisting of the productivity, yield, and resin utilization during the load phase of a capture-stage in a chromatographic purification setup, whereas in Paper IV, the performance was defined by the variance of the oscillating product stream. It could also be noted that the economic criteria used in Paper IX was based on the performance of the blue-green system in terms of their impact on the number of flooded manholes, which serves as an example of economic criteria being based on performance criteria.

### 2.1.2 Decision variables

To accomplish the objectives we define for ourselves, we need to make a decision (or several). During optimization studies, decision *variables* are any number of variables identified to be the most appropriate to tune in order to improve the objective function. We systematically perturb the decision variables in order to evaluate and minimize the objective function. The most fundamental demands on the decision variables can therefore be summarized as that they need to be something that we can manipulate, and they need to have an impact on the result of an objective function evaluation. Given these demands, the selection of the decision variable will thus depend on what the aim is and what the possibilities are (both in terms of reflecting the real life situation as well as in terms of what is possible to do with the model) during a given study. In a study on blue-green systems, Zhou et al. (2019) used the reduced impervious fraction of an urban area together with the increased fraction of pipe diameter as their decision variables for minimizing the cost of combining pipe networks with infiltration based sustainable urban drainage systems; comparing this study with the study performed in Paper IX, we used only the reduced impervious fraction (framed as the implementation extent of blue-green systems) of 30 different subcatchments in Malmö since modifying the piping network was out of our scope. Two other examples to compare are those of Papers III and IV. In the former study, the steam consumption of a polyalcohol evaporator system and the steam consumption of a downstream methanol distillation column were used as constant decision variables to minimize the cost induced by the steam consumption in these systems. In the latter study, a time-variant steam consumption was applied to minimize the oscillations the evaporator system was subject to. As can be seen, the decision variable(s) must be selected to fit the purpose and possibilities of the study.

Beyond this "Rule of Fit", selecting decision variables is often made difficult by the complexity that engineers face (both in terms of the models but also in the real world), but it is an important endeavor, nonetheless. However, the selection is often left solely to the intuitions of engineers (De Godoy and Garcia, 2017), whilst a systematic approach may be warranted to ensure feasibility and good control. For the purpose of proper selection, much can be learned from the subject of control structure selection, which focuses on the selection of manipulated variables, controlled variables, and their interconnections (Yelchuru and Skogestad, 2012). This subject has over the years gained a great amount of interest from both academia and industry, and many methods have been developed to facilitate decision-making and selection. In an extensive review, De Godoy and Garcia (2017) divide these methods into three categories:

(i) the process-oriented approaches that are easily understood and implemented, but may lead to suboptimal solutions;

(ii) the mathematical and optimization-based approaches that are difficult to formulate and perform but rigorous and lead to reliable solutions; and

(iii) the hybrids that attempt to mix the best of the two former worlds by utilizing the process knowledge of engineers together with systematic mathematical programming.

A prominent example of a hybrid method is self-optimizing control, which is used for the selection of controlled variables (Skogestad et al., 1999) and has garnered a lot of attention over the years (Jäschke et al., 2017). Other examples include the mathematically based Single-Input Effectiveness proposed by Cao and Rossiter (1997) (in recent years applied to input-output pairing by Jolevski et al. (2017)), the rigorous branch-and-bound method of Kariwala and Cao (2010), and the integrated selection and controller tuning proposed by De Godoy and Garcia (2017). Continuing within the field of computer-aided optimization specifically, there have been efforts towards systematically selecting decision variables via different *mathematically* based methods. For instance, Brady and Yellig (2005) developed a matrix-based simulation data mining method to rank decision variables in complex models from a single simulation run. Zhao et al. (2016) used digraphs to select decision variables that would require the least amount of simultaneous equations in equation-oriented models. Another method was used by Asadollahi and Naevdal (2010), who studied large-scale production optimization problems and employed multiscale estimation to dynamically modify the number of variables during optimization runs by grouping the variables. Finally, a more *process-oriented* method was applied in Paper I, where a subset selection algorithm, first proposed by Cintron-Arias et al. (2009) and applied to parameter estimation procedures by Andersson et al. (2014), was used for the *in-silico* selection of decision variables. Although based on an algorithm to score the different potential combinations of decision variables that could be used for optimization, it is deemed as more process-oriented since the method, as applied, requires a process model and a great deal of insight into both the process and the model.

### 2.1.3 Constraints

In everyday life, we are sometimes restricted in our possibilities when trying to accomplish our objectives. In optimization, restrictions of this kind are called *constraints*. Generally, we have *equality* constraints and *inequality* constraints, and these can be seen as definitions of extra relationships (in addition to the objective function) between the decision variables, $\boldsymbol{u}$, and the model output, $\boldsymbol{y}$, as shown in Figure 2.1 (Dutta, 2016). These constraints are evaluated alongside the objective function during optimization, as schematically presented in Figure 2.2, and the constraints must be satisfied, i.e. $\boldsymbol{C}_{eq}(\boldsymbol{u}) = 0$ and $\boldsymbol{C}_{ineq}(\boldsymbol{u}) \leq 0$, for the optimal solution to be considered feasible. Here it should be noted that one view on the process model, which needs to converge, is that it can be considered a set of constraints creating a feasible region for the solution (Biegler, 2010). However, unconstrained optimization does not refer to a lack of a predictive model but instead indicates that the objective function is the only condition or relationship between the decision variables and the model output that must be considered.

Constrained optimization problems can have constraints on the model, and/or lower and upper bounds on the decision variables (also known as box constraints). Constraints can be formulated to make the solution satisfy any criterion, which goes for both the model itself and its output, the real-life counterpart of the model, as well as the decision variables. For instance, in addition to considering the model as a set of constraint equations, constraints can be formulated for safety conditions (e.g. temperatures and pressures cannot exceed a maximum value), for operational/economical feasibility (e.g. yield cannot be below a minimum value), or for physical correctness (e.g. flow rates cannot exceed system capacities). For instance, in Larsson et al. (2012), the authors formulate a range of different constraints for optimizing economical grade changes in a polyethylene plant. They use box constraints to reflect real-life conditions caused by minimum cooling duties and maximum pump capacities, as well as the rate of change of the manipulated variables for equipment limitations and safety. Furthermore, they

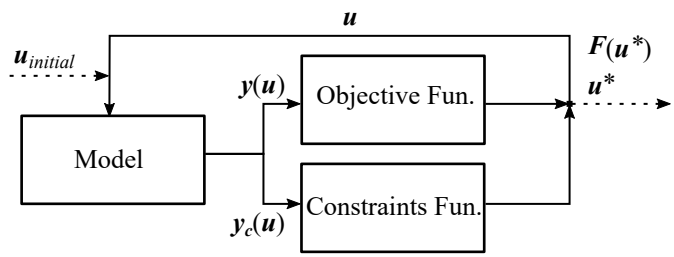



**Figure 2.2** An abstract schematic of a general evaluation of an objective function in a constrained problem where the model itself is *not* considered as a constraint. The decision variable, $\boldsymbol{u}$, will be given an initial value ($\boldsymbol{u}_{initial}$) by the user and used to initially evaluate the model. The objective function in turn takes the model output for the specific value of $\boldsymbol{u}$, here called $\boldsymbol{y}(\boldsymbol{u})$, as input. Model output will also be used to evaluate a set of non-linear constraints, and the model output relevant for the constraint is here called $\boldsymbol{y}_c(\boldsymbol{u})$; however, sometimes the constraints can be evaluated without the use of a model. Furthermore, the constraints need to be satisfied for convergence. These evaluations will be iterated until an optimal (minimal) value, $\boldsymbol{F}(\boldsymbol{u}^*)$, has been reached within the feasible region.

set constraints on e.g. reactor pressure and reactant concentration in order to guarantee safe operation of the plant. Another interesting example is that of Negrellos-Ortiz et al. (2018), who performed a model-based optimization study of product transitions in a cryogenic air separation unit. They selected an algorithm that could not handle constraints other than box constraints, and therefore used only such constraints. However, they state that they formulated their objective function so that it not only considered the target transition objectives, but also the dynamic manipulations to the zero-order hold control actions. This means that instead of setting hard constraints on the control actions, the authors recast their constraints as part of the objective to be minimized. A similar strategy but with a different reasoning was used in both Papers VI and IV, where we used the control actions as part of the objective in order to avoid the extreme behavior of bang-bang/on-off optimization and to ensure stability. In addition to this, in Paper VI, constraints were formulated on operating parameters such as loading time and flow rates, as well as for model convergence. In Paper IV, constraints were formulated in comparison to the nominal case. For instance, box constraints were used to limit the decision variables so that they were held between half and double that of their nominal values, and inequality constraints were formulated on the product stream to ensure the process did not perform worse than the nominal case.

## 2.2 Optimization approaches

Whilst all problems have their own degree of uniqueness, a general six-step (almost classic) method can be used to solve virtually any problem in process engineering. This method starts by analyzing the process itself, followed by formulating the optimization problem. Once formulated, there is a wide range of different approaches towards solving the optimization problem. For instance, if a problem is formulated as a multi-objective optimization problem, the approach will be different than if a simpler single objective is to be minimized. In a similar fashion, unconstrained and constrained problems may require different methods in order to be solved, and different algorithms may be needed in order to take the constraints into consideration. There are also alternate ways of approaching problems when it comes to dealing with any dynamic nature of the problem, as well as when it comes to making use of gradient information or not. However, regardless of the approach, all algorithms iterate, with more or less finesse to the method to produce the next candidate solution – and the optimal solution. This section is intended to introduce and discuss the aforementioned procedure as well as the approaches that are most relevant to the current thesis.

### 2.2.1 General six-step method for solving optimization problems

Edgar and Himmelblau (1989) presented a general procedure for solving optimization problems in six steps, which can be considered a classic[2]. The steps are summarized in Figure 2.3. The first three steps are primarily concerned with the mathematical definition of the problem, and the last three steps focus on finding a solution.

[2]This six-step method was kept for the second edition of the book published in 2001, which is considered to be a standard reference work by Biegler and Grossmann (2004).

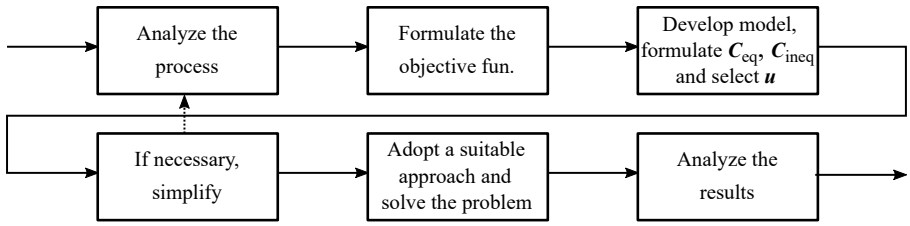



**Figure 2.3** A summary of the six step method by Edgar and Himmelblau (1989), who proposed that we need to start by analyzing the process at hand, followed by formulating the optimization problem along with the model, before solving the problem and analyzing the results.

We start an optimization study by analyzing the process itself so that variables and characteristics of interest can be identified and defined, and list them (Edgar and Himmelblau, 1989). The analyses and preparations also help us gain insight into the process and will serve us well down the line, since knowing the important factors of the process will be helpful when preparing a process model, formulating objective functions and constraints, and selecting decision variables. Conversely, it is extremely difficult to describe a process mathematically if you know nothing at all about it, so we basically *must* start here.

Continuing, we need to decide what we want to accomplish and formulate a mathematical version thereof in terms of the listed variables from the first step (Edgar and Himmelblau, 1989). This was covered with provided examples in section 2.1.1.

Following the formulation of the objective function(s), the constraints have to be formulated (section 2.1.3), and the decision variables must be selected (section 2.1.2); a valid predictive model of the process also has to be developed (Edgar and Himmelblau, 1989). Developing a model can be performed by making use of first principles, empirical relationships, implicit concepts, and external restrictions. In addition to creating custom-made models from scratch in a programming language (e.g., Python), there are many tools available to help with modeling (and simulation). Examples of such tools include flowsheeting environments for production processes (e.g., the Aspen Plus and gPROMS environments), hydrodynamic modeling and simulation tools for urban drainage (e.g., MIKE Urban and SWMM), and machine learning technologies that require large amounts of data (e.g., scikit-learn and TensorFlow). Regardless of the chosen tool, a systematic approach towards the development of a new model that incorporates knowledge about the process (obtained from the first step) is required (Rasmuson et al., 2014).

If the optimization problem and/or the model that we formulated in the first three steps is too large in scope or too complicated, there may be a need to simplify, in which case we should revisit steps 1-3 and either break the problem into manageable parts or simplify the problem as a whole (Edgar and Himmelblau, 1989). This can be done by splitting an model into smaller parts, or by reducing the dimensionality (and thus, to some extent, the difficulty of finding the optimum) of the problem by reducing the number of decision variables. We used these strategies for Papers III and IX, respectively. In Paper III, the full dynamic model of a two-stage evaporator and a methanol column would not converge due to stiffness that was presumably caused by differences in the

dynamics of the used unit-operation models, which led to us separating the two different systems into two different modules that were co-simulated; this is more thoroughly described in Paper V. In Paper IX, the system originally contained a huge number of small subcatchments, and the implementation ratios of blue-green systems to impervious areas in each subcatchment were used as decision variables. This number was reduced by creating larger subcatchments, leading to a smaller number of decision variables (but still quite large, with 30 subcatchments).

A suitable approach to solve the problem must then be adopted, with the goal of ensuring a good fit between your problem and your selected algorithm. A range of approaches are considered in the remainder of this section, and may require consideration of whether the problem is single- or multi-objective, whether it is constrained, of a static or dynamic nature, and/or whether we can, need, or wish to make use of gradient information.

Finally, when the problem has been solved, we need to analyze the results and ensure that the solution is sane and useful, before moving back to the real world for validation and implementation.

### 2.2.2 Single-objective & multi-objective optimization

A single-objective optimization problem is fairly easy to solve (Dutta, 2016), primarily as there will only be one objective to consider without any compromises necessary between different competing goals. An abstract schematic of the straightforward sequential procedure and the flow of information for single-objective optimization is presented in Figure 2.4. Essentially, the user provides the optimizing algorithm with an initial set of values for the decision variables, $\boldsymbol{u}_{initial}$, which the algorithm uses as a starting point in order to iterate its way towards the optimal solution. Whilst there may still be issues to consider regarding objective function convexity, local and global optima, and potential discontinuities, a well-posed single-objective optimization problem yields only one meaningful direction for the selected optimization algorithm: downwards, towards a – hopefully, *the* – minimum. For a constrained single-objective NLP, we thus define a feasible point in the decision space ($\boldsymbol{u}^*$) to be a *global* minimizer if no $\boldsymbol{u}$ exists such that:

$$F(\boldsymbol{u}) \leq F(\boldsymbol{u}^*),\ u \in R^{N_u}$$

In other words, if the function value $F(\boldsymbol{u}^*)$ is less than $F(\boldsymbol{u})$ for all $\boldsymbol{u}$ in the full feasible region as defined by the optimization problem, $\boldsymbol{u}^*$ is a global minimizer. We have thus found our global optimum. We further define $\boldsymbol{u}^*$ as a *local* minimizer if $F(\boldsymbol{u}^*) \leq F(\boldsymbol{u})$ for all $\boldsymbol{u}$ in the vicinity of $\boldsymbol{u}^*$, which means that several local optima may exist, but only one global optimum exists in the feasible region depending on the convexity of

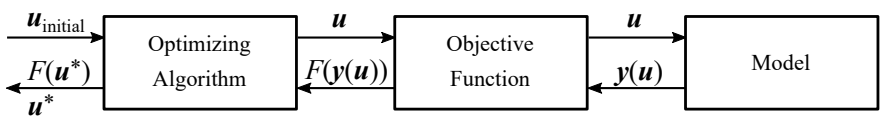



**Figure 2.4** A schematic of a single-objective, multi-variate optimization approach.

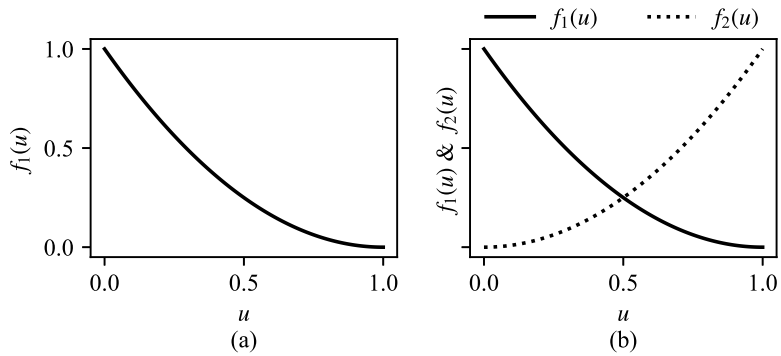



**Figure 2.5** An illustrative example of (a) a single objective and (b) a multi objective problem and the compromises that may be necessary for the latter. For instance, $f_1(u)$ can be imagined to represent price and $f_2(u)$ can be imagined as *bad* quality - both of which we would like to minimize, and thus we have to compromise and make a decision regarding which is most important.

the NLP, as discussed in section 2.1. A single-objective optimization is exemplified in Figure 2.5a, where we can consider $f_1(u)$ to represent the price of a product. If we are interested in the cheapest product possible, without regards to anything else, we should choose $u = 1$. Whilst formulating an economical objective function such as this example does create a single objective function, it may not always be possible to fit a price tag to everything. Furthermore, practical problems from the physical world formulated on a performance form are often ill-cast as true single-objective problems since there are myriad of compromises that need to be made. These kind of compromises may also require information that is not readily available to the practitioner and only accessible by decision-makers in an organizational hierarchy, and the problem structure thus needs to be adaptable to such information. As such, most of the problems we encounter are more aptly formulated as multi-objective problems.

However, a multi-objective optimization problem can be rather complex to approach, because the existence of multiple objectives mean that there is more than one direction in which to push. For instance, expanding the example shown in Figure 2.5a into the multi-objective problem in 2.5b, we may still want a low price ($f_1(u)$), whilst also avoiding *bad* quality, represented by $f_2(u)$. Choosing the low-cost product at $u = 1$ yields a product with a high measure of bad quality, which is something we also want to minimize, but choosing a high quality product at $u = 0$ forces us to pay more. Even for this simple example, the problem of finding an optimal solution is more difficult compared to the single-objective example. In general, for multi-objective optimization, two main concerns should be considered. Firstly, and perhaps most importantly, we need to define what optimality is in the presence of several objectives, since it is not as straightforward as in the single-objective case. Secondly, the objectives may be conflicting or incommensurable (in which case improving one is impossible without worsening the other), whilst we need to express our objectives in units that mix well, i.e., using the same unit or making the objective dimensionless (Dutta, 2016). Within this, how to weigh the different objectives against each other is also important to consider.

#### 2.2.2.1 Optimality of multi-objective problems

One of the most common and practical ways to define an optimal point in multi-objective optimization is through the concept of Pareto optimality (Marler and Arora, 2004). By this definition, a feasible point in the decision space ($\boldsymbol{u}^*$) is Pareto optimal if, and only if, there exists no other feasible point ($\boldsymbol{u}$) such that (Marler and Arora, 2004):

i. $\boldsymbol{F}(\boldsymbol{u}) \leq \boldsymbol{F}(\boldsymbol{u}^*)$, and

ii. $F_i(\boldsymbol{u}) \leq F_i(\boldsymbol{u}^*)$ for at least one of the objective functions $F_i \in \boldsymbol{F},\ i = [1,\ 2,\ ...,\ m]$.

This is a logical extension of the single-objective definition of optimal points. In other words, $\boldsymbol{u}^*$ is Pareto optimal if no $\boldsymbol{u}$ exists that improves any (at least one) objective function. *Weakly* Pareto optimal is similarly defined as $\boldsymbol{u}^*$ being weakly Pareto optimal if no point $\boldsymbol{u}$ exists that improves all (every single one) of the objective functions (Marler and Arora, 2004). Provided that $\boldsymbol{u}^*$ is Pareto optimal, the resulting objective vector $\boldsymbol{F}(\boldsymbol{u}^*)$ is also Pareto optimal (Dutta, 2016). In addition to Pareto optimality, the concepts of dominance (Deb, 2009; Marler and Arora, 2004) and efficiency (Marler and Arora, 2004) are quite prominent. These concepts are, for all practical purposes, indistinguishable from Pareto optimality in their respective definitions (Marler and Arora, 2004). Using these definitions, a Pareto-optimal set of solutions (i.e., all solutions that cannot be improved in any objective function) can then be constructed and visualized. When the objectives are graphically presented against each other, these sets commonly take the form of sweeping fronts. We call these *Pareto fronts* (Degerman, 2009), and an illustrative example of a Pareto front retrieved using two different methods is presented in Figure 2.6.

#### 2.2.2.2 Conflicting objectives in multi-objective problems

Multi-objective optimization can be simple if the objectives are non-conflicting, al-

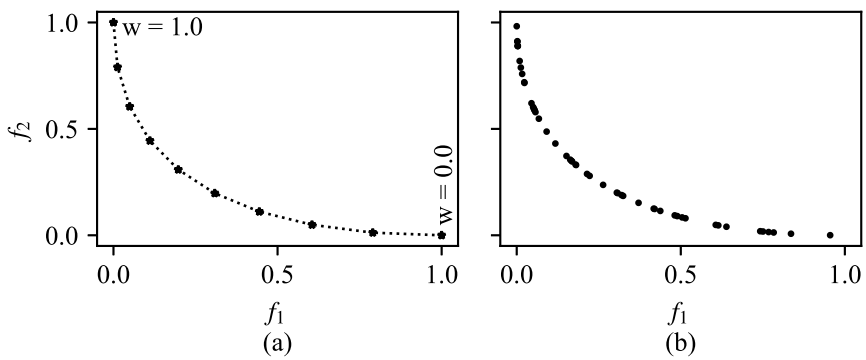



**Figure 2.6** An illustrative example of a front created by the Pareto set, here solved in two ways. In (a), the weighted sum method (Marler and Arora, 2004) is used. With an increase in the weight $w$, $f_1(u)$ becomes more important to consider and more important to minimize. For large $w$, $f_1(u)$ is therefore small. In (b), the NSGA-II multi-objective optimizer (Deb, 2009) was used to create the Pareto front.

though it may not always be obvious if they are. In those cases, the Pareto-optimal set contains only a single optimal solution (Deb, 2009). However, when we have conflicting or incommensurable objectives, full fronts of Pareto-optimal sets will exist, reflecting the need to make compromises and to decide how important the objectives are in relation to each other, which is an important endeavor. The quantification of the weights needs be done by the practitioner or decision-maker injecting or inserting their preferences into the process of optimization, and this can be done either *a priori* or *a posteriori* (Marler and Arora, 2004). These two approaches are schematically presented in Figure 2.7. The former approach means that we need to set our preferences, denoted $\boldsymbol{w}$ (short for *weights*), before the optimization begins and then re-run the optimization for each set of preferences that we are interested in examining. By doing so, we essentially recast the multi-objective problem into a single-objective formulation. Conversely, the latter approach means treating the multi-objective problem as is and producing a palette of Pareto-optimal solutions from which the practitioner can make a selection based on their preferences. This approach thus requires a full matrix of initial decision variable settings, $\boldsymbol{U}_{initial}$. There are also methods for which no articulation of preference is required, and these three categories of methods are all based on reformulating a multi-objective function into one that can be solved by standard single-objective optimization algorithms (Marler and Arora, 2004). Other approaches are also available that can solve multi-objective problems as they are, such as evolutionary algorithms

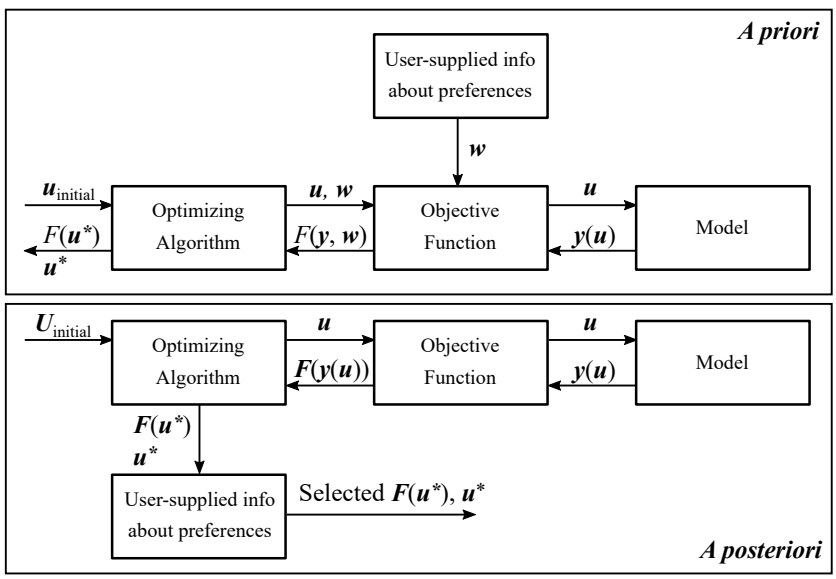



**Figure 2.7** A schematic of two multi-objective, multi-variate optimization approach: *a priori* (upper), meaning preferences are set before the optimization algorithm is run to find the Pareto-optimal solution (and repeated for each set of preferences), and *a posteriori* (lower), meaning that preferences are set after the algorithm has found all the Pareto-optimal solutions. Note that $y(u)$ has been omitted for brevity in the output from the objective function to the optimizing algorithm in the *a priori* approach.

with biologically inspired search heuristics based on stochastic elements (Deb, 2009; Emmerich and Deutz, 2018).

The examples in Figure 2.6 were created by solving the multi-objective optimization problems presented in Problem 2.2 using two methods: (a) using the weighted sum method to recast the problem as a single-objective formulation and applying the SciPy implementation of the SLSQP single-objective method (Kraft, 1988), and (b) as a true multi-objective optimization problem with a convenient Python implementation of the algorithm NSGA-II (Deb, 2009) created by Khan and Tiwari (2017). As seen by analyzing the two problems (or by revisiting figure 2.5b), the objectives are in conflict in the search space, since $f_1(u)$ is minimized for the same $u$ that $f_2(u)$ is maximized, i.e. at $u = 1$.

$$
\begin{array}{rll}
(a) & & \\
\min\limits_{u} & F(u,w) = wf_1(u) + (1-w)f_2(u) & \\
\text{where} & f_1(u) = (u-1)^2 & \\
& f_2(u) = u^2 & \\
\text{s.t.} & 0 \leq u \leq 1 & \\
& 0 \leq w \leq 1 &
\end{array}
\qquad
\begin{array}{rll}
(b) & & \\
\min\limits_{u} & \boldsymbol{F}(u) = [f_1(u), f_2(u)] & \\
\text{where} & f_1(u) = (u-1)^2 & \\
& f_2(u) = u^2 & \\
\text{s.t.} & 0 \leq u \leq 1 &
\end{array}
\tag{2.2}
$$

Using the weighted sum method to solve Problem 2.2a (results presented in Figure 2.6) we see that with an increase in the weight, $f_1(u)$ becomes a more important part of $F(u)$ and is therefore more important to minimize, which in this case means that $f_1(u = 1) = 0$. When the weight is low, then $f_2(u)$ is important to minimize. These kind of Pareto fronts are used in the optimization results of Papers VI and VII, and are useful to illustrate the compromises that are necessary between different objectives as well as to show the possible variety among the different solutions of a multi-objective optimization problem. It should be noted that all solutions found via the weighted sum method are Pareto optimal, but not all Pareto optimal points can be found (Deb, 2009). For instance, optimal points in a non-convex region of a Pareto front cannot be found (Emmerich and Deutz, 2018) and an even distribution in the weight does not guarantee an even spread on the Pareto front (Das and Dennis, 1997). When applying the NSGA-II algorithm to Problem 2.2b, we retrieve essentially the same front, as seen by comparing Figure 2.6a and b. Similar limitations concerning the distribution of the Pareto-optimal solutions exist, exemplified by the fact that $f_1(u) = 1$ is not found, which stems from the stochastic nature of the algorithm. In contrast to the weighted sum method, the evolutionary algorithms can find Pareto-optimal points even on concave regions of the Pareto front, but the stochastic approach lacks strict convergence criteria (Holmqvist, 2013).

Many more methods similar to those visited here exist for dealing with multi-objective optimization problems. The curious reader is directed towards Marler and Arora (2004), Emmerich and Deutz (2018), and Deb (2009) for thorough reviews.

### 2.2.3 Unconstrained & constrained optimization

There are many approaches and algorithms to find the optimal solution to our optimization problem. These algorithms can be divided into two categories based on their capability to deal with unconstrained and constrained problems (Biegler, 2010). An abstraction for solving unconstrained problems is presented in Figure 2.8. This is nigh-on identical to that for single-objective problems in Figure 2.4 (but here generalized for multi-objective problems by using $\boldsymbol{F}(y(u))$); since there are no extra relationships to consider, the procedure becomes very straightforward, and the optimizer can focus on minimizing the objective function.

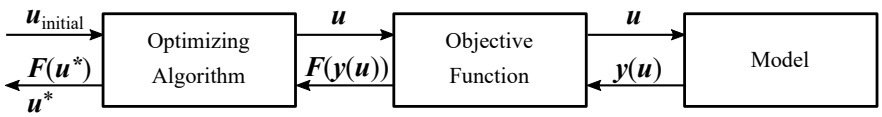



**Figure 2.8** A schematic of an unconstrained optimization approach.

In the case of constrained optimization, schematically presented in Figure 2.9, the optimizer needs to take these constraints into account and evaluate them to ensure they are satisfied. The constraints function is dotted to the model for this, since not all constraints in all problems necessarily require the model to be evaluated. The approach and choice of algorithm thus must take into account whether or not the problem has constraints since not all algorithms accept constraints. For instance, focusing on some of the algorithms available in the scientific computation package available for Python known as SciPy, the Nelder-Mead simplex method (Nelder and Mead, 1965) is intended for unconstrained problems only, and does not accept any constraints at all. The SLSQP algorithm (Kraft, 1988) accepts both constraints to create feasible and unfeasible regions in the decision space as well as decision variable bounds for limiting the search space. The COBYLA algorithm (Powell, 1994) accepts constraints but does not accept bounds on the decision variables, which thus have to be formulated as constraints to be evaluated after the objective function (an example of constraints that do not require the model to be evaluated). Here, it can be noted that constrained algorithms often work for unconstrained problems as well, as seen in the examples above where SLSQP was used

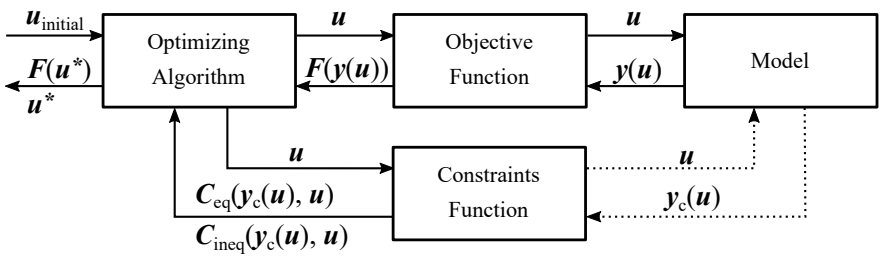



**Figure 2.9** A schematic of a constrained optimization approach; constraints may require some output from the model ($\boldsymbol{y}_c(\boldsymbol{u})$, but not necessarily so – for instance, box constraints need only the bounds for evaluation, and sometimes the model may be formulated entirely as the constraints in the optimization problem.

to solve both the unconstrained problem and the constrained problem (both bounded). Also note that similar variations in constraint-handling capabilities exist amongst evolutionary algorithms as well (Deb, 2009).

Truly unconstrained and/or unbounded problems may be rare in the real world (Deb, 2009), but problems can still be formulated as such for various purposes and thus need to be approached in a suitable manner. For instance, in Paper I, a number of unconstrained and unbounded problems with the same objective function were solved in order to investigate whether the decision variables selected by the Subset Selection Algorithm were effective. For this purpose, the Nelder-Mead algorithm was used. Another example includes that of the previously mentioned study presented by Negrellos-Ortiz et al. (2018), who cleverly formulated an unconstrained problem reflecting a restricted reality in order to work with the algorithm of their choosing, BOBYQA, which does not handle constraints beyond decision variable bounds (Powell, 2009). In Paper IX, the COBYLA algorithm was used to optimize the siting and sizing of blue-green systems in Malmö, using no constraints but only decision variable bounds. However, COBYLA does not accept decision variable bounds. Our approach was therefore to input the decision variable bounds as constraints for the algorithm to accept them. The practical difference is that the algorithm checks feasibility of solutions using the constraints instead of limiting the space that is searched beforehand, which unfortunately may lead to an unnecessarily high number of function evaluations.

Revisiting (the unconstrained) Problem 2.2a and reformulating it into a constrained problem gives us an illustration of how constraints may reduce the feasible region, and thus why it is important to take these into account. The constrained version of Problem 2.2 is formulated as Problem 2.3 below:

$$
\begin{aligned}
\min_{\mathbf{u}} \quad & F(u) = w \cdot f_1(u) + (1-w) \cdot f_2(u) \\
\text{where} \quad & f_1(u) = (u-1)^2 && f_2(u) = u^2 \\
\text{s.t.} \quad & f_1(u) - 0.5 \leq 0 && f_2(u) - 0.5 \leq 0 \\
& 0 \leq u \leq 1 \\
& 0 \leq w \leq 1
\end{aligned}
\tag{2.3}
$$

Inequality constraints have been introduced to the functions $f_1(u)$ and $f_2(u)$, stating that they have to be below 0.5 for a solution to be considered feasible. Approaching this as a constrained problem, the SLSQP algorithm can still be used, and the results of using the weighted sum method is presented in Figure 2.10. As can be seen in Figure 2.10, the area that was formerly fully feasible has now been considerably limited; it is now divided into feasible (white) and unfeasible (gray) regions, and solutions in the regions of $f_1(u) > 0.5$ or $f_2(u) > 0.5$ that were previously feasible are now considered not to be. It is, in other words, very important to choose a suitable approach and algorithm to the formulated problem, especially as constrained problems are very common within engineering. Good examples of studies with constrained problems from this thesis include Papers IV and VI. In the former paper, oscillations in an evaporator system were minimized with inequality constraints formulated to ensure the desired performance of the system with respect to the product quality. COBYLA was selected as the choice of optimizer to deal with the constraints. In the latter paper, we investigated flow rate

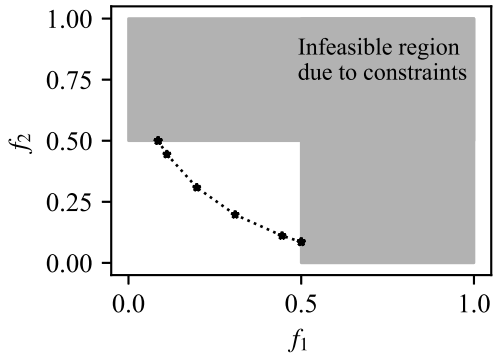



**Figure 2.10** The same problem as presented in Figure 2.6 but now constrained so that the two functions $f_1(u)$ and $f_2(u)$ have to be below 0.5 for a feasible solution.

trajectories to optimize the loading sequence of a chromatographic capture step, and while no constraints were set on the performance of the system (e.g., minimum yield or productivity), the model itself was treated as an equality constraint.

### 2.2.4 Derivative-free & gradient-based optimization

Constrained and unconstrained algorithms alike come in different varieties, and both can be further categorized into algorithms that are either derivative-free or gradient-based. These two classes of algorithms require different approaches to solve the problem at hand.

Derivative-free optimization (DFO) algorithms do not require information about the derivative of the problem in order to solve it; instead, they evaluate the objective function directly (Biegler, 2010). This property makes DFO a no-frills approach for the practicing engineer towards solving optimization problems, and its abstraction is – for our intents and purposes – identical to that for the unconstrained schematic presented in Figure 2.8 above (given an unconstrained problem). It also means that DFO algorithms are very well-suited for studies where the derivatives are difficult to formulate or retrieve, which is the case of many practical studies (Negrellos-Ortiz et al., 2018). For instance, first principle models with all the necessary derivative information may not always be fully available, either because the process is not well-understood or the model is data-driven; the available model may not have been originally tailored for optimization studies, and therefore may not have been prepared to express the required information; or modeling tools with embedded simulation procedures where access to the required information is restricted due to proprietary reasons. The evaluation of the objective function may at times also be computationally expensive or noisy, in which case DFO algorithms are more suitable than methods relying on numerically estimated derivatives (Rios and Sahinidis, 2013). According to Biegler (2010), the DFO approach is easy to implement since it requires relatively little insight (compared to using gradient-based methods) into the otherwise important minutiae of the optimization problem; in other

words, DFO algorithms allow the practitioner to perform value-adding studies by simple means. Furthermore, since DFO algorithms do not deal with derivatives neither do their termination criteria, leading to DFO algorithm favoring the search for a global instead of a local minimum (a risk of getting stuck in local minima still exists). However, no strict convergence criteria exist for these methods, and the performance does not scale well with the number of decision variables, leading to studies with more than a few dozen decision variables being rare (Biegler, 2010).

DFO algorithms can be further classified as *stochastic* or *deterministic* algorithms, depending on whether or not an algorithm takes random steps in its minimization method (Rios and Sahinidis, 2013). Stochastic examples include evolutionary algorithms such as NSGA-II, whereas an example of deterministic DFO algorithms is the Nelder-Mead simplex algorithm.

Stochastic algorithms rely to some degree on randomization in their approach towards solving the optimization problem. The hit-and-run algorithms reviewed by Rios and Sahinidis (2013) serve as good and relatively simple examples of this. Essentially, in the procedure of finding the optimum, the hit-and-run algorithms will randomly generate new candidates based on two random components; a *direction* and a *step* are generated from uniform distributions such that the step is feasible (within constraints). Another good example is that of the evolutionary (a.k.a. genetic) algorithms, which are bio-inspired and designed to mimic the evolution of a population. Randomization stems from crossover and mutation (Biegler and Grossmann, 2004), and Darwin's law of the survival of the fittest is applied. In other words, in each iteration, the solutions with the best objective function values have higher chances of surviving to the next iteration and to contribute towards (re)producing new points in their vicinity. Other common methods include the Simulated Annealing and Particle Swarm Optimization algorithms. Generally, stochastic algorithms are easy to use (Rios and Sahinidis, 2013) and are able to find the global optimum without getting stuck in local optima (Dutta, 2016). Stochastic algorithms can further be used for *a posteriori* multi-objective optimization. For instance, in the case of processes in urban environments, Eckart et al. (2018) used the BorgMOEA algorithm to find Pareto-optimal sets of reduced peak flows and runoff volumes from low-impact developments intended for managing urban stormwater. We also saw the use of NSGA-II for multi-objective optimization in the example presented in Figure 2.6b. However, it should be noted that stochastic algorithms naturally are not limited to multi-objective optimization problems but can be applied to single-objective optimization as well, with a good example being Javaloyes-Antón et al. (2013) and their minimization of the total annual cost of a complex distillation column using the Particle Swarm Optimization algorithm.

Deterministic algorithms, in contrast, do not use randomization elements in setting the next value for the decision variables and will instead rely on some heuristic to find the search direction for the next point in the vicinity of the current point. Two of Powell's algorithms, COBYLA and BOBYQA, serve as good examples for this purpose. The former works by making linear approximations of the objective and constraint functions whereas the latter makes a quadratic approximations. Both algorithms use a trust region subproblem for constraints, bounds, and variable changes. Deterministic DFO was suggested by Negrellos-Ortiz et al. (2016) to be an effective approach for solving optimization problems at the engineering level. Negrellos-Ortiz et al. (2016)

were also able to show the effectiveness of deterministic DFO for dynamic problems in particular, utilizing the BOBYQA algorithm (Powell, 2009) to optimize dynamic product transitions for two different cases, concerning a set of reactors (Negrellos-Ortiz et al., 2016) as well as an air separation unit (Negrellos-Ortiz et al., 2018), respectively. In both these studies, they employed models built in Aspen Plus Dynamics as black-box models, and were only concerned with the inputs and the outputs. Another example of DFO algorithms being applied for trajectory optimization is that of Paper IV, where we used the constrained DFO algorithm COBYLA to minimize the oscillations of the investigated evaporator system. The mentioned study is a good example of the difficulty of retrieving gradient information. In Aspen Plus Dynamics, the simulation procedures are namely built into the modeling software, and the software provides only limited opportunities for accessing the gradient information in a reliable manner, which is necessary for gradient-based techniques. As a final note on deterministic DFO, similarly to stochastic algorithms, there are global algorithms as well, with the interval analysis-based algorithm presented by Perez-Galvan and Bogle (2015) being a good example.

In comparison to derivative-free methods, gradient-based methods are significantly quicker to reach local optima than derivative-free methods, and these methods require the objective function to be smooth, i.e. at least three times continuously differentiable; here, *differentiable* means that all the partial derivatives of the objective function with respect to the decision variable(s) exist, and *continuously differentiable* means that all those partial derivatives are also continuous (Biegler, 2010). This property of the objective function and its derivatives is necessary because gradient-based algorithms utilize the first- and second-order partial derivatives of the objective function with respect to the decision variables in order to locate and classify the optimum. An abstraction of the gradient-based approach is presented in Figure 2.11.

The fundamental idea behind gradient-based algorithms is to determine where the optimum is located by using the gradient information. The Steepest Descent algorithm serves as a good example since its working principle is showcased in its name – the algorithm determines the direction of the steepest possible descent at the current point using gradient information, and takes as long a step as possible. For any problem with more than one decision variable, the gradient information is retrieved in a matrix form, which is called the Jacobian ($\boldsymbol{J}$ in Figure 2.11). The second-order derivatives can, for instance, be used to determine whether a found optimum is a minimum or a maximum.

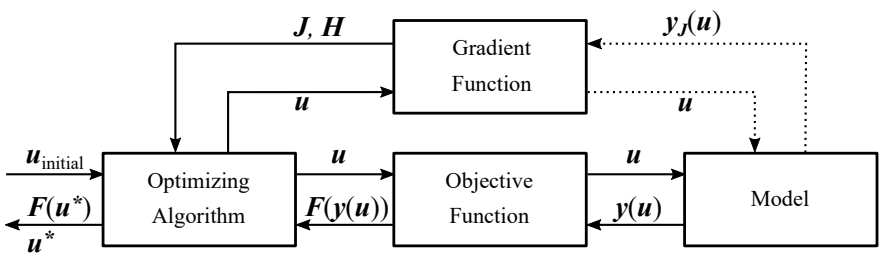



**Figure 2.11** A schematic of a gradient-based approach to optimization. The Jacobian, $J$, may be retrieved using information provided by the user to the algorithm, or the algorithm can approximate it numerically by finite difference perturbation of the model; the latter is indicated by the dotted line.

This information is also retrieved in a matrix form, called the Hessian (***H*** in Figure 2.11). Here it should be noted that while a multi-objective extension of the Steepest Descent method has been developed by Désidéri (2012), gradient-based algorithms are classically applied to single-objective problems (e.g. economic objectives, or multi-objectives solved via the weighted sum method), which therefore will be the focus here as well. Thus, for a single-objective problem with $N_u$ decision variables, the Jacobian and Hessian are generally defined according to Equation 2.4 (Andersson et al., 2014) and Equation 2.5 (Yang, 2017), respectively:

$$\boldsymbol{J} = \frac{\partial F}{\partial \boldsymbol{u}} = \begin{bmatrix} \frac{\partial F}{\partial u_1} & \cdots & \frac{\partial F}{\partial u_{N_u}} \end{bmatrix} \tag{2.4}$$

$$\boldsymbol{H} = \frac{\partial^2 F}{\partial u_i \partial u_j} = \begin{bmatrix} \frac{\partial^2 F}{\partial u_1^2} & \cdots & \frac{\partial^2 F}{\partial u_1 \partial u_{N_u}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial u_{N_u} \partial u_1} & \cdots & \frac{\partial^2 F}{\partial u_{N_u}^2} \end{bmatrix} \tag{2.5}$$

As alternatives to the Steepest Descent algorithm, some algorithms use the gradient to determine the *best* descent towards the optimum. What is meant by best can be explained by taking a valley as an example, where the minimum is located somewhere in the valley. The algorithms that proceed via the best route take the straightest feasible path towards the optimum, instead of finding and taking the steepest descent straight into the bottom of the valley and then moving on towards finding the optimum from there (still by steepest descent, which may not necessarily be so steep in the bottom of the valley). Examples of such algorithms include Newton's method, which is based on finding the zeros of the Jacobian given a continuously differentiable objective function by utilizing a Taylor expansion (Biegler, 2010; Yang, 2017). Further unconstrained examples include Newton-type methods such as Line Search method, and Trust Region method, with examples of constrained methods being the Sequential Quadratic Programming (e.g. SLSQP) and Interior Point (e.g. IPOPT) methods (Biegler, 2010). There are also methods that rely on Hessian update approximations, yielding a lower linear algebra cost compared to using the full Hessian; an example of a category of these methods is the Quasi-Newton methods, including the Broyden-Fletcher-Goldfarb-Shanno algorithm (Biegler, 2010).

Regardless of the choice of gradient-based algorithm, information about the gradient is required, and there are different ways of retrieving it. For instance, an analytical or symbolic Jacobian and Hessian can be supplied by the practitioner to the algorithm in the form of an auxiliary function, which may require a toolset designed for that purpose. This approach was adopted in Paper VI, where a toolchain including an automatic differentiation package was used to perform the optimization study; the purpose of the package was to efficiently compute the derivatives, which were supplied to IPOPT, which was the optimizer of choice in the study. There are also algorithms that, lacking a user-defined Jacobian function, are able to numerically approximate the Jacobian. Visualizing this potential relationship is the purpose of the dotted line between the model

and the gradient information in Figure 2.11, as the model can be – but does not need to be – numerically evaluated for the Jacobian. This approach was taken in Paper VII, as the SciPy implementation of the SLSQP algorithm was used for optimization, which uses numerical approximation by finite differences in a vicinity of the current point, i.e., by taking small steps to evaluate the effect on the objective function, in each iteration.

### 2.2.5 Static & dynamic optimization

Optimization approaches can further be divided into static and dynamic optimization. During static optimization, we are primarily concerned with steady-state behaviors and inputs, whereas dynamic optimization entails also considering the time-dependency or time-variance of variables in the models and processes.

Static optimization is an important application that utilizes constant decision variables and steady-state models. It has been widely used for design purposes within both production systems, exemplified by the optimal heat exchanger network design by Tanozzi et al. (2019), as well as urban environments, exemplified by the optimal sizing and siting of storage units in sustainable urban drainage system by Cunha et al. (2016). In addition to design, the approach can also be used to optimize operation, planning (Biegler, 2010), or some combination of these three applications (Chibeles-Martins et al., 2016). Solving a generic static problem such as Problem 2.1 has a procedure identical to that presented in Figure 2.9, assuming DFO is applied (otherwise the gradient information is necessary as well). The static approach was applied in Papers III, VII, and IX. In Paper III, constant flow rates of steam were used to optimize the cost induced by steam consumption in an evaporator system and a methanol column. In Paper VII, optimal integrated sequences of chromatographic column were designed using constant column volume and flow rate ratios. In Paper IX, the implementation extent of blue-green systems in 30 different subcatchments in Malmö were used to optimize net-present value of the investment to construct blue-green systems, utilizing a dynamic piping network flow model. Since dynamic models were used to capture the dynamic system behavior, the studies presented in Papers III and IX can perhaps be considered border-line cases between static and dynamic optimization.

Dynamic optimization utilizes time-variant decision variables and dynamic models. It is therefore a bit more convoluted than steady-state optimization, because – in addition to the concepts explored so far in the current chapter – we must consider how to deal with a variable decision variable (or multiple). We may also need to consider specific optimality conditions for optimal control problems as well as the handling of potential path constraints (Biegler, 2010). However, dynamic optimization is very powerful for controlling processes, and has for instance been applied for trajectory optimization in order to perform open-loop optimal control of chromatographic separation processes (Sellberg, 2018). Dynamic optimization can also be used for model-based predictive control (MPC or MBPC) as described by Maciejowski (2002), which is especially relevant for systems under regular disturbances (Santos et al., 2020), for start-up/shutdown scenarios (Sindareh-Esfahani et al., 2017), and for finding the optimal path during production change from set point A to set point B (Negrellos-Ortiz et al., 2016). However, using dynamic optimization for real-time forecasting of the optimal decision variable path sets a time limit on the required computations, which in turn sets high demands on how the problem is formulated and solved (Biegler, 2010).

In general, a dynamic optimization problem can be formulated similar to Problem 2.1, with the added dimension of time. There are two main ways in which to approach such a constrained and bounded dynamic problem: (i) by partial discretization, and (ii) by full discretization (Biegler and Grossmann, 2004).

In terms of partial discretization, one relatively simple way to approach a dynamic problem is by what Biegler and Grossmann (2004) call the *direct sequential* method because the optimization problem and model are solved in sequence. This approach comprises temporal discretization of *only* the decision variable (e.g., according to a zero-order hold (ZOH) strategy), whereas the model is kept as a whole and solved using an embedded differential-algebraic equation (DAE) set solver through single-shooting (e.g., a numerical integrator in the modeling and simulation environment of choice solves the model as an initial-value problem) (Biegler and Grossmann, 2004). This means to divide the decision variable, $\boldsymbol{u}$, into a set of $N_\tau$ number of time horizons, $\tau$, each of which can assume independent values (within set bounds). By doing so, the originally infinite-dimensional problem – infinite due to the potentially infinite amount of points in the continuous time between an initial and a final point in time ($t_0$ and $t_f$, respectively) – is recast as a finite-dimensional problem with only $N_\tau$ decision variables distributed over the optimization time horizon. This discretization approach creates a relatively small optimization problem compared to full discretization (Cervantes and Biegler, 2008), and a generic decision variable time profile is visualized in Figure 2.12 to visualize what the ZOH discretization may result in.

A schematic of the optimization procedure is presented in Figure 2.13, where the discretized decision variable as a function of time ($t$) is used in the model. Discretization according the direct sequential approach can be practically performed in different ways, e.g., by using an *if* statement within the model itself, or by using a *for* loop in the objective function, to have the model simulated over each time horizon with the corresponding decision variable used. Thus, in each time horizon the discretized con-

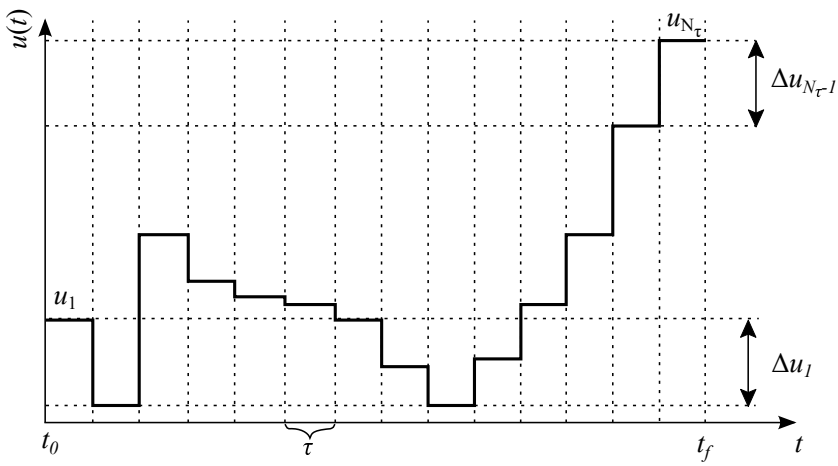



**Figure 2.12** A generic example of a control signal, $u(t)$, during dynamic optimization utilizing discretization of the decision variable into $N_\tau$ zero-order hold sections.

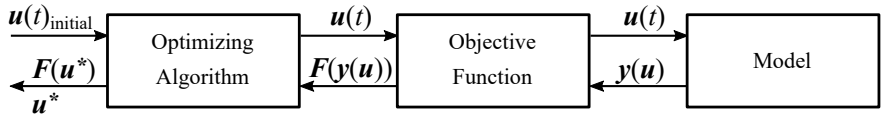



**Figure 2.13** A schematic of a direct sequential optimization approach to solve a dynamic optimization problem, where the decision variables are time-variant and have been temporally discretized to create a finite-dimensional optimization problem. This utilizes an embedded DAE solver to evaluate the model at each iteration. The time-dependency is omitted in the outputs for the sake of brevity.

trol signal can assume an individual value, thus representing one degree of freedom each for the optimizer. This means that the discretized control signal has $N_\tau$ degrees of freedom. One problem with this is that it could lead to difficulties in converging on an optimal solution. Furthermore, since each discretized horizon is independent from the previous and following, "bang-bang optimization" may be an issue, which can be mitigated by adding $\sum_{i=1}^{n} R\Delta u_i$ (where $R$ is a scaling parameter and $\Delta u_i$ is the difference between two neighboring ZOH signals, as visualized in Figure 2.12) to the objective, thereby punishing large differences between two consecutive ZOH signals. An example of this approach from the work presented in this thesis is Paper IV. This study utilized a toolchain presented in further detail in Paper V in order to employ the direct sequential approach with the ZOH strategy. Using this approach, an open-loop trajectory optimization with the purpose of minimizing process-disturbing oscillations was successfully performed by applying the DFO algorithm COBYLA. Alternatives to the ZOH strategy would be higher-order hold methods such as linear multistep methods as presented by Biegler (2010) as well as parsimonious input parametrization (Rodrigues and Bonvin, 2018), but these alternatives have not been further explored and applied in practice in the current thesis.

In terms of full discretization, not only is the decision variable discretized, but so are all the variables of the model (Biegler and Grossmann, 2004). Comparing to the direct sequential method, the full discretization approach is considered as an everything-at-once, or *simultaneous* approach (Biegler, 2010). The model is now formulated fully as constraints, which is visualized in the procedure schematic presented in Figure 2.14. This approach typically yields a very large AE problem that is not solved at each iteration, but only at the optimum (Biegler and Grossmann, 2004). There are two main methods to consider: (i) multiple-shooting, and (ii) collocation methods (Biegler and Grossmann, 2004). However, when applying multiple-shooting, the problem can be relaxed so that a sequential-like approach is applicable, making use of an embedded DAE solver to instead solve $N_\tau$ initial-value problems (i.e. one for each time horizon) (Biegler and Grossmann, 2004), which are coupled only by their initial and final states. This problem-solving structure thus requires continuity constraints in the form of equality constraints, meaning that the model state at the end of one horizon must be equal to the model state at the beginning of the following horizon (Cervantes and Biegler, 2008). Approaching the problem like this further means that there are possibilities for parallel computations, which is very useful for problems requiring many simulations (Andersson, 2014). Using collocation, the continuous time problem is instead recast

into an NLP by using polynomial representations to approximate the model state and the decision variable profiles (Biegler and Grossmann, 2004). This recasting can be done on a local scale, where the low-order polynomials are used to approximate these profiles in each collocation time horizon (note that there may be several collocation time horizons in one ZOH time horizon), or on a global scale, where a single higher-order polynomial is used (Magnusson, 2016). Thus, collocation methods create the largest NLP problems out of the three concepts explored here, and the collocation formulations are practically unsolvable via DFO due to the sheer size of the decision variable set. Specialized large-scale NLP algorithms, e.g. IPOPT, are required in conjunction with capitalizing on the structure of the collocation equations (Biegler, 2010). This approach was applied to solve the optimization problem presented in Paper VI, in which the optimization problem and the model was solved simultaneously by using a local collocation method to create an NLP, which in turn was solved using the IPOPT algorithm (Wächter and Biegler, 2006). Comparing this to the sequence presented in Figure 2.9, this simultaneous approach left no box for the model at all as in Figure 2.14, since the model was formulated entirely as constraints to be satisfied by the optimizing algorithm. Furthermore, specific equality constraints were implemented on every decision variable in each time step, or communication point, ensuring continuity. Thus, the objective function does not need the model to be evaluated, and the evaluation thereof relies solely on the decision variables. However, in said study, an initial-value problem solver was used to provide the optimizer with feasible initial conditions for the decision variables, as well as for model-based verification of the optimal solution.

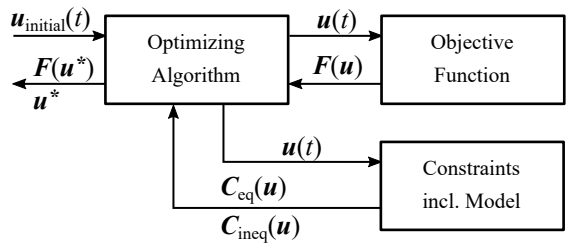



**Figure 2.14** A general schematic of the full discretization approach, where the model is cast as a set of equality constraints and is thus solved by the optimizing algorithm without the need for an embedded DAE/initial-value problem solver. The time-dependency is omitted in the outputs for the sake of brevity.

### 2.2.6 Summary

In this chapter, we visited the foundational concepts for an optimization problem, namely the objective function, the decision variables, and the constraints. We have furthermore explored the approaches that have the most relevance to the work presented in this thesis, including the general six-step method towards problem solving; single- and multi-objective optimization; unconstrained and constrained optimization; derivative-free and gradient-based optimization; and static and dynamic optimization.

To summarize, a comparison is offered between the different types of studies and their degrees of difficulty in terms of applying the necessary tools. For instance, a

static single-objective, unbounded, and unconstrained problem, solved by adopting a derivative-free and unconstrained (i.e. without recasting the problem) approach may well be the easiest to implement, due to the minimal gradient information and constraint formulations required to perform such a study. A good example of a study with a very similar approach can be found in Paper IX, with the only difference being that it used decision variable bounds. The study offered its own unique obstacles that required solving, for instance concerning decision variable reduction and selection – going from 1,000 subcatchments to 30 – as well as the coupling of the different software environments that were used, i.e. Python and the collection of software produced by DHI Sverige AB, available in the MIKE software suite. At the other end of the spectrum, with a dynamic multi-objective, bounded, and constrained problem, solved by adopting a gradient-based, fully discretized approach may be the most difficult, since a lot of technical footwork and insight into the minutiae of the problem is required. A good example of this is Paper VI, which required the use of a complex toolchain to formulate the model and to extract the gradients, to make solving the problem practically feasible.

# 3

# The interplay between optimization, modeling, and simulation

The formulations and approaches for computer-aided optimization described in the previous chapter all rely on the utilization of a mathematical process model that needs to be simulated in order to gain knowledge about how the process behaves for certain inputs. However, if no model exists, the practitioner has to create one; and if no simulation strategy exists, the practitioner has to implement one as well. This chapter is intended to provide an overview of general procedures for creating and solving models, i.e. modeling and simulation. These procedures will then be tied together, or integrated, with the general procedure for solving an optimization problem presented in Section 2.2.1, enabling a discussion on the impact and interplay of our choices during modeling, simulation, and optimization. The integrated procedure is then complemented by a purpose-centric framework, with the aim being that these tools can help and be useful to practitioners performing optimization studies in order to provide foundations for process improvements.

## 3.1 Modeling

A model used in computer-aided optimization is a mathematical representation of a process we are interested in studying that possesses predictive capabilities (i.e. the model can predict what the result (output) caused by specific conditions (input) will be). A conceptual example of a model reflecting a real process is presented in Figure 3.1. According to Hangos and Cameron (2001), a mathematical model can be created via a first principles approach. This way, engineering knowledge is used to describe what is happening in the process in order to establish a model structure and estimate model parameters, which yields a so called white-box model. The opposite, the black-box model, is instead built solely on process data, and it usually requires huge amounts of data to capture the behavior of a process into a model structure with parameters. A hybrid approach is the gray-box model, where we create a model structure using what we know

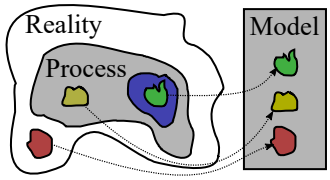



**Figure 3.1** Into a model go all identified essential process characteristics (green), often only a subset of all the essential characteristics (blue). Sometimes non-essential parts (yellow) of the process as well as parts that we thought were important but have actually been wrongly identified as important (red) are incorporated into the model.

about the process and its dominating phenomena and then use process data in order to determine the model parameters. Most process models are of a gray-box type (Hangos and Cameron, 2001). Combined into a model goes all the knowledge about the characteristics of the process that have been identified as essential (green), but most often not all the actual essential characteristics (blue). Sometimes also non-essential parts (yellow) of the process are used in the model, and sometimes even parts are incorporated that we thought were important but have actually been wrongly identified as important (red). This means that a model can never be a perfect representation of reality or a process; however, we can create models that are abstraction of the real process (Rasmuson et al., 2014), usable for our intents and purposes (Edgar and Himmelblau, 1989).

The orchestrated activities associated with model creation are referred to as modeling, and it is at least as much of an art as it is a science; how a process can (or should) be described mathematically depends on the context within which the model is developed, e.g. the problem formulation as well as how the problem is to be solved (Nilsson, 1993). Similarly to optimization, good practice requires the modeling to follow a systematic procedure, such as the seven step method presented by Hangos and Cameron (2001). This seven step procedure is schematically depicted in Figure 3.2, and whilst modeling procedures in literature are rarely described according to this exact structure, a good example of the procedure being explicitly applied is described by Degerman (2009), who applied it for the successful design of robust preparative chromatography. A similar seven-step procedure has also been presented by Rasmuson et al. (2014).

The seven step procedure by Hangos and Cameron (2001) starts by having the practitioner define the problem. In general, this step consists of defining the degree of detail relevant to the modeling goal, which needs to fit with the optimization problem formulation, meaning that our optimization problem feeds into the modeling procedure. Setting goals can be done by considering what kind of questions that need answers, which the model will help provide (Rasmuson et al., 2014). Furthermore, defining the degree of detail includes specifying e.g. inputs and outputs, model hierarchies, range and accuracy, as well as temporal characteristics of the model (Hangos and Cameron, 2001). Temporal characteristics refers to whether the model is to be static or dynamic, and this is particularly important since problems and questions that depend on the dynamics of a given process are difficult to solve and answer using static models (Pedersen and Wik, 2020).

The second step is to identify controlling factors, which entails investigating the

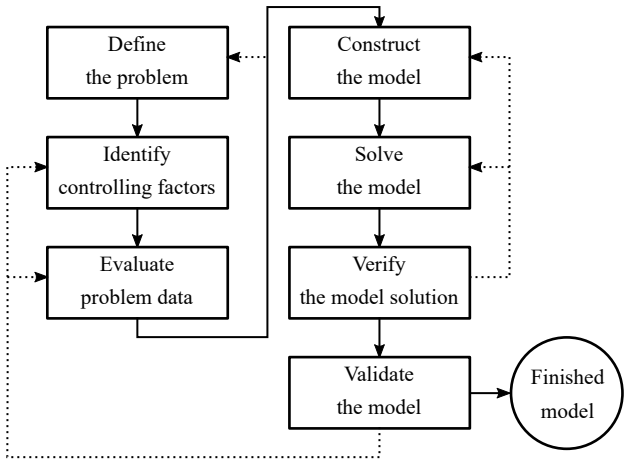



**Figure 3.2** The seven-step modeling procedure presented by Hangos and Cameron (2001). It all begins with a problem definition (top left), which we are familiar with from Chapter 2.1.

physical properties and phenomena of the process to be described. The most important factors to include are the governing, dominating factors; an example could be the temperature of a reactor in which a highly temperature-dependent reaction takes place. Revisiting Figure 3.1, this means that implementing the non-essential factors (yellow) may lead to unnecessarily complex models that are prohibitively expensive to simulate, which therefore should be avoided. A good example of this is that of Paper III, where formaldehyde was included in one of the models but not the other, due to its real-life dominance in the evaporator system and insignificant contribution to the physical behavior of the methanol column. However, the selection of the most important characteristics to incorporate should not be done arbitrarily, but should follow logically from the problem definition (Hangos and Cameron, 2001), and it should be based on gathered data and experience about the modeled process (Rasmuson et al., 2014).

During the third step, the practitioner needs to evaluate the process data relevant for the problem at hand. This evaluation should be done in conjunction with the parameter values and uncertainties/precision, and if the conclusion is that no suitable parameter values can be found in either literature or by analyzing the available data, then decisions made in steps 1 and 2 may require reconsideration (Hangos and Cameron, 2001).

For the fourth step, the model is to be translated into a set of equations, and each important characteristic needs to be formulated into an appropriate mathematical entity, such as a variable or a function (Rasmuson et al., 2014). The equation set may consist of ordinary differential equations (ODEs), partial differential equations (PDEs), and/or algebraic equations (AEs) (Hangos and Cameron, 2001), often collected into a PDAE system. Physical feasibility conditions (e.g. concentrations are always positive), boundary conditions (i.e. how the process behaves at its edge, e.g. no spatial change in concentration at the end of a chromatographic column), and initial values (e.g. starting concentrations) are also important to consider and define (Rasmuson et al., 2014). This step can, according to Hangos and Cameron (2001), be successfully completed by

first identifying the (sub)system boundaries where mass and energy is likely to accumulate, followed by defining the characterizing variables (e.g. mass flows). The mass and heat balances can then be formulated, followed by any mass and heat transfer rate specifications. Property relations (e.g. AEs describing thermodynamics or rain-water retention) and balance volume relations (e.g. vapor-liquid equilibrium equations) are also needed to be specified, before equipment and control conditions (e.g. pressures and temperatures) and modeling assumptions can be specified. However, the assumptions are usually made throughout the process of formulating the model, and should be noted explicitly since they will in a sense delimit the range in which the model is valid.

The fifth step consists of solving the model; in other words, the model is to be simulated. For this purpose, a simulation strategy must be chosen and implemented, and this generally requires consideration of all the decisions made in the first four steps (Hangos and Cameron, 2001). We will elaborate on this step in Section 3.2 below.

Continuing, the solution from the fifth step is to be verified in the sixth step, which entails determining that the simulation behaves as intended; if not, the code needs to be reviewed to see that it has been implemented correctly (Hangos and Cameron, 2001). This review may mean that it is necessary to revisit steps four and/or five. Determining whether the simulation works as intended can be done by comparing the solution with previously known results (Rasmuson et al., 2014). A good example of this from the work presented in this thesis is that of Paper VIII. During the development of the Python implementation of the model for simulating blue-green systems, the results were compared to the pre-existing Matlab-Excel implementation. By doing so, we could see that the Python implementation gave the exact same results, but was more useful since it was simulated in a fraction of the time of the old implementation. After verifying the model, Rasmuson et al. (2014) recommend parameter estimation (also called model calibration) to be carried out. This means to use real data in order to find values for the model parameters that yield the best fit between the model and the available data, which we did for the model in Paper VIII.

Finally, in the seventh step, the model is to be validated by comparing model output with real data – preferably not the same data as for the parameter estimation mentioned in the previous step (Rasmuson et al., 2014). This validation should be performed with the decisions from the first two steps in mind, including the modeling goal, and the validation method to choose depends on these decisions as well as the process and the model itself (Hangos and Cameron, 2001). Examples for how to validate the model include comparison with experiments or by comparing model behavior with process behavior for the same conditions (e.g. temperatures, input concentrations, or rainfall). Important tools include a sensitivity analysis of the input as well as parameters (Hangos and Cameron, 2001; Rasmuson et al., 2014), and important measures to consider include (but are not limited to) accuracy, precision, robustness, generality, and fruitfulness of conclusions from the model (Rasmuson et al., 2014).

Once having gone through these seven steps, it is easy to believe that we will have developed a perfect model – but we should expect iterations to be required (Hangos and Cameron, 2001). Changing the model structure may be required, or recalibration of the model may be necessary as new data becomes available. Having eventually finished a model, it is important to remember Figure 3.1 – no model is perfect, but some models are useful for the intents and purposes for which they were designed and created.

## 3.2 Simulation

The fifth step of the modeling procedure by Hangos and Cameron (2001) concerns solving the model, i.e. simulation, which also requires a systematic approach. For this purpose, Ramirez (1997) presented a five-step procedure for process simulation. This procedure is presented in Figure 3.3.

Out of these five steps, we recognize three steps from previous sections of the current thesis. The first step follows from the first steps of the optimization procedure (Section 2.2.1) and the modeling procedure (Section 3.1), respectively and combined, whereas the second step here is a summary of the second, third, and fourth steps from the modeling procedure. The fifth and final step in the simulation procedure is the same as the sixth step of the modeling procedure.

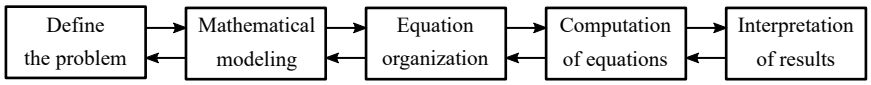



**Figure 3.3** The five-step simulation procedure by Ramirez (1997). Note the similarities with the previously presented optimization and modeling procedures.

The third and fourth steps of the five-step simulation procedure are therefore of special interest to this current section. In the third step, we must organize the equations into a solution strategy so that they can be efficiently and effectively solved in a manner that is congruous with our goals for the study. This means that decisions must be made regarding which variables to solve for in each equation, and the equations should be arranged so as to reflect the causality of the real process (Ramirez, 1997). The fourth step of simulation in the context of computer-aided optimization is then carried out with the help of a computer. This means that a numerical strategy or method is required to solve the DAE system consisting of ODEs, PDEs, and AEs.

Today, there exist tools for modeling that help engineers with these steps for simulation. Examples of such tools are the Aspen Plus suite by AspenTech and the MIKE suite by DHI, for process flowsheet simulation and urban drainage simulation, respectively. In the case of using such software, the engineer is able to rely on the software automatically organizing and solving the equations when prompted to do so, whilst the default options for simulation may need reviewing should convergence issues occur.

However, if a practitioner makes a model from scratch and wants to design a custom-made solution strategy to simulate it, then the decision tree approach presented in Figure 3.4 is a useful guide to ensure consideration of the most important elements[1]. In this decision tree, the first step is to identify whether the model is spatially distributed. This means that we have designed the model to be concerned with the spatial behavior of some property, e.g. the temperature gradient in a tank or concentration gradient in a chromatographic column, and thus includes of a set of PDEs. The alternative to this is a spatially lumped model, which is unconcerned with the spatial behavior, instead summarized as a single value (e.g. the integral measure of the temperature in a tank, i.e. the thermometer value); a lumped model is comprised

[1] This decision tree is part of the advanced course Process Simulation (KETN01) taught by Professor Bernt Nilsson to master students specializing in process design at the chemical and biotech engineering programs.

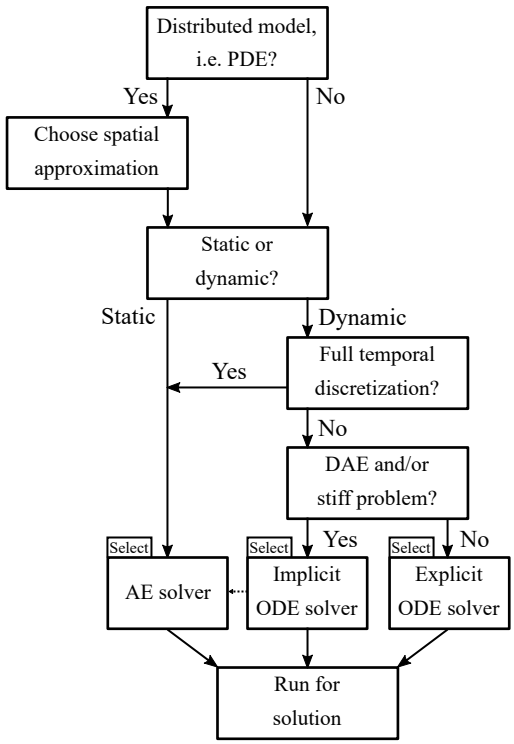



**Figure 3.4** A decision tree to use when creating a simulation strategy for solving a process model.

of a set of ODEs and AEs. If the model is distributed, then we first need to choose a spatial approximation. There are several well-proven alternatives, ranging from the finite difference methods to the method of lines (Kurtz et al., 1978), and the choice is not always obvious since many factors – including implementation complexity, accuracy and stability requirements, as well as computational expensiveness – need to be considered and therefore some experimentation may be required (Hangos and Cameron, 2001). Then we need to decide if we are to evaluate the model statically or dynamically, which is given by our optimization problem formulation and how we built the model. If a static simulation is chosen, we need to choose an AE solver such as the Newton-Raphson method to solve our problem (Ramirez, 1997). If a dynamic simulation is chosen, then we need to decide whether to perform a full temporal discretization or not. If a full discretization is desirable, then an AE solver can be used again. In the context of optimization, the AE solver can be replaced by formulating the model as a set of equality constraints, as discussed in Section 2.2.5; however, for pure simulation purposes, an AE solver is necessary. If a full temporal discretization is not desired, then we need to consider whether we have a DAE and/or a model containing vastly different dynamics, i.e. a stiff problem. If the model is stiff, then an implicit ODE solver (e.g. an implicit-explicit additive Runge-Kutta method (Meyer et al., 2018)) will be required, since these are able to solve a set of AEs for each

time step; otherwise, an explicit solver that requires no iteration for the integration, e.g. the explicit Euler method, will do well (Hangos and Cameron, 2001). Also note, that in order for any solver to perform the necessary computations, the initial values determined in the fourth step of the modeling procedure also need to be supplied to the solving algorithm. Finally, going back to the top of the decision tree, if instead the model is lumped we need to decide whether we need a static or dynamic solution to the problem, which again is given by our optimization problem and the way we built our model. From there the decisions to be made are the same as for the distributed model. Having gone through the steps from the five step procedure for process simulation by Ramirez (1997) supported by the decision tree, the simulation strategy is in place, and the model can subsequently be solved.

As a final note on simulation, there is an interesting special case of simulation where there are multiple models (that we therefore can call modules) describing, for instance, different physical domains or modeling environments (such as in the case of Bulian and Cercos-Pita (2018) as well as Mikkonen et al. (2017)). Moreover, models may also be divided into different modules in order to mitigate convergence issues of a large, complex model Lin et al. (2017), or to protect important know-how that may have been embedded into a model (Andersson, 2016). These modules still need to be simulated together, and this is referred to as co-simulation. For each of these modules, the modeling and simulation procedures have been worked through to establish all that is required for retrieving the necessary information from the modules, and preparing them for interchange of information. What remains is to couple the modules and/or their results together to make the most use of them as possible, which is a difficult task that has yielded some very interesting research over the years, with Andersson (2016) serving as a comprehensive example. In terms of this thesis, Papers III and IX utilized different co-simulation strategies to perform the optimization studies. The former paper optimized the resource consumption induced costs of an evaporator system and a methanol column with respect to the steam consumption. Models of these system were created and co-simulated using a strategy described in Paper V. The reason the models were divided into modules was to mitigate convergence issues that are believed to be caused by the stiffness of the combined model of the two systems. In the latter paper, two modules describing entirely different phenomena were used to optimize the siting and sizing of blue-green systems in Malmö, where the first module described the performance of the blue-green systems themselves, and the second module used the results of the first to predict the effects on the urban drainage network.

## 3.3 Mutual interplay

This brings us to how our choices of methods and tools during optimization, modeling, and simulation interplay and impact our study. The common theme in the three procedures for optimization, modeling, and simulation is that they are problem-centric, with each of the three beginning with a problem definition. This starting point is very common to problem-solving in general, since interpreting, understanding, and abstracting a problem is very important in bringing a higher probability of success in solving it (Salim and Diefes-Dux, 2012). The classic six step method of solving optimization

problems presented by Edgar and Himmelblau (1989) is well-proven. It contains a feedback loop to consider if the process, model, or problem is too complex, and in doing so, we can reformulate the problem – including the model and the simulation strategy – in order to solve it with reasonable effort. It is now a good idea to integrate this six step method with the seven step method for modeling, presented in Section 3.1, and the five step method for process simulation, presented in Section 3.2. An interpretation of how to integrate the procedures is presented in Figure 3.5, where the first step of the modeling procedure is aligned with the optimization problem formulation, and the first two as well as the last step of the simulation procedure are considered to be parts of the previous procedures as stated in Section 3.2.

Beginning by looking at the methods we select to use during an optimization study, the most salient part of this integrated procedure is that developing a model and a simulation strategy is done in the third step of the optimization procedure, while an optimization approach is to be adopted not until the fifth step of the optimization procedure. However, if this approach is not considered during modeling and simulation strategy creation, the practitioner may suffer from a setback in the form of a mismatch between the purpose of the study and the developed tools, as well as a mismatch between the developed tools themselves. For instance, working with the toolchain developed by Sellberg et al. (2017) for the purpose of modeling, simulation, and trajectory optimization of a chromatographic purification process, meant that the model, simulation strategy, and optimization approach had to be in harmony from the beginning when applying this toolchain in Paper VI. This was because the model equations were organized in order to be simulated as a set of equality constraints in the optimization problem via direct collocation. In this case, direct collocation can be seen as a simultaneous technique for optimization and simulation, and it serves as an example of how working through the modeling and simulation procedures without a clear view of the optimization approach would force the practitioner to loop back to the starting square *ad nauseam*. Another interesting example is when there is already a preexisting model and simulation procedure that we want to use for optimization studies, meaning that the optimization approach could not be taken into account during model and simulation strategy development. The utilization of an existing model and simulation strategy means that these steps (along with the first step of the optimization procedure, i.e. the process analysis) have already been performed at some point in time. However, for the purposes of a current study that an engineer is working on, it can be seen as injecting the model and simulation strategy straight in to the second step of the optimization procedure. This situation was the case of Paper IV. For the study, the evaporator system model that had been developed previously and used for *in silico* studies presented in Papers II and III (with the embedded simulation strategy provided by the AspenTech software, Aspen Plus Dynamics), was used with an entirely new purpose, for which a new optimization problem was formulated, and a new optimization approach was adopted. Taking an example from literature, Lindholm and Giselsson (2012) describe a procedure in which they start with modeling and then go on to formulate their optimization problem as they propose a method for modeling the relation of utility operation and production, meaning that they change places of parts of the integrated procedure.

Focusing a bit more on the impact of the chosen tools, and using Paper IV again as an example, by utilizing a process flowsheet model with an embedded simulation strat-

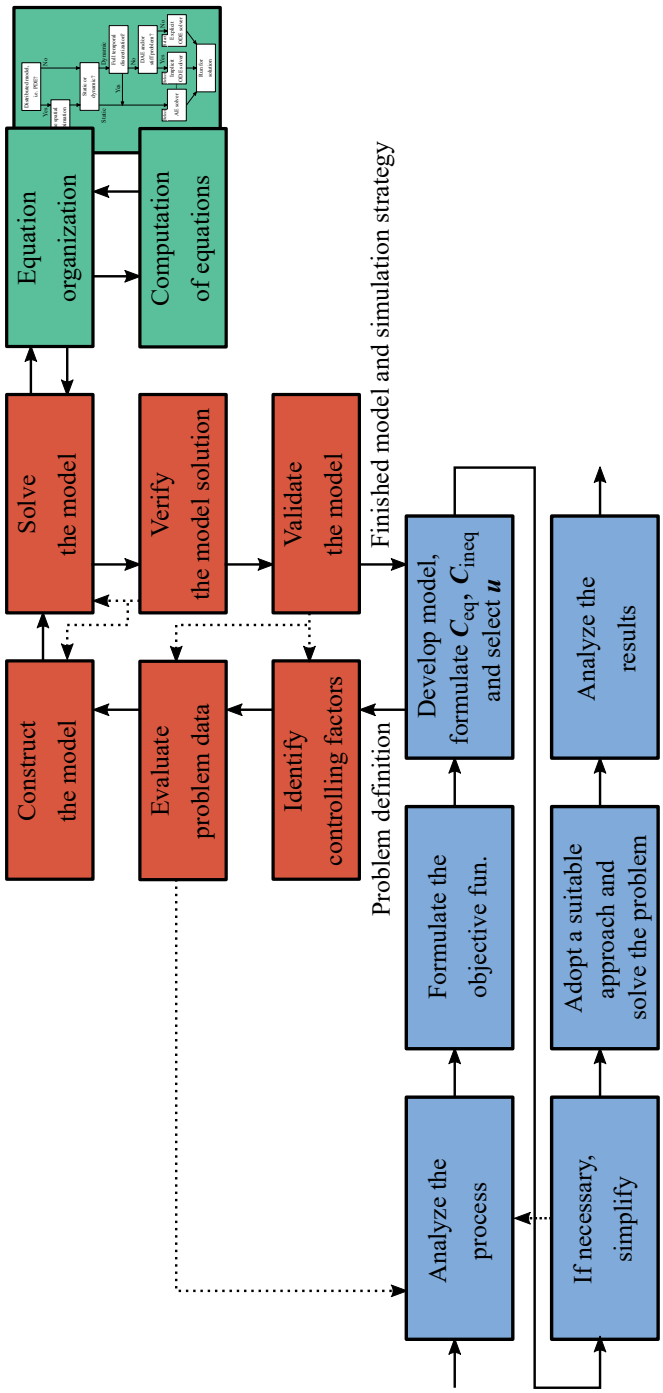



**Figure 3.5** An integration of the three different procedures for optimization (blue), modeling (red), and simulation (green).

egy, it was practically unfeasible with the available resources to implement advanced simulation and modeling techniques such as the collocation technique from Paper VI, and even automatically extracting an exact Jacobian was difficult. However, choosing the AspenTech suite as a toolset for modeling and simulation did allow for efficient and effective development of a usable model of a large complex process by making use of a library of physical properties for the required components and a library of unit operation submodels – libraries which otherwise would have been custom-made. By coupling it with Python, we were thus able to bring together the best of two worlds: the generalizability of Python with freely available state-of-the-art optimization algorithms, backed by the component information and modeling-simulation tools available in the AspenTech software suite. This was also the case of Paper III, where the very same toolchain (consisting of a series of Python packages and Aspen Plus Dynamics modules) was used for an optimization study with a different purpose. Similarly, making use of the preexisting blue-green systems model (and simulation strategy) from Paper VIII required a subsequent optimization problem formulation, a complementary model to describe the piping network dynamics with its own simulation strategy, as well as a co-simulation strategy in order to perform the optimization study in Paper IX. Here, too, the choice of modeling and simulation tools in some ways limited us in our choice of, and in other ways directed us towards a method for solving the problem, which can be summarized as derivative-free, unconstrained optimization based on co-simulation of models implemented and simulated in different environments. Similarly, but conversely, the methods chosen for Paper VI (e.g. direct collocation) demand a certain structure of the equation set as well as that the Jacobian and Hessian matrices are easily accessible and accurate, which thus had to influence the development of the utilized toolchain.

To better reflect the interplay between the choices of methods and tools, we can consider another interpretation of how to integrate the procedures for optimization, modeling, and simulation. This abstract interpretation is presented in Figure 3.6, and can be used to provide an overview of the purpose of a study, along with the chosen methods, tools, and/or approaches for optimization, modeling, and simulation. The colored parts from Figure 3.5 are preserved in their respective entirety and can be seen as simply slotted into Figure 3.6. Thus, in this perspective, the required work is purpose-centric instead of problem-initiated, and can be viewed as an extension of purpose-driven mod-

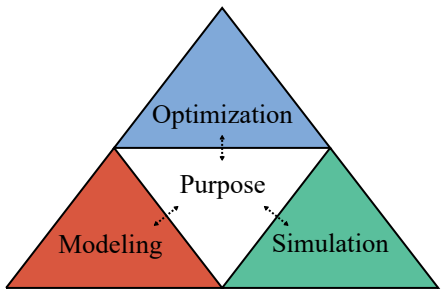



**Figure 3.6** A purpose-centric view of the integration of procedures for performing a computer-aided optimization study.

eling (Thaysen, 2006), in the sense that we need to establish a clear purpose to guide our choices in the different procedures. We can start with any part that is necessary for our optimization study, and switch tasks almost arbitrarily, as long as we revisit the overarching purpose of the study and perform each required task. This perspective also provides a more true-to-life perspective on how we can utilize preexisting models, as well as take the desired optimization approach into consideration during modeling and simulation strategy creation. As exemplified above, this is what we did in the work leading up to the studies presented in Papers III, IV, and IX. Naturally, at some point in time, the processes were analyzed and modeled – that is simply something that has to be done if a process is to be abstracted into a model. However, given those available models and a general purpose of wanting to find out how to best improve the studied processes, we were able to formulate interesting optimization problems, and adopt suitable approaches in conjunction with developing (co)simulation strategies. Thus, combining the purpose-centric, free-moving view on performing an optimization study with the three procedures for optimization, modeling, and simulation – or the integrated procedure as presented in Figure 3.5 – provides a flexible framework for staking out a purpose-guided path through the myriad of decisions that need to be made, encouraging consideration of their impacts and interplay.

# 4

# Computer-aided optimization of complex processes

As we have seen in the previous chapters, performing an optimization study may require plenty of considerations, decisions, and technical footwork in terms of creating a model, developing a simulation strategy, and adopting an appropriate optimization approach. In this chapter, the abstractions, framework, and procedures from the second and third chapters are used to describe, analyze, and contextualize the work presented in this thesis regarding the industrial purification of a polyalcohol as well as blue-green systems, appended as Papers I-V and Papers VIII-IX, respectively. The interrelations of these papers are presented in Figure 4.1, which shows the grouping of papers into the two cases. Paper I laid the technical foundation for Paper II, which in turn spawned Papers III and IV. Here, the simulation structure from Paper III was modified and used for optimization in Paper IV. Paper V is a technical report encompassing and describing the details of the work behind Papers I-IV. Also highlighted in Figure 4.1 is that the co-simulation approach used in Paper III inspired the same kind of approach in Paper IX, only with different (and more appropriate) modeling and simulation tools that were inherited from Paper VIII. Finally, as stated in the introduction, the case of chromatographic purification – Papers VI and VII – is included in this thesis primarily due to the pertinence of the utilized optimization tools and approaches to the previous chapters, and will not be revisited in this chapter. For a thorough review with great insights into open-loop optimization of chromatographic purification, please see Sellberg (2018).

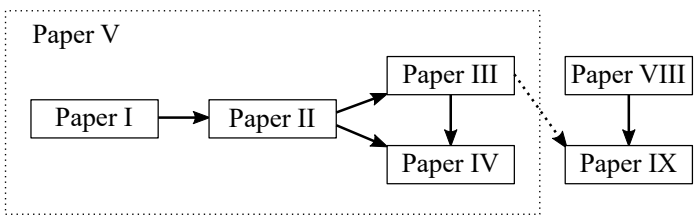



**Figure 4.1** The relationship of the collected work presented in this thesis regarding industrial purification of a polyalcohol, and blue-green systems for sustainable urban drainage.

## 4.1 Production systems – purification of a polyalcohol

The work concerning the industrial purification of a polyalcohol began with Paper I. The main purpose of the paper, as presented in Figure 4.2, was to apply a Subset Selection Algorithm (SSA) presented by Cintron-Arias et al. (2009) for the purpose of selection decision variables for computer-aided optimization, and to analyze its performance by performing optimization runs with the decision variable subsets suggested by the algorithm to be most effective at optimizing the process. To accomplish this purpose, a process model was required. The model built for this study was a simplified version of the two-stage evaporator system presented in Figure 1.1 including the stripper, but without the mechanical vapor recompression loop and the aforementioned oscillations, two factors that were implemented during later developments of the model. Having performed a process analysis with the help of experienced engineers and formulated an initial objective function with the intent to minimize direct variable costs whilst maximizing production, the integrated procedure from Figure 3.5 could be continued in order to create the model. The controlling factors of the evaporator system were identified to be the vapor-liquid equilibrium of the polyalcohol and several other compounds in conjunction with the Maurer kinetics. Problem data was evaluated and for the VLE, the vast AspenTech libraries for components and unit operations were used; for the Maurer kinetics, parameters that were found in literature as well as provided by AspenTech were used. Then, Aspen Plus was used to construct a steady-state model provided with input data regarding feed flow rates, compositions, temperatures, and pressures. The steady-state model was then converted into a dynamic model using the built-in tool for the conversion (following the recommendations from AspenTech), and simulated using the embedded default strategy of Aspen Plus Dynamics. This essentially meant that once the modeling tool was chosen, the freedom in creating a simulation strategy was heavily restricted since the modeling tool allowed for decisions at the level of selecting solving algorithms, and did not really allow for following the steps for creating

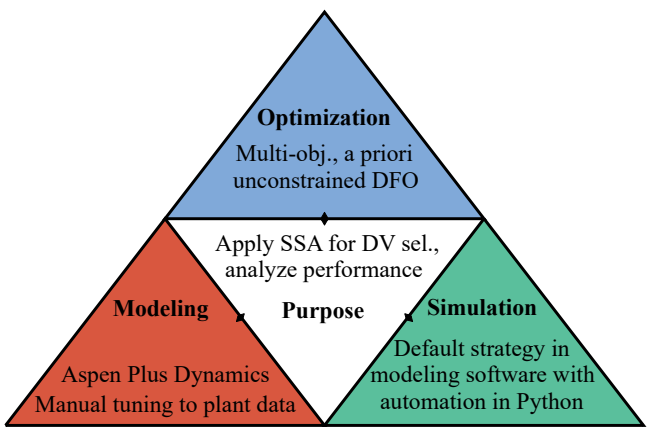



**Figure 4.2** A purpose-centric view of Paper I, where we aimed to apply the Subset Selection algorithm for decision variable selection, and to analyze the performance of the recommended subsets.

a simulation strategy presented in Figures 3.3 and 3.4. This has both advantages and disadvantages. For instance, one advantage is that modeling and simulation is made rather convenient using the component and unit operation libraries with the embedded simulation strategy, available at the fingertips with much of the work done almost for free. However, there is no such thing as a free lunch, since troubleshooting of complex models with little insight into the solution strategy may become very difficult, and flexibility for the user to adopt an optimization approach is very limited since tailoring of the solution strategy is not possible at the required level of equation organization. Finally, the model behavior was verified through iteration of previous steps, and when behaving as expected, it was validated by comparing model output to plant data, complemented via manual tuning of parameters for the model to match plant data within $\pm 3\%$.

Regarding the SSA, in order to make calculations of key numerical quantities used to rank subsets of decision variables, the SSA requires a sensitivity matrix (i.e. a Jacobian), which was estimated using a three-point central finite differences approach in the neighborhood of a nominal operating scenario. The simulation automation and computations required for the SSA was set up using Python, utilizing the COM protocol for communication between Aspen Plus Dynamics and Python. The optimization problems that were run were all formulated as *a priori* handled multi-objective (i.e. cast as single-objective), unconstrained problems solved with a DFO algorithm, namely the SciPy implementation of Nelder-Mead. In other words, the optimization was also handled in Python. An abstraction of the optimization problem is presented in Figure 4.3, with the Python-Aspen Plus Dynamics border being drawn to highlight the combined use of different software. This combination thus allowed us to efficiently model and simulate the complex process, making use of the AspenTech library of physical property data and methods required for the components present in the evaporator system, whilst simultaneously enabling the custom-made study that we aimed to perform. The optimization problem was formulated as Problem 4.1 below; here, $Q$ is the steam consumption used as a measure of the energy consumption, $X$ is the product purity, subscript *nom* refers to the nominal operating scenario, and the weight, $w$, was set to 0.7 to reflect on-site process economics. Furthermore, each set of decision variables, $u$, was part of the full

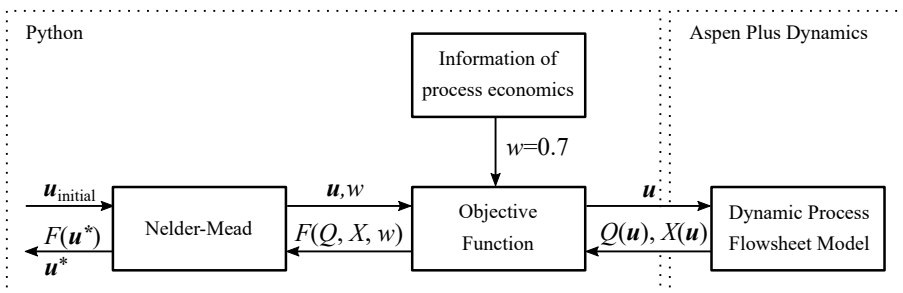



**Figure 4.3** An abstraction of the optimization problem(s) in Paper I, where Nelder-Mead was used to solve the multi-objective function, scalarized using the weighted sum method. The objective function evaluated the dynamic process flowsheet model, which for each decision variable setting returns the energy consumption, $Q$, and the product purity, $X$.

set of the 50 recommended sets of decision variable combinations by the SSA, $u_{SSA}$.

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & F(\boldsymbol{u}) = w\frac{Q(\boldsymbol{u})}{Q_{nom}} - (1-w)\frac{X(\boldsymbol{u})}{X_{nom}} \\ \text{for each} \quad & \boldsymbol{u} \in \boldsymbol{u}_{SSA} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_{50}] \\ \text{s.t.} \quad & \text{Dynamic process flowsheet model} \end{aligned} \tag{4.1}$$

Because the optimization problem was solved without constraints and bounds, only the physical feasibility of the process was ensured, as the problem was only subject to the convergence of the process flowsheet simulation. The practical feasibility, e.g. in terms of pump-controlled flow rates etc., was not guaranteed through additional optimization constraints, which is a clear weakness of Paper I. Bearing this in mind, the results of the combined SSA and optimization study was that the recommended decision variables were generally more effective in terms of minimizing the objective function compared to the real-life controlling inputs of the process. These conditions and results highlighted the necessity to develop the model as well as the optimization problem formulation, and the need to reconsider the decision variables chosen in the real process.

Continuing, a need for further process analysis was identified, concerning in particular the oscillations in the evaporator system. The oscillations were begotten by the combined flow of intermittent concentrated feed and the continuous diluted downstream recycling, both directed into one balance tank preceding the stripper, which is how the disturbing phenomenon was also modeled. These combined flows thus created oscillations that began a new cycle each time the intermittent feed became active. The negative effects of the oscillations and their propagations on the overall process performance were investigated and analyzed through *in-silico* sensitivity analysis in Paper II. Here, the downstream recycling mass flow was perturbed by $\pm 2.5\%$, and the resulting effects were scrutinized with regards to the four parameters of (i) balance tank product mass fraction, (ii) the product mass flow as well as (iii) the product purity after the second stage, and (iv) the steam consumption. For this purpose, the modeling and simulation tools were inherited from Paper I, as seen in Figure 4.4. However, the model was developed to also include the MVR circuit, as presented in the schematic in Figure 4.5. The oscillatory behavior was also implemented as described above.

The objective function from Paper I was also reused for the study, but not for performing an optimization study; the objective was used for aiding in the process analysis, in order to quantify the system performance over the course of the oscillatory cycles. The major insight was that the way the feed and recycling mass flows were combined excited a $\pm 15\%$ oscillation in the balance tank product mass fraction, which in turn propagated and affected performance throughout the system. Since equipment such as pumps need to be able to manage the peaks, this meant a reduction of throughput as well as an increase in steam consumption – in other words, a reduced production quantity with an increased resource consumption, impacting both profitability and environmental performance.

As indicated in Figure 4.1, the quantification of the performance loss in Paper II led to investigations into how to best improve the situation. In Papers III and IV, the modeling and simulation tools along with the models used for the previous papers were

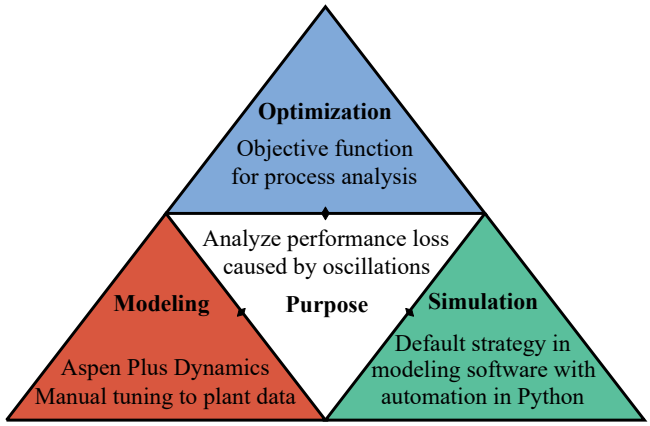



**Figure 4.4** A purpose-centric view of Paper II, where we aimed to analyze and quantify the performance loss due to the oscillations in the system.

again inherited and utilized for the new optimization studies.

The purpose of Paper III was to investigate how to minimize the costs induced by the steam consumption, as seen in the abstraction presented in Figure 4.6. However, the steam consumption in the evaporator system was not the only concern; in order to provide a more holistic overview of the effects on the cost drivers across several subsystems, the methanol distillation column (that also uses steam at a high rate) directly connected to the evaporator system was of interest as well. Furthermore, due to stiffness-caused convergence failures when attempting to simulate a dynamic flowsheet model consisting of both the two-stage evaporator system and the methanol column, the decision was made to go with modularization, i.e. to set up the models as individ-

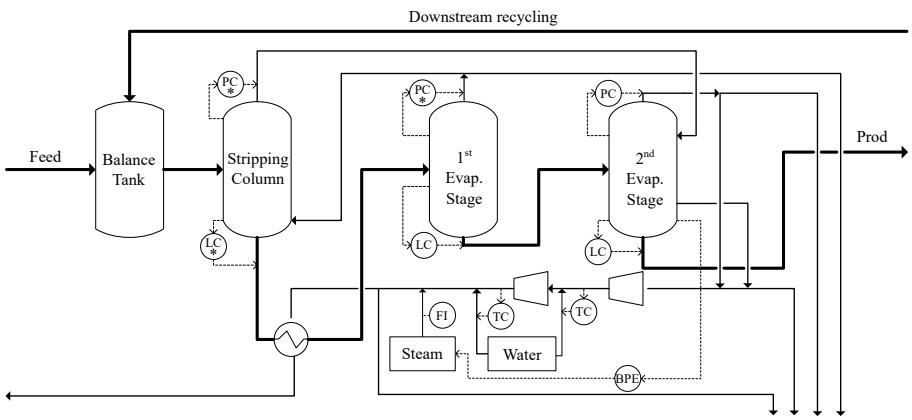



**Figure 4.5** A schematic of the process flowsheet and the model used in Papers II, III, and IV. Note that in Paper IV, the BPE control was removed in order to let the optimizer change the steam consumption over time.

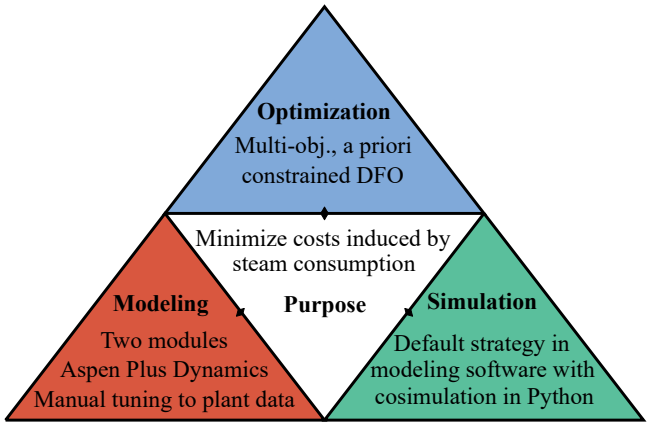



**Figure 4.6** A purpose-centric view of Paper III, where we aimed to minimize the costs induced by the steam consumption in two modules by formulating a pseudo-economic objective with cost factors for the two modules set *a priori*, i.e. used as weights.

ual modules. The modules used for Paper III are presented in Figure 4.7. The decision to create modules required a co-simulation strategy, which was developed in Python (described by increasing degree of detail in Papers III, IV, and V). In summary, the developed co-simulation strategy was implemented in Python and called the Python Module Coupler (PyMoC), and was used to transfer the stream information over the course of the dynamic simulations from any source stream to the corresponding destination stream. PyMoC does this automatically by coupling together identically named streams in the different modules, and identifies which is source and which is destination by checking the variable property in Aspen Plus Dynamics to see whether it is free (i.e. calculated as output, thus the source) or fixed (i.e. set as input, thus the destination). In this case, the source streams from the evaporator passed through a translation module (as indicated in Figure 4.7), the purpose of which was to remove formaldehyde, which was negligible in the real process.

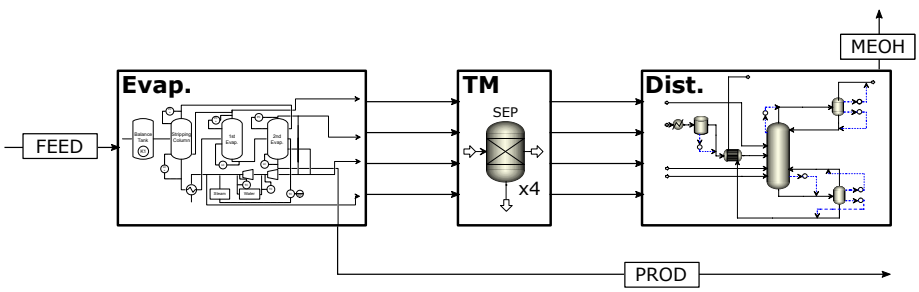



**Figure 4.7** The investigated system in Paper III, presented in the form of the utilized modules. Note that the evaporator system module is inherited from Paper II, whereas the translation module (TM) and the methanol distillation column were created for the study.

The optimization problem in Paper III was to minimize the steam costs in the two modules, whilst still satisfying operating constraints on average production throughput and product purity on both the polyalcohol and the methanol (as well as a minimum flow rate on the methanol, since it was used as fuel in other parts of the process). To do this, a pseudo-economic function was formulated with approximated normalized cost factors (which can be seen as an *a priori* approach to multi-objective optimization). The problem was unbounded, leaving the decision variables to freely take any value. However, since the objective function required no information from the model, the model was simulated only as part of the constraints. Together with the *a priori* approach to multi-objective optimization, this yields the optimization problem abstraction presented in Figure 4.8. The optimization problem was mathematically formulated as Problem 4.2 below, and solved using the SciPy implementation of COBYLA:

$$
\begin{aligned}
\min_{u_1,\, u_2} \quad & F(\boldsymbol{u}) = \sum_{i=1}^{2} \frac{u_i}{u_{i,nom}} \cdot c_i \\
\text{s.t.} \quad & \text{Modularized flowsheet model} \\
& \overline{w}_{PROD} - C_{w_{PROD}} \geq 0, \qquad \overline{P}_{PROD} - C_{P_{PROD}} \geq 0 \\
& \overline{w}_{MEOH} - C_{w_{MEOH}} \geq 0, \qquad \overline{P}_{MEOH} - C_{P_{MEOH}} \geq 0 \\
& w_{MEOH,\, min} - C_{w_{MEOH,\, min}} \geq 0 \\
& u_1 = w_{steam,\, evap} \in \mathbb{R} \qquad u_2 = w_{steam,\, dist} \in \mathbb{R}
\end{aligned}
\tag{4.2}
$$

Here, $c_i$ for $i = [1,2]$ refers to the cost factors for the steam consumption in the evaporator system and the methanol distillation column, set to $c_1 = 1.0$ and $c_2 = 0.7$ to reflect the distinct cost levels incurred by the different steam pressure levels; $w$ is mass flow rate; $P$ is purity; $\overline{w}$ and $\overline{P}$ denote the average of the respective parameter over the time period; subscripts $PROD$ and $MEOH$ refer to the streams thus named in Figure 4.7; $u_{i,nom}$ is the nominal decision variable values; $C_{k_j}$, $k = (w,P)$, $j = (PROD, MEOH)$ de-

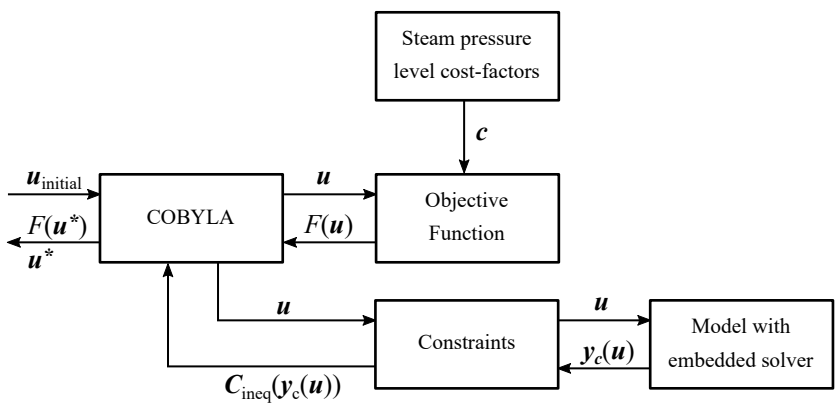



**Figure 4.8** The abstraction of the optimization problem as solved in Paper III. Here, $\boldsymbol{y}_c(u)$ includes $i_j$, where $i = (\overline{w}, \overline{P})$, $j = (PROD, MEOH)$, and $i = w$ for $j = MEOH$, $min$.

note the respective values of the *mean constraints*; and $C_{w_{MEOH,\,min}}$ denotes the constraint value for the minimum flow rate in the MEOH stream. Whilst parts of the abstraction in Figure 4.8 may be recognized from the discussion regarding full discretization in section 2.2.5, it should be noted that the embedded solver in the software was still used for solving the problem in Paper III.

The optimization results for the mass flow and purity of the product stream are presented in Figure 4.9. In the upper graph of Figure 4.9, we can see the effective increase of the product stream flow rate to a level that satisfies the average flow rate constraint. However, as can be seen in the lower graph of Figure 4.9, the increase in flow rate came at the expense of a purity reduction, which was allowed by the constraints set to reflect the real-life process and product demands. In the case of the methanol distillation column, the optimized flow rate and purity results are presented in Figure 4.10. For this subsystem, both the purity and the mass flow rate was increased – looking especially at the purity, this was in fact increased with a significant margin to the constraints.

In addition to the decreased steam consumption visible in Figure 4.9, there is an obvious change not only in the level but also in the behavior of the oscillations in the mass flow rate. The oscillations in the methanol purity were also fully eliminated. These results meant that even a time-invariant steam consumption could impact the oscillatory cycles of the system, which led us to believe that finding a steam consumption trajectory to minimize the oscillations on the product mass flow rate would be possible. This became the purpose of the study presented in Paper IV, as seen in the center of the abstraction in Figure 4.11.

In Paper IV, the same model as in Paper II was used, albeit run at a different nominal case. The default strategy in the simulation software was once again used, but to solve the trajectory optimization problem, PyMoC was modified slightly in order to perform a partial discretization instead of co-simulation, more specifically employing the direct sequential approach. This showcases the versatility of the tools available in Python. However, we did opt to perform partial discretization in part due to being un-

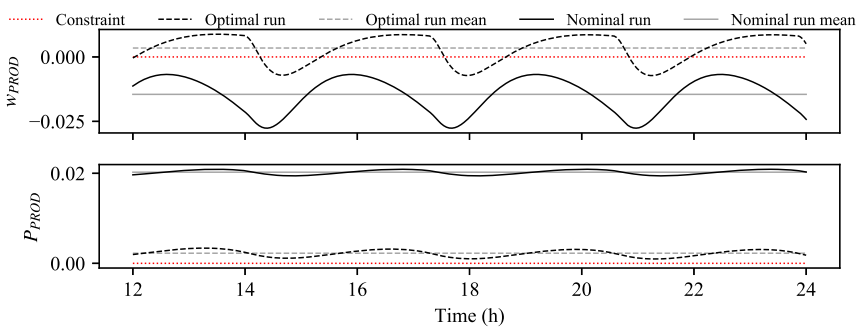



**Figure 4.9** Mass flow rate and purity for the product stream marked PROD in Figure 4.7. The optimal solution shows an increased product flow rate but a reduced purity compared to the nominal case since the constraints allowed for a reduction in purity.

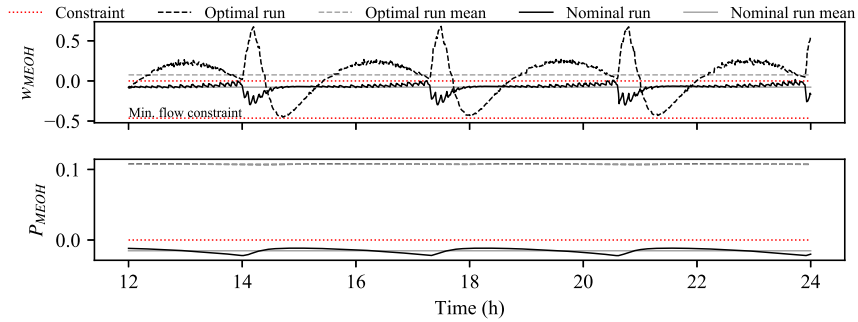



**Figure 4.10** Mass flow rate and purity for the product stream marked MEOH in Figure 4.7. The optimal solution shows an increased product flow rate and an increased purity compared to the nominal case, satisfying all constraints.

able to find a way to perform a full discretization using the model implemented in Aspen Plus Dynamics. The primary obstacle was retrieving the necessary model equations to use as constraints as well as the gradient information required to efficiently solve the optimization problem. This serves as an example of how the tools chosen for a study are then inherited and enables another study while also restricting it, showing what the impact and interplay of the tools and methods chosen for optimization may be across several studies. An additional reason for selecting the direct sequential approach with a DFO algorithm was the previous successful research published on the subject by e.g. Negrellos-Ortiz et al. (2018). Nevertheless, the direct sequential approach meant tem-

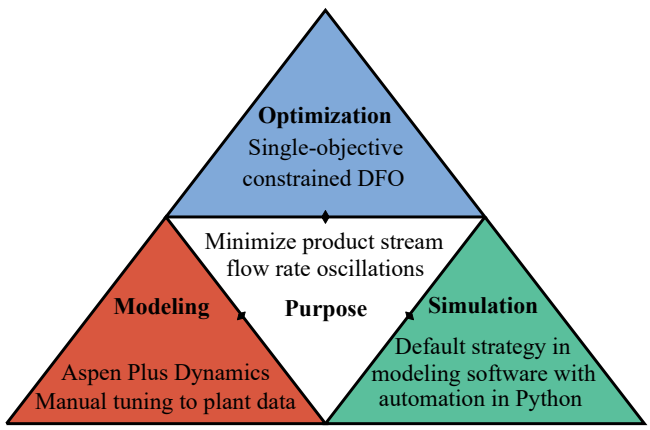



**Figure 4.11** A purpose-centric view of Paper III, where we aimed to minimize the disturbing oscillations by utilizing the previously developed Python-Aspen Plus Dynamics tool-chain in a single-objective constrained optimization problem, which was solved using the DFO algorithm known as COBYLA.

poral discretization of the decision variable (here the steam consumption), which was handled by the PyMoC based algorithm, and letting Aspen Plus Dynamics evaluate the model – an abstraction of the solution method is presented in Figure 4.12. The oscillations that were to be minimized were quantified utilizing the variance ($\sigma^2$) of the mass flow of the product stream ($w_{prod}(t,\boldsymbol{u})$). Furthermore, in order to minimize the risk of so called bang-bang optimization, a penalty function ($\rho(\boldsymbol{u})$), which sums the squared difference of the values of neighboring discretized decision variables, was added to the objective function. In order to make sure that the optimal solution would satisfy real-life operating conditions, whilst giving the optimizer some space to work with, constraints on the average mass flow rate and the product purity were formulated so that they had to be at least 99.8% of the nominal case averages. The discretized decision variable was also bounded between half and double the nominal (time-invariant) decision variable value; however, since COBYLA only accepts constraints and not bounds, these were formulated as constraints. The optimization problem was thus mathematically formulated as Problem 4.3 below:

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & F(\boldsymbol{u}) = \frac{\sigma^2(w_{PROD}(t,\boldsymbol{u})) + \rho(\boldsymbol{u})}{F_{nom}} \\ \text{s.t.} \quad & \text{Flowsheet model} \\ & \overline{w}_{PROD} - \beta \cdot \overline{w}_{PROD,\,nom} \geq 0 \\ & \overline{P}_{PROD} - \beta \cdot \overline{P}_{PROD,\,nom} \geq 0 \\ & 0.5 \cdot u_{nom} \leq u \leq 2 \cdot u_{nom},\ \boldsymbol{u} = [u_1, ..., u_{N_u}] \end{aligned} \tag{4.3}$$

Here, $\beta = 0.998$, and $\rho(\boldsymbol{u}) = R \cdot \sum_{i=1}^{N_u - 1} \Delta u_i^2$, where $R$ is a case-specific weight set to scale the penalty; also, the objective function formulation was normalized with respect to its nominal value ($F_{nom}$), i.e. with a time-invariant decision variable yielding $\rho(\boldsymbol{u}) = 0$.

The results from the optimization are presented in Figure 4.13, where the mass

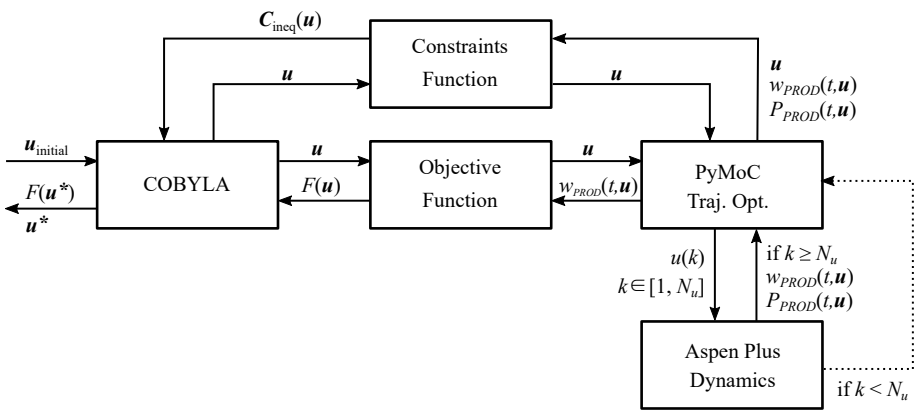



**Figure 4.12** A schematic of the optimization problem in Paper IV. The objective function utilizes PyMoC to discretize the decision variable, which in turn uses the embedded strategy Aspen Plus Dynamics for simulation of the process model.

flow rates of product and steam have been normalized. In Figure 4.13a, the nominal case is presented, and in Figure 4.13b, the optimized scenario is presented. As can be seen, the oscillations are significantly reduced, in fact to an objective function value corresponding to 0.3% of that of the nominal case. In addition to that, the average steam consumption was reduced to 69% of the nominal level whilst satisfying the purity constraint of 99.8% of the nominal purity. Minimizing the oscillations in this manner leads to a opportunities for financial gain, since throughput can be increased as the constraining impact of the oscillation peaks has been minimized. However, practical implementation in the real process most probably requires some sort of closed loop control, which means that this open-loop optimization approach needs further work.

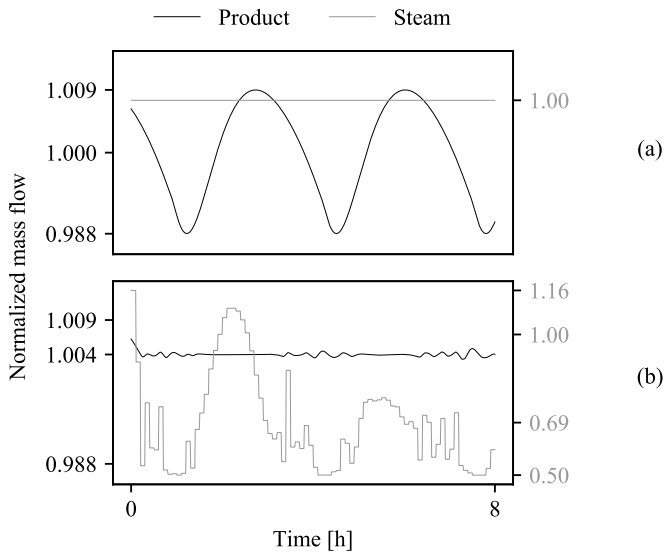



**Figure 4.13** The results of the optimization in Paper IV. The mass flow of product as well as steam have been normalized to their respective nominal values.

As shown throughout these four publications (Papers I-IV) as well as the supplemental technical report (Paper V), with relatively simplified yet effective means, we were able to create rigorous process models in an efficient manner. We were further able to reuse them in different applications of computer-aided optimization that are common within the field of process systems engineering. These applications have led to a greater understanding of the process, which in turn facilitates implementing physical improvements. In addition to the findings directly connected to the industrial purification of a polyalcohol at Perstorp AB, through abstraction we were also able to generalize and apply the used methods and approaches to other fields and processes; this proved useful for investigating blue-green systems for sustainable urban drainage.

## 4.2 Urban environments – blue-green systems

The blue-green systems of the neighborhood Augustenborg in Malmö were studied in depth, analyzed, and conceptualized by Haghighatafshar et al. (2018a) and Haghighatafshar et al. (2018b), where controlling factors and governing equations were identified and formulated. The work in these two papers thus comprise the first step of the optimization part as well as the first two steps of the modeling part of the integrated procedure presented in Figure 3.5. The conceptualization of the process can be summarized as having individual stormwater control measures (SCM) be organized into blue-green systems, where the SCMs will collect and retain rainwater, passing it downstream when full. Essentially, three characteristics of the blue-green systems can be considered the controlling factors, namely the retention ($R$); the stormwater control measure area ($A_{SCM}$); and the directly connected impervious area ($DCIA$) to the SCM.

The purpose of Paper VIII, as presented in Figure 4.14, was therefore to develop this physically based model together with a swift and sturdy simulation strategy, facilitating inexpensive simulation of the existing blue-green systems, with particular respect to the estimation of lag times and peak discharges. In order to do this, continuing to go by the integrated procedure in Figure 3.5, the model was constructed purely in Python using the four governing equations presented in Equations 4.4-4.7. The first step of the model is to use the system characteristics and a rainfall time series to create an equidistant time series of the rain depth and to calculate the effective retention capacity, $R_e^{i,t}$, for each time step, $t$, according to Equation 4.4. Following this, the vessel discharge volume, $V_{out}^{i,t}$, can be calculated according to Equation 4.5.

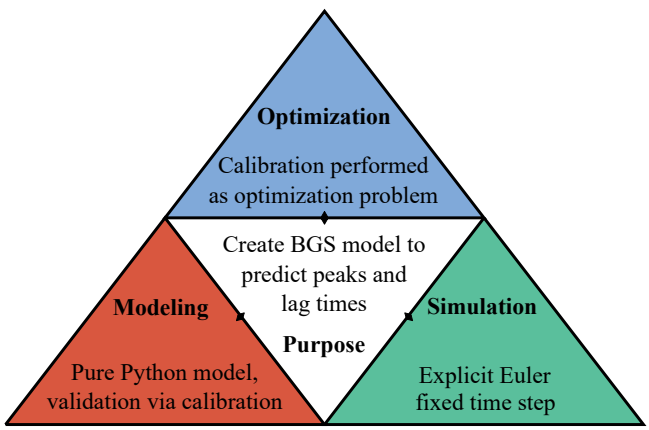



**Figure 4.14** A purpose-centric view of Paper VIII, where we aimed to create a quick and robust model to inexpensively predict peaks and lag times of the volumetric discharge from blue-green systems as a result of (heavy) rainfall. This was done entirely in Python, arranging the equations for solving with a fixed-step explicit Euler method. The model was calibrated, the problem of which was posed as an optimization problem, and whilst this may be considered to be part of the validation in the modeling procedure, it is presented as part of the optimization due to the approach during the study, as presented in the paper. BGS is here used as shorthand for blue-green systems.

$$R_e^{i,t} = \frac{(S_{fb}^i + S_{inf}^i) \cdot A_{SCM}^i}{DCIA^i} \tag{4.4}$$

$$V_{out}^{i,t} = \frac{(R - R_e^{i,t}) \cdot DCIA^i}{1000} + \sum_{j=1}^{a} V_{out}^{i-j,t} \tag{4.5}$$

Here, $S_{fb}^i$ and $S_{inf}^i$ represent the storage depth in the freeboard and the infiltration layer, respectively, of an SCM as depicted in Figure 4.15. The physical representation of the effective retention capacity and the vessel discharge volume and their relationship to the DCIA for each SCM is visualized in Figure 4.16a. These two equations are volume-based, and by using the equidistant time series of accumulated rain depth, the results of Equation 4.5 can be used to calculated the flows from each SCM, $q^{i,t}$, according to 4.6. Then, Equation 4.7 can be used to calculate the dynamic storage volume in the virtual reservoir, $S$, residing at the end of the blue-green system as depicted in Figure 4.16b.

$$q^{i,t} = \frac{V_{out}^i - V_{out}^{i-1}}{\Delta t} \tag{4.6}$$

$$\frac{dS}{dt} = q^{n,t} - \hat{Q}_{out} = q^{n,t} - k \cdot S^m \tag{4.7}$$

In equations 4.4-4.7, the index $i$ represents each SCM in the full blue-green system, $a$ is the number of directly connected upstream SCMs to the current SCM, and $n$ is the total number of SCMs in the blue-green system, and $k$ and $m$ are non-linear reservoir model parameters. It should also be noted that Equation 4.6 is in some sense the dynamic version of Equation 4.5, in Paper VIII temporally discretized with $\Delta t = 5$ min. The choice of the time step followed from the input data that was transformed into an equidistant time series of accumulated rain depth. These governing equations were arranged in order to reflect the process as conceptualized in Figure 4.16b, with Equation 4.7 representing the non-linear reservoir model. The model algorithm as described above is schematically presented in Figure 4.17. The modeled system discharge, $\hat{Q}_{out}$ was simulated utilizing a custom-made fixed-step explicit Euler method. The choice of this simulation method was based on the same line of thinking as the decision tree in Figure 3.4, and – whilst certainly not the most sophisticated method for numerical integration – ensured inexpensive simulation of the model equations whilst also facilitating

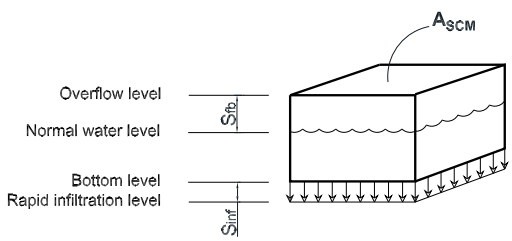



**Figure 4.15** An abstraction of a single SCM as used in the model for the blue-green system.

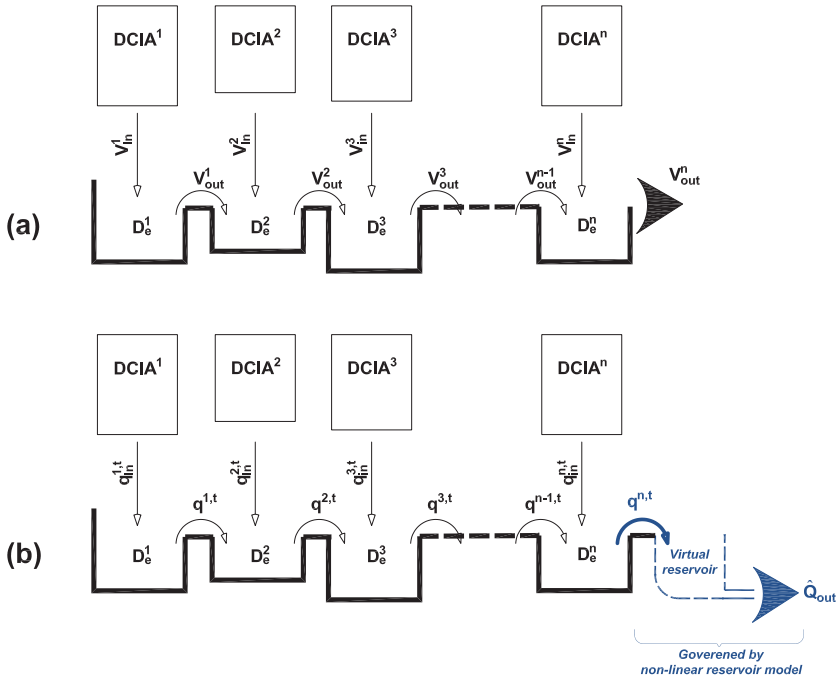



**Figure 4.16** (a) A schematic representation of the conceptual model as developed by Haghighatafshar et al. (2018a), able to predict the accumulated discharge volume from the system. (b) A schematic representation of the conceptual model as developed in Paper VIII, which is able to also predict the dynamic discharge of water due to the temporal discretization of Equation 4.5 into Equation 4.6, enabling prediction of peak discharge and lag times.

easy comparison of the model output with the temporally equidistant measurements used as input to the model.

Verification of the Python implementation of the model was performed by comparing the output with a preexisting toolchain using Excel in conjunction with Matlab. When identical results were retrieved with the two implementations, the Python implementation was considered finished. Furthermore, since it performed the necessary calculations in less than a second – which was several orders of magnitude quicker than the previous implementation – and worked for several different rainfalls and system configurations, the implementation was considered to be both swift and sturdy. The model was then calibrated, and as presented in Figure 4.14, this was handled as an optimization problem. The formulated objective was to minimize the Nash-Sutcliffe Efficiency Index ($NSEI$), as defined in Equation 4.8 below. Essentially, the $NSEI$ quantifies the fit of the model output ($\hat{Q}_{out}$) as compared to the measured time series ($Q_{out}$), where $NSEI = 1$ means a perfect fit, $NSEI = 0$ means that the predicted model output is as good as the mean value of the observations, and $NSEI < 0$ means that the model is worse than the mean value of the observations ($\overline{Q}_{out}$) at predicting the system behavior.

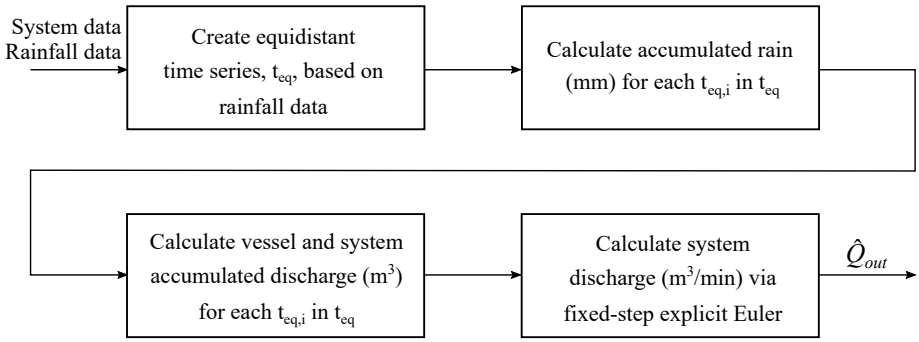



**Figure 4.17** A schematic of the developed model and the flow of information.

$$NSEI = 1 - \frac{\sum_{i=1}^{N}(\hat{Q}_{out} - Q_{out})^2}{\sum_{i=1}^{N}(Q_{out} - \overline{Q}_{out})^2}, \quad -\infty < NSEI \leq 1 \tag{4.8}$$

In order to simplify calibration, the non-linear reservoir model parameter $m$ was set to 1, meaning that it was instead considered a linear reservoir model, and the calibration was performed only with respect to $k$, which was bounded to be greater than zero. The calibration problem was thus formulated as Problem 4.9 below, and an abstraction of the calibration cast as an optimization problem is presented in Figure 4.18.

$$\begin{aligned} \min_{k} \quad & -NSEI \\ \text{s.t.} \quad & \text{Equations 4.4-4.7} \\ & k > 0 \end{aligned} \tag{4.9}$$

Solving this calibration problem with COBYLA for two different configurations – reflecting the Northern and Southern parts of Augustenborg, respectively – yielded a model capable of producing the results presented in Figure 4.19. These results indicate a satisfying fit with regards to lag times and peak flows between the model output and

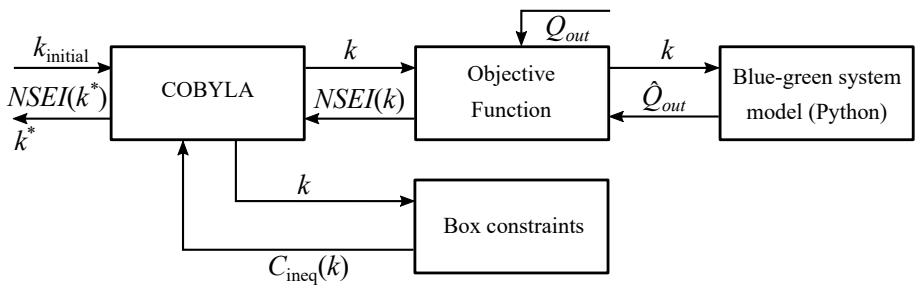



**Figure 4.18** An abstraction of the calibration problem, cast in the now familiar shape of a box-constrained derivative-free optimization problem.

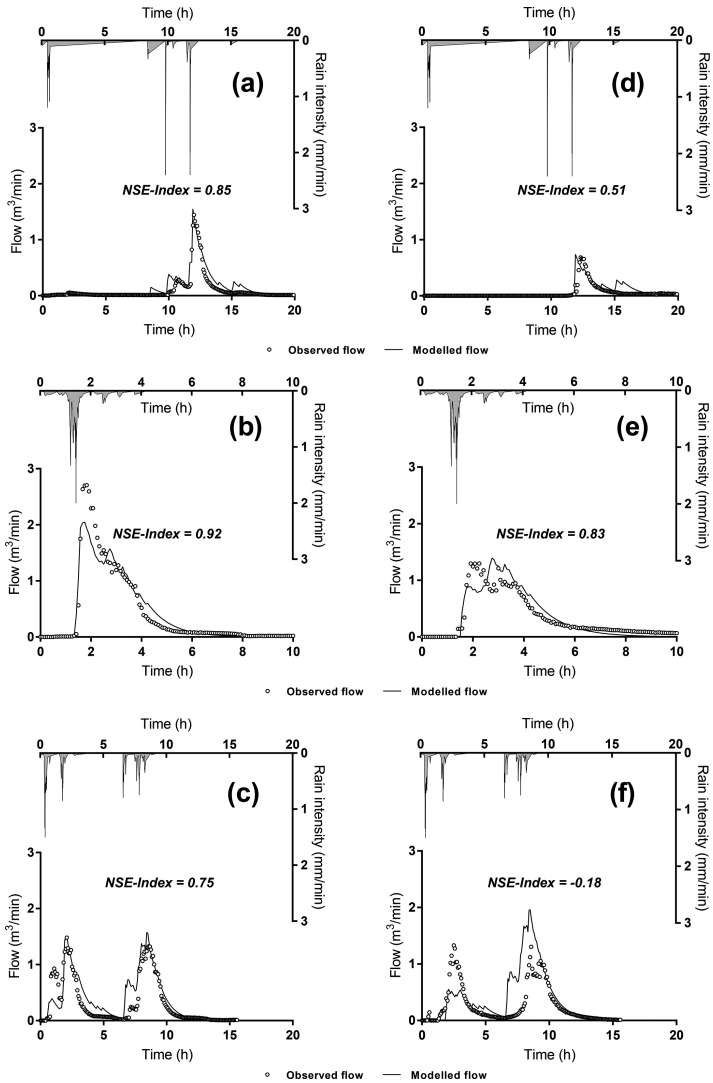



**Figure 4.19** Measured versus modeled discharges from the Northern (a, b, c) and Southern configuration (d, e, f) in Augustenborg, Malmö, three different rainfall scenarios (one scenario per row).

the real-life measurements for both of the system configurations and several different rainfalls.

With a finished model that could satisfyingly predict lag times and discharges, the purpose of Paper IX, as presented in Figure 4.20, became to study the sizing and siting of blue-green system retrofits across the city of Malmö. However, the blue-green system model was not enough on its own as there was also a need to describe the behavior of the piping network in order to see what effects the blue-green effects had in terms of

mitigating flooding in the pipe network. Therefore, preexisting models for the impervious area (i.e. area not retrofitted with blue-green systems) rainfall runoff and piping network behavior were used in conjunction with the BGS model to predict the number of flooded manholes for a 10-year rainfall, and these two modules were co-simulated as according to the rightmost part of Figure 4.21. The idea to cosimulate the models was inspired by the previously described co-simulation-optimization study in Paper III, and the co-simulation itself was handled in the MIKE suite provided by DHI Sverige AB.

The issue of sizing and siting was cast as an optimization problem, where we chose a simplistic, capitalistic view on implementation. An economic objective function was formulated to take into account both the implementation costs of blue-green systems, as well as the saved costs due to reduced flooding. These costs were annualized using the capital recovery factor and the sinking fund factor (similar to NPV), according to Equations 4.10 and 4.11 presented below:

$$C_{act} = \sum A_{BG} \left( c_{BG} \cdot \frac{i(1+i)^L}{(1+i)^L - 1} + \frac{M}{L} \cdot \sum_{l=0}^{L-1} (1+\alpha)^l \right) \tag{4.10}$$

$$C_{flood} = c_{fn} \cdot n_{fn}(\boldsymbol{u}) \cdot \frac{i}{(1+i)^T - 1} \tag{4.11}$$

Here, $\sum A_{BG}$ refers to the total area of retrofitted blue-green systems (approximated based on the Augustenborg implementation); $c_{BG}$ is the average cost of implementation of blue-green systems per area unit; $i$ is the average interest rate (assumed to 3%); $L$ is the technical lifespan of the retrofit (assumed to 50 years, which is the local standard for infrastructure investments); $M$ is the maintenance cost; $\alpha$ is the average annual pay

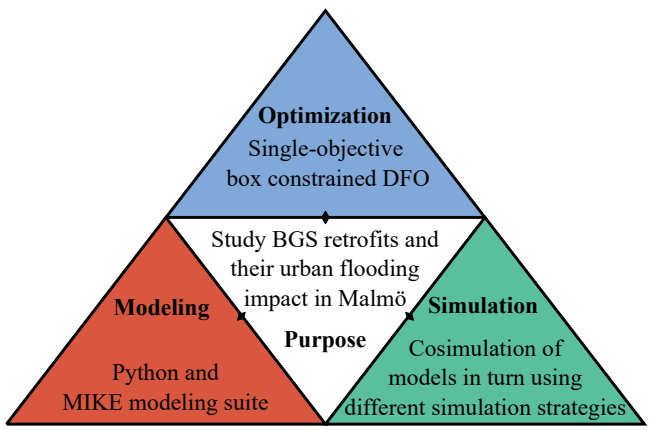



**Figure 4.20** A purpose-centric view of Paper IX, where we aimed to study the size and location of blue-green system retrofits across the city of Malmö. This was done by using the blue-green system model as implemented in Paper VIII in conjunction with preexisting impervious area rainfall runoff and piping network models in the MIKE suite. The simulation of the latter models, as well as the co-simulation strategy, was handled in the MIKE suite, whilst the optimization was implemented in Python. BGS is here used as shorthand for blue-green systems.

raise (assumed to 2%); $c_{fn}$ is the average cost per flooded node (approximated by using the cost incurred by the Malmö flooding of August 31 2014); $n_{fn}$ is the model output in terms of number of flooded nodes, or manholes; and $T$ is the recurrence interval of the storm used for optimization (i.e. 10 years). Furthermore, the majority of the city of Malmö was in the study divided into 30 different areas, or subcatchments, and the implementation extent of blue-green systems in each of these 30 subcatchments were used as the decision variables. These decision variables were subject to box constraints specifying that the implementation must be between 0-100%, thus constraining the decision variables within physically possible bounds. The optimization problem was thus formulated as Problem 4.12 below:

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & F(\boldsymbol{u}) = \frac{C_{total} - C_{nom}}{C_{nom}} \\ \text{s.t.} \quad & C_{total} = C_{act} + C_{flood} \\ & \boldsymbol{u} = [u_2, u_2, ..., u_{30}],\ u_i = [0,1] \text{ for } i \in [1,2,...,30] \end{aligned} \tag{4.12}$$

Here, subscript *nom* refers to the nominal case, i.e. the actual flooding of August 31 2014 in Malmö where no blue-green systems were implemented in the investigated subcatchments, and $u_i,\ i \in [1,2,...,30]$ is the implementation extent factor of blue-green systems in each catchment. The problem was solved by applying the DFO algorithm COBYLA, and a schematic of the optimization problem solution structure is presented in Figure 4.21, resulting in the map presented in Figure 4.22. The optimal solution entails modifying the surfaces from impervious to blue-green systems to an extent of 56.6% of the area in the first catchment and 63.6% of the area in the nineteenth catch-

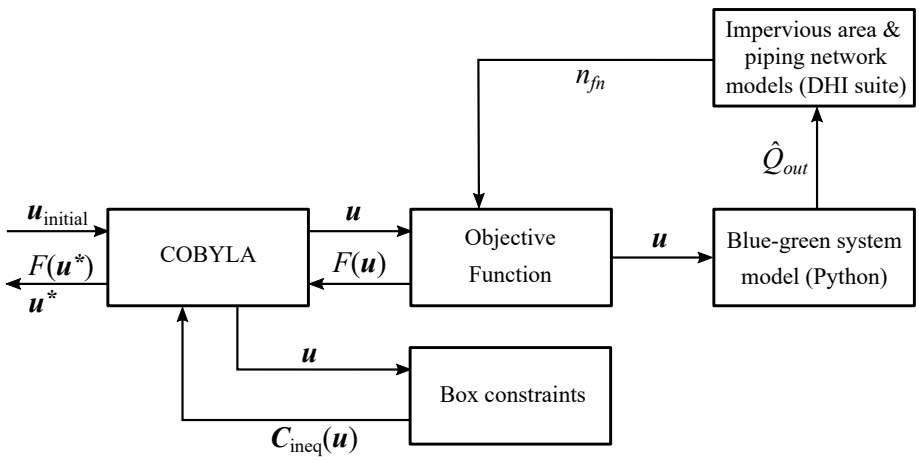



**Figure 4.21** A schematic of the optimization problem in Paper IX. The objective function required the evaluation of the blue-green system model and models for the impervious area as well as the piping network in series. This series of evaluations returned the number of flooded manholes, $n_{fn}$, which was used to calculate the economic objective function.

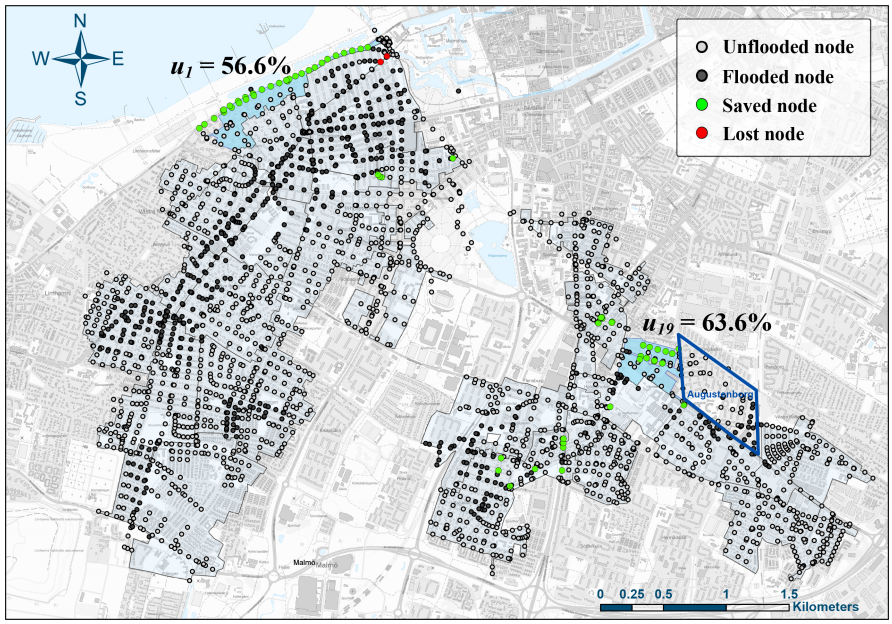



**Figure 4.22** A map of the optimal solution that minimizes the total cost. For easier comparison with the nominal case, the manholes have been marked out as flooded (filled markers), unflooded (unfilled markers), as well as saved (green-filled markers), and lost (red-filled markers) as compared to the nominal case. Background picture: dimmed topographic web map, courtesy of The Swedish Mapping, Cadastral and Land Registration Authority, ©Lantmäteriet.

ment, *caeteris paribus*. This reduces the number of flooded nodes by 8.5%, and results in a cost saving of 3% for the simulated scenario. Whilst this is a modest cost reduction, it should be noted that this is a very simplified objective that is not intended to take into account, for instance, the urban beautification provided by the implementation blue-green systems. However, this is a factor that may be quantified and taken into account by using available tools developed for that specific purpose.

In Paper IX, the DFO algorithm COBYLA was once more applied. This was chosen primarily because the Jacobian was difficult to formulate and retrieve due to the complex behavior of the piping network and its model, leading to difficulties in determining how the number of flooded nodes changes with respect to the implementation of blue-green systems. As an alternative, a numerical estimation of the Jacobian could have been performed, e.g. by simply using the built-in function of SLSQP so that the practical difference compared to using COBYLA would have been insignificant. However, the familiarity of COBYLA from the calibration of Paper VIII made it a comfortable choice. Whether this was the most efficient option has not been explored *ex post facto*, but it can be satisfactorily stated that the algorithm was effective for the intents and purposes of the study in Paper IX. As such, by choosing to work with the modeling and simulation tools that we used profoundly impacted the available options in terms of

optimization approaches. It should be noted that the reverse is also true; had we desired a certain optimization approach that was incompatible with the available models, then we would have needed to create new models by using tools that facilitate the desired approach. However, choosing to work with the preexisting runoff and piping models enabled us to effectively gather interesting results and insights into flood mitigation in Malmö in a short period of time.

# 5

# Concluding remarks

In conclusion, an integrated procedure and a purpose-centric framework have been provided in this thesis and used to describe the presented and contextualized cases, in which there was a need for improving the particular processes. These improvements have subsequently been found through the application of different approaches for computer-aided optimization, that in turn rely on the simulation of mathematical models for the prediction of how the process in question would behave under certain conditions. As we have seen, there exist procedures that can be very useful when performing an optimization study, including procedures for creating mathematical models representing the process as well as simulation strategies to solve these models, which have been assembled in a provided integrated procedure. There is also the interplay that needs to be considered between the models, their simulation strategies, and the optimization approaches, which therefore need to be chosen and/or developed keeping each in mind; this is a gap that can successfully be bridged by using the provided framework and taking a purpose-centric view on the optimization study at hand. As such, the purposes of this thesis have been achieved.

Furthermore, by taking a complex real-life process, breaking it down and simplifying through abstraction, similar optimization methods have been applied to cases that used different modeling and simulation tools. The work presented in this thesis thus shows that the level of abstraction facilitated by the purpose-centric framework is very useful combined with the integrated procedure for solving optimization problems within different fields of research in collaboration with various research teams and companies. As such, an extra conclusion is that abstraction beats complexity; making (not keeping!) things simple makes possible studies that can in turn enable us to make better decisions in order to improve any conceivable situation across fields of research, business, and interest.

## 5.1 Future work

The framework and integrated procedure at the level of abstraction provided in this thesis should be considered the kind of tools that can always be improved, and will thus benefit from further work. If these tools are to be beneficial and useful to practitioners, it is important to ensure they are (and become increasingly) flexible in terms of being

general yet specific enough, clear to understand, and easy to use. Essential factors for the purpose of improving the tools include dissemination, application, and feedback as well as contributions by practitioners.

In terms of future work for the purification of a polyalcohol at Perstorp AB, the modeling work that was described in Paper V should be continued in order to build and finish a digital twin of the full plant. The methods used may also be complemented by utilizing the huge amounts of data generated by the operation of the plant, where machine learning techniques may for instance be applied for time-series forecasting of the plant behavior. However, any optimization study undertaken to improve the plant does not amount to anything if not properly implemented. It is therefore important to continue the work on the technical issues surrounding practical implementation and automation, using appropriate methods available in literature for the purpose of controlling the plant and keeping it at optimal operating conditions in real time. However, it is important to remember that the social perspective matters as well. For instance, implementing the results from a computer-aided optimization study in the real-life process may come down to the willingness of the process operator to diverge from truly tested operating conditions that empirically always work. Similarly, it is important to rely on and learn from those with the most knowledge and experience of the process, which is gathered on an everyday basis. Thus, learning and change intersects with the operation of a production plant, and here the many levels of process systems engineering and management meet in a cross-section containing a vast number of interesting problems to study.

Within the field of chromatographic purification of pharmaceuticals, there has already been some very interesting work regarding process control, as well as integrated and continuous processes. However, much more remains at different levels of implementation for a number of applications. For instance, optimizing the flow-rate during the loading phase of the capture step of mAb purification would be interesting. Furthermore, developing processes that can automatically find optimal operating conditions (either based on first principles or data-based models) would be a great step towards the automation that may be necessary to realize the potential residing in integrated and continuous processes. However, this also demands an entirely new view on how to build processes at different steps in clinical trials, which may also warrant scrutiny, since it is easier to scale up from already integrated and continuous processes than to file with authorities for approval of a new operating point.

Regarding the work with urban flooding mitigation using blue-green systems presented in this thesis, there are several areas that can be developed further. For instance, the model may be extended with infiltration and evaporation in order to shift the functionality of the model from event-based to be able to predict flow dynamics across longer periods of time. This would most probably also warrant a revision of the simulation strategy currently in place, which may benefit from variable-step methods if longer periods are to be simulated. The effectiveness of blue-green systems to deal with extreme rainfalls has been qualitatively studied for individual implementations up to city-scale. However, a methodology that also considers the existing urban infrastructure needs to be developed that can be used to determine how these systems should be implemented on a larger scale. Particularly interesting would be to incorporate changes to the urban infrastructure over time in order to investigate any potential impact thereof.

# Bibliography


Andersson, C. (2016). *Methods and Tools for Co-Simulation of Dynamic Systems with the Functional Mock-up Interface*. PhD thesis, Lund University.

Andersson, N. (2014). *Parallel computing in model-based process engineering*. PhD thesis, Lund University.

Andersson, N., Åkesson, J., Link, K., Yances, S. G., Dietl, K., and Nilsson, B. (2014). Parameter selection in a Combined Cycle Power Plant. *Proceedings of the 10th International Modelica Conference*, pages 809–817.

Asadollahi, M. and Naevdal, G. (2010). Selection of Decision Variables for Large-Scale Production Optimization Problems Applied to Brugge Field. *Society of Petroleum Engineers Russian Oil and Gas Conference and Exhibition 2010*.

Aumann, L. and Morbidelli, M. (2007). A continuous multicolumn countercurrent solvent gradient purification (MCSGP) process. *Biotechnology and Bioengineering*, 98(5):1043–1055.

Biegler, L. T. (2010). *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Biegler, L. T. and Grossmann, I. E. (2004). Retrospective on optimization. *Computers and Chemical Engineering*, 28(8):1169–1192.

Brady, T. F. and Yellig, E. (2005). Simulation Data Mining: A new Form of Computer Simulation Output. *Proceedings of the Winter Simulation Conference 2005*, pages 285–289.

Bulian, G. and Cercos-Pita, J. L. (2018). Co-simulation of ship motions and sloshing in tanks. *Ocean Engineering*, 152:353–376.

Cao, Y. and Rossiter, D. (1997). An input pre-screening technique for control structure selection. *Computers and Chemical Engineering*, 21(6):563–569.

Cervantes, A. and Biegler, L. T. (2008). Optimization Strategies for Dynamic Systems. In Floudas, C. and Pardalos, P., editors, *Encyclopedia of Optimization*, pages 2847–2857. Springer, Boston, MA (US).

Chibeles-Martins, N., Pinto-Varela, T., Barbosa-Póvoa, A. P., and Novais, A. Q. (2016). A multi-objective meta-heuristic approach for the design and planning of green supply chains - MBSA. *Expert Systems with Applications*, 47:71–84.

Cintron-Arias, A., Banks, H. T., Capaldi, A., and Lloyd, A. L. (2009). A Sensitivity Matrix Methodology for Inverse Problem Formulation. *Journal of Inverse and Ill-Posed Problems*, pages 545–564.

Cunha, M. C., Zeferino, J. A., Simões, N. E., and Saldarriaga, J. G. (2016). Optimal location and sizing of storage units in a drainage system. *Environmental Modelling and Software*, 83:155–166.

Das, I. and Dennis, J. E. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69.

De Godoy, R. J. C. and Garcia, C. (2017). Plantwide Control: A Review of Design Techniques, Benchmarks, and Challenges. *Industrial and Engineering Chemistry Research*, 56(28):7877–7887.

Deb, K. (2009). *Multi-Objective Optimizaiton using Evolutionary Algorithms*. Wiley, Chichester, 2nd edition.

Degerman, M. (2009). *Design of Robust Preparative Chromatography*. PhD thesis, Lund University.

Désidéri, J. A. (2012). Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318.

Diab, S. and Gerogiorgis, D. I. (2018). Process modelling, simulation and technoeconomic optimisation for continuous pharmaceutical manufacturing of (S)-warfarin. In *Computer Aided Chemical Engineering*, volume 43, pages 1643–1648. Elsevier Masson SAS.

Díaz-Trujillo, L. A. and Nápoles-Rivera, F. (2019). Optimization of biogas supply chain in Mexico considering economic and environmental aspects. *Renewable Energy*, 139:1227–1240.

Dimian, A. C., Bildea, C. S., and Kiss, A. A. (2014). Chapter 19 - Economic Evaluation of Projects. In Dimian, A. C., Bildea, C. S., and Kiss, A. A., editors, *Integrated Design and Simulation of Chemical Processes*, volume 35 of *Computer Aided Chemical Engineering*, pages 717–755. Elsevier.

Dutta, S. (2016). *Optimization in Chemical Engineering*. Cambridge University Press, Cambridge, 1st edition.

Eckart, J., Sieker, H., Vairavamoorthy, K., and Alsharif, K. (2012). Flexible design of urban drainage systems: Demand led research for Hamburg-Wilhelmsburg. *Reviews in Environmental Science and Biotechnology*, 11(1):5–10.

Eckart, K., McPhee, Z., and Bolisetti, T. (2018). Multiobjective optimization of low impact development stormwater controls. *Journal of Hydrology*, 562(April):564–576.

Edgar, T. F. and Himmelblau, D. M. (1989). *Optimization of Chemical Processes*. McGraw-Hill, Singapore.

Emmerich, M. T. and Deutz, A. H. (2018). A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17(3):585–609.

Gomis-Fons, J., Löfgren, A., Andersson, N., Nilsson, B., Berghard, L., and Wood, S. (2019). Integration of a complete downstream process for the automated lab-scale production of a recombinant protein. *Journal of Biotechnology*, 301(April):45–51.

Grossmann, I. E. and Harjunkoski, I. (2019). Process Systems Engineering: Academic and Industrial Perspectives. *Computers and Chemical Engineering*, 126:474–484.

Guiochon, G. (2006). The limits of the separation power of unidimensional column liquid chromatography. *Journal of Chromatography A*, 1126(1-2):6–49.

Haghighatafshar, S. (2019). *Blue-green stormwater systems for citywide flood mitigation: Monitoring, conceptualization, modeling, and evaluation*. PhD thesis, Lund University.

Haghighatafshar, S., Jansen, J. l. C., Aspegren, H., and Jönsson, K. (2018a). Conceptualization and schematization of mesoscale sustainable drainage systems: A full-scale study. *Water (Switzerland)*, 10(8):1–16.

Haghighatafshar, S., Nordlöf, B., Roldin, M., Gustafsson, L.-G., la Cour Jansen, J., and Jönsson, K. (2018b). Efficiency of blue-green stormwater retrofits for flood mitigation - Conclusions drawn from a case study in Malmö, Sweden. *Journal of Environmental Management*, 207:60–69.

Haghighatafshar, S., Yamanee-Nolin, M., Klinting, A., Roldin, M., Gustafsson, L.-G., Aspegren, H., and Jönsson, K. (2019a). Hydroeconomic optimization of mesoscale blue-green stormwater systems at the city level. *Journal of Hydrology*, 578.

Haghighatafshar, S., Yamanee-Nolin, M., and Larson, M. (2019b). A physically based model for mesoscale SuDS – an alternative to large-scale urban drainage simulations. *Journal of Environmental Management*, 240:527–536.

Hangos, K. M. and Cameron, I. T. (2001). *Process Modelling and Model Analysis*. Academic Press, London, UK.

Hekmat, D., Mornhinweg, R., Bloch, G., Sun, Y., Jeanty, P., Neubert, M., and Weuster-Botz, D. (2011). Macroscopic investigation of the transient hydrodynamic memory behavior of preparative packed chromatography beds. *Journal of Chromatography A*, 1218(7):944–950.

Hoang, L. and Fenner, R. A. (2016). System interactions of stormwater management using sustainable urban drainage systems and green infrastructure. *Urban Water Journal*, 13(7):739–758.

Holmqvist, A. (2013). *Model-based Analysis and Design of Atomic Layer Deposition Processes*. Phd thesis, Lund University.

Jäschke, J., Cao, Y., and Kariwala, V. (2017). Self-optimizing control - A survey. *Annual Reviews in Control*, 43:199–223.

Javaloyes-Antón, J., Ruiz-Femenia, R., and Caballero, J. A. (2013). Rigorous design of complex distillation columns using process simulators and the particle swarm optimization algorithm. *Industrial and Engineering Chemistry Research*, 52(44):15621–15634.

Jolevski, D., Bego, O., and Sarajcev, P. (2017). Control structure design and dynamics modelling of the organic Rankine cycle system. *Energy*, 121:193–204.

Kariwala, V. and Cao, Y. (2010). Branch and bound method for multiobjective pairing selection. *Automatica*, 46(5):932–936.

Khan, H. A. and Tiwari, M. K. (2017). NSGA-II, https://github.com/haris989/NSGA-II, date accessed: 2020-01-20.

Knutson, H.-K. (2016). *Robust Multi-objective Optimization of Rare Earth Element Chromatography*. PhD thesis, Lund University.

Kraft, D. (1988). A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*.

Kurtz, L. A., Smith, R. E., Parks, C. L., and Boney, L. R. (1978). A comparison of the method of lines to finite difference techniques in solving time-dependent partial differential equations. *Computers and Fluids*, 6(2):49–70.

Larsson, P. O., Åkesson, J., Carlsson, N., and Andersson, N. (2012). Model-based optimization of economical grade changes for the Borealis Borstar ® polyethylene plant. *Computers and Chemical Engineering*, 46:153–166.

Leonzio, G., Bogle, D., Foscolo, P. U., and Zondervan, E. (2020). Optimization of CCUS supply chains in the UK: A strategic role for emissions reduction. *Chemical Engineering Research and Design*, 155:211–228.

Lin, Z., Wang, J., Nikolakis, V., and Ierapetritou, M. (2017). Process flowsheet optimization of chemicals production from biomass derived glucose solutions. *Computers & Chemical Engineering*, 102:258–267.

Lindholm, A. and Giselsson, P. (2012). Formulating an optimization problem for minimization of losses due to utilities. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, volume 8, pages 567–572. IFAC.

Löfgren, A., Yamanee-Nolin, M., Tallvod, S., Fons, J. G., Andersson, N., and Nilsson, B. (2019). Optimization of integrated chromatography sequences for purification of biopharmaceuticals. *Biotechnology Progress*, 35(6).

Maciejowski, J. M. (2002). *Predictive Control with Constraints*. Prentice Hall.

Magnusson, F. (2016). *Numerical and Symbolic Methods for Dynamic Optimization*. PhD thesis, Lund University.

Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395.

Maurer, G. (1986). Vapor-liquid equilibrium of formaldehyde-and water-containing multicomponent mixtures. *AIChE Journal*, 32(6):932–948.

Meyer, K., Huusom, J. K., and Abildskov, J. (2018). Efficient Implicit-Explicit Time Stepping for Accurate and Rapid Simulation of Chromatographic Models. In *Computer Aided Chemical Engineering*, volume 44, pages 853–858. Elsevier.

Mikkonen, H., Lappalainen, J., Pikkarainen, T., and Kuivalainen, R. (2017). Modelling and Dynamic Simulation of the 2nd Generation Oxy Fired Power Plant - Oxidant Fan Failure Case. *Energy Procedia*, 114:561–572.

Negrellos-Ortiz, I., Flores-Tlacuahuac, A., and Gutiérrez-Limón, M. A. (2016). Product Dynamic Transitions Using a Derivative-Free Optimization Trust-Region Approach. *Industrial & Engineering Chemistry Research*, 55(31):8586–8601.

Negrellos-Ortiz, I., Flores-Tlacuahuac, A., and Gutiérrez-Limón, M. A. (2018). Dynamic optimization of a cryogenic air separation unit using a derivative-free optimization approach. *Computers & Chemical Engineering*, 109:1–8.

Nelder, J. and Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, 7:308–313.

Nilsson, B. (1993). *Object-Oriented Modeling of Chemical Processes*. PhD thesis, Lund University.

Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2017). Unbiased Selection of Decision Variables for Optimization. In Espuña, A., Graells, M., and Puigjaner, L., editors, *27th European Symposium on Computer Aided Process Engineering*, volume 40 of *Computer Aided Chemical Engineering*, pages 253–258. Elsevier.

Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2018). Analysis of an Oscillating Two-Stage Evaporator System through Modelling and Simulation: an Industrial Case Study. *Chemical Engineering Transactions*, 69:481–486.

Novak Pintarič, Z. and Kravanja, Z. (2015). The importance of proper economic criteria and process modeling for single- and multi-objective optimizations. *Computers and Chemical Engineering*, 83:35–47.

Pedersen, S. and Wik, T. (2020). A comparison of topologies in recirculating aquaculture systems using simulation and optimization. *Aquacultural Engineering*, 89(February):102059.

Perez-Galvan, C. and Bogle, I. D. L. (2015). Deterministic Global Dynamic Optimisation using Interval Analysis. *Computer Aided Chemical Engineering*, 37(June):791–796.

Powell, M. J. D. (1994). A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. In Gomez, S. and Hennart, J.-P., editors, *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer, Dordrecht.

Powell, M. J. D. (2009). The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, pages 26–46.

Ramirez, W. F. (1997). *Computational Methods for Computer Simulation*. Butterworth-Heinemann, Oxford, 2nd edition.

Rasmuson, A., Andersson, B., Olsson, L., and Andersson, R. (2014). *Mathematical Modeling in Chemical Engineering*. Cambridge University Press, Cambridge.

Rios, L. M. and Sahinidis, N. V. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293.

Rodrigues, A., Pereira, C., Minceva, M., Pais, L., Ribeiro, A., Ribeiro, A., Silva, M., Graça, N., Santos, J., Rodrigues, A., Pereira, C., Minceva, M., Pais, L., Ribeiro, A., Ribeiro, A., Silva, M., Graça, N., and Santos, J. (2015). Principles of Simulated Moving Bed. *Simulated Moving Bed Technology*, pages 1–30.

Rodrigues, D. and Bonvin, D. (2018). Parsimonious Input Parameterization for Dynamic Optimization Problems. In *Computer Aided Chemical Engineering*, volume 44, pages 769–774. Elsevier.

Salim, W. W. A. W. and Diefes-Dux, H. A. (2012). Problem Formulation within Open-ended Problems: Looking through the Structure-Behavior-Function (SBF) and Novice-Expert (NE) Frameworks. *Procedia - Social and Behavioral Sciences*, 56(Ictlhe 2012):160–174.

Santos, P., Pitarch, J. L., Vicente, A., de Prada, C., and García, Á. (2020). Improving operation in an industrial MDF flash dryer through physics-based NMPC. *Control Engineering Practice*, 94(September 2019):104213.

Schmidt-Traub, H., Schulte, M., and Seidel-Morgenstern, A. (2012). *Preparative Chromatography*. Wiley-VCH, Weinheim, 2nd edition.

Sellberg, A. (2018). *Open-Loop Optimal Control of Chromatographic Separation Processes*. PhD thesis, Department of Chemical Engineering, Lund University.

Sellberg, A., Holmqvist, A., Magnusson, F., Andersson, C., and Nilsson, B. (2017). Discretized multi-level elution trajectory: A proof-of-concept demonstration. *Journal of Chromatography A*, 1481:73–81.

Sellberg, A., Nolin, M., Löfgren, A., Andersson, N., and Nilsson, B. (2018). *Multi-flowrate Optimization of the Loading Phase of a Preparative Chromatographic Separation*, volume 43 of *Computer Aided Chemical Engineering*, pages 1619–1624. Elsevier B.V.

Sindareh-Esfahani, P., Tabatabaei, S. S., and Pieper, J. K. (2017). Model predictive control of a heat recovery steam generator during cold start-up operation using piecewise linear models. *Applied Thermal Engineering*, 119:516–529.

Skogestad, S., Halvorsen, I. J., Larsson, T., and Govatsmark, M. S. (1999). Plantwide control: the search for the self-optimizing control structure. *IFAC Proceedings Volumes*, 32(2):6932–6937.

Sörensen, J. and Emilsson, T. (2019). Evaluating flood risk reduction by urban blue-green infrastructure using insurance data. *Journal of Water Resources Planning and Management*, 145(2):1–11.

Tanozzi, F., Sharma, S., Maréchal, F., and Desideri, U. (2019). 3D design and optimization of heat exchanger network for solid oxide fuel cell-gas turbine in hybrid electric vehicles. *Applied Thermal Engineering*, 163(August):114310.

Thaysen, M. (2006). *Hybrid Modelling for Enhanced Biorector Performance*. PhD thesis, Technical University of Denmark (DTU).

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-searchalgorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Yamanee-Nolin, M. (2019). Modularization, co-simulation, and optimization of complex dynamic production processes in the ProOpt project. Technical report, Department of Chemical Engineering, Faculty of Engineering, Lund University, P.O. Box 124 SE-22100 LUND.

Yamanee-Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2020). Trajectory optimization of an oscillating industrial two-stage evaporator utilizing a Python-Aspen Plus Dynamics toolchain. *Chemical Engineering Research and Design*, 155:12–17.

Yamanee-Nolin, M., Löfgren, A., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2019). Single-shooting optimization of an industrial process through co-simulation of a modularized Aspen Plus Dynamics model. In Kiss, A. A., Zondervan, E., Lakerveld, R., and Özkan, L., editors, *29th European Symposium on Computer Aided Process Engineering*, volume 46 of *Computer Aided Chemical Engineering*, pages 721–726. Elsevier.

Yang, X.-S. (2017). Chapter 23 - Optimization. In Yang, X.-S., editor, *Engineering Mathematics with Examples and Applications*, pages 267–283. Academic Press, Cambridge, MA (US).

Yelchuru, R. and Skogestad, S. (2012). Convex formulations for optimal selection of controlled variables and measurements using Mixed Integer Quadratic Programming. *Journal of Process Control*, 22(6):995–1007.

Zhao, F., Chen, X., and Zhu, L. (2016). Simplification of Equation-Oriented Models through the Digraph Method. In Kravanja, Z. and Bogataj, M., editors, *26th European Symposium on Computer Aided Process Engineering*, volume 38 of *Computer Aided Chemical Engineering*, pages 973–978. Elsevier.

Zhou, Q., Lai, Z., and Blohm, A. (2019). Optimising the combination strategies for pipe and infiltration-based low impact development measures using a multiobjective evolution approach. *Journal of Flood Risk Management*, 12(2):1–14.

# Paper I

Antonio Espuña, Moisès Graells and Luis Puigjaner (Editors), Proceedings of the 27th European Symposium on Computer Aided Process Engineering – ESCAPE 27
October 1st - 5th, 2017, Barcelona, Spain 
http://dx.doi.org/10.1016/B978-0-444-63965-3.50044-1

# Unbiased Selection of Decision Variables for Optimization

Mikael Nolin[a*], Niklas Andersson[a], Bernt Nilsson[a], Mark Max-Hansen[b], Oleg Pajalic[b]

[a] *Lund University, Getingevägen 60, 221 00 Lund, Sweden*
[b] *Perstorp AB, Industriparken, 284 80 Perstorp, Sweden*
mikael.nolin@chemeng.lth.se

## Abstract

Complex chemical processes require complex simulation models. Selecting decision variables for optimization is increasingly difficult. This paper presents a study of a Subset Selection Algorithm (SSA) applied to the selection of decision variables to facilitate a reduction of the decision variable combination sets to consider for a process designer, aimed towards improving said selection, optimization, and thereby resource efficiency. The results help conclude that SSA is able to reduce the consideration set of decision variable combinations for the process designer, and selects combination sets that are more effective in terms of minimizing the objective.

**Keywords**: Subset Selection Algorithm, decision variables, optimization.

## 1. Introduction

In order develop a sustainable and competitive process industry, the efficient use of resources is key in combination with developing process control, modelling, and simulation (Process IT Innovation and Automation, 2012). The present study aims to contribute to these areas through a case study of a trimethylolpropane (TMP) separation stage at Perstorp, Sweden, as a method for subset selection has been applied for the mathematical and unbiased selection of decision variables for optimization. The method would serve an important purpose since decision variables are traditionally chosen through the intuition and experience of the responsible process designer, arguably limiting the quality of the solution as models and simulations grow in complexity (Brady & Yellig, 2005). However, in recent years, different methods have been developed which could be useful tools in decision variable selection/handling (Asadollahi & Naevdal, 2010; Zhao et al., 2016). Especially interesting to this study has been the work Cintrón-Arias et al. (2009) whose Subset Selection Algorithm (SSA) has in the present work been implemented in Python to work with an Aspen Plus Dynamics model and applied to aid in the selection of decision variables, which have then been used in optimization runs.

## 2. Theory

### *2.1. General*

The SSA proposed by Cintrón-Arias et al. (2009) can, provided a sensitivity matrix J, calculate two key numerical quantities: $\alpha$, the selection score, and $\kappa$, the condition number. These indicate the accuracy of the optimization solution and the condition of the

problem posed by the chosen subset, respectively, based on the analogous reasoning by Andersson et al (2014) who also introduced Θ, the SSA score. α is calculated as the norm of the square roots of the diagonal elements of the inverse of the $J^TJ$ matrix, whilst κ is calculated as the condition number of J, defined as the ratio of the largest ($s_l$) and smallest ($s_p$) singular values of J. Θ is then the sum of the common logarithms of the two quantities.

$$\alpha = \left| \frac{1}{\sigma} \sqrt{(J^T J)^{-1}_{kk}} \right| \qquad \text{for} \quad k = 1, \dots, n_D \tag{1}$$

$$\kappa = \frac{s_l}{s_p} \tag{2}$$

$$\Theta = \log_{10}(\alpha) + \log_{10}(\kappa) \tag{3}$$

In equation 1, $n_D$ refers to the number of decision variables of the data set. A low value of κ indicates a well-conditioned problem, and a low value of α indicates high accuracy to be expected of the solution, i.e. the result from the optimization runs should be reliable.

## 3. Methods

### *3.1. Evaporator System Model in Aspen Plus Dynamics*

The TMP evaporator system model was built in Aspen Plus v8.8 using the software's standard blocks. It was then converted into an Aspen Plus Dynamics v8.8 model through the flow driven mode, without controllers, after which it was tuned to real process data. An overview of the process is presented in figure 1 below.

The system contained formaldehyde, which motivated the use of AspenTech's formaldehyde/water/methanol system property and chemistry data package from aspenONE Exchange, with SYSOP7K as property method and NRTL as base method. A simplification was made in that the data package's reaction set was reduced to only the first four reactions, which include the methylene glycol reaction as well as polyoxymethylene reactions for n ≤ 4 as presented by Maurer (1986).

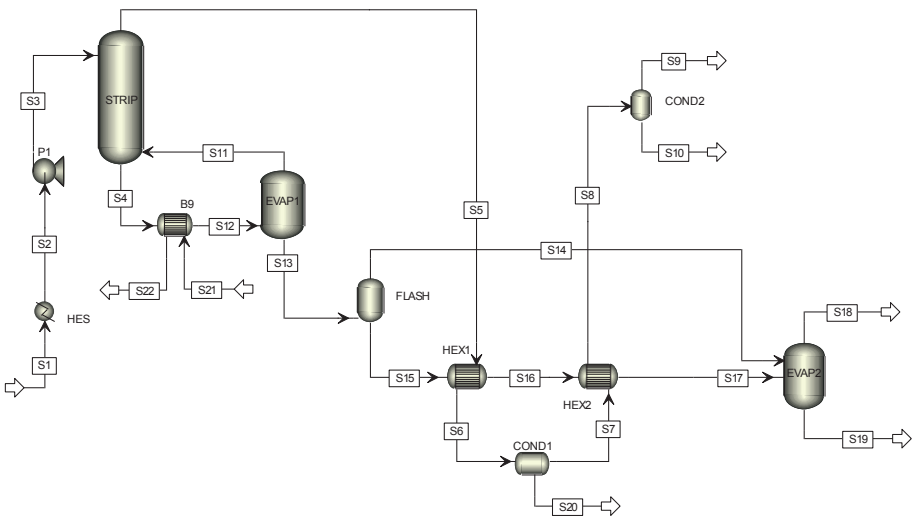



Figure 1: The flowsheet as built in Aspen Plus using standard blocks.

### *3.2. Subset Selection Algorithm in Python*

SSA was implemented in Python, and the information flow is presented in figure 2. This starts with the user supplying the necessary input, such as a specification of the decision variables and the objective to be studied, as well as a COM link to the model itself. In this case, 14 different variables were investigated, presented in table 1. These were chosen as they were 'fixed' in Aspen Dynamics as well as possible decision variables in reality. The algorithm will then run the model repeatedly, perturbing each variable in order to get the sensitivity matrix J through central finite differences approximation, essentially retrieving a Jacobian. After the sensitivity matrix is retrieved, the algorithm will create every possible subset of decision variables and objectives through combination in order to evaluate these subsets by calculating α, κ, and Θ. The combination sets are ranked by lowest Θ and presented to the user in tables and figures, to facilitate the selection of decision variables for further investigation.

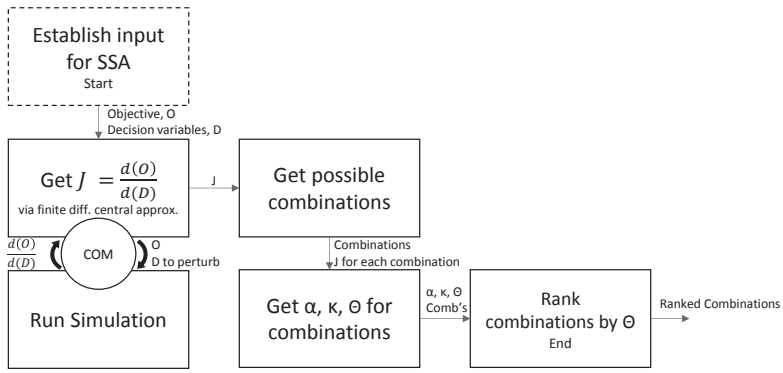



Figure 2: A schematic of the information flow for the implementation of SSA. The dashed starting box represents a phase where user input is needed.

Component Object Model (COM) – a platform-independent and object-oriented system – was used to enable communication between Aspen Plus Dynamics and Python. The Python extension win32com was used, but other methods supported by Python and the simulation software should be applicable, which makes the method generalizable.

### *3.3. Optimization*

For the optimization study, the objective was a scalarized multi-objective that signifies the direct variable costs using the objective parameters from the SSA analysis. This was expressed as in equation 4, as a weighted difference of the simulations' objective parameter time integrals from the SSA – the evaporation duty, Q, which should be low, and the product stream purity, $X_{prod}$, which should be high.

Table 1: The fixed variables in Aspen used in SSA. The labels refer to those in figure 1.

| **Flow (F)** | S4 | S5 | S7 | S9 | S11 | S13 |
|---|---|---|---|---|---|---|
| | S14 | S15 | S18 | S19 | S20 | S21 |
| **Pressure (P)** | S1 | S21 | | | | |

$$min.\ F_{obj} = -\left((1-w)\cdot\frac{\int_{t=0}^{t_{sim}} X_{prod}\ dt}{\int_{t=0}^{t_{sim}} X_{prod,nom}\ dt} + w\cdot\frac{-\int_{t=0}^{t_{sim}} Q\ dt}{\int_{t=0}^{t_{sim}} Q_{nom}\ dt}\right) \quad (4)$$

In equation 4, *w* is a scalar weight set to 0.7 to reflect on-site economics. The objective function is subject to physical constraints and bounds in Aspen, and is scaled with the nominal operation point (subscript nom). The decision variables used for each optimization run were the top 10 variable combinations as suggested by the SSA combination level 1-5 (i.e. number of decision variables in the combination). Optimization runs were performed using the Python package SciPy's implementation of the Nelder-Mead algorithm, with the nominal operating point as guess point

## 4. Results and Discussion

### *4.1. Model Tuning & Accuracy*

The Aspen Plus Dynamics model was tuned and the model output was, where possible, compared to the plant's measured data at a nominal operating point. The validation results are presented in figure 3 below, normalized with respect to the measured data. As can be seen, the model is accurate within ± 3 %, accepted as accurate enough for the purpose of this study. Please note that these validation variables are not the same as the decision variables presented in table 1.

### *4.2. Subset Selection Algorithm*

The SSA evaluates all combination sets in a matter of seconds and presents the user with the best selection, ranked by the lowest Θ. In figure 4 below, the results from the SSA are presented as the common logarithm of α and κ plotted against each other for the "top 10" combinations in terms of lowest Θ per combination level 2-5. Low values are desirable for both α and κ, giving low values to Θ as well. For combination level 1, i.e. only using one decision variable, the common logarithm of κ was as expected always 0 and the common logarithm of α varied between [-0.237; 3.12]. An interesting trend of a concentration of the sets with an increased number of decision variables in a set is apparent from figure 4, which would indicate increasing similarities of the sets. This also seems to be the case

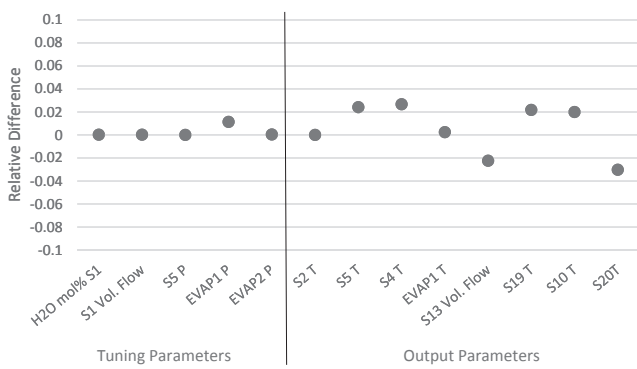



Figure 3: Measurements compared to the model outputs – the model is accurate within ±3 %. Horizontal axis labels refer to figure 1, with P for pressure, and T for temperature.

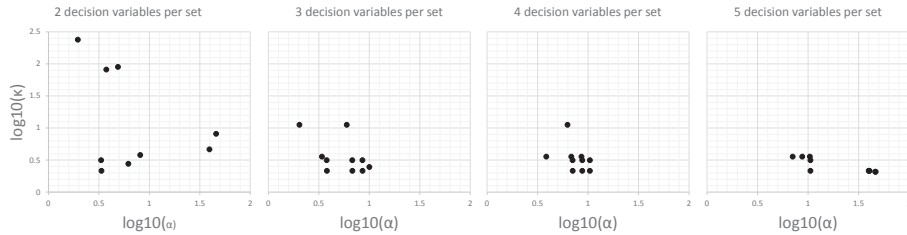



Figure 4: The common logarithms of α and κ for the top 10 combinations in combination levels 2-5. • denotes one combination set.

upon inspection of the suggested combination sets, since the top sets selected by SSA often feature the same, prominent decision variables across the levels. The frequency of the decision variables in the combination sets on levels 1-5 is presented in table 2 below. What can be seen is that decision variables close (in terms of where they are found in the process model) to the objective function variables are selected more often, due to their lower Θ scores. Evident as well from table 2 is that some decision variables are rarely (or never) selected. These particular variables are often those that logically should have a minimal effect on the process – e.g. S9 flow rate – which indicates that the SSA filters out such variables, and highlights more effective variables.

Using SSA for the first five combination levels has in this case narrowed down the amount to consider from the 3472 combination sets to 50 combination sets – ten per combination level. This can be expanded or reduced further depending on user requirements and will naturally be helpful for a process designer, especially when designing and optimizing processes that are more complex, leading to an increased number of possible combination sets. However, the SSA is very dependent on the quality of the sensitivity matrix, which potentially could be a weak point of the method.

### *4.3. Optimization*

The results from the optimization runs are presented in figure 5 below. The "top 10" combinations in terms of lowest Θ per combination level 1-5 were used for optimization runs. As can be seen, the optimizations using SSA selected decision variable combinations get lower objective scores than optimization runs using combinations chosen to represent reality as close as possible, by comparing model with schematics of the process. The decision variables the real life combinations consist of are marked (*) in table 2, and the fact that they are mostly very low in the frequency helps explain their lesser performance. This was an unanticipated but interesting effect, which probably has to do with SSA's reliance on a *J* matrix and its design. The effect, though, indicates unforeseen potential with using SSA for finding the most effective decision variable combinations.

Table 2: The frequency of decision variables in the top 10 comb. sets for combination levels 1-5.

| Dec. Var. | Counts |
|---|---|
| S18 F | 39 |
| S11 F | 35 |
| S4 F | 34 |
| S15 F | 27 |
| S13 F* | 25 |

| Dec. Var. | Counts |
|---|---|
| S5 F* | 25 |
| S7 F | 11 |
| S14 F | 9 |
| S21 P | 3 |
| S19 F | 1 |

| Dec. Var. | Counts |
|---|---|
| S21 F | 1 |
| S1 P* | 0 |
| S20 F* | 0 |
| S9 F | 0 |

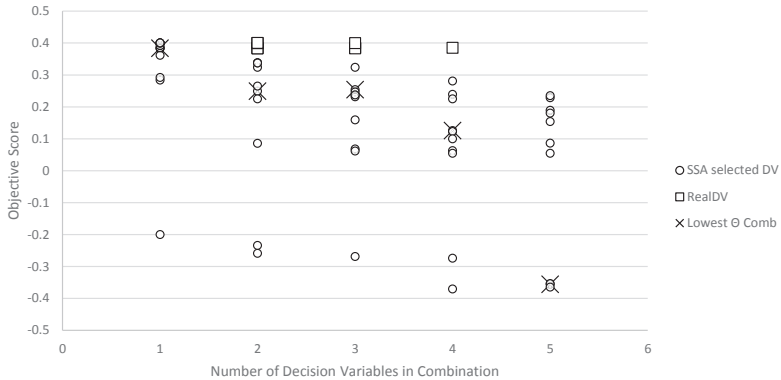



Figure 5: The optimization results using the top 10 SSA decision variables for each combination level compared to the real decision variables results.

## 5. Conclusions

The SSA is able to reduce the number of decision variables to consider, highlighting the most effective variables and filtering out those with minimal effect on the process, facilitating the choice for the responsible process designer. The lower objective scores from the optimization results using SSA selected sets, when compared with optimization runs using combinations not suggested by the SSA but chosen to represent reality, indicate the appropriateness of the suggested decision variables. This is especially interesting since it indicates that the SSA also selects variables depending on their effectiveness in terms of influencing the studied process. Future research may be directed towards establishing this, as well as the reliability of the optimizations using the SSA combination sets.

## References

Andersson, N., Åkesson, J., Link, K., Gallardo Yances, S., Dietl, K., & Nilsson, B. (2014) "Parameter selection in a Combined Cycle Power Plant", *Proceedings of the 10th International ModelicaConference*, pp 809-817.

Asadollahi, M. & Naevdal, G. (2010) "Selection of Decision Variables for Large-Scale Production Optimization Problems Applied to Brugge Field", *Society of Petroleum Engineers Russian Oil and Gas Conference and Exhibition 2010*.

Brady, T.F., & Yellig, E. (2005). "Simulation Data Mining: A new Form of Computer Simulation Output", *Proceedings of the Winter Simulation Conference 2005*, pp. 285-289.

Cintrón-Arias, A., Banks, H.T., Capaldi, A., & Lloyd, A.L. (2009) "A Sensistivity Matrix Methodology for Inverse Problem Formulation", *Journal of Inverse and Ill-Posed Problems*, vol. 17, pp. 545-564.

Maurer, G. (1986) "Vapor-Liquid Equilibrium of Formaldehyde- and Water-Containing Multicomponent Mixtures", *AIChE JOURNAL,* vol. 32, no. 6, pp. 932-948.

Process IT Innovation and Automation (2012). "National pooling of resources for industrial process automation". [pdf] Available Online: http://www.processitinnovations.se/Sve/Nyheter/Filer/national%20pooling_120619.pdf [Accessed 2016 October 26].

Zhao, F., Chen, X., & Zhu, L. (2016) "Simplification of Equation-Oriented Models through the Digraph Method", *Proceedings of ESCAPE26 Part A*, pp. 973-978.

# Paper II

CHEMICAL ENGINEERING TRANSACTIONS
VOL. 69, 2018
A publication of AIDIC
The Italian Association of Chemical Engineering
Online at www.aidic.it/cet
Guest Editors: Elisabetta Brunazzi, Eva Sorensen

**ISBN** 978-88-95608-66-2; **ISSN** 2283-9216
DOI: 10.3303/CET1869081

# Analysis of an Oscillating Two-Stage Evaporator System through Modelling and Simulation: an Industrial Case Study

Mikael Nolin[a]*, Niklas Andersson[a], Bernt Nilsson[a], Mark Max Hansen[b], Oleg Pajalic[b]

[a] Lund University, P.O. Box 124, 221 00 Lund, Sweden
[b] Perstorp AB, Industriparken, 284 80 Perstorp, Sweden
mikael.nolin@chemeng.lth.se

With increasing demands on the industry for resource efficiency, processes are often built or retrofitted with recycle streams in order to decrease energy and raw material demands. However, this also increases the complexity of the process as a whole and may bring unexpected effects. In this contribution, such a case, consisting of a two-stage evaporator system, fed with the product stream of an upstream batch system and a continuous recycling stream from downstream separation processes, was analyzed. This evaporator system was subject to potentially performance-limiting oscillating disturbances. The purpose of this study was to, through modelling and simulation, expand the knowledge of the system by analyzing the system dynamics, and to discover any co-oscillations and their extent in the process, as well as their effect on process parameters such as product purity and the energy usage in terms of steam consumption. The investigation was performed by modelling using Aspen Plus Dynamics. Simulation, data-extraction, and analysis was performed via a COM enabled Python interface. The results of the study highlight the full-system propagation of oscillations along with co-oscillation of selected key parameters, and support the conclusion that there is potential for cost-reductions by decreasing steam consumption by 1.1 %, without any investments.

## 1. Introduction

Evaporator systems are integral parts of many different purification processes, e.g. sugar or kraft pulp production, but exhibit the inherent trait of being energy intense. With globally increasing demands on resource efficiency (Pitarch et al., 2017), recycling of matter and energy may be implemented in order to achieve these demands (Márquez-Rubio et al., 2012). However, even though the multistage evaporator system is common, it is still complex to model and control (Adams et al., 2008), and adding recycling has long been known to risk making the overall system dynamics even more complex – even so if the subsystems are simple (Trierweiler et al., 1998). Thus, understanding the evaporator system dynamics, as well as their interaction with other subsystems in a process is a prerequisite in order to reach the aforementioned demands on resource efficiency.

As such, efforts have been aimed at understanding and improving both the systems, and especially how to improve their degree of process integration (Walmsley et al., 2017). Previous studies have also highlighted the importance of understanding and controlling disturbances in evaporator systems. Adams et al. (2008) designed a new control architecture to solve the suboptimal performance of an evaporator system due to three different cyclic behaviors of the plant, and Pitarch et al. (2017) employed an optimization strategy to improve process control and reduce the effects of disturbances.

The present study was similarly focused on discovering how disturbances propagate and affect an evaporator system, and an industrial case study was performed of a system for polyalcohol purification at Perstorp AB. The investigated system, presented in Figure 1, consists of a two-stage evaporator, including a two-stage mechanical vapor recompression with intercooling, as well as a preceding stripping column. The evaporator system is fed from a balance tank (located between the upstream batch system and the continuous evaporator system) into which a continuous, dilute recycle stream is connected, introduced in order to increase resource-efficiency.

Please cite this article as: Nolin M., Andersson N., Nilsson B., Max-Hansen M., Pajalic O., 2018, Analysis of an oscillating two-stage evaporator system through modelling and simulation: an industrial case study, Chemical Engineering Transactions, 69, 481-486
DOI: 10.3303/CET1869081

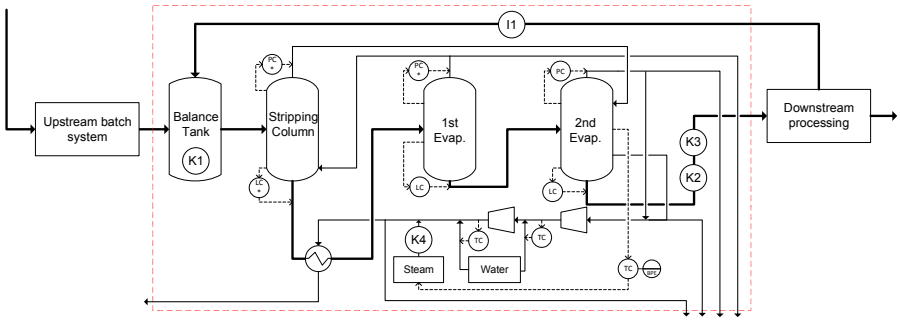



*Figure 1: A schematic overview of the studied evaporator system (in dashed box), including mechanical vapor recompression and a stripping column, and its connections to the upstream batch system and downstream processes.*

As the upstream batch system intermittently fills the balance tank with material high in product mass fraction, the continuous (dilute) recycle stream has a diluting effect on the contents in the balance tank. Investigating this system is particularly interesting since the disturbance is cyclical (potentially limiting the process to a cyclic steady state at best) and therefore of an oscillating nature, which commonly creates performance-worsening propagations throughout entire systems (Yuan & Qin, 2014).
To examine the process, a dynamic model of the evaporator system was created via Aspen Plus Dynamics, using a COM-enabled Python interface for simulation control, data-extraction, and graphical representation of the data to be analyzed. The analysis employs an approach based on the combination of one-factor-at-a-time (OFAT) sensitivity analysis and phase plane analysis, and is thus able to provide an increased understanding of how process parameters in the system co-oscillate, or co-vary.
The purpose of the present contribution was to analyze the dynamics of the system, to discover and highlight important co-oscillating parameters and their effects on the process, especially regarding the swing-effects on product purity and the energy usage (in terms of steam consumption), thereby improving the understanding of the system's dynamic behavior,. The present study was intended as a precursory study and has the potential of serving as a decision basis for further studies regarding e.g. root cause diagnosis and/or process improvements (e.g. in terms of operations or refurbishments) in order to minimize the oscillation propagation throughout the system.

## 2. Theory

For the study, an OFAT sensitivity analysis was combined with a phase plane analysis, where different input variables were manipulated in order to analyze their respective impacts on the phase plane of key process parameters, and to explore if and analyze how the key process parameters co-oscillate. An OFAT sensitivity analysis is the practice of changing one variable whilst keeping the others fixed, and is useful for qualitatively identifying input parameter effects on the output of a model (Delgarm et al., 2018). The OFAT approach to sensitivity analysis is simple to implement and functions well in a close proximity to the nominal case, but it carries the risk of deceiving the analyst who could well be tempted to rank input variables' importance based on their impact on the output (Saltelli, 1999). However, in the present study, the aim is not to make a ranking, but rather to illustrate the effects of varying certain input parameters on certain key parameters. As such, the OFAT approach was chosen for this study.
To visualize and analyze the propagation of the oscillations throughout the system, phase planes of the key process parameters were produced and analyzed according to Bequette's (1998) writings. Instead of examining how a process variable changes in the temporal domain, pairing variables in a phase plane facilitates the analysis of the system's stability. It further makes it easier to understand how variables co-vary. This may in turn increase the understanding of the safety margins regarding product quality, and potentially their required quantity. This would serve as an enabler for removing oscillations and/or minimizing swing-effects, which according to Yuan and Qin (2014) would improve operation of the process and lead to financial gains, e.g. through sustaining product quality demands more frequently at a lower cost.

## 3. Method

To create a dynamic model able to highlight the dynamics of the swings, a steady state model of the evaporator system presented in the dashed box in Figure 1 was first created in Aspen Plus v8.8, using standard blocks. Formaldehyde oligomerization was modeled by applying the Maurer model, which provides high quality representations of the aqueous formaldehyde systems (Ott et al., 2005). Specifically, AspenTech's own implementation of the Maurer model was used, with SYSOP7K as property method with NRTL as base method. This was implemented with small modifications in the same way as in Nolin et al. (2017).
The steady state model was then converted into a flow-driven dynamic model, which made use of PI controllers to reflect the real process, as well as *PtoT* and *Delta* blocks to model boiling point elevation (BPE) control. The controllers denoted by asterisks in Figure 1 were implemented in order to keep the simulation within operationally possible bounds. This functioned as a solution towards qualitatively catching pressure-driven phenomena in a flow-driven simulation, and eliminated issues with unrealistic pressure gains. However, it might have unexpected effects on the simulation results and thus needs to be considered during analysis.
The phase planes were constructed for key process parameters from the sensitivity analysis simulations, for which an input variable had been perturbed individually. The input variables to vary and key process parameters to examine were chosen based on the experience of process engineers working with the process, as well as using the Subset Selection Algorithm (SSA) as applied in Nolin et al. (2017), based on Andersson et al. (2014). Some of the choices and SSA recommendations overlapped, and are presented together with the relative perturbation in table 1 below, with overlapping parameters denoted by ‡. The shorthand names given to the input variable and key parameters, i.e. I1 and K1 through K4, are also mapped in Figure 1.
To control the simulations, and to gather and present the data, a COM-enabled Python interface was used. This utilized the Python extension win32com, but using other methods, e.g. OPC, should be possible to use as well depending on Python and simulation software support, adding generalizability in terms of software. The perturbation was performed by making a positive and a negative step change to the input variable. Data collection was started after the perturbed simulations had reached a cyclic steady state, as it was desired to investigate how the changes affect the cyclic steady state and its position in the phase planes.

*Table 1: Input variables and key process parameters for the sensitivity and phase plane analyses. ‡ denotes overlap between SSA recommendations and process engineer choices.*

| Input variable | Unit | Relative Perturbation [%] | Key process parameters | Unit |
|---|---|---|---|---|
| I1: Recycle stream mass flow | [kg/h] | ± 2.5 | K1: Balance tank product mass fraction | [-] |
| | | | K2: 2nd stage liquid product stream mass flow ‡ | [kg/h] |
| | | | K3: 2nd stage liquid product stream product mass fraction, i.e. product purity | [-] |
| | | | K4: Added steam mass flow, i.e. steam consumption ‡ | [kg/h] |

The temporal domain data that was gathered for each parameter during 24 simulation hours was processed in two different ways before presentation in order to visualize the findings according to Bequette (1998), as well as to visualize the relative movements of the cycles in the different phase planes. The first way meant normalization with respect to its own mean value before the phase planes were produced, and the second way was done through scaling with the variable's nominal run mean value, both of which render the results dimensionless. Furthermore, adding an objective function similar to that in Nolin et al. (2017) allowed for analysis of the system's performance during the cycles. The objective is defined in equation 1 as a function of the discrete temporal element $t_i$, i.e. the $i^{th}$ simulation run sample, where $\overline{K3}_{nom}$ and $\overline{K4}_{nom}$ are the parameters' corresponding mean values for the nominal run, and $w$ is a weight set to 0.7 to reflect process economics.

$$F_{obj}(t_i) = -\left((1-w)\cdot\frac{K3(t_i)}{\overline{K3}_{nom}} + w\cdot\frac{-K4(t_i)}{\overline{K4}_{nom}}\right) \quad (1)$$

## 4. Case study results and discussion

The scaled times series that the analysis is built upon are presented in Figure 2 below. As can be seen, the model response is oscillating for all four parameters, and small changes in I1 may have great effects on steam consumption (K4). However, the way in which the parameters co-oscillate is difficult to discern from Figure 1.

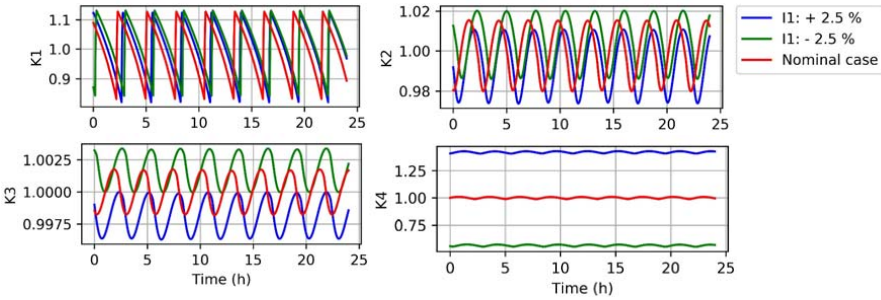



*Figure 2: The scaled time series for K1 (upper left), K2 (upper right), K3 (lower left), and K4 (lower right).*

This is more easily understood from the phase planes. The scaled and normalized phase planes for the balance tank product mass fraction, K1, and the 2nd stage liquid product stream mass flow, K2, when perturbing the recycle stream mass flow, I1, by ± 2.5 %, are presented in Figure 3a and 3b, respectively. As can be seen in both Figures 2 and 3, the balance tank product mass fraction (K1) varies significantly during the cycles, by roughly ±15 percentage points. In Figure 2, the cycle can be seen to start at the lowest point in each oscillation, when the upstream batch system fills the balance tank, rapidly increasing the product mass fraction in the balance tank (as well as the liquid level), which subsequently is lowered during the rest of the cycle as the evaporator system is being fed, and low-concentrated material is recycled to the balance tank. This means that both the liquid level and the product mass fraction in the balance tank is reduced, until the cycle starts anew. In Figure 3, this cycle starts at the curvature's left-most point, and travels to the right along the horizontal axis. The balance tank disturbance is clearly shown to propagate all the way through the very end of the evaporator system by the ±2 percentage points of variation in the 2nd stage liquid stream mass flow, K2, seen on the vertical axis in Figure 3. Inferred from this is the strain put on the PI controls through the system, that are put in place to keep the process in check and at its set point. Furthermore, it can be seen that only small relative changes are created during the input variable perturbation during the sensitivity analysis, but as shown in Figure 3b, decreasing (green) or increasing (blue) the recycle stream mass flow, I1, will lead to a lower balance tank product mass fraction, but in either end of the cycle. Further indicated in Figure 3a is that increasing the recycle stream mass flow will generally yield lower production rates (and vice versa), which is coupled to a higher dilution factor in this case, i.e. there is comparatively more water in the feed that needs to be evaporated in relation to the desired product.

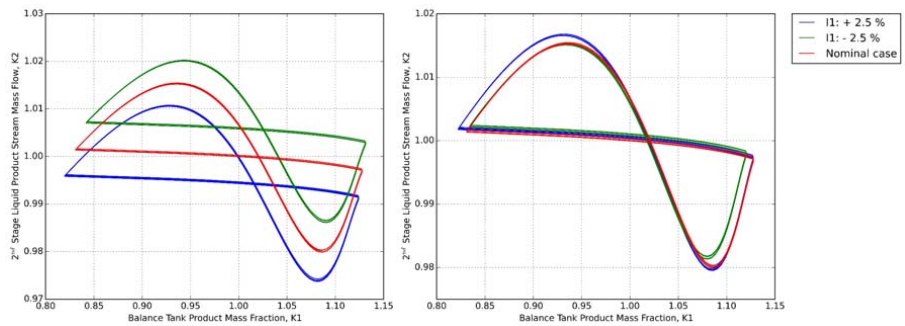



*Figure 3: (a) Scaled and (b) normalized phase planes for the balance tank product mass fraction versus the 2nd stage liquid stream mass flow, for the nominal (red), positively (blue), and negatively (green) perturbed case.*

The suggestion that more water needs to be evaporated when perturbing I1 by +2.5 %, thus requiring more steam, agrees with the results in the scaled phase plane in Figure 4, in which the phase planes are presented for the objective parameters, K3 and K4. In Figure 4a, the positive perturbation of I1 (blue) can be seen to

require more steam than the other two cases. The overall cyclical appearance of these furthermore shows that there is a strong covariation between the two parameters. This variation is most probably primarily caused by the balance tank disturbance visible along the horizontal axis in Figure 3.

The scaled results are shown in Figure 4a, which indicate that even relatively small changes in the recycle stream mass flow may force large adjustments to the added steam mass flow in order to keep the BPE in the second stage at its set point. Whilst the qualitative effect is logical and expected, i.e. an increase in steam consumption with an increase in evaporator feed stream water content and vice versa, the quantity was much greater than expected. This might have been influenced by some unreal behavior the model catches with its use of extra PI controllers for simulation control. Thus, further work will be needed to determine the accuracy, e.g. through big-data analysis or pilot plant experiments. Similarly, the extremely small changes in product purity during a cycle is attributed to the strict control exerted by the model's control of second stage BPE.

Continuing, the model displays a center behavior in the normalized phase plane in Figure 4b, which implies the existence of an equilibrium point. Furthermore, the size increase of the green loop, i.e. for the case of the negative perturbation of -2.5 % to the recycle stream mass flow (I1), shows that the relative difference between peaks and valleys for this run is greater than for the other two. Together with the band widening for the same run, this suggests that this simulation case is relatively difficult to control in a consistent fashion, probably due to other phenomena becoming more pronounced and having a greater effect on the simulation.

The objective score was added to the nominal case phase planes presented above in Figures 3-4 (red), and the results are presented in Figure 5. In Figure 5a, the nominal case 2$^{nd}$ stage liquid stream mass flow (K2) and balance tank product mass fraction (K1) is presented with the objective score.

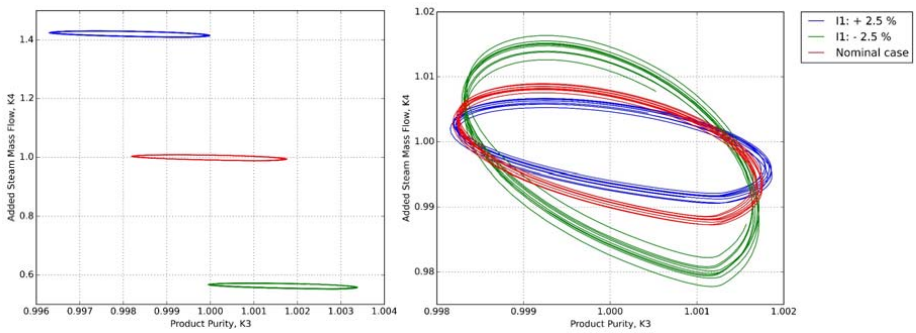



*Figure 4: (a) Scaled and (b) normalized phase planes of the objective parameters for the nominal case (red), the positively (blue), and negatively (green) perturbed cases.*

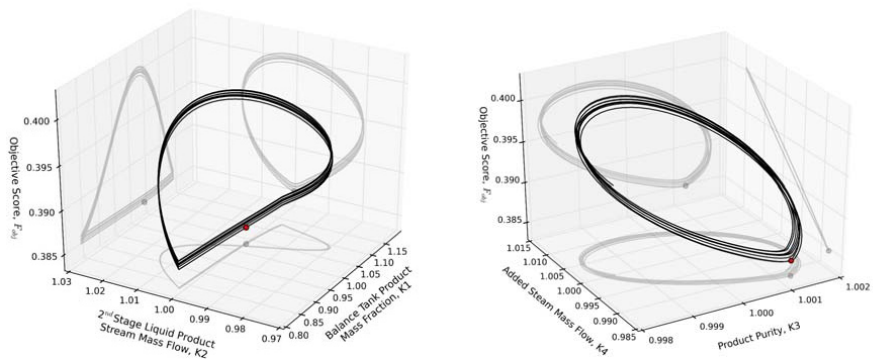



*Figure 5: The objective function with (a) balance tank product mass fraction and 2$^{nd}$ stage mass flow and (b) to the objective parameters. The red dot marks the minimum point of the objective during the cycles.*

As marked by the red dot, the position with the lowest objective score is along the straight line during the batch system emptying sequence, close to the average value for both parameters. This indicates that minimizing the swing effects would not need grand adjustments since the best operating point during a cycle is close to the mean, and would presumably have a stabilizing effect on downstream separation processes, into which oscillations in the 2nd stage liquid stream mass flow would propagate. In Figure 5b, the addition of the objective score to the nominal case cycle is presented for product purity (K3) and added steam mass flow (K4), with a red, filled circle to mark the minimum of the objective during the cycles. This shows a strong dependability on the added steam mass flow, which exerts a larger influence than the extremely small variation in product purity during the cycle. It can further be seen that the best point for the nominal case needs roughly 1.1 % less steam than the average steam usage during the cycles. This fact, together with the small changes in product purity, means that there is potential for significant financial gain (via a lower steam usage) to be found in the way the process is operated today, even without any drastic adjustments or refurbishments. One way to achieve this could be by fixating the mass flow of added steam to the marked point in Figure 5b, which could also have a minimizing effect on the oscillations in the 2nd stage liquid stream mass flow rate, K2, something that may be confirmed through further simulation.
Finally, an important point to reiterate and highlight is that the oscillations that seemingly start in the balance tank, with parameter K1, further propagate throughout the entire flowsheet, to parameter K2. This indicates that identifying and rectifying the actual root cause of the disturbance may have beneficial effects on the performance of the entire system by removing or reducing the oscillations.

## 5. Conclusions & Future Work

The results show that swings in steam consumption are present in the nominal case, but that the product purity varies only very little over a cycle, which indicates that by fixating the added steam mass flow to the optimal point in the current way of operation, steam consumption can be decreased by roughly 1.1 %. This translates into significant cost reductions with virtually costless changes to the process. Furthermore, the quantitative effects of the recycle stream mass flow on the steam consumption need to be properly assessed, in order to determine whether there is any real potential for cost reductions of which to take advantage. Finally, the results of the study support the conclusion that there are co-oscillations for the selected key parameters, and that oscillations extend throughout the evaporator system. This means that finding and rectifying the root cause can be a very important next step in improving process performance, and should be performed in follow-up studies.

### References

Adams G.J., Burke B.J., Goodwin G.C., Gravdahl J.T., Pierce R.D., Rojas A.J., 2008, Managing steam and concentration disturbances in multi-effect evaporators via nonlinear modelling and control, Proceedings of the 17th World Congress of The International Federation of Automatic Control, Seoul, Korea, July 6-11, 2008, 13919-13924.

Andersson N., Åkesson J., Link K., Gallardo Yances S., Dietl K., Nilsson B., 2014, Parameter selection in a Combined Cycle Power Plant, Proceedings of the 10th International Modelica Conference, pp 809-817.

Bequette B. W., 1998, Process dynamics: modeling, analysis, and simulation, Prentice Hall PTR, New Jersey, US.

Delgarm N., Sajadi B., Azarbad K., Delgarm S., 2018, Sensitivity analysis of building energy performance: A simulation-based approach using OFAT and variance-based sensitivity analysis methods, Journal of Building Engineering, 15, 181-193.

Márquez-Rubio J.F., del Muro-Cuéllar B., Velasco-Villa M., Cortés-Rodríguez D., Sename O., 2012, Control of delayed recycling systems with unstable first order forward loop, Journal of Process Control, Volume 22, 729-737.

Nolin M., Andersson N., Nilsson B., Max-Hansen M., Pajalic O., 2017, Unbiased Selection of Decision Variables for Optimization, Editor(s): Antonio Espuña, Moisès Graells, Luis Puigjaner, In Computer Aided Chemical Engineering, Elsevier, 40, 253-258, ISSN 1570-7946, ISBN 9780444639653, DOI:10.1016/B978-0-444-63965-3.50044-1.

Ott M., Schoenmakers H., Hasse H., 2005, Distillation of formaldehyde containing mixtures: laboratory experiments, equilibrium stage modeling and simulation, Chemical Engineering and Processing: Process Intensification, 44, 6, 687-694.

Pitarch J.L., Palacín C.G., De Prada C., Voglauer B., Seyfriedsberger, G., 2017, Optimisation of the resource efficiency in an industrial evaporation system, Journal of Process Control, 56, 1-12.

Saltelli A., 1999, Sensitivity analysis: Could better methods be used?, Journal of Geophysical Research, 104, 3789-3793.

Trierweiler J.O., Schulte B., Engell S., 1998, Dynamics and control of a process with recycle streams, Journal of Process Control, 8, 507-515.

Walmsley T.G., Atkins M.J., Ong B.H.Y., Klemeš J.J., Walmsley M.R.W, Varbanov P.S., 2017, Total Site Heat Integration of Multi-Effect Evaporators with Vapour Recompression for Older Kraft Mills, Chemical Engineering Transaction, 61, 265-270, DOI:10.3303/CET1761042

Yuan T., Qin S.J., 2014, Root cause diagnosis of plant-wide oscillations using Granger causality, Journal of Process Control, 24, 450-459.

# Paper III

Anton A. Kiss, Edwin Zondervan, Richard Lakerveld, Leyla Özkan (Eds.)
Proceedings of the 29th European Symposium on Computer Aided Process Engineering
June 16th to 19th, 2019, Eindhoven, The Netherlands. 
http://dx.doi.org/10.1016/B978-0-12-818634-3.50121-1

# Single-shooting optimization of an industrial process through co-simulation of a modularized Aspen Plus Dynamics model

Mikael Yamanee-Nolin[a,*], Anton Löfgren[a], Niklas Andersson[a], Bernt Nilsson[a], Mark Max-Hansen[b] and Oleg Pajalic[b]

[a]*Lund University, Naturvetarvägen 14, 221 00 Lund, Sweden*
[b]*Perstorp AB, Industriparken, 284 80 Perstorp, Sweden*
*mikael.yamanee-nolin@chemeng.lth.se*

## Abstract

The Python Module Coupler (PyMoC) is a tool for co-simulation of Aspen Plus Dynamics modules that together make up an overall process flowsheet. The tool requires only user input in the form of file paths to Aspen Plus Dynamics modules, and it is able to automatically make the required connections there between, and keep track of the simulation whilst updating the streams regularly. This contribution briefly discusses the implementation and mechanisms of PyMoC, and then applies it to a multi-module, single-shooting constrained optimization problem, where an industrial set-up consisting of an evaporator system coupled to a distillation column is studied. This serves as a showcase of PyMoC's functionality and usability, as well as its potential in serving as a helpful tool for practitioners of model-based studies who could benefit from modularizing their models. Utilizing PyMoC for this purpose, the optimization results indicate that the operating costs induced from the steam consumption can be reduced by 54% compared to a nominal operating case, but a holistic, full-process study is necessary to understand the full set of possibilities, causes, and effects.

**Keywords:** Python, Aspen Plus Dynamics, co-simulation, optimization, PyMoC

## 1. Introduction

Model-based studies have a wide range of important uses, and can help practitioners save both time and money, whilst improving quality and safety of a process (Oppelt et al., 2015). Especially dynamic model-based studies have many different applications due to their capabilities of investigating not only steady-state conditions but also transient patterns (Skorych et al., 2017). Thus, performing such studies, e.g. single-shooting optimization, plays an important role in improving production processes in order to satisfy demands regarding e.g. environmental aspects whilst increasing production capacity.

When modeling and simulating large, complex flowsheets, there are advantages to be found in dividing the overall model into smaller sub-modules (i.e. modularization), such as simplifying the addition of complementing models and opening up possibilities for parallel work-flows (Felippa et al., 2001). It also makes overview easier if the models would follow existing P&IDs. Modularization is furthermore a solution to convergence issues that large-scale complex process flowsheet models may suffer from (Lin et al., 2017), which would otherwise obstruct model-based studies.

However, modularization comes with its own challenges, as there will thus be a need to connect and co-simulate the modules, which poses a challenge in itself (Andersson, 2016). Doing so manually is obviously not feasible considering the sheer number of spatial and temporal connections necessary for continuous simulation. Commercially available tools such as the AspenTech Operator Training Simulator (OTS) could be used, but the lack of publications utilizing the OTS for model-based studies (Ahmad et al., 2016) supports the conclusion that this tool may not be appropriate to use for such studies. To this end, a tool called the Python Module Coupler (PyMoC) has been developed to help practitioners with co-simulation of modules developed in Aspen Plus Dynamics (APD), utilizing the process flowsheeting software as the engine for calculations. The Python implementation of PyMoC allows for advanced model-based studies by automating connections and data transfer through the application of a naming convention and a zero-order hold analogy (ZOHA), respectively.

In this contribution, PyMoC has been used to optimize a multi-module model of an oscillating polyalcohol separation process, consisting of a two-stage evaporator system and a methanol distillation column. PyMoC will first be introduced briefly, before the modules and the optimization problem is presented, followed by results and conclusions.

## 2. The Python Module Coupler - PyMoC

PyMoC is based on the COM enabled interface first presented in Nolin et al. (2017), and works by taking as input the paths of the APD files that are to be co-simulated, and then automatically connects these and co-simulates them. The nature of its Python implementation further simplifies customizing model-based studies and applying useful 3rd party software. In this section, two of the main mechanisms behind the function of PyMoC are briefly explained; (i) the naming convention, and (ii) how the transfer of data between modules is performed during runtime.

### *2.1. The naming convention*

In order to minimize the amount of work required by the user, connections between modules are made automatically. This is based on a naming convention, i.e. streams with identical stream names are connected. PyMoC automatically recognizes which stream is the source and the destination, based on the 'Fixed'/'Free' stream variable property used within APD. This means that the user only needs to make sure that the streams to be connected are named identically during modeling.See figure 1 for an illustrative, where the streams 'FEED' will be connected with the algorithm recognizing that the Module A stream is the source to the destination in Module B. The naming convention is also used for deciding which results that are to be presented. The user may provide an *interesting stream-name prefix*, which the algorithm will use to recognize for which streams the user wants results. This is practical especially for large systems, as the user can name the interesting streams with that prefix instead of naming specific streams, and the stream results will be extracted automatically for all interesting streams. Using figure 1 as an example again with e.g. "I-" as the prefix, streams I-IN, I-V, and I-L will have their results extracted automatically.

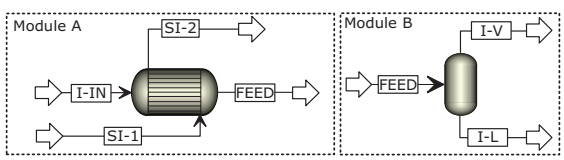



Figure 1: An illustrative example for the naming convention.

### *2.2. Data transfer using zero-order hold*

A zero-order hold (ZOH) strategy, which has recently been successfully applied for the control of chromatographic separation (Sellberg et al., 2017), is utilized to transfer data between modules, and is essentially a piece-wise constant method. It works by taking the output of the source stream and setting that as the input of a destination stream, keeping this constant over a simulation time horizon, $\tau$. The length of the simulation time horizon is chosen by the user, who should consider the trade-off between accuracy and speed - a smaller $\tau$ gives more frequently updated modules, increased accuracy since modules can work with fresher data, as well as longer run-times. The main advantage is the simple implementation, and the main disadvantage is that the sudden step-changes to a variable may introduce discontinuities, something of which the user needs to be aware.

## 3. Optimization of a modularized model using PyMoC

The problem entails optimizing the two parts of the polyalcohol separation system modeled in two independent flowsheets in order to grant the aforementioned advantages of modularization. The modules are described here, followed by the formulation of the optimization problem.

### *3.1. The models*

The modularized process is presented schematically in figure 2. The evaporator system module has been presented in Nolin et al. (2018). The methanol distillation column was modeled using standard blocks of *RADFRAC*, *MHEX*, and *FLASH*2 in Aspen Plus and then converted to Aspen Plus Dynamics, since the evaporator system was oscillating with propagations into the distillation system. The default controllers added during the conversion to an APD model were kept. An important aspect to note of both processes is that the separation was driven by steam at different pressures, inducing different process costs. Furthermore, of special interest are the streams named PROD and MEOH in figure 2, representing the target product stream of the main component in each module.

The distillation module shares four connections with the evaporator module, all of which drawn from the top of process units in the evaporator module, thus containing the most volatile components at each instance. These streams are passed through a 'translation module' (TM in figure 2) since the evaporation and distillation modules do not share identical Aspen Property Definition files. The difference there-between, c.p., is that formaldehyde was excluded from the distillation module component/property sets during modeling (and subsequently also the Maurer reactions (Hasse and Maurer, 1991) of the evaporator system), as its presence was assumed to be negligible. Essentially, this means that the distillation module cannot handle formaldehyde since its component/property sets do not contain the component. The translation module makes sure that formaldehyde is bypassed, but that the rest of the stream states (in terms of temperatures, pressures, mass flows, and compositions) are transferred from the evaporation to the distillation module. The translation module consists of one *SEP* block per stream, with the singular task of separating formaldehyde into a bypass stream, available for use if necessary later on.

### *3.2. The optimization problem*

The steam consumption was one of the major cost-drivers in a nominal operating case of the overall, oscillating process. Thus, the goal of the optimization was to minimize the steam consumption cost incurred during daily operation while managing process and product demands (which are averaged due to the process oscillations captured in the modules). These demands entail the evaporation module product stream (PROD in figure 2) purity and mass flow, as well as the mass flow and methanol concentration in the top stream of the distillation column (MEOH in figure 2). In

addition to these constraints, another constraint on the minimum mass flow rate from the top of the distillation column was set at half of the nominal run mean value, to keep the column from running dry. This gives a total of five inequality constraint functions. Furthermore, each simulation included a pre-processing simulation period of 12 simulation hours to have any potential (cyclic) steady-state established before the optimization run. The problem was approached by employing a single-shooting strategy with $\tau = 0.1h$, and formulated as unbounded optimization subject to non-linear constraints:

$$
\begin{aligned}
\underset{u}{\text{minimize}} \quad & \phi(u) = \sum_{i=1}^{2} \frac{c_i}{u_{i,nom}} \cdot u_i \\
\text{w.r.t.} \quad & u_1 = w_{steam,\ evap} \in \mathbb{R} \\
& u_2 = w_{steam,\ dist} \in \mathbb{R} \\
\text{s.t.} \quad & \text{Modularized flowsheet model} \\
& C_1 = \overline{w}_{PROD} - C_{w_{PROD}} \geq 0 \qquad C_2 = \overline{P}_{PROD} - C_{P_{PROD}} \geq 0 \\
& C_3 = \overline{w}_{MEOH} - C_{w_{MEOH}} \geq 0 \qquad C_4 = \overline{P}_{MEOH} - C_{P_{MEOH}} \geq 0 \\
& C_5 = w_{MEOH,\ min} - C_{w_{MEOH,\ min}} \geq 0
\end{aligned}
$$

where $w$ is mass flow rate, $P$ is purity, $\overline{x}$ denotes the average of a variable over the time period, subscripts $PROD$ and $MEOH$ refer to the streams thus named (figure 2) $u_{i,nom}$ is the decision variable values , and $c_i$ is a price factor relating the mass flow of steam to the incurred cost thereof; $c_1 = 1$ and $c_2 = 0.7$, reflecting the cost ratio between the two steam pressure levels. $C_{i_j},\ i = (w,P),\ j = (PROD, MEOH)$ denote the respective values of the *mean constraints*, while $C_{w_{MEOH,\ min}}$ denotes the constraint value for the minimum flow rate in the MEOH stream. The problem was solved by utilizing the scientific computing package *SciPy*'s implementation of the optimization algorithm Constrained Optimization By Linear Approximation (COBYLA), using the nominal point of the simulation as initial values for the decision variables, and running the model as part of the constraint function rather than the objective function.

## 4. Results and discussion

The objective value for the nominal run, $\phi(u_{nom}) = 1.7$, represents a normalized hourly steam consumption cost. By reducing $u_1$ by 69% and $u_2$ by 33%, the optimization was able to reduce the objective value by 54% to $\phi(u_{opt}) = 0.78$, whilst successfully satisfying the constraints. The

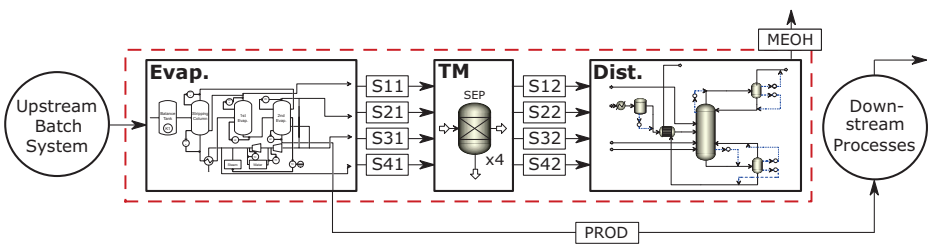



Figure 2: A schematic of the modularized process model, marked by the dashed box; the evaporator system ("Evap.") is described in Nolin et al. (2018), whilst the translation module ("TM") and the distillation column ("Dist.") are introduced in this contribution. Note the target product streams PROD and MEOH. The decision variables, i.e. the steam streams, are fully embedded in each module.

latter can be seen in figures 3 and 4, as the mean of variables in the optimal run is greater than the respective mean constraint; the same is true for the minimum flow constraint regarding $\overline{w}_{MEOH}$.

To gain insight into the dynamic behavior of the process, the PROD and MEOH stream results (i.e. $w$ and $P$) of the nominal run as well as the optimal run are presented together with their mean values and the relevant constraints in figures 3 and 4, respectively. The results have been scaled to be centered around the mean constraint for each variable, rendering the results as dimensionless quantities. Neither graph show sudden, discontinuous responses at the start/end of the simulation time horizons, suggesting that the ZOHA strategy has not created any significant discontinuities.

As can be seen in figure 3, the mass flow and purity for the product stream vary somewhat during the nominal run, as previously reported in Nolin et al. (2018), and such is still the case for the optimal run. The size (i.e. difference between minumum and maximum) of the oscillations in the mass flow have been reduced by 27%, which should lead to simplified control and mitigated effects downstream. Furthermore, the optimal solution will effectively lower the purity by almost 2 percentage units. Together, the increased production rate as measured by $w_{prod}$ and the decreased purity as measured by $P_{prod}$ indicate that the optimal solution essentially produces more of a less concentrated product stream. It is therefore possible that increased costs will be incurred downstream to evaporate any excess water. Exactly how these interact with each other and how they in combination will affect downstream processes may require further studies.

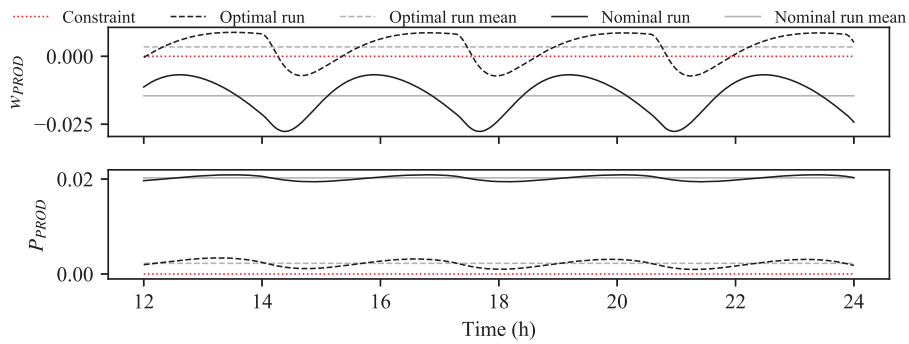



Figure 3: The PROD stream results from the nominal and optimal runs.

The mass flow results for the MEOH stream presented in figure 4 show that the optimal run yields massive oscillations over cycles, with a "jittery" model response, the latter of which presumably caused by the fast dynamics of a pressure controller at the top of the distillation column. The content of this stream is primarily used as fuel for the process, and the effects of propagations here will thus presumably not propagate to the same extent as will oscillations in the PROD stream, even though they are relatively large. Furthermore, the purity of the MEOH was increased significantly in the optimal run. This, combined with the reduced steam consumption for both modules, is further support for the natural conclusion that less water is evaporated in the evaporator module, and subsequently less water is passed on to the distillation column via the top-drawn streams.

## 5. Conclusions

The optimization shows that it is possible to reduce steam costs by 54 % compared to the nominal case whilst adhering to process demands. However, this may be a globally suboptimal solution

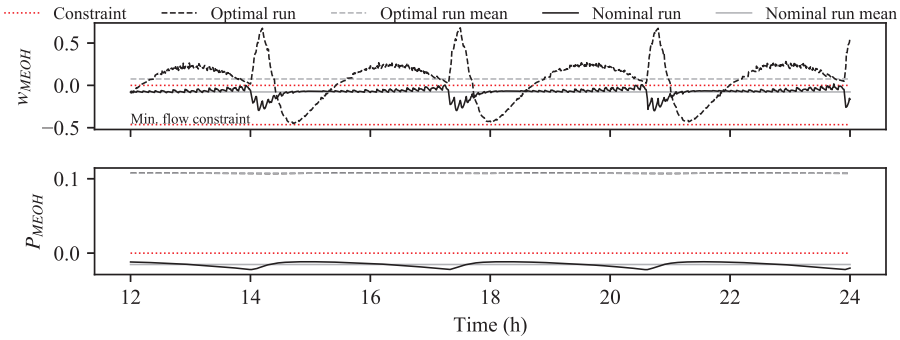



Figure 4: The MEOH stream results from the nominal and optimal runs; note the additional constraint regarding minimal mass flow rate.

when looking at the full process, since less water is evaporated in the investigated modules. This leads to as of now uncertain consequences downstream, but it is possible that the savings in the evaporator and distillation modules presented in the current work may be negated by increased costs downstream. As such, it is imperative to perform a holistic investigation of the full process to prepare for improved decision-making. However, the current work shows that, for the investigated systems, reducing the steam consumption whilst still adhering to production and product demands is indeed possible. Furthermore, oscillatory propagations from any module to other parts of the process need to be studied further. Negative effects thereof are especially necessary to study, and these can potentially be mitigated through an optimal control trajectory optimization possibly combined with improved tuning of the relevant controllers. PyMoC should be a useful tool to use for the former, due to the nature of its Python implementation as well as the implementation of the ZOHA strategy for data transfer.

## References

Ahmad, Z., Patle, D. S., Rangaiah, G., 2016. Operator training simulator for biodiesel synthesis from waste cooking oil. Process Safety and Environmental Protection 99, 55 – 68.

Andersson, C., 2016. Methods and tools for co-simulation of dynamic systems with the functional mock-up interface. Ph.D. thesis, Lund University.

Felippa, C. A., Park, K., Farhat, C., 2001. Partitioned analysis of coupled mechanical systems. Computer Methods in Applied Mechanics and Engineering 190 (24), 3247 – 3270, advances in Computational Methods for Fluid-Structure Interaction.

Hasse, H., Maurer, G., 1991. Vaporliquid equilibrium of formaldehyde-containing mixtures at temperatures below 320 k. Fluid Phase Equilibria 64, 185 – 199.

Lin, Z., Wang, J., Nikolakis, V., Ierapetritou, M., 2017. Process flowsheet optimization of chemicals production from biomass derived glucose solutions. Computers & Chemical Engineering 102, 258 – 267.

Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., Pajalic, O., 2017. Unbiased selection of decision variables for optimization. In: Espuña, A., Graells, M., Puigjaner, L. (Eds.), 27th European Symposium on Computer Aided Process Engineering. Vol. 40 of Computer Aided Chemical Engineering. Elsevier, pp. 253 – 258.

Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., Pajalic, O., 2018. Analysis of an oscillating two-stage evaporator system through modelling and simulation: An industrial case study. Accepted for publication in Chemical Engineering Transactions, vol. 69.

Oppelt, M., Wolf, G., Urbas, L., 2015. Life cycle simulation for a process plant based on a two-dimensional co-simulation approach. Vol. 37 of Computer Aided Chemical Engineering. Elsevier, pp. 935 – 940.

Sellberg, A., Holmqvist, A., Magnusson, F., Andersson, C., Nilsson, B., 2017. Discretized multi-level elution trajectory: A proof-of-concept demonstration. Journal of Chromatography A 1481, 73 – 81.

Skorych, V., Dosta, M., Hartge, E.-U., Heinrich, S., 2017. Novel system for dynamic flowsheet simulation of solids processes. Powder Technology 314, 665 – 679, special Issue on Simulation and Modelling of Particulate Systems.

# Paper IV

Contents lists available at ScienceDirect

Chemical Engineering Research and Design

journal homepage: www.elsevier.com/locate/cherd

# Trajectory optimization of an oscillating industrial two-stage evaporator utilizing a Python-Aspen Plus Dynamics toolchain

*Mikael Yamanee-Nolin*[a,*], *Niklas Andersson*[a], *Bernt Nilsson*[a], *Mark Max-Hansen*[b], *Oleg Pajalic*[b]

[a] *Lund University, Faculty of Engineering, Department of Chemical Engineering, P.O. Box 124, SE-221 00 Lund, Sweden*

[b] *Perstorp AB, Industriparken, 284 80 Perstorp, Sweden*

ARTICLE INFO

*Article history:*
Received 20 September 2019
Received in revised form 18 November 2019
Accepted 14 December 2019
Available online 24 December 2019

*Keywords:*
Aspen Plus Dynamics
Python
Dynamic optimization
Derivative-free optimization
Evaporator system
Oscillations

ABSTRACT

Evaporators are integral parts of many separation processes across production industries, and they need to be well understood in order to be operated well, thereby enabling high resource-efficiency and productivity. In a previous investigation, the effects of disturbing oscillations in a two-stage evaporator system were quantified. In the current study, these oscillations were reduced through trajectory optimization using steam consumption as a temporally discretized decision variable, taking advantage of a dynamic process flowsheet model in Aspen Plus Dynamics (APD) employed as if it were a black-box model. The optimization was performed utilizing a Python-APD toolchain with the SciPy implementation of COBYLA. The optimal trajectory was able to successfully reduce the objective function value (including the product stream mass flow variance and a bang-bang penalty on the trajectory itself) to slightly less than 0.3 % of that of the nominal case, in which a time-invariant steam consumption was employed. This in turn grants opportunities to increase throughput of the process, leading to significant financial gains.



## 1. Introduction

Evaporator systems are key components of production processes across many different industries and disturbances (such as unexpected feed concentration perturbations) can become a real issue for process engineers. Considering concurrent demands on companies to increase productivity and resource-efficiency, and that evaporators are inherently energy intense, proper control and disturbance management is essential. In order to manage these demands better, model-based studies lend themselves as viable options.

Whilst there is plenty of published research on evaporator systems utilizing steady-state models, far fewer studies have been published involving dynamic simulation and control (Luyben, 2018); in addition to those listed by Luyben (2018), there are a few other studies of particular relevance to the current work, especially with regards to managing disturbances. For instance, Kumar et al. (2013) created a generalizable dynamic model that they used to gain insights into the behavior of a multi-effect evaporator system subjected to disturbances, whilst Adams et al. (2008) designed a control architecture to deal with oscillatory disturbances. Pitarch et al. (2017) implemented a real-time optimization strategy to boost the resource-efficiency of an evaporator system, showcasing the usefulness of non-linear programming (NLP) (Biegler, 2010). Moreover, disturbances of an oscillating nature can

* Corresponding author.
E-mail address: mikael.yamanee-nolin@chemeng.lth.se (M. Yamanee-Nolin).
https://doi.org/10.1016/j.cherd.2019.12.015

become especially problematic when they propagate through and decrease performance of an entire process (Yuan and Qin, 2012). This in turn prompted a previous in-silico investigation of an industrial evaporator system (Nolin et al., 2018), in which disturbing oscillations of the industrial system were analyzed and quantified for the first time. The oscillations were further found to be present throughout the whole system, effectively increasing steam consumption by 1 % and reducing productivity by 2 %, which is of economic concern due to the scales of the processing industries. Furthermore, since the evaporator system was bottlenecking the entire process, eliminating the oscillations at a low cost would translate into financial gains.

Considering the problem accounted for above, the aim of the current study was to eliminate the oscillations in the evaporator system reported by Nolin et al. (2018) through model-based dynamic NLP, taking advantage of the existing Aspen Plus Dynamics model used in that study. This optimization based solution would serve as an alternative to expensive retrofits or reconstruction. However, the built-in tools of the AspenTech software lack the necessary capabilities for advanced, model-based optimization studies. For instance, Aspen Plus does not include algorithms for multiobjective optimization, which severely limits the options for investigating a process (Muñoz López et al., 2018). Furthermore, it is unclear how to deal with dynamic optimization problems such as grade transitions in Aspen Plus Dynamics, which motivates the use of an external optimization architecture in conjunction with a dynamic flowsheet model (Negrellos-Ortiz et al., 2018). This way, it is possible to take advantage of the component and unit-operation libraries available in commercial flowsheeting software in combination with generalizable optimization tools. Therefore, the pre-existing Aspen Plus Dynamics model was employed as if it were a black box type of model in the sense that the input and output data of the model are the main concerns (in a similar fashion as Negrellos-Ortiz et al. (2018)). To perform the optimization, the model was coupled to a modified version of the COM-based Python Module Coupler (PyMoC) cosimulation tool, which was first presented in a cosimulation-based optimization study that included a version of the current evaporator system model (Yamanee-Nolin et al., 2019). In order to easily implement general inequality constraint equations as well as decision variable boundaries specifically, the SciPy implementation of the COBYLA algorithm (Powell, 1994) was used to perform the optimization.

The results show that it is possible to minimize the oscillations through trajectory optimization, leading to an increase in throughput, a reduction in steam consumption, while product purity was kept at a satisfying level. In addition to the performance improvements in the evaporator system, the minimization of the oscillations further means that throughput can be increased even further, and the downstream propagation of the oscillations will pose less of a problem, potentially leading to positive effects in the subsequent subsystems as well.

The remainder of this paper is structured as follows: section 2 introduces the process and the model that is used for the current study. This is followed by the presentation of the mathematical problem formulation along with a brief description of the modified PyMoC architecture with which the optimization problem is solved. Results are provided and discussed in section 3, with a summary of the major conclusions presented in the final section.

## 2. Materials and Methods

This section begins with a description of the investigated process along with the Aspen Plus Dynamics model used for the study. The formulation of the trajectory optimization problem then follows, along with a description of how it was solved by using the PyMoC tool to transfer data between the optimizer and the model.

### 2.1. *Process and model*

The studied evaporator system is the first stage of a series of separation processes for purifying a polyalcohol component at Perstorp AB and it is presented schematically in Fig. 1. The system starts with a balance tank working as a buffer vessel between the upstream batch-based system and the evaporator system; the upstream system intermittently feeds the balance tank with a solution that is highly concentrated with respect to the desired product (The 'Feed' stream in Fig. 1). The balance tank is also fed by a dilute recycling stream from a part of the downstream processing. The balance tank mixture feeds a stripping column, the purpose of which is to remove water. The bottom stream of the stripping column is then heated by a stream from a mechanical vapor recompression (MVR) circuit, which also contains some added steam and water (the latter for direct intercooling purposes), before it is fed to the first evaporator stage. The top stream of the first evaporator stage drives the stripping column, whereas the bottom is fed to the second stage. The bottom of the second stage is the product stream, named 'Prod' in Fig. 1, which is sent downstream for further purification.

The process was subject to oscillatory disturbances, which have been analyzed and quantified in-silico previously (Nolin et al., 2018). These disturbances are generated by a combination of how the process was originally constructed, then retrofitted and currently operated. The main cause is the fact that the concentrated feed stream is intermittent, whilst the diluted recycling stream and stripping column feed stream are both continuous. This will affect the liquid level in the tank, as well as how the concentration changes over time. Essentially, the feed from the batch system will make the concentration of product in the balance tank increase rapidly, and the material in the balance tank will subsequently be diluted by the recycling stream. As the stripping column feed stream is continuous, this diluting effect propagates into the whole system as the stripping column feed is successively diluted over the course of a cycle. This cycle restarts the next time the feed sequence is activated, which leads to the concentration increasing again, giving rise to performance-worsening oscillations that can propagate, leading to unnecessarily high steam consumption as well as production rate margin.

To study these dynamic disturbances, a dynamic model of the process was needed. Aspen Plus, along with its dynamic counterpart Aspen Plus Dynamics, were used as modeling and simulation tools for this purpose. This is a combination which has been used as an efficient computer-aided process engineering tool kit for many different studies concerned with the dynamics of vapor-liquid systems, with two good examples being the studies presented by Bildea and Kiss (2011) and Zhang et al. (2018). The current model was based on the one presented by Nolin et al. (2018), albeit run at a different nominal case setting. The model was constructed using the standard blocks available in Aspen Plus; *HeatX* and *Flash2* were

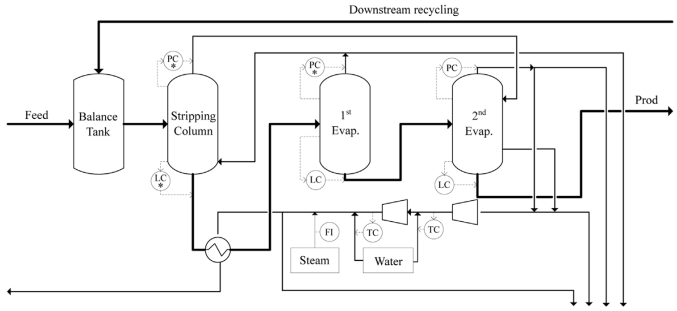



**Fig. 1 – The modeled process.**

used to model the evaporator stages, and a *RadFrac* column was used for the stripper. The MVR circuit was modeled using a combination of compressors, and mixers for adding water and steam. The steady state model built in Aspen Plus was then converted as a *flow-driven* simulation to Aspen Plus Dynamics using the built-in tool for that purpose. During conversion, controllers were automatically added and were decided to be kept and used at their default settings, as their positions for the most part reflected the real plant. The controllers that did not reflect real controller positions are marked by asterisk in Fig. 1; these are used in the model to mimic pressure-driven flow related phenomena, which otherwise would risk unrealistic behavior in terms of mass flow rates and pressure changes in the simulation. Furthermore, there was no boiling-point elevation control in place in the current model (as opposed to the previous study (Nolin et al., 2018)), as the use of steam consumption as the decision variable was desirable. Finally, the intermittent stream feeding the balance tank was modeled using a 'Task' in Aspen Plus Dynamics, which was programmed to mirror the real-life behavior where the balance tank is nearly filled at a specific low liquid level condition.

### 2.2. *Trajectory optimization using PyMoC*

The goal of the study was to minimize the product stream mass flow oscillations over eight hours by utilizing the steam consumption as a decision variable discretized in time. This was chosen as a decision variable due to expectations (based on a development of the conclusions in a previous study (Yamanee-Nolin et al., 2019)) that a form of counter-phase on the steam consumption could be found that would minimize the oscillations. The variance ($\sigma^2$) of the product stream mass flow was used as a measure of the oscillations for the objective function. Furthermore, a bang-bang trajectory (e.g. a trajectory consistently alternating between and only between its lower and upper bounds (Lang and Biegler, 2007)) could lead to long-term premature equipment fatigue from regular and significant changes in temperatures and/or pressures in the system. Therefore, an additive penalty term was introduced in the objective in order to make the resulting trajectory smoother. The term penalizes the square of the signal difference of two neighboring horizons. Finally, the sum of the variance and the penalty is normalized with respect to the nominal objective value, $\phi_{nom} = \sigma^2(w_{PROD}(t, u_{nom}))$.

To account for process requirements in place, inequality constraints were formulated for the averages of the mass flow and purity of the product stream. In order to give the optimizer some leniency whilst maintaining process demands, these averages were required to be greater or equal to 99.8 % of the average of the nominal run, giving $\beta = 0.998$. Furthermore, bounds were set on each of the discretized decision variables so that the steam consumption would be kept between half and twice the average steam consumption of the nominal case at all times.

The problem was solved utilizing a single-shooting approach and was mathematically formulated as follows:

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & \phi(\mathbf{u}) = \frac{\sigma^2(w_{PROD}(t, \mathbf{u})) + \rho(\mathbf{u})}{\phi_{nom}} \\ \text{s.t.} \quad & \text{Flowsheet model} \\ & \bar{w}_{PROD} - \beta \cdot \bar{w}_{PROD,\ nom} \geq 0 \\ & \bar{P}_{PROD} - \beta \cdot \bar{P}_{PROD,\ nom} \geq 0 \\ & 0.5 \cdot u_{nom} \leq u \leq 2 \cdot u_{nom}, \mathbf{u} = [u_1, ..., u_{N_u}] \end{aligned}$$

where $\phi$ is the objective function and $u$ the (discretized) decision variables; $\rho(\mathbf{u}) = \Sigma_{k=1}^{N_u - 1} \Delta u_k^2$ is the bang-bang penalty and $R_u$ is a case-specific weight to scale it; $w$ is mass flow rate; P is purity; $\bar{w}$ and $\bar{P}$ denote the average of those variables over the simulated time period; subscript *PROD* refers to the stream thus named in Fig. 1; and subscript *nom* refers to the nominal case.

To solve the problem, derivative-free optimization (DFO) was considered. DFO algorithms can be classified as *stochastic* or *deterministic* algorithms, depending on whether or not an algorithm takes random steps in its minimization procedure (Rios and Sahinidis, 2013). An example of deterministic DFO algorithms is the Nelder-Mead simplex algorithm, whereas stochastic examples include simulated annealing and particle swarm algorithms (Rios and Sahinidis, 2013). As presented by Negrellos-Ortiz et al. (2016), DFO has previously been suggested to serve as an effective alternative in solving dynamic optimization problems at the engineering-level (as defined by Biegler (2010)), where gradient information may be unavailable or unreliable in practice. Negrellos-Ortiz et al. furthermore showed the effectiveness of deterministic DFO techniques, as they utilized Powell's BOBYQA algorithm (Powell, 2009) to perform dynamic product transitions for a set of different reactors (Negrellos-Ortiz et al., 2016) as well as for an air separation unit (Negrellos-Ortiz et al., 2018). In both these

studies, they utilized Aspen Plus Dynamics models as a type of black-box model, only concerned with inputs and outputs. Another successful use of DFO algorithms for trajectory optimization has been presented by Mohd Fuad et al. (2012), who employed Powell's COBYLA algorithm (Powell, 1994) to find the optimal trajectory for long-term catalyst deactivation. The deterministic DFO approach was thus chosen to solve the current optimization problem and stochastic approaches were left for future studies. Whilst both BOBYQA and COBYLA were considered as potential algorithms, COBYLA was eventually chosen in order to easily implement the constraints and bounds of the problem. The SciPy implementation of COBYLA was successfully used to solve the problem with the final accuracy tolerance left at the default value of $10^{-6}$, the absolute constraint violation tolerance set to zero, and the nominal trajectory (i.e. flat) used as an initial trajectory.

In the current work, the PyMoC algorithm first presented in (Yamanee-Nolin et al., 2019) was modified in order to employ a direct, sequential approach to trajectory optimization (Biegler, 2010). In this approach, the originally infinite-dimensional problem is recast as a finite-dimensional problem through discretization of the decision variable along the temporal axis. For cosimulation purposes, PyMoC utilizes a zero-order hold analogy for data transfer during cosimulation. This data transfer strategy makes use of synchronization points (SPs) at which simulation is paused for data transfer; values in different modules are updated and then held constant for a horizon ($\tau$). For trajectory optimization purposes, PyMoC and the direct, sequential approach were as such found to work nearly seamlessly together, as the SPs in PyMoC were possible to use for updating the decision variable signal for the next horizon instead of transferring data between modules. Due to this good fit between PyMoC and the chosen optimization approach, it was sufficient to add a trajectory optimization wrapper to PyMoC that accepts the decision variable vector from the optimizer and injects the relevant information at the correct SPs. Furthermore, the dynamic optimization problem in the current work was solved in an open-loop fashion similar to Sellberg et al. (2018), for eight simulation hours with ten decision horizons per simulation hour giving $N_u = 80$ and $\tau = 0.1$ h.

In order to provide better insight into how a user can establish the COM based link between Python and Aspen Plus Dynamics, a basic example is provided in Plate 1. This example includes how to set up such a connection (given some model), as well as reading data from and writing data to generically named parts (blocks/streams) of the model. These functionalities are essential and can be further built upon in order to carry out a customized optimization study, where an objective function can be composed and supplied to any of the algorithms that are freely available to all Python users. As for the current study, a schematic overview of the optimization process, starting at the optimizer call, is presented in Fig. 2. COBYLA is used for evaluating the objective function, which in turn utilizes PyMoC for transferring data between the optimizer and the model during trajectory optimization. PyMoC will take the full decision variable vector as an input, and it will then loop through the vector and advance the simulation one synchronization point and horizon at a time, until the set final time of eight hours is reached. The simulation output will then be sent back to the objective function for evaluation and updating of the decision variable vector until convergence is achieved.

```python
import win32com.client

if __name__ == " __main__ ":
    path = 'C:\\filepath\\'
    model_name = 'filename.dynf'

    # Create the Python -Aspen connection via COM
    adyn = win32com.client.Dispatch('AD Application')

    # Make Aspen window visible, activate,
    # and load specified model
    adyn.Visible = True
    adyn.activate()
    adyn.OpenDocument(path+model_name)

    # Create model object handles
    sim     = adyn.Simulation
    fsheet  = sim.Flowsheet
    streams = fsheet.Streams
    blocks  = fsheet.Blocks

    # Enable recording of variable history
    sim.options.TimeSettings.RecordHistory = True

    # Set end time to 2 hours and run simulation
    sim.endtime = 2
    sim.run(1) # Synchronous (1), runs to end.

    # Retrieve temperature in Block B1
    Block1_T = blocks('B1').T.Value

    # Set Stream S1 mass flow rate to 100 kg/h
    streams ('S1 ').FmR.Value = 100
```

**Plate 1 – An example of how to establish the COM based connection between Python and Aspen Plus Dynamics, as well as for reading and writing data. This can be extended into a non-linear program by calling an available optimization algorithm and supplying a custom-made objective function to it.**

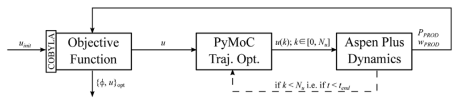



**Fig. 2 – A schematic of the optimization process: COBYLA evaluates the objective function, which in turn makes use of PyMoC. PyMoC accepts the decision variable vector as a whole as input, and uses a zero-order hold strategy to implement the discretized decision variable at the correct synchronization points of the simulation, essentially through looping of the vector.**

## 3. Results and discussion

In Fig. 3, the results from the nominal and the optimized runs are presented, normalized with respect to the mean value of the nominal run. The oscillations of the nominal operating conditions using a constant steam consumption are shown in Fig. 3a. For the current nominal operating point, the oscillations were quantified to be at around ±1%, thus having a negative effect on the production rate due to capacity constraints and safety margins of the process. However, as seen in Fig. 3b, the optimal trajectory is able to increase the average production rate slightly; it also brings the oscillations to a much lower level whilst lowering the average steam consumption to 69 % of the nominal operating point, which in part is made possible given that the average purity was allowed to decrease. That a reduction of the steam consumption is possible further indicates that the method for dealing with the

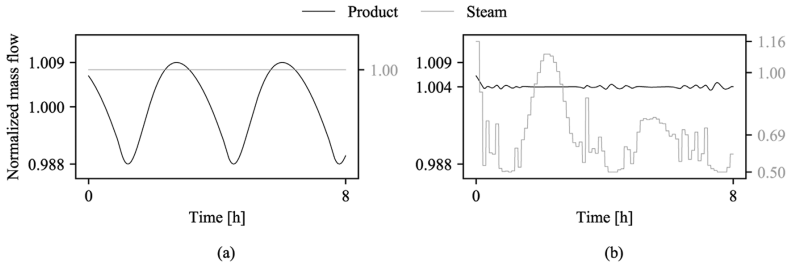



**Fig. 3 – The stream results from the nominal and optimal runs. The black line is the product stream, and the gray line is the steam consumption (i.e. the discretized decision variable). With an optimized steam trajectory, the product mass flow oscillations have been dramatically reduced.**

oscillations in the plant thus far has been to use an excessive amount of steam. This means that by using an optimal trajectory on the steam consumption in order to minimize the oscillations, the process can be improved in terms of both production rate as well as resource-efficiency whilst the product purity constraint is in place and satisfied. This in turn translates to significant financial and environmental gains.

However, even though the objective was formulated to minimize the occurrence of bang-bang optimization, we still see this at times, especially towards the end of the optimal trajectory. This is most probably due to the fact that the effect of the decision variables does not have time to propagate through the system before the final time of the whole optimization horizon, promoting relatively extreme behavior. The behavior where the signal approaches a constant value and at the end of the whole horizon makes short and sudden moves away from that value is called the turnpike effect, according to Faulwasser et al. (2017). Rawlings and Amrit (2009) also describe and provide an example of the turnpike effect. There have been many studies regarding how to avoid or remedy stability issues of the turnpike effect. Methods that have been suggested include, for instance, employing an infinite-horizon approach (Würth et al., 2009) or by formulating terminal conditions, e.g. equality constraints ensuring closed-loop stability, as reviewed by Mayne et al. (2000). Whilst it is an interesting issue, implementing the aforementioned methods for counteracting the turnpike effect was not performed in the current study.

Another interesting detail in the results is the amplitude reduction between the two major peaks in the optimal steam trajectory. The reduction indicates that at the start of the simulation, the process carries a lot of momentum that is counteracted by the first peak, which in turn reduces the workload required of the second peak. This difference in workload is due to an integrating effect of the product dilution happening as a result of a reduced steam consumption. There is thus potential for lowering the average steam consumption even more over time, which has to be studied further. However, it is important to note that this would still be subject to the product purity constraint.

The purity results are presented in Fig. 4, again normalized with respect to the mean value of the nominal run. The purity constraint based on downstream requirements is satisfied in the optimized scenario, with an average of 99.8 % of the nominal average. The purity decrease in the optimal case is to be expected as a result from reducing the average steam consumption, since using less steam will evaporate less water in the system. This decreasing trend is not expected

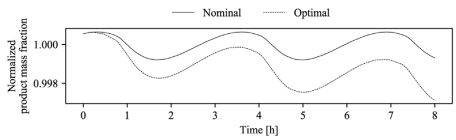



**Fig. 4 – The product purity from the nominal and optimal runs.**

to constitute a major issue, since the previously performed sensitivity analysis (Nolin et al., 2018) showed that the purity only changes very little with the different perturbation cases. However, since the results of the sensitivity analysis were assumed to depend to a certain extent on the tight boiling-point elevation control, which is overridden in the model of the current study, the trend should be further investigated in order to ensure the purity is kept at a required level. The downstream effect of the purity oscillations, which are present in the nominal case as well as in the optimal case, should also be investigated in order to ascertain to what extent the oscillations influence the rest of the purification process and to determine how to minimize any potential negative impact.

The nominal cases of the previous cosimulation-based study (Yamanee-Nolin et al., 2019) and the current work are not the same, but it is still both possible and interesting to compare the results between the current trajectory (time-variant) optimization and the previous time-invariant optimization. In the cosimulation study (Yamanee-Nolin et al., 2019), the max-min difference of the oscillations were reduced by a substantial 27 % simply by setting the steam consumption at a specific (and time-invariant) level, leading to possibilities for increasing production rate whilst satisfying all purity and production rate constraints. In the current work, the oscillations have been reduced even further by utilizing a trajectory, with the max-min difference having been reduced by 85 %. This reduction is actually in one sense slightly conservatively stated since the maximum value appears only once in the optimized case of the current work (at the start); this is in contrast to the nominal case of the current work, as well as the optimal case in the cosimulation study (Yamanee-Nolin et al., 2019), in both of which it appears in each oscillatory cycle. Furthermore, the objective function, i.e. the sum of the variance and the additive bang-bang penalties, was in the current work decreased by 99.7 % in the optimal case compared to the nominal case. It should here be duly noted that the nominal case essentially does not include the bang-bang penalty as it

employs a time-invariant value on the steam consumption, whilst the optimal case actually does include the penalty.

## 4. Conclusions

The optimal trajectory is able to minimize the oscillations, reducing them by 99.7 % compared to the nominal case, whilst satisfying constraints. By using a steam trajectory, it is furthermore possible to reduce the average steam consumption by 31 % over the simulated period. This reduction is connected to less evaporated water in the system, but with satisfied constraints on both productivity and purity, the product requirements are met at this stage. As an alternative to reconstructing the evaporator system, implementing steam trajectories should therefore be regarded an idea to consider. Future studies should focus on implementation of the trajectories on a practical level, since the open-loop method applied in this contribution is not practically feasible for real-time operation. Finally, the results from this study show that PyMoC with a wrapper can be successfully used for trajectory optimization of an Aspen Plus Dynamic model employed as if it were a black box.

## Acknowledgments

The authors gratefully acknowledge the financial support from VINNOVA [grant number 2015-02421].

## References

Adams, G., Burke, B., Goodwin, G., Gravdahl, J., Peirce, R., Rojas, A., 2008. Managing steam and concentration disturbances in multi-effect evaporators via nonlinear modelling and control. IFAC Proceedings Volumes 41 (2), 13919–13924, http://dx.doi.org/10.3182/20080706-5-KR-1001.02356, 17th IFAC World Congress.

Biegler, L.T., 2010. Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Bildea, C.S., Kiss, A.A., 2011. Dynamics and control of a biodiesel process by reactive absorption. Chemical Engineering Research and Design 89 (2), 187–196, http://dx.doi.org/10.1016/j.cherd.2010.05.007.

Faulwasser, T., Korda, M., Jones, C.N., Bonvin, D., 2017. On turnpike and dissipativity properties of continuous-time optimal control problems. Automatica 81, 297–304, http://dx.doi.org/10.1016/j.automatica.2017.03.012.

Kumar, D., Kumar, V., Singh, V., 2013. Modeling and dynamic simulation of mixed feed multi-effect evaporators in paper industry. Applied Mathematical Modelling 37 (1), 384–397, http://dx.doi.org/10.1016/j.apm.2012.02.039.

Lang, Y.-D., Biegler, L., 2007. A software environment for simultaneous dynamic optimization. Computers & Chemical Engineering 31 (8), 931–942, http://dx.doi.org/10.1016/j.compchemeng.2006.10.017, 7th World Congress of Chemical Engineering.

Luyben, W.L., 2018. Dynamic simulation of multi-effect evaporators. Chemical Engineering and Processing - Process Intensification 131, 106–115, http://dx.doi.org/10.1016/j.cep.2018.07.005.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P.O.M., 2000. Survey constrained model predictive control: Stability and optimality. Automatica 36 (6), 789–814, http://dx.doi.org/10.1016/S0005-1098(99)00214-9.

Mohd Fuad, M.N., Hussain, M.A., Zakaria, A., 2012. Optimization strategy for long-term catalyst deactivation in a fixed-bed reactor for methanol synthesis process. Computers & Chemical Engineering 44, 104–126, http://dx.doi.org/10.1016/j.compchemeng.2012.05.003.

Muñoz López, C.A., Telen, D., Nimmegeers, P., Cabianca, L., Logist, F., Impe, J.V., 2018. A process simulator interface for multiobjective optimization of chemical processes. Computers and Chemical Engineering 109, 119–137, http://dx.doi.org/10.1016/j.compchemeng.2017.09.014.

Negrellos-Ortiz, I., Flores-Tlacuahuac, A., Gutiérrez-Limón, M.A., 2016. Product dynamic transitions using a derivative-free optimization trust-region approach. Industrial & Engineering Chemistry Research 55 (31), 8586–8601, http://dx.doi.org/10.1021/acs.iecr.6b00268.

Negrellos-Ortiz, I., Flores-Tlacuahuac, A., Gutiérrez-Limón, M.A., 2018. Dynamic optimization of a cryogenic air separation unit using a derivative-free optimization approach. Computers and Chemical Engineering 109, 1–8, http://dx.doi.org/10.1016/j.compchemeng.2017.10.020.

Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., Pajalic, O., 2018. Analysis of an oscillating two-stage evaporator system through modelling and simulation: an industrial case study. Chemical Engineering Transactions 69, 481–486, http://dx.doi.org/10.3303/CET1869081.

Pitarch, J., Palacín, C., Prada, C.D., Voglauer, B., Seyfriedsberger, G., 2017. Optimisation of the resource efficiency in an industrial evaporation system. Journal of Process Control 56, 1–12, http://dx.doi.org/10.1016/j.jprocont.2017.04.002.

Powell, M.J.D., 1994. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. Springer Netherlands, Dordrecht, pp. 51–67.

Powell, M.J.D., 2009. The BOBYQA algorithm for bound constrained optimization without derivatives. University of Cambridge, Cambridge, pp. 26–46, Cambridge NA Report NA2009/06.

Rawlings, J.B., Amrit, R., 2009. Optimizing Process Economic Performance Using Model Predictive Control. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 119–138.

Rios, L.M., Sahinidis, N.V., 2013. Derivative-free optimization: a review of algorithms and comparison of software implementations. Journal of Global Optimization 56 (3), 1247–1293, http://dx.doi.org/10.1007/s10898-012-9951-y.

Sellberg, A., Nolin, M., L“ofgren, A., Andersson, N., Nilsson, B., 2018. Multi-flowrate Optimization of the Loading Phase of a Preparative Chromatographic Separation. Vol. 43 of Computer Aided Chemical Engineering, http://dx.doi.org/10.1016/B978-0-444-64235-6.50282-5.

Würth, L., Rawlings, J.B., Marquardt, W., 2009. Economic dynamic real-time optimization and nonlinear model-predictive control on infinite horizons. IFAC Proceedings Volumes 42 (11), 219–224, http://dx.doi.org/10.3182/20090712-4-TR-2008.00033, 7th IFAC Symposium on Advanced Control of Chemical Processes.

Yamanee-Nolin, M., Löfgren, A., Andersson, N., Nilsson, B., Max-Hansen, M., Pajalic, O., 2019. Single-shooting optimization of an industrial process through co-simulation of a modularized aspen plus dynamics model. In: Kiss, A.A., Zondervan, E., Lakerveld, R., Özkan, L. (Eds.), 29th European Symposium on Computer Aided Process Engineering, Vol. 46 of Computer Aided Chemical Engineering. Elsevier, pp. 721–726, doi:doi.org/10.1016/B978-0-12-818634-3.50121-1.

Yuan, T., Qin, S.J., 2012. Root cause diagnosis of plant-wide oscillations using granger causality. IFAC Proceedings Volumes 45 (15), 160–165, http://dx.doi.org/10.3182/20120710-4-SG-2026.00172, 8th IFAC Symposium on Advanced Control of Chemical Processes.

Zhang, Z., Wu, Z., Durand, H., Albalawi, F., Christofides, P.D., 2018. On integration of feedback control and safety systems: Analyzing two chemical process applications. Chemical Engineering Research and Design 132, 616–626, http://dx.doi.org/10.1016/j.cherd.2018.02.009.

# Paper V

# Modularization, co-simulation, and optimization of complex dynamic production processes in the ProOpt project

Mikael Yamanee-Nolin

*Department of Chemical Engineering, Lund University, P.O. Box 124, SE-221 00 Lund, Sweden*

## Abstract

This technical report is intended to supply details about the work and findings of the ProOpt WP1 project. The goal of the project was two-fold:

- To develop a method for the effective modeling of complex process systems, and to create process flowsheet models to be used for model-based studies.
- To develop a method for the effective control and automation of simulations of complex process system models, enabling advanced model-based studies.

Utilizing the AspenTech process flowsheeting software suite, especially Aspen Plus (AP) and Aspen Plus Dynamics (APD), in combination with Python 2.7, the listed goals have been achieved.

The method for effective modeling essentially means modularization according to existing piping and instrumentation diagrams (P&IDs). Breaking down the overall model into smaller modules allows for efficient and effective modeling where modules can be exchanged and used in other situations where meaningful. Modularization further means avoiding convergence issues that a one-as-a-whole process model may suffer from when it grows too large and complex. However, modularization requires other means of simulating the overall flowsheet. Therefore, a dynamic co-simulation tool was developed to facilitate simulation of several modules (which is infeasible to perform manually). This is part of the method to control and automate the APD simulations using Python. At the time of writing, two publications based on the simulation control tool-chain, Nolin et al. (2017) and Nolin et al. (2018), show that the Python-Aspen tool-chain called PyMoC can be used to enable model-based studies to better understand and control the process.

*Keywords:* Co-simulation, Aspen Plus Dynamics, Python Module Coupler

*Email addresses:* mikael.yamanee-nolin@chemeng.lth.se (M. Yamanee-Nolin)

⋆NOTICE: This is the author's version of a technical report produced at the conclusion of the ProOpt project. Report completed January 2019; editorial updates performed in January 2020.

## 1. Introduction

The ProOpt Work Package 1 project was a cooperation between Perstorp AB, primarily represented by Oleg Pajalic, Mark Max-Hansen, and John Berggren, and the Department of Chemical Engineering at LTH, primarily represented by Bernt Nilsson, Niklas Andersson, and Mikael Yamanee-Nolin. In order to be able to implement production changes of complex and dynamic processes, an advanced tool-chain was needed to help in planning and implementation. This project focused on the planning phase, especially on modeling and simulation.

Process optimization is usually carried out by performing parameter studies using commercial flowsheeting tools. However, these tools often lack appropriate methods for implementing advanced process optimizations (Muñoz López et al., 2018). In addition, the process models and the modeling philosophy behind both process design and process simulation are often inappropriate for advanced model-based studies (such as multi-objective or trajectory optimization) since they contain specifications, or discontinuities or are static models. Furthermore, during modeling of complex processes, the process flowsheet models may become very large and/or complex, which increases difficulty of use in at least two ways: (i) by increasing the likelihood of convergence issues (Lin et al., 2017), and (ii) by decreasing "overviewability" for a user.

Therefore, the project focused on developing a holistic and easily accessible approach to process optimization at the engineering level, as defined by Biegler (2010). This was done by introducing a user methodology for modeling and simulation, coupled with a computational platform, which in combination can facilitate the application of non-proprietary optimization methods on state-of-the-art flowsheeting models. Case studies thereof were also published.

During the project, modularization of process flowsheet models according to P&IDs was found to increase process overview and decrease convergence issues, and this is detailed in section 2. The modules were then coupled and co-simulated as an aggregated system. The main tool was the open source programming language Python, allowing for generalizable optimization methods with state-of-the-art algorithm available in numerical libraries such as SciPy, together with the dynamic process flowsheet simulation Aspen Plus Dynamics (APD). This technology enabled the creation of the Python-Aspen tool-chain called the Python Module Coupler (PyMoC). PyMoC facilitates co-simulation, which is required if modularization of a process model is performed, which in turn allows for utilization of different and complementary modeling techniques for different process steps. The details about PyMoC and its building-blocks are presented in section 3. As PyMoC made it possible to co-simulate modularized process models, it also facilitated the optimization of larger process sections, which is described in section 4.

## 2. Modeling

This section describes the findings and conclusions, along with the major considerations regarding and during modeling of the complex processes pertinent to the project. The models that were produced during the project are also presented in this section.

## 2.1 Modeling complex production processes

Model-based studies are very useful, but complex processes may demand complex models for accurate description. For instance, dynamic models are needed to understand the time-dependent behavior of a system (Skorych et al., 2017). However, large-scale, complex process flowsheet models may suffer from convergence problems and thereby cannot be simulated (Lin et al., 2017), preventing the overall system from being studied. In such a case, one course of action is to divide the overall system into smaller modules, which may work since the modules will not suffer from the same convergence issues as the full model (Lin et al., 2017). For instance, during the work behind Nolin et al. (2017) and Nolin et al. (2018), in which only one module of a larger process was presented, attempts were made to create a model of the overall process in one single flowsheet. However, convergence issues arose both in Aspen Plus as well as in Aspen Plus Dynamics that were very difficult to solve, and therefore a modularized approach was chosen. The modules did not suffer from convergence issues (as the whole model did), which was along the findings of Lin et al. (2017), and the models were more easily executable in both of the aforementioned pieces of simulation software.

However, the modularized approach described above entails its own challenges. Dynamically simulating a modularized complex model by hand is not viable as it eventually takes too much manual effort to update all the connections with the necessary states (e.g. temperature, pressure, flows, and compositions). Furthermore, the risk of errors is expected to be relatively high. Therefore, the simulations need to be automated in some manner, and one solution is co-simulation with the developed tool-chain PyMoC, which is covered in section 3.2.

Whilst modularized models may be of different physical domains or runtime environment implementations, e.g. Bulian and Cercos-Pita (2018) and Mikkonen et al. (2017), there can be other reasons to modularize an overall system model. For instance, key know-how embedded in a model may need protection (Andersson, 2016), and modularization simplifies modifications according to project needs, e.g. the addition of complementing models (Bulian and Cercos-Pita, 2018; Felippa et al., 2001). Dividing a model into several modules also simplifies the general overview of a digital twin of a complex chemical process as models can be created to mirror existing piping and instrumentation diagrams. It also opens up possibilities for work-flow parallelization, and as Felippa et al. (2001) state, testing and validation of smaller changes to a model can be performed modularly.

Modularization offers three more main advantages, according to Felippa et al. (2001): customization, independent modeling, and the connection of non-matching models. Firstly, customization entails being able to apply the numerical solvers most fitting for a module, which leads to efficient solving of the overall system. Secondly, independent modeling concerns the possibility of connecting non-matching models, as well as having the modeling work divided between engineers who are not necessarily working together physically. Thirdly, and finally, software and model reuse is important as it saves time and money since the wheel does not have to be reinvented every time it needs to be used. Felippa et al. (2001) argue that this is relevant especially to academia, who may study a process in many different projects. However, with the increasingly volatile, uncertain, complex, and ambiguous environment many companies across

industries operate in (Bennett and Lemoine, 2014), modularization and co-simulation is pertinent for industrial users in order to understand a process better.

Modularization is thus very useful since it alleviates the problem of initialization/-convergence issues in APD along several other advantages. However, in order to maximize usefulness and facilitate simulation utilizing PyMoC, the modules need to be prepared for co-simulation already during modeling, and especially two factors need to be considered: (i) there needs to be some stream naming convention by which to consistently name streams that are inter-modularly connected, and (ii) the modules may need to be supplemented with internal equations, algorithms, and/or translation modules in order to actually work in conjunction.

#### *2.1.1 Stream naming convention*

During modeling, it is important to remember that specific streams in different modules are to be connected during simulation. This makes it important to have some stream naming convention to follow strictly, in order to minimize confusion and maximize utility from modularization and co-simulation e.g. via automation of connections. Stream naming conventions can be formulated to fit a specific purpose, if necessary. Examples of stream naming conventions follow:

i All streams are named corresponding to their P&ID tag, i.e. intermodularily connected streams are given identical names across different modules.

ii Intermodularily connected streams are given identical names (possibly independent of the P&ID tag), other streams are irrelevant and disregarded.

iii Intermodularily connected streams are given identical names (possibly independent of the P&ID tag), interesting streams from which to retrieve results begin with specific string, other streams are irrelevant and disregarded.

Note that the current implementation of PyMoC has assumed a stream naming convention on the form of example iii. This means that it is important for the process designer to choose and set identical names to the connected streams. Furthermore, it is possible to set a specific string (default: 'S-') that PyMoC will use to identify streams the user has deemed as interesting to extract results from.

#### *2.1.2 Model supplements*

The modules may sometimes need some supplemental instructions or translation modules in order to function well when connected. Supplemental internal algorithms within a module may for instance also be necessary to simulate the dynamics of a pseudo-continuous process with balance tanks.

*Internal algorithms*

Dynamic modeling using APD is powerful in the sense that it is possible to add so called 'tasks' in a module, which will be carried out during simulation. These can be set to occur at specific times, or when specific conditions have been met. For instance, in order to simulate generic a fed-batch reactor, the following *Task* could be added in the 'Flowsheet' menu of an appropriate APD model:

```
Task FedBatchReactor runs at 0 // <Trigger>
 // # start feed; 3600 kg/h, ramp up over 18 s = 0.005 h
 SRAMP(STREAMS("TO_REAC").FmR, 3600, 0.005);

 // # wait for liquid level to rise to 1 m
 WAIT FOR BLOCKS("REAC").level < 1;

 // # stop feed; 0 kg/h, ramp down over 3.6 s = 0.001 h
 SRAMP(STREAMS("TO_REAC").FmR, 0, 0.001);

 // # wait 1 h for the reaction to take place
 WAIT 1;

 // # purge reactor contents
 SRAMP(STREAMS("FROM_REAC").FmR, 36000, 0.005);

 // # insert waiting time to consider e.g. cleaning

 // restart
 RESTART;
End
```

The listed algorithm will start the feed to the fed-batch reactor, and stop it when the liquid level has reached a specified height. It will then wait for the reaction to occur, and end by purging the reactor of the product. Then it repeats the cycle. The APD documentation contains all the necessary details regarding available commands and keywords of which to make use during the creation of custom-made algorithms.

*General translation modules*

A translation modules is any module built for the specific purpose of connecting one or several modules, and may not be part of the actual modeled processes. They may thus be very useful, or even necessary, in order to properly connect modules e.g. with different component lists. This could be when a component is critical in one module but negligible and/or troublesome in the following. In this case, a translation module can be designed with the purpose to filter the specific component from the stream, separating it into a pure bypass stream. One example of this is presented in section 2.2.5. Translation modules could potentially also be used to perform other functions, such as reintroducing a bypassed component, adding a new, or for testing process modifications without making changes to modules.

## 2.2 The ProOpt modules

A model of the trimethylolpropane separation process immediately following the reactor, schematically presented in the dashed box in figure 1, was created during the course of ProOpt. The model was decided to be modularized due to the findings presented in section 2.1, and followed stream naming convention example iii in subsection 2.1.1. The modules include the evaporator system, the methanol column, and the crystallizer with centrifuges, whilst the distillation system remained to be modeled at the end of the project. In general, the aim was to use as much as possible of the AspenTech standard libraries of components, physical properties, and unit operations. However, to get a better model fit, some modifications to density calculations had to be made, as well as adding the Maurer kinetics for formaldehyde oligomerization. This is described in section 2.2.1, along with considerations regarding pressure/flow driven simulations.

In general, the input values to the model were retrieved from the Process Explorer

software. The streams were mapped to the available and relevant indicators in the PID, and values thereof were taken as an average of the latest 12 months at the time of modeling. The period of 12 months was chosen in order to avoid just capturing a snapshot of the system, and to average out seasonal effects. However, the length of the period can be considered for modification when doing similar modeling in the future, and should fit the needs and goals of the modeling. For instance, if a certain seasonal effect is of particular interest in a study, then that should affect how the period is chosen.

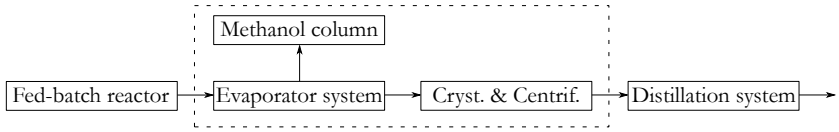



**Figure 1.** A schematic overview of the modeled process. The modules within the dashed box have been produced during the course of the project.

### *2.2.1 Major considerations during modeling*

In this section, some salient points that were considered during the project will be highlighted. These include the Maurer kinetics, pressure driven vs flow driven simulations in Aspen Plus Dynamics, as well as the mixture density regressions that were carried out.

*Maurer kinetics for formaldehyde oligomerization*

Formaldehyde was present as a component in parts of the separation system, and if the oligomerization thereof is not accounted for, the computational accuracy of simulations will suffer greatly. This motivated the use of a model for vapor-liquid equilibrium of multicomponent mixtures containing water and formaldehyde (Maurer, 1986), which AspenTech has implemented in a package and made available at the aspenONE Exchange. The package utilizes SYSOP7K as property method and NRTL as base method. However, to increase computational performance, a simplification was made in that the package's reaction set was reduced to only the first four reactions, which include the methylene glycol reaction as well as polyoxymethylene reactions for $n \leq 4$ as presented by Maurer (1986).

*Pressure/flow driven simulations*

During modeling, the question whether to use pressure-driven or flow-driven dynamic simulations was considered. The two methods are different in how they calculate flows in the simulation. Essentially, the flow-driven dynamic simulation calculates flow rates from simple mass balances - what goes in either accumulates in or leaves a vessel - and the governing specification is the mass flow rate into a model as a whole. This makes for rather simple and effective modeling, and is expected to be accurate especially for processes where the flow rates and pressures are not affected by downstream conditions, i.e. can be set by pumps and compressors (Yang et al., 2009). The flows in the pressure-driven dynamic simulation are, as the name suggests, driven by the fact that there are different pressures at different points of the process. The pressure-driven is generally considered more rigorous than the flow-driven mode, as hydraulics and fluid mechanics

are taken into account (Luyben, 2002). The flows are thus always calculated (and never set) from pressure differences. This means that modeling becomes more difficult but also more realistic since all unit operations need to be separated by a pressure changer, e.g. a pump, compressor, or a valve. Due to this, convergence during simulations is expected to also become more difficult. However, whilst uncertain which method will produce the model with greater fidelity, it is suspected that the added complexity of the pressure-driven simulation will make it possible to account for more situations, e.g. the occurrence of back-flows. The two methods thus carry different advantages and disadvantages, and it is important to find a good balance between the difficulties brought on by added complexity and the pay-off of added complexity. Since the improvements that a pressure-driven model would bring were in themselves difficult to ascertain at the time of modeling, and due to some initial problems with convergence, the decision was made to simplify the modeling and work with flow-driven models.

However, there was – and in general, there is – a need to account for some pressure-driven phenomena in the flow-driven simulations. For example, consider the vapor stream of some condensation vessel in a plant, which may or may not be controlled using a PI controller. Even in the latter case though, the vapor stream flow rate will essentially be limited by the conditions in the vessel and the pressure drop caused by the piping. This may in turn be impacted by the temperature, composition, pressure and mass flow rate of the material entering the vessel, as well as the pressure drop in the piping system downstream on the vapor side, which combined will induce a mass flow of vapor out of the vessel. In APD on the other hand, whilst the outgoing flow would be correctly calculated and thus adjusted if prompted by a change in the vessel conditions in the pressure-driven simulation, it may become a problem when choosing a flow-driven simulation. We noticed during the project that when a module has internal recycling loops, i.e. with internal tear streams, the vapor stream mass flow rate leaving the condensation vessel will be set to a constant value. This is probably a decision made by the vendor in order to deal with the tear streams. However, using a constant value for the outgoing vapor stream may result in extremely unrealistic pressure gains in the simulation if accumulation of vapor becomes extreme, and this is an error that will propagate (potentially unnoticed) throughout the system. However, this can be remedied by using PI controllers, and luckily, Aspen adds these by default in the conversion process from a static to a dynamic simulation. As found during the work behind Nolin et al. (2018), it is important to keep these PI controllers, and to use them in order to capture the pressure-driven phenomena of the vapor stream flow rates in flow-driven simulations.

*Mixture density calculation modifications*

The default way in which Aspen calculates mixture densities when using the NRTL physical property set is by going via the calculation of the liquid molar volume applying the Rackett equation directly. This is implemented in Aspen in the route called VLMX01. For details regarding calculation routes, please refer to the Aspen Plus V10 manual.

However, it was discovered that this method was not accurate for mixtures containing sodium formate (SoF). Therefore, an investigation in the effectiveness of changing routes and performing regressions using experimental data was performed, and the

conclusions were to perform regressions on a route called VLMX26. The underlying method of VLMX26 for mixtures is to calculate the density by using the mole-fraction average of the pure components' liquid molar volumes given by the Rackett equation. It can thus be considered to be an effective extension of VLMX01.

VLMX26 can be successfully fit to data by performing a standard data regression in Aspen Plus. This is done by inserting the desired data-set into Aspen, and then selecting Regression as the Run Mode, followed by creating a Regression case for the parameter named DNLCOSTD's all five elements. This needs to be done for all required components, and was thus done for both TMP and SoF. The Parameters tab for the TMP regression is shown as an example in figure 2.

Parameters to be regressed

| Type | Parameter | Parameter | Parameter | Parameter | Parameter |
|---|---|---|---|---|---|
| Name | DNLCOSTD | DNLCOSTD | DNLCOSTD | DNLCOSTD | DNLCOSTD |
| Element | 1 | 2 | 3 | 4 | 5 |
| Component or | TMP | TMP | TMP | TMP | TMP |
| Group | | | | | |
| | | | | | |
| | | | | | |
| Usage | *Regress* | *Regress* | *Regress* | *Regress* | *Regress* |
| Initial value | 0,375978 | 0,46523 | 818 | 340 | 777,1 |
| Lower bound | -3,3838 | -4,18707 | -7362 | -3060 | -6993,9 |
| Upper bound | 4,13576 | 5,11753 | 8998 | 3740 | 8548,1 |
| Scale factor | 1 | 1 | 1 | 1 | 1 |
| Set Aji = Aij | *No* | *No* | *No* | *No* | *No* |

**Figure 2.** The SoF-water mixture density: The parameters used for the regression.

In the upper subplot of figure 3 the experimental data for the density of SoF-water mixtures is presented along with the calculated values using the default route VLMX01 (not regressed), and the selected route VLMX26 (regressed). In the lower subplot in figure 3, the residuals from the upper subplot is presented. It can be seen that selecting VLMX26 and performing a data regression provided great benefit to the accuracy of the density calculations. These modifications are used in all modules described below.

The same type of results but for TMP are presented in figure 4, where the VLMX01 shows greater fidelity than for the SoF-water case, but still performs worse than VLMX26. Due to this, VLMX26 was chosen as the route for calculating liquid molar volumes for mixtures, and thereby densities.

### *2.2.2 The evaporator system*

The final dynamic model is presented in figure 5 below. It should be noted that this module makes use of the Maurer reactions described in section 2.2.1.

The buffer vessel after the reactor, B1, was decided to be included in the model even though it is not part of the PID, since it contributes with important dynamics in real life. The dynamics were modeled using a task setting the incoming flow from the reactor to be intermittent with respect to the liquid level in B1, reflecting real life operation. The following pre-heater train actually using hot streams from the evaporator system was in the module instead modeled using a *Heater* block for the sake of simplicity. This is followed by a pump.

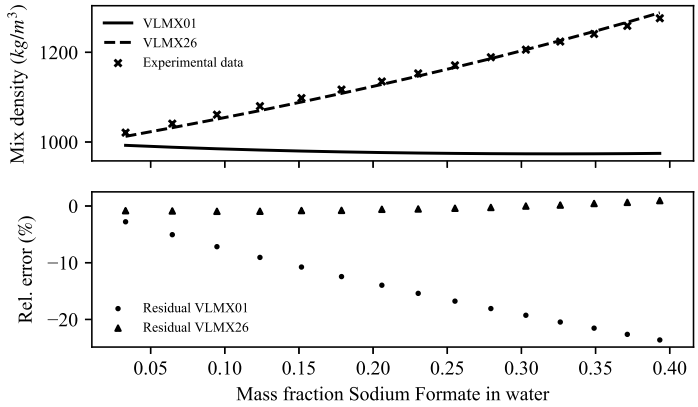



**Figure 3.** The SoF-water mixture density: experimental and the two Aspen routes VLMX01 (default) and VLMX26 (selected route).

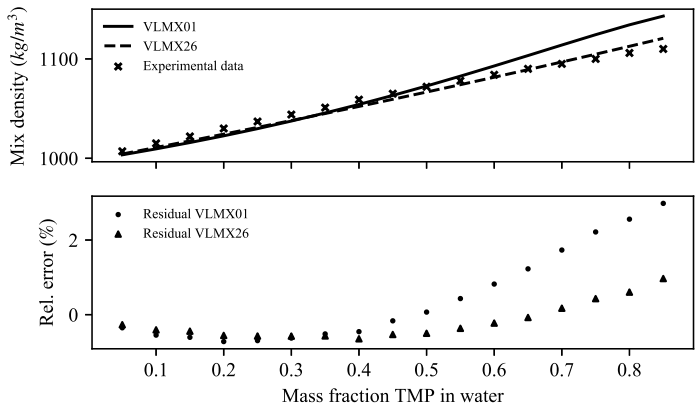



**Figure 4.** The TMP-water mixture density: experimental and the two Aspen routes VLMX01 (default) and VLMX26 (selected route).

The stripper, B2, was modeled using a *RadFrac* block using an estimated number of 13 theoretical stages. This does not include a condenser (which there is none) nor a reboiler (which technically is the first evaporator stage). The first evaporator stage, B1, is modeled in two parts. Firstly, by using a *HeatX* block taking the bottom stream from the stripper, heats it using the MVR-loop steam, and feeds it to the second part, which is a *Flash*2 block used to model the evaporating part of the first stage. Part of the vapor stream is passed back to the bottom of the stripper, and a part is relayed to the methanol column.

The second evaporator stage, B3, was modeled in a similar fashion as to the first

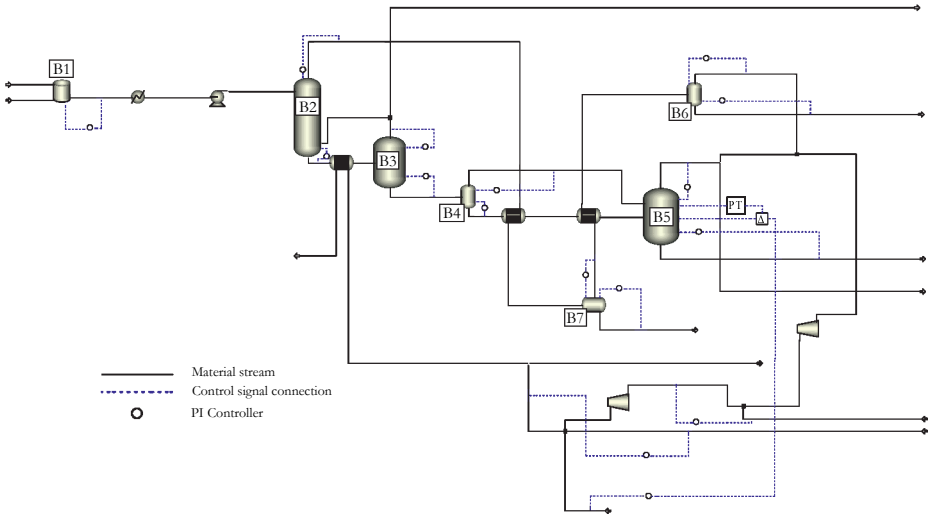



**Figure 5.** The two-stage evaporator model in Aspen Plus Dynamics.

stage, but is modeled using two heat exchangers, which was done in order to capture the fact that we draw condensate at different stages. The evaporator system is controlled by regulating the steam consumption via the boiling-point elevation (BPE) in the second stage. The second stage BPE is modeled using a so called *SteamPtoT* block and a *Delta* block. The *SteamPtoT* calculates the saturation temperature of steam with respect to the current pressure in the second stage, information which is fed to the *Delta* block together with the current temperature in the second stage. The BPE is calculated as the difference between these values, and forwarded to the PI controller regulating the added steam at the bottom of figure 5.

The MVR system was modeled using *Compr* blocks for the two compressors, and *Mixer* blocks for the mixing vessels in the MVR loop, where steam/water is added. The control using the intermediate and final temperature of the MVR loop is modeled using PI controllers regulating the flow rate of steam/water added to the loop.

Several streams in figure 5 are directed to the methanol column module. In fact, all outgoing streams in figure 5 save for the B5 bottom stream and the hot stream leaving the B3 heat exchanger are intended for the methanol column module. This makes for four streams going to the methanol column module: the bottom stream of B7, the bottom stream of B6, part of the top stream of B5, as well as part of the top stream of B4.

### *2.2.3 The methanol column*

The final dynamic model of the methanol column is presented in figure 6 below. It should be noted that this module does not make use of the Maurer reactions described in section 2.2.1, as the presence of formaldehyde was neglected and disregarded; besides this aspect, the physical properties remain identical to the evaporator system. However, the streams from the evaporator module actually do contain formaldehyde, which motivated the use of a translation module to filter the component from the input streams.

The translation module is described in subsection 2.2.5.

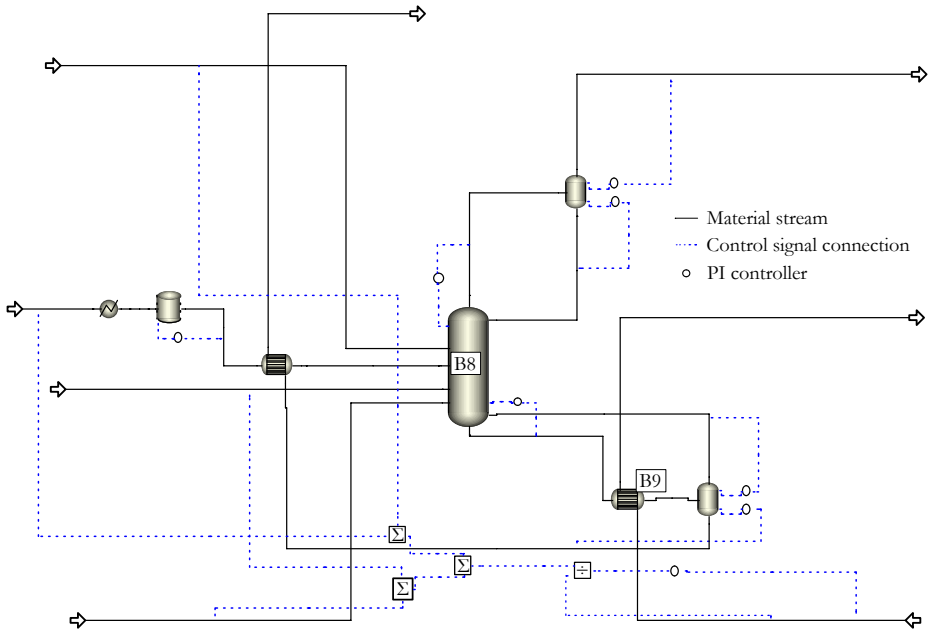



**Figure 6.** The methanol column module in Aspen Plus Dynamics.

Essentially, the system is modeled according to Sundqvist (2014). Having passed the translation module, the input streams from the evaporator system are directed onto the corresponding stage of the methanol column, B8, which in turn is modeled using a *RadFrac* block. The reboiler, B9, is modeled as a *HeatX* block followed by a *Flash*2 block, and the condenser is modeled using only a *Flash*2 block. The amount of steam used in the reboiler is controlled, calculated using a ratio between the added steam mass flow rate and the total mass flow rate of the streams from the evaporator. This is modeled using the *Sum* block as well as the *Ratio* block available in APD.

### *2.2.4 The crystallizer and centrifuges*

The final dynamic model is presented in figure 7. It should be noted that this module does not make use of the Maurer reactions described in section 2.2.1, as the presence of formaldehyde was neglected and disregarded. Furthermore, a solubility data regression was performed for this model - it was similar to the previously described density regression, but considerably less complex.

The product stream from the evaporator system, i.e. the B5 bottom stream, is mixed in B10 with many different streams, which are modeled as only one stream as seen in figure 7. The crystallizers are then modeled using a combination of *HeatX* and *Flash*2 blocks, similar to how the evaporator system was modeled. The centrifuges, B11, are modeled using a shortcut method with the *Sep* block, since APD v10 cannot handle particle separation; thus, they need further work to have proper dynamic responses.

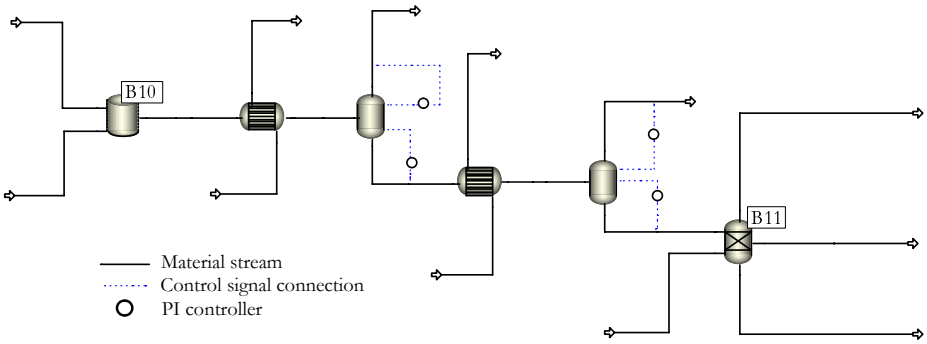



**Figure 7.** The crystallizers and centrifuges module in Aspen Plus Dynamics.

Water is also added to the centrifuges to account for the washing procedure taking place there.

In order to focus on the important aspects of crystallization, the underlying physical properties used for this module differ significantly from both the evaporator system and methanol column modules. The Maurer oligomer components were removed from the component list, and instead the solid form of sodium formate and the respective ions were added. Furthermore, the appropriate salt reaction was implemented using the AP tool for the purpose, which is important to account for the boiling-point elevation due to the presence of salt. Through this, the model was able to replicate the mass and energy balances from the real process to a satisfying degree in APD, with a model that works in APD.

### *2.2.5 The evaporator-methanol column translation module*

The translation module for the evaporator system and the methanol column, presented in figure 8, was created in order to filter out formaldehyde from the streams, the mere presence of which otherwise would make the methanol column model deviate from the real process. The translation module is very simplistic, and consists of one *Sep* block per stream that is to be connected between the modules. The stream names are given according to the stream naming convention, i.e. identical to the streams found in the respective upstream and downstream modules.

By default, this setup will showcase unexpected behavior, where it will fail to pass along the correct temperatures and pressures to the outgoing streams. However, this can be remedied by adding what AspenTech calls constraints to the simulation, which is done by using the simulation explorer in APD to find the constraints editor (shown in figure 9). For each of the four individual *Sep* in the translation module, the following code has been implemented:

```
1 CONSTRAINTS
2  // Flowsheet variables & equations...
3  BLOCKS("B12").T("S-B3-BP") = BLOCKS("B12").In_F.Connection("S-B3-1").T;
4  BLOCKS("B12").T("S-B3-2") = BLOCKS("B12").In_F.Connection("S-B3-1").T;
5  BLOCKS("B12").P("S-B3-BP") = BLOCKS("B12").In_F.Connection("S-B3-1").P;
6  BLOCKS("B12").P("S-B3-2") = BLOCKS("B12").In_F.Connection("S-B3-1").P;
7 END
```

Here, and as shown in figure 8, B12 is a *Sep* block, which receives stream S-B3-1 and separates the formaldehyde into a bypass stream (S-B3-BP) and a stream directed to the methanol column module (S-B3-2). By setting up the code in the listing above, the expected and desired behavior that the temperature and the pressure of the outgoing streams will be equal to that of the incoming stream. In other words, this ensure that nothing but the formaldehyde separations happens in this module. This will surely be necessary to implement in similar translation modules that are to be created in the future.

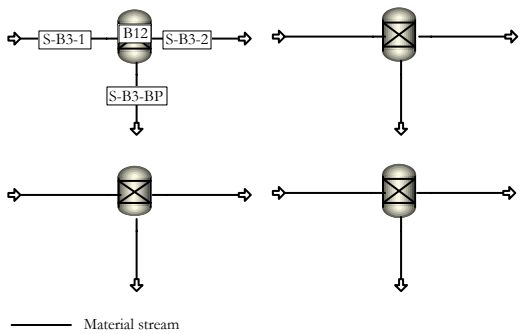



**Figure 8.** The evaporator system-methanol column translation module in Aspen Plus Dynamics.

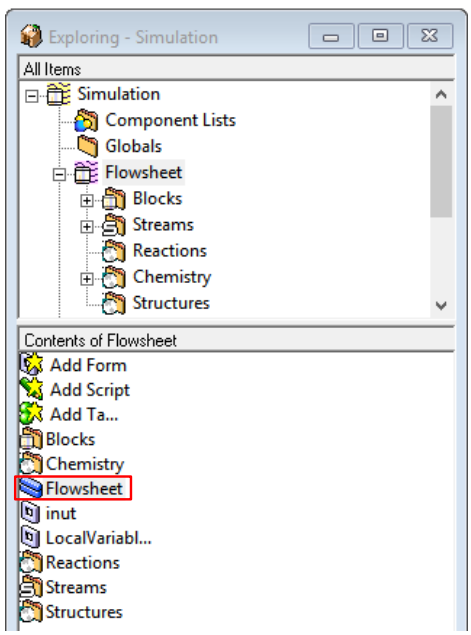



**Figure 9.** The evaporator system-methanol column translation module in Aspen Plus Dynamics.

## 3. Simulation

The development of the Python-Aspen tool-chain for modeling and simulation began with the desire to harness the power of two worlds that would complete each other. Process flowsheet simulators such as Aspen Plus (AP) and APD are very powerful modeling tools and relatively simple to use, given the large libraries of both unit operations and physical properties, and that the user can point-and-click whilst being guided through each step. However, performing model-based studies using the internal methods of AP or APD demands significant knowledge of the inner workings of the software. This is a limiting factor to a practitioner used to working with more general methods and state-of-the-art algorithms that can easily be used to automate the studies. Using an external programming language such as Python for simulation control and automation to overcome this obstacle would thus enable the combination and simultaneous utilization of the best of the two worlds - ease of modeling combined with simplified/generalized, powerful model-based studies.

The Python-Aspen tool-chain began as a set different tools applied together but evolved into the Python Module Coupler, which has come to encompass the whole tool-chain including co-simulation capabilities, presented schematically in figure 10. This chapter describes the building blocks of PyMoC, i.e. the utilized software along with basic instructions on the most important modifications and aspects of use, before moving on to the details and considerations of PyMoC itself.

It may here be noted that there does exist alternatives to the modeling-simulation approach described in this work (i.e.modularization and co-simulation using PyMoC or other tool-chains like it). For instance, utilizing the built in *Section* functionality of AP allows a user to create models that can hold different component lists, whilst still having only one single all-encompassing flowsheet. However, this demands some specialized training in the use of AP, whilst using PyMoC requires rather more basic knowledge of AP. Both approaches still require some degree of knowledge of Python in the case that e.g. optimization studies are of interest. The choice of approach may thus come down to the available resources for a specific project; in the ProOpt project, the focus became oriented towards a Python-heavy solution.

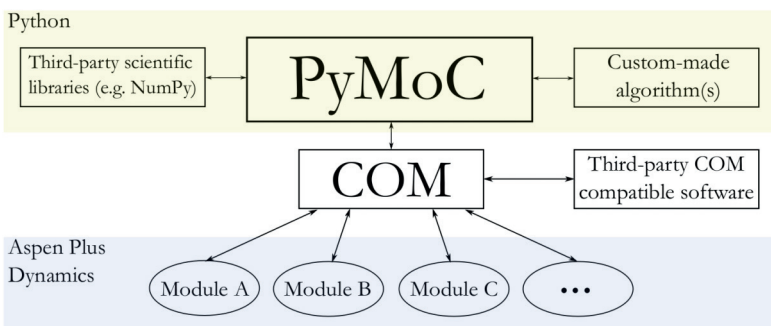



**Figure 10.** The COM enabled PyMoC tool chain.

## 3.1 Utilized Software

### *3.1.1 Dynamic Flowsheeting Software*

Flowsheeting simulators are tools that can be used to conveniently model and simulate the important macroscopic phenomena in a complex production process (Haus et al., 2018). Dynamic flowsheeting allows for investigations of transient scenarios, in contrast to steady-state flowsheeting, which is not able to replicate the time-dependent behavior of a real-life process (Skorych et al., 2017). This makes dynamic flowsheeting a key tool when modeling and simulating for instance (semi-) batch systems, start-up and shut-down sequences, or unexpected disturbances. Insights gathered from such models can in turn be used to understand e.g. how plant improvements can be implemented, what effects a certain change will have on a process, or for training operators (Mikkonen et al., 2017). Its usefulness has led to dynamic flowsheeting being applied more frequently in fields like computer-integrated manufacturing as well as real-time optimization (Dimian et al., 2014).

A variety of dynamic flowsheeting software is available, both commercially as well as open-source. APD was used as the flowsheeting program during development of the PyMoC algorithm. APD is a powerful dynamic flowsheet simulator based on steady-state models created in Aspen Plus, which means that it can leverage the wide range of components, physical properties, and unit operations available with the software. This has been done in several research studies such as Alobaid et al. (2015a), Alobaid et al. (2015b), Ahmad et al. (2016), and Mikkonen et al. (2017) applying Aspen Plus Dynamics, and others utilizing Aspen Plus (Corbetta et al., 2016).

PyMoC is flexible in terms of choice of process flowsheet simulator due to the application of the Component Object Model (COM) standard, meaning that any COM compatible software should be applicable for use instead of Aspen Plus Dynamics (potentially after modifications for certain function keywords, e.g. to run the simulation, depending on differences between software). The range of potential replacement software includes process modeling environment software that make use of the interoperability that the CAPE-Open standard provides, since this is based on COM/CORBA (Soares and Secchi, 2004). Examples of other potential flowsheeting programs include gPROMS by PSE, and UniSim Design by Honeywell (CAPE-OPEN Laboratories Network, 2018). All in all, this means that PyMoC carries potential for generalizability through customizability.

### *3.1.2 Python*

Python is an object-oriented, high-level programming language combining power with readable syntax (Python Software Foundation, 2018b). Being open-source, the engine itself as well as all third-party modules/packages are free to use and distribute even commercially (Python Software Foundation, 2018a), and this has led to a significant following in both engineering and scientific communities, with a range of success-stories as testament (Python Software Foundation, 2018c). Furthermore, as Shukla and Parmar (2016) states, there are convenient software suites available to support users in installing and maintaining an updated system of packages, e.g. Anaconda (Anaconda, Inc., 2019). Employing such software suites allows the user to focus on value-adding activities such as process modeling and model-based studies.

The main advantages of utilizing Python in a research context include that it is easy to learn, it is platform independent and thus allows for teamwork across large teams with a heterogeneous system architecture (e.g. a mixed setup of MacOS, Windows, and Linux), and that it is open-source with countless third-party modules freely available for application in any custom-made algorithm. This makes it an efficient and low-cost alternative to consider as e.g. a 'glue language', or even the main language in a programming-heavy project.

The software suite Anaconda with Python 2.7 was used for constructing the Python-Aspen tool-chain. This simplified the creation process since all of the necessary packages to run PyMoC are included. A list of the necessary packages and their respective main functions in the PyMoC algorithm are presented in table 1.

**Table 1.** A list of the necessary packages for PyMoC to run.

| *Package* | *Main function for PyMoC* |
|---|---|
| NumPy | Numerical computations |
| matplotlib | Graphical representation of data |
| win32com | Communication between Python and Aspen Plus Dynamics |
| sys | Printing exit messages |
| datetime/time | Timing calculations and simulations |

An important aspect of saving the user time and effort during co-simulation using APD and PyMoC is the ability to have the algorithm automatically load the modules of which file paths have been provided by the user. However, the *win*32*com* package of Python 2.7 cannot do this by default, due to what must be considered a bug in the error handling system of said package. As such, the user needs to perform one modification to the code of the *win*32*com* package before using properly PyMoC the first time.

During initialization, PyMoC will invoke the *OpenDocument* function of the Aspen Plus Dynamics COM object that the algorithm has established. Unless *win*32*com* has been modified (either by versions released after the time of writing, or by a local user), this will throw an error, and provide an error code to the user. In order to have PyMoC automatically load the modules, this error code needs to be included in the list of 'bad context' errors in the file *dynamic.py* found in the *win*32*com* client installation folder. The installation folder is by default named something to the effect of: "C:/Users/[USERNAME]/Anaconda2/pkgs/pywin32-[...]/Lib/site-packages/win32com/client" (provided Anaconda with Python 2.7 is used as a software suite). The file *dynamic.py* should be modified by adding the error code to the list called `ERRORS_BAD_CONTEXT`, beginning on row 38. Following this modification, the automated loading of module files will function as intended, and will save lots of time for the user.

Python can create a COM object of the APD flowsheet. What this metaphorically does is that it allows Python to open up the gates of the APD flowsheet, to navigate freely along a structure similar to a folder tree using dot notation. To show how the dot notation navigation works, a simple model is presented along with part of its COM object tree in figure 11, and the accompanying listing shows the code to find the duty

for the block HEX and the temperature of the stream SI-1. This is the general way of navigating through dot notation, which depends on the names given to the object within the flowsheet. Unfortunately, Python is forced to do the navigation blindly since there is seemingly no function to reveal actual paths or functions that are available. This is where the practitioner needs to have the relevant module visible on screen, at least in the beginning, to be able to figure out “where to go next”. The positive thing here is that most things that are visible in the flowsheet are accessible via dot notation simply by following the structure with the same names that are showing in the flowsheet, as shown in figure 11 and the accompanying listing. This means that once the COM gate is opened, it is not only possible but - when the method is understood and second-nature - it is actually rather convenient to access everything in the flowsheet without any additional work required, such as creating or activating tags (which is actually required when using Aspen OTS and OPC - for every single potentially interesting variable).

```
# Establishing the COM object in Python, naming it adyn
adyn = InitCOM()
# Retrieving the duty of HEX
Q = adyn.simulation.flowsheet.blocks("HEX").Q.value
# Retrieving the temperature of SI-1
T = adyn.simulation.flowsheet.streams("SI-1").T.value
```

It should be noted that accessing options related to the simulation engine etc. whilst possible may be difficult. The built-in Help in APD (accessible via F1) is often of great use, and shows often more or less exactly how to navigate using dot notation, similar to when navigating the flowsheet through dot notation - but so is not always the case. Part of the reason is because it is actually intended towards the use of Visual Basic and Excel, and another part of it is simply general ambiguities (or perhaps even errors). Sometimes, some imaginative guesswork and trial-and-error is necessary to reach the desired final destination through dot notation.

## 3.2 The Python Module Coupler - PyMoC

PyMoC is a very useful Python-Aspen tool-chain that allows for any conceivable model-based study possible since it is compatible with other third-party Python

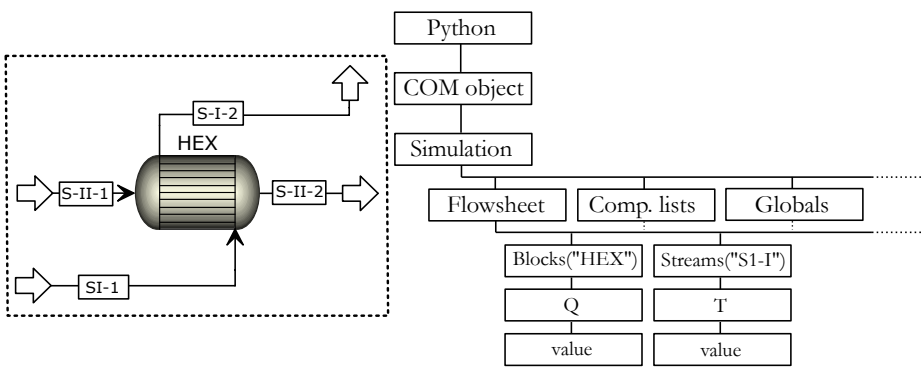



**Figure 11.** The simple model together with part of its COM object tree.

packages as well as custom-made wrappers. It further sports co-simulation capabilities, and is able of orchestrating the simulation of several Aspen Plus Dynamics modules. As such, it fills the needs of the ProOpt project, which was to control simulations of the models that were decided to be modularized through the implementation of a co-simulation strategy.

Co-simulation involves the coordination of the independent dynamic simulation of several models that have been developed and implemented separately from each other (Steinbrink et al., 2017), and can be used to analyze and optimize the overall system (Zitney, 2010). Regardless of the motivation for dividing an overall system into smaller pieces (covered in chapter 2), the need to connect and co-simulate modules remains a challenge (Andersson, 2016). Bulian and Cercos-Pita (2018) argue that, in general, a co-simulation strategy has three parts: (i) the coupling, (ii) the communication layer, and (iii) the implementation. The coupling refers to how the data exchange between the different models is performed during the simulation runtime. For instance, Bulian and Cercos-Pita (2018) themselves utilize two simulation environments with significantly different time steps, which requires a special strategy of moving one environment temporally ahead of the other and then transfer data 'back in time' to the lagging model. The communication layer is simply how the models are linked, which can be performed through e.g. dynamic linking or a network connection. Finally, the implementation concerns how the co-simulation algorithm is actually implemented with regards to e.g. language and data flows.

Working with APD specifically, there is the option of utilizing the Aspen Operator Training Simulator (OTS) Framework as a co-simulation tool. However, Aspen OTS is somewhat arduous to use as it demands 'manual labor' (i.e. tedious pointing-and-clicking) to set up a co-simulation, something that has to be repeated when a module is changed, with no obvious way to automate this. It is also quite a task to setup inputs/outputs and retrieve these, as the OTS/OPC combination explored during the project demands manual setting up of all interesting variables before usage. This is something that PyMoC does better, as the COM functionality (as previously described) is much more convenient. Furthermore, adding custom-made tools for advanced model-based studies is also difficult as there is no clear way to latch on to the OTC/OPC setup. Finally, the general lack of published research using the Aspen OTS (Ahmad et al., 2016), especially for advanced studies of co-simulated models, is interpreted as another testimony that it might not be an appropriate tool for the purpose.

The purpose of this section is to present PyMoC, beginning with an in-depth explanation of the considerations behind PyMoC in section 3.2.1, on the form of the three aforementioned main parts of the co-simulation strategy (Bulian and Cercos-Pita, 2018). In connection to this, the four advantages (Felippa et al., 2001) are discussed as well, and they are considered to be important to achieve in order for the tool to be useful. A use-case example of co-simulating an illustrative modularized model using PyMoC is presented in in section 3.3.

### *3.2.1 PyMoC algorithm details*

The purpose of the PyMoC tool is to minimize the amount of manual labor required of the user in order to connect modules of a full model, as well as to grant them full control and customizability for further advanced model-based studies. By these means,

the advantages of modularization, customization, independent modeling, and software reuse presented by Felippa et al. (2001) are fulfilled.

Firstly, modularity is achieved inherently by dividing the overall system into smaller models, which can be modified independently in turn allowing for differences such as the aforementioned physical properties. Also, by making small changes to several modules, a new module could be implemented, having been developed independently. Secondly, PyMoC allows for customization of the different Aspen Plus Dynamics models, as they can utilize different numerical solvers and methods that are available in the runtime environment. Thirdly, PyMoC supports independent modeling. Part of this can be exemplified by the free usage of physical properties most fitting the simulated scenario, e.g. the usage of the formaldehyde model by Hasse and Maurer (1991) as a global chemistry in a pertinent evaporator model but switching to a salt precipitation global chemistry in a crystallizer model that follows. Finally, keeping the modules apart also allows for the possibility of developing modules at different production sites of a company. Sharing these modules to other production sites then gives rise to software reuse.

All of this is achieved partly by virtue of its Python implementation, and by taking advantage of the strengths of dynamic process flowsheet modeling with Aspen Plus Dynamics. Furthermore, and more specifically, the use of COM in combination with the application of a very simple stream naming convention together allow for effective communication, along with the coupling applying a piecewise-constant data exchange between the different modules.

This section will be structured around the three parts of co-simulation strategy that Bulian and Cercos-Pita (2018) reported, namely coupling, the communication layer, and the implementation.

*Coupling*

Data transfer between the different modules during co-simulation is of the utmost importance, since without communication they will only be run individually and independent from each other, which is the opposite of what co-simulation is intended to achieve. However, communication must be synchronized to avoid ambiguities between the data used for calculations in the different modules (Walker, 2018). According to Oppelt et al. (2015), synchronization can be achieved by coordinating the running of modules in a step-by-step fashion and having data exchange be performed at discrete time intervals. This is further very similar to conclusions drawn from the Base Parallel Algorithm that Andersson (2016) presents. Thus, a strategy was developed to couple modules and exchange data during PyMoC runtime. This is based on synchronization points (SPs) at the beginning/ends of the simulation time horizons ($\tau$), in combination with piecewise-constant data transfer.

The synchronization points are similar to those proposed by Oppelt et al. (2015) and Andersson (2016) in that the algorithm will make all the modules take one global time step, which means that the SPs will be at the beginning of a given simulation time horizon, and the end of the preceding. During co-simulation, PyMoC will pause modules at an SP - i.e. after a simulation time horizon - and wait for all modules to arrive at the same SP before data transfer occurs. This method forces quick modules to wait for slow ones, which will have consequences if there is a significant difference in computational time for the different modules. Essentially, the method makes the whole

co-simulation run at the speed of the slowest module, which could pose a disadvantage, but it is at the same time a very straightforward way of synchronizing the simulations. The data transfer is performed at the SP, and signals between different modules are kept constant over the course of one simulation time horizon, i.e. the data transfer is piecewise-constant. The strategy as described is depicted schematically in figure 12.

The piecewise-constant data transfer means simple implementation and is the same concept as zero-order hold (ZOH). ZOH has classic applications within digital control of dynamic systems (Sun et al., 1992) and has also during recent years been successfully applied to open-loop control of chromatographic separation processes (Sellberg, 2018). Since control issues like these are concerned with transferring some input signal at the right time, the zero-order hold analogy (ZOHA) for the similar purposes of PyMoC.

There are two drawbacks inherent to the piecewise-constant data transfer that need to be considered, though; (i) loss of resolution and stability since the input/output of the modules is not transferred instantaneously, and (ii) introduction of discontinuities into the simulation that may propagate throughout. Loss of resolution can be remedied through choosing a smaller simulation time horizon for the global time step; similarly, the stability issue may require a small communication interval. This will also affect overall co-simulation speed, as more pauses for data transfer means more intervals where modules will be paused, which is a trade-off to consider. In other words, this allows the user to make their own decision regarding whether to prioritize simulation resolution or speed. Furthermore, the smallest time step (by default) in Aspen is 0.01 hours, which makes the smallest available global time step 0.02 hours for the same settings, since a module needs to take a minimal time step in order to have changes calculated and then propagated in a timely manner throughout the co-simulation. It should also be noted that the issue of stability may actually hinge on the modeling, and more precisely, the module boundaries. These boundaries should be considered carefully so that model sections with tightly coupled dynamics/physics are not disrupted beyond recognition by the PyMoC communication intervals.

The introduction of discontinuities is a more difficult problem to solve, and could require another strategy entailing some kind of horizon-variant data transfer. Another possible way to solve it is by using some kind of correction strategy, similar to the coupling strategy of Bulian and Cercos-Pita (2018); however, since the different Aspen modules would need to be restarted (or rewound), a correction strategy may lead to significantly extended simulation times. Regardless, discontinuities is an area of PyMoC that has room for improvements in the future, e.g. by adding another data transfer approach as an alternative to the ZOHA.

*Communication Layer*

There are two main parts of the communication layer of PyMoC. The first part is the communication interface known as Component Object Model (COM). COM is a platform-independent and object-oriented application binary interface (i.e. an ABI) (Microsoft, 2018). This means that COM allows for software written in different languages to connect and communicate directly, and that it can be implemented in any language and for any operating system, according to Ungerer and Goodchild (2002). In turn, using COM for PyMoC therefore increases its usability by allowing it to be coupled with other software available on other platforms or in other languages besides

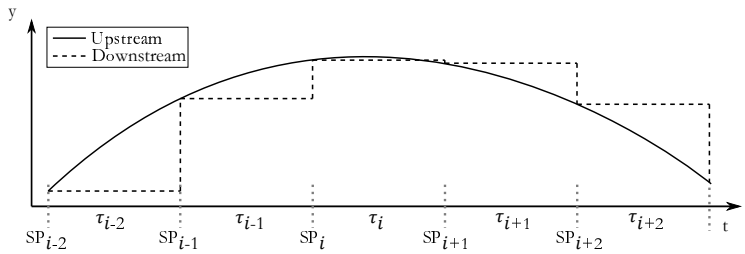



**Figure 12.** A schematic of the zero-order hold analogy during PyMoC runtime. A variable, *y*, with respect to time, *t*, is common between the two modules and therefore transferred from an upstream to a downstream module at a sync point (SP) and then kept piecewise-constant in the downstream module over a simulation time horizon ($\tau$).

Windows, Python, and Aspen Plus Dynamics. This fits the purpose of PyMoC very well, as it is intended to give users the possibility of adding any other customized software or third-party algorithm for advanced studies; COM's ability to do this may be exemplified by Nolin et al. (2017) and Nolin et al. (2018).

PyMoC applies COM to establish a dynamic linking between different modules, and how the tool-chain connects the different software, libraries, and algorithms are is visualized in figure 10. More specifically, the Python package win32com is utilized in PyMoC, and the communication could to some extent be compared to how Aspen OTS utilizes Open Platform Communications (OPC), but without the necessity of having communicating via a server, such as Aspen OTS requires its OPC server. Instead, COM enables direct communication between the modules. This does carry with it one major practical difference for the user. Where utilizing Aspen OTS and OPC requires the user to tag any variable that would be interesting to study - be it a reactor temperature or a stream mass flow rate - the COM interface instead opens up a tree in which all parameters and variables existing in the flowsheet is automatically accessible through dot notation. This is an advantage since it allows for simple access to all variables and parameters without adding extra steps.

The second part of the communication layer of PyMoC is the stream naming convention established in the algorithm. This convention is a essentially a rule-set serving as a simple yet effective solution towards having the algorithm automatically connect streams from different modules. It makes the setup of the co-simulation effortless for the user instead of demanding manual, time-consuming pointing and clicking that may have to be repeated, e.g. due to remodeling or due to rerunning a slightly different setup in Aspen OTS.

The naming convention is basically implemented as part of the modeling, from the beginning, and can be expressed as the combination of two sub-strategies; the user needs to (i) specify a string by which interesting streams' names begin, and (ii) to give streams that are to be connected by PyMoC identical names. The algorithm will scan all modules for streams starting with the 'beginning string', and automatically label them as source (upstream) or destination (downstream) streams by whether the mass flow rate variable of the stream is 'fixed' or 'free', respectively. In Aspen Plus Dynamics, this signifies if a stream is supposed to be an input to a module, or calculated within

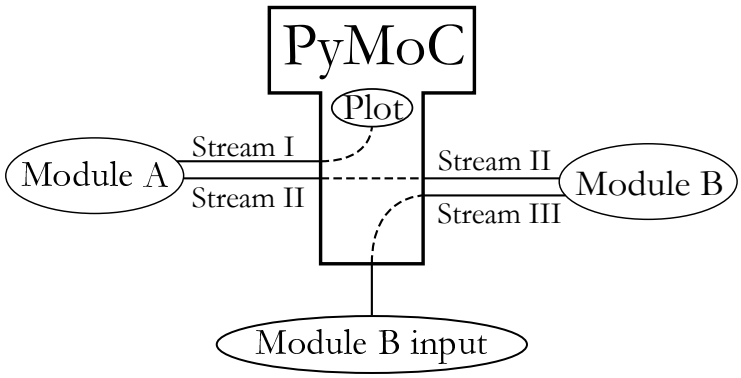



**Figure 13.** A graphic representation of how the naming convention functions; it may take a user-supplied stream and plot the results, it will automatically connect identically named streams, and there are possibilities to provide input to modules.

(and thus potentially, but not necessarily, an output stream). This string specification can also be used for highlighting important streams interesting to study, which might not be necessarily connected to another module. During co-simulation, the algorithm will keep track of the source and destination streams, and automatically transfer data (e.g. states such as temperature, pressure, flows, compositions) from the sources to the destinations that are identically named. The concept of identically named streams across modules is a reflection of reality as streams remain intact as one and thus only has a single name, which works well for modeling purposes where existing P&IDs or established rulesets/agreements for naming streams are available. The overall idea is represented in figure 13, where the output streams from module A, stream I and stream II, end up going different paths due to the fact that both are considered interesting (and thus marked by the user), but only Stream II has a corresponding stream in Module B. Therefore, the values of stream II are transferred to module B, whilst the results from Stream I are plotted. Furthermore, the algorithm is currently capable of handling fixed-value inputs to either module, which in figure 13 is represented by the module B input to stream III.

A drawback of the naming convention is that the engineer performing the modeling needs to make sure that two streams that are not meant to be connected are not named identically (e.g. through the use of the automatic ‘stream namer’ in Aspen Plus), which can pose an issue for large and complex models spanning several modules. However, this is not a major problem in practice due to the recognition of source and destination streams, and ability of Aspen Plus Dynamics to override changes in input to stream variables that are not ‘fixed’. This leads to the only streams likely to be affected by this issue are modules’ input streams meaning that there are relatively few streams that could be subjected to this, since all other streams are calculated within a module.

Furthermore, the automated model connection through the stream naming convention combined with use of COM enables PyMoC to connect to any Aspen Plus Dynamics model without the need for the user to tailor an interface between two specific modules. This means that PyMoC is a generic framework for co-simulation, according to the classification scheme presented by Steinbrink et al. (2017).

### 3.2.2 Algorithm Implementation

The information flow of the Python implementation, which requires the packages listed in table 1, is schematically represented in figure 14. The algorithm requires the user to input the paths to the .dynf files that are to be co-simulated. The user should also provide a time at which to end the simulation, and is also able to choose the simulation time horizon length. The PyMoC algorithm will then start one instance of Aspen Plus Dynamics per .dynf file, load all the modules, and establish the COM connection. Before simulation, interesting streams will be identified by their names, which should be named according to the naming convention, and establish the necessary connections between identically named streams. The simulation will then begin by performing the first transfer of states at $t = 0$, followed by simulation through to the first SP, where connections will be updated. This will repeat until $t = t_{end}$, at which time the co-simulation will be stopped, and data for the interesting streams will be gathered and presented to the user. It may here also be noted, as each APD instance is run as an independent process, parallel execution of the modules is possible and the simultaneous running of several modules is handled automatically by the entire setup.

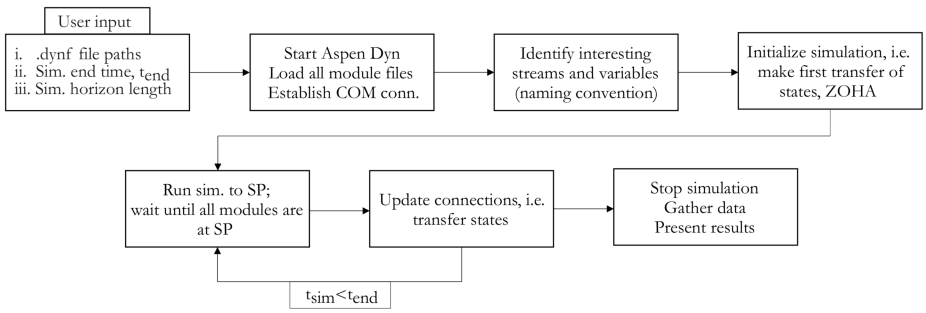



**Figure 14.** Schematic overview of the PyMoC algorithm.

## 3.3 Example of application - simulation

In this section the functionality of PyMoC will be demonstrated. To do this, an illustrative use-case example has been constructed, where a very simple modularized model will be co-simulated using the PyMoC - the model is intended to be very simple in order to not distract from PyMoC and to make it easier to follow the case. To further show the simplicity in utilizing PyMoC for advanced model-based studies, the modularized model will then be the subject of an optimization study.

The example model - the Aspen Plus Dynamics version of which is presented in figure 15 - is that of a simplistic separation of ethanol and water, and consists of a heat exchanger, modeled using the Aspen Plus *HEATX* block, and a flash vessel, modeled by the *FLASH2* block. The purpose of the heat exchanger is to use condensing steam ("SI-1") to heat an incoming 30/70 (mass) mixture of ethanol/water at 1.3 bar, 55 °C to 93 °C ("S-II-1"). The heated stream ("S-II-2") is mixed with a colder but significantly smaller stream ("S-III"), which is a 40/60 (mass) mixture of ethanol/water at 1.3 bar, and 45 °C. The mixed stream is subsequently fed to the flash vessel. The flash vessel

is two meters high and one meter in diameter, and operates at 1 bar. This is controlled by the PI controller "FLASH PC". There is also a liquid level controller, "FLASH LC", the set point of which is at one meter, i.e. half the height of the flash vessel. UNIQUAC was chosen as a property model, fully following the AspenTech flowchart for choosing a property method (Aspen Technology, 2016) for the process, which contains a polar, non-electrolyte mixture below 10 bar with binary interaction parameters available.

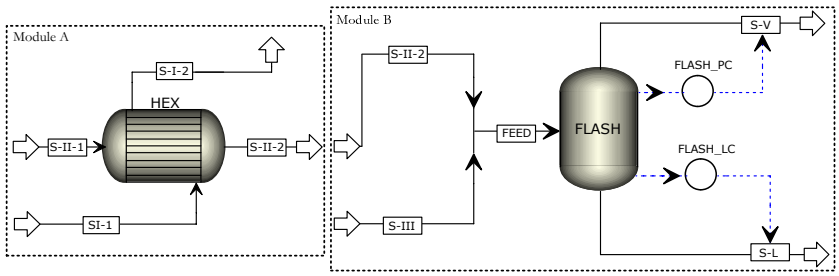



**Figure 15.** The simple Aspen Plus Dynamics model used for the demonstration.

The overall model reflects the structure of figure 13 in terms of stream names and module inputs/outputs. As presented in figure 15, the model has been modularized into module A and B. Module A's stream I ("SI-1" and "S-I-2" in figure 15) will not be transferred to module B, since there is no identically named stream in the latter. However, the outbound part (together with every other stream except for "FEED" and "SI-1") will be treated as an interesting stream according to the naming convention (here having been specified as streams starting with "S-"). Stream II leaving module A will be carried over to stream II in module B as they have identical tags, "S-II-2". Finally, stream III ("S-III") in module B is open for input from the Python environment as it is an input stream to the module.

In order to provide some interesting transient results, disturbances were programmed to occur during the co-simulation, in the form of sinusoidal disturbances in the mass flow rate, temperature, pressure, and composition of stream S-II-1. The results are presented in figure 16, and the way in which these disturbances propagate through the co-simulation and how the data transfer is performed over module boundaries is shown in figures 17 and 18, which show the model response for the stream leaving module A and entering module B (S-II-2), respectively.

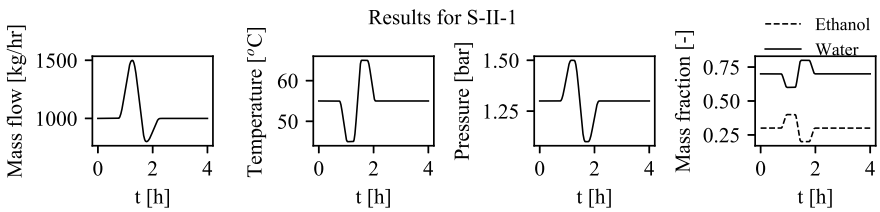



**Figure 16.** The results of co-simulation showing disturbances programmed to the stream S-II-1.

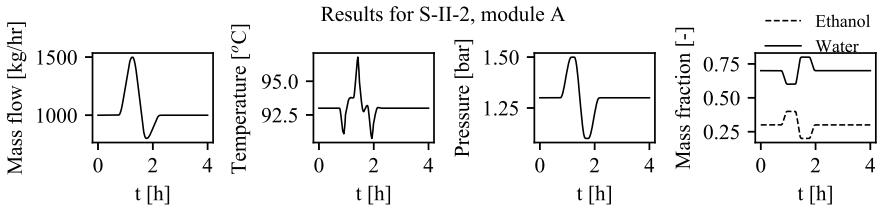



**Figure 17.** The results of the simulation showing the propagation of disturbances programmed to the stream S-II-1 for the module A outbound stream S-II-2.

The co-simulation applied a simulation time horizon, $\tau = 0.1$ h (black in figure 18), equal to 10 times Aspen Plus Dynamics' internal solver's default step. This yielded simulation times of roughly 1 minute per 4 simulated hours (Intel Core i7-6700 3.40 GHz, 16 GB RAM), and also helps to clearly show that the ZOHA strategy functions as intended, as it is the source of the curves' jaggedness in figure 18. Naturally, the curves would be smoother with shorter simulation time horizons, e.g. $\tau = 0.02$ h (red in figure 18), but this comes at the cost of longer simulation times since the transferring of data is performed during a brief pause in the simulation in order to keep the modules in synchronization. In this case, $\tau = 0.02$ h yielded simulation times of roughly 3 minutes for 4 simulation hours, i.e. a factor 3 increased simulation time compared to the longer horizon. Thus, this balance will become a question of priority, which the user may decide for themselves.

To demonstrate that S-III can be changed independently of the simulation results in process flowsheeting software (much like in figure 13), it was subjected to step changes that can be seen in figure 19.

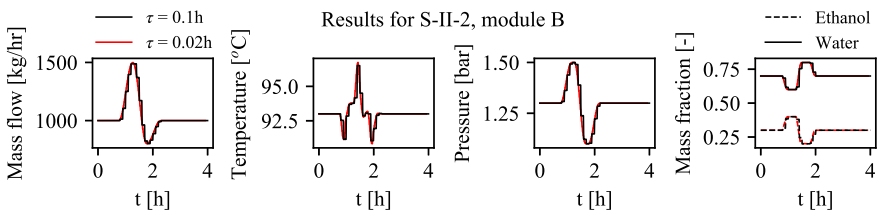



**Figure 18.** The results for the module B inbound stream S-II-2, to which data are transferred using the ZOHA strategy. This is visible especially for the case when $\tau = 0.1$ h, as it creates the jaggedness of the curves.

Finally, to show the effects of the ZOHA strategy on the final results, in figure 20 a comparison of the stream called "S-V" for $\tau = 0.1$ h and $\tau = 0.02$ h is presented. It can be seen that a lower $\tau$ will yield a finer resolution also for the end results, which is expected since there are more frequent updates to the input. However, the results do not differ greatly for this scenario, which indicates that the issue of discontinuities in general is not as prevalent as believed initially; it should be kept in mind that this may not be the case for other scenarios. Furthermore, error control in the manner that

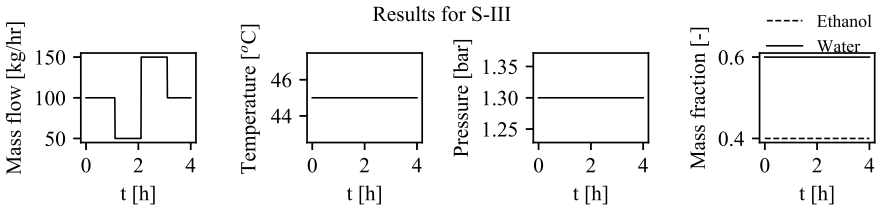



**Figure 19.** The results from the module B S-III stream, which had its input changed independently of the simulation.

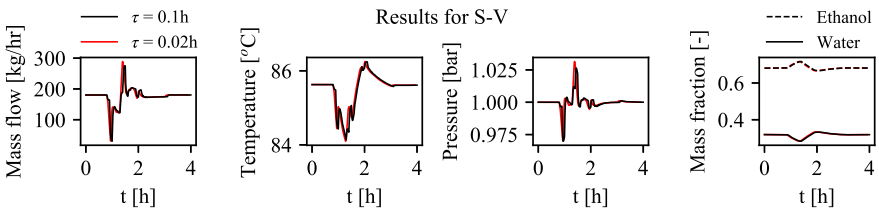



**Figure 20.** The results for the module B outbound stream S-V, for $\tau = 0.1$ h (black) and $\tau = 0.02$ h (red).

the Matlab ODE solver *ode*45 has is practically infeasible in PyMoC, since it would require backing up the simulation to a certain point and comparing the results with a higher order prediction. Error control is therefore left to any possible extent for the internal APD engine to handle.

The main conclusion to be drawn from this example is that the co-simulation works as intended. PyMoC should thus be able to support practitioners who perform model-based studies of complex chemical processes.

It should be duly noted that, when co-simulating more complex modules, there may be a need for some *manual finishing touches* before having PyMoC couple the modules, arising from the fact that the results of the dynamic simulation may differ significantly from the steady state simulations. Therefore, the dynamic simulations may crash early on if two connected streams differ too much and thus too large a step change is performed during the running of PyMoC. According to Luyben (2013), the best way to approach a dynamic simulation is to let it run through time and converge to a steady state (which is something that I found through experience and agree with). This state should then be saved as the initial condition of the simulation file. Another finishing touch that should be made, is to make sure then that the output of one module that will be the input of another has converged at such a steady state, and then manually ramp the input in the second module to the correct value. This should increase chances of convergence and decrease risk of integrator corrector failures in APD.

### 3.4 Summary on practicalities during use

PyMoC was created with the goal of minimizing the incessant pointing-and-clicking inherent in Aspen Operator Training Simulator, to simplify advanced model-based studies. As such, there are a but a few things that need to be considered when using PyMoC, and some of them require diving into either the code of Python itself, or the code of PyMoC. These are:

- Before using PyMoC the first time, make sure to fix the error that otherwise will occur when loading modules. Instructions for this are available in section 3.1.2.
- To have PyMoC load the desired modules automatically, supply the file paths as strings to the variables *docpathbase*, *docpathlist*, and *docnamelist*. The division into three variables is purely for supporting cleaner code and multiple modules. However, as the code is currently constructed, the variables *docpathlist*, and *docnamelist* are necessary to use, whereas *docpathbase* is used to have cleaner code if several modules have identical paths to a certain point in the folder tree.
- During modeling, choose a stream naming convention and be consistent with it throughout the workflow of modeling, simulation, and model-based studies. This will make PyMoC automatically connect the provided modules.
- For the simulation, the final time and horizon lengths can be set via the parameters named *FinalTime* and *interval*, respectively. The interesting stream name indicator can be set as the parameter *interestingstreams* (a string).
- Remember to add the finishing touches to the modules, if necessary to stabilize the modules!

When all of this has been considered, co-simulation should be as simple as running PyMoC and waiting for the results.

## 4. Optimization

This chapter is intended to give some insight into process optimization in general. Special focus is then lent towards optimization using PyMoC, with an illustrative example.

### 4.1 Optimization in general

In practice, performing an optimization study is defined as finding the best solution to a given system within some constraints, and this requires three elements according to (Biegler, 2010). Firstly, an *objective function* is required, which needs to be well-defined (Sellberg, 2018). The purpose of formulating the objective function is to represent a scalar quantity reflecting one or multiple performance criteria of the given system (Floudas, 1995). Formulating the objective function is one of the key issues in order to perform advanced model-based optimization studies, and demands knowledge of the fundamental principles of the optimization algorithms as well as existing software to use as tools (Biegler, 2010). Examples of typical objective functions in the field of

chemical engineering include minimizing costs, maximizing profits, or minimizing the environmental impact.

Secondly, it is necessary to create a *mathematical model* of the process or system, which is translated into a set of constraints constituting a feasible region of the solution (Biegler, 2010). Essentially, the model is used to predict the relevant behavior of the system in order to use that information to evaluate the objective function, which is to be minimized, as well as the constraint functions, which are to be satisfied. Mathematical models can be created using three different approaches: fundamental principles, empiricism, and analogical modeling (Floudas, 1995). This means that mathematical models can be built by using continuity equations (Luyben, 1989), by 'black-box modeling' using statistical analysis such as machine/deep learning (Lee et al., 2018), or by utilizing established knowledge about a similar system (Floudas, 1995), as did Sellberg et al. (2017a). Models can also be created through a combination of the aforementioned approaches, for which process flowsheeting software such as Aspen Plus allows via the customizable libraries described in section 2.1.

Thirdly, a set of adjustable *decision variables* is needed that can be used as input to the process model (Biegler, 2010). The specifications of the decision variables need to define different states of the model (Floudas, 1995), in order for them to have an effect on the objective function. Examples within chemical engineering include using the reactor temperature as a decision variable in order to find the highest selectivity of the reactor; as a side note, the reactor temperature could further be dependent on e.g. the steam consumption, which would be dependent on a valve setting, and so on. Selecting the proper set of decision variables is often difficult, and can be very reliant on the decision-making engineer's experience, and understanding of the process and the data. Therefore, Nolin et al. (2017) used an early version of the Python-Aspen tool chain to apply a subset selection algorithm to reduce the number of potential decision-variable set candidates. This was shown to be an effective tool in terms of filtering out ineffective and recommending effective sets of decision variables in terms of the effects on the objective function.

Basically, during an optimization study, the decision variable is adjusted in order to evaluate the model response. The model response is used as a prediction of the real process behavior, and further used to evaluate the objective function. Optimization problems can thus generally be formulated algebraically as:

$$\begin{aligned} \min \quad & f(x,u) \\ \text{w.r.t.} \quad & u \in \mathbb{R} \\ \text{s.t.} \quad & h(x,u) = 0 \\ & g(x,u) \leq 0 \\ & x \in \mathbb{R} \end{aligned}$$

where $f$ is the objective function, $x$ variables and parameters, $u$ the decision variables, $h(x,u) = 0$ represents the model equations that need to converge, and $g(x,u) \leq 0$ represent constraint functions that need to be satisfied.

Optimization can further be divided into steady-state optimization and dynamic optimization (DO). Since PyMoC is intended to co-simulate dynamic process flowsheet

models (in the time domain), further exploration of DO is warranted (also, the most important basics of steady-state optimization have already been captured above). DO is more complex than steady-state optimization but very powerful, and can be used for e.g. trajectory optimization (Sellberg, 2018). DO can also be used for model-based predictive control (MPC or MBPC) as described by Maciejowski (2002), which is especially relevant for systems under regular disturbances, for start-up/shutdown scenarios, and for finding the optimal path during production change from set point A to set point B. However, using DO for forecasting the optimal decision variable path in real time sets a time limit on the computations that needs to be considered, which in turn sets high demands on how the problem is formulated and solved (Biegler, 2010).

In general, a DO problem such as trajectory optimization can be seen as the optimization problem formulated above, with the added dimension of time. One relatively simple way of approaching a trajectory problem is then to discretize the control signal in time according to a zero-order hold strategy, as presented in Sellberg et al. (2017). This simply means to temporally divide the decision variable into a set of *n* time horizons, each of which can take their own values (within any bounds set). It should be duly noted that PyMoC by nature of its zero-order hold analogy for data transfer has excellent support for this approach. Another approach would be to choose a decision variable, and then recast it as a mathematical function, e.g. a polynomial or trigonometric function. The next step would then be to use the parameters of the mathematical function as the actual decision variables for optimization in order to find the optimal control signal. Both the discretization and the parametrization approaches are visualized in figure 21. Applying either approach in real time for MPC would mean that also the recent history of the actual process (and not just the process model) is taken into account to find the optimal path over the next period of time.

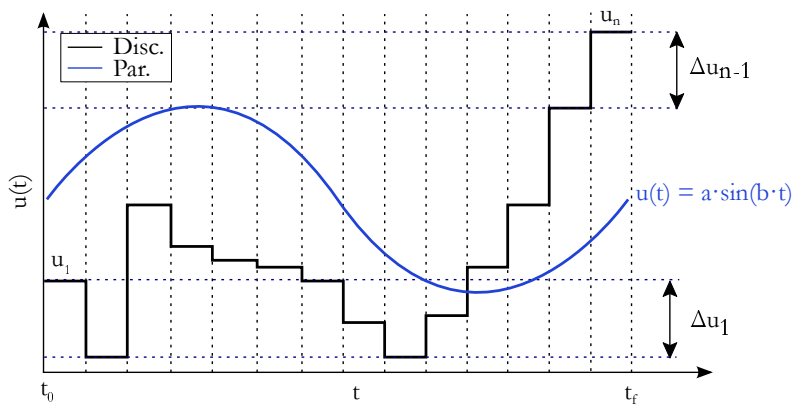



**Figure 21.** Visualizing dynamic optimization using discretization and parametrization of the decision variable u(t).

The parametrized control signal in figure 21 only carries two degrees of freedom in total, i.e. the parameters *a* and *b*, whilst each horizon for the discretized control signal can take on an individual value, thus representing one degree of freedom each for the optimizer. This means that the discretized control signal has *n* degrees of freedom,

which on the one hand leads to greater difficulties in converging on an optimal solution. Furthermore, since each discretized horizon is independent from the previous and following, "bang-bang optimization" may be an issue, which can be mitigated by adding $\sum_{i=1}^{n} R_u \Delta u_i$ (where $R_u$ is a scaling parameter and $\Delta u_i$ is the difference between two neighboring ZOH signals) to the objective, thereby punishing large differences between two consecutive ZOH signals. On the other hand, little to no knowledge or insight into the shape of the optimal solution is necessary since the optimizer has a good chance to find at least a local minimum given enough iterations. Reversely, the parametrization approach demands knowledge about what the desired trajectory should look like. Otherwise, finding an appropriate optimal solution may be nigh on impossible, e.g. when the optimal trajectory is of a sine wave shape whilst the control signal has been parametrized as a straight line.

## 4.2 Optimization with PyMoC

Solving dynamic non-linear programming problems with PyMoC assumes that a process flowsheet model has previously been created and modularized. PyMoC is then used to couple and co-simulate these modules, meaning that PyMoC is a tool developed for the engineering level of optimization studies (Biegler, 2010). PyMoC is capable of transferring and extracting information about the states of the modules, e.g. temperatures, pressures, flows, and compositions. However, it is not capable of finding and retrieving exact, analytical Jacobians from the modules. This essentially means that, from the point of view of the optimization algorithm, PyMoC is simply a black box that takes some input, generates and outputs some model response that can be used to evaluate the objective function. This is not the most efficient method (Biegler, 2010), but it is simple to implement since nothing but the inputs and outputs need to be defined. The simplicity further helps practitioners take advantage of the power of both the simulation engines as well as the unit operation/property libraries embedded in process flowsheeting software. However, should there be a need for increasing computational performance during PyMoC optimization, it will be necessary to study how to retrieve the required information such as exact/analytical Jacobians out of the modules, or how to create customized functions to express this using readily available information.

The major advantage of using PyMoC for optimization, rather than the internal Aspen methods (which presumably would not work with modularized models anyways, since co-simulation is necessary), is that it allows for the formulation of customized, more complex optimization problems on a general form. This means that multiple inputs, outputs, constraints, and bounds can be considered without special knowledge in how to operate the AspenTech products. Such optimization problems are furthermore relatively straightforward to implement if the user has general experience in optimization with other programming languages, such as Matlab or similar. Also, by virtue of its Python implementation, PyMoC allows for the utilization of state-of-the-art optimization algorithms through third-party packages such as SciPy.

### *4.2.1 Program structure*

The modularized nature of the process model used in PyMoC needs to be considered in order to successfully perform an optimization study. This means that there is need for a certain structure to use PyMoC for optimization. The most simplistic approach

is to go along with the aforementioned perspective of the optimization algorithm, and to treat PyMoC as a black box that takes some module input *u* and returns the module response *x*. PyMoC thus needs to be enveloped in an optimization wrapper, which finds the optimal point. This is for all intents and purposes the same logic by which most non-linear programming is performed, as described in section 4.1, and the general structure has been visualized and is presented in figure 22.

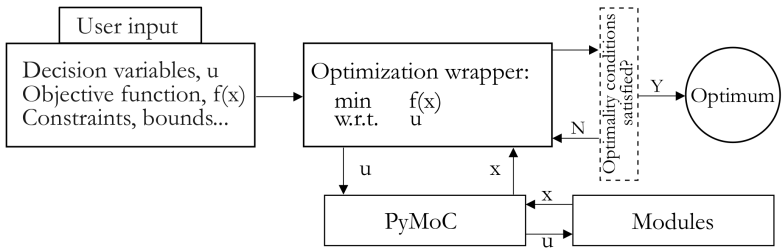



**Figure 22.** A visualization of the code structure necessary to solve a multi-module optimization problem using PyMoC.

Since an optimization wrapper around PyMoC is required, the user will have to create this and supply it with information about the optimization problem, e.g. decision variables and the objective function. The optimization wrapper will then start PyMoC, which in turn will carry out co-simulations of the modules provided to it, until the relevant optimality conditions have been met. A pseudo-code example reflecting the structure of figure 22 is presented in the listing below, using functions defined within the framework of PyMoC as well as functions provided by the COM interface:

```
from scipy.optimize import minimize

### Supply file paths
docpathlist = ['filepath1', 'filepath2']
docnamelist = ['filename1', 'filename2']

### Initialize Aspen files
# This creates the COM objects necessary to navigate
# the "Aspen flowsheet tree"
adyn, sim, fsheet, STREAM, BLOCKS = InitAspen()

### Initialize co-simulation
# This establishes the connections before simulations begin
InitCoSimulation()

### Define decision variables and initial values
u_vars = [u_var_1, ..., u_var_n]
u_init = [u_var_1.value, ..., u_var_n.value]

### Define a scalefactor as some function of u_init
scalefactor = F(u_init)

### Run optimization
DV_opt = minimize(objfun, u_init/scalefactor,
                  bounds=bounds,
                  constraints=confun)
```

The objective function, $objfun$, needs to be defined before it is called by the optimization algorithm. This can be done in the pseudo-code listing as follows:

```
def objfun(u_in):
    u = u_in*scalefactor
    PyMoCAlgorithm(u)
    # Defining the objective as some cost
    # incurred by the objective function variable x
    f = x.value*costfactor
    return f
```

Similarly, any constraint function will have to be defined before beginning the optimization. With all pieces in place, the optimization would proceed as suggested by the visualization in figure 22, until an optimal solution is found.

### *4.2.2 Example of application - optimization*

This example of how to use PyMoC for an optimization study uses the model described in section 3.3, with the described disturbance having been removed. This optimization example will instead focus on a dynamic, constrained optimization problem. The steam consumption, i.e. the steam flow rate in module A ("SI-1"), figure 15, is used as a single-value decision variable to optimize the net profit of the process.

The objective function, i.e. the net profit, is formulated as the difference between the income ($I$) generated by the product stream ("S-V") and the costs ($C$) generated by the steam consumption. These are expressed as an income factor (1.1) or cost factor (0.8) multiplied by the respective stream mass flow rate. The decision variable, i.e. the steam consumption, is left unbounded. Furthermore, the objective function is subject to the model itself, and constraints on the product stream S-V: it shall not contain less than 70 % ethanol by mass and the mass flow shall not be below 90 kg/h. However, changing the steam flow rate from the nominal point will introduce a transient behavior, taken into account through the use of the integrals of the state variables. These integrals are taken only over the second half of the simulation in order make convergence less difficult, by not "punishing" the objective or constraint functions too hard by considering the early stages of the transient behavior. The full optimization problem can be formulated as:

$$
\begin{aligned}
\underset{u}{\text{minimize}} \quad & \phi(I(t), C(u)) = \int_{t=2}^{4} [I(t) - C(u)] \ \mathrm{dt} \\
\text{where} \quad & I = 1.1 \cdot F_{mass,\ S-V}(t) \\
& C = 0.8 \cdot u \\
\text{w.r.t.} \quad & u = F_{mass,\ S-I1} \in \mathbb{R} \\
\text{s.t.} \quad & \text{Modularized flowsheet model} \\
& \int_{t=2}^{4} [X_{EtOH,\ S-V}(t) - 0.70] \ \mathrm{dt} > 0 \\
& \int_{t=2}^{4} [F_{mass,\ S-V}(t) - 90] \ \mathrm{dt} > 0
\end{aligned}
$$

The optimization problem was solved using the scientific Python package *SciPy*'s implementation of the COBYLA algorithm, with the initial value for the decision variable

set to be at half of its nominal value, $F^{init}_{mass,\ S-I} = 100.0\ kg/h$. The optimal point was subsequently found to be $F^{opt}_{mass,\ S-I} = 116.3\ kg/h$, with all constraints satisfied within the COBYLA default tolerance. The product stream ("S-V") for the optimal run is presented in figure 23. The results show that dynamic optimization studies may be successfully completed through PyMoC. It should be noted that even though it has not been shown here, trajectory optimization would work very well with PyMoC. Due to the implementation of the zero-order hold analogy for data transfer, only minor modifications to the standard PyMoC are necessary, and a trajectory optimization problem can be set up using one (or several) decision variable(s) per time horizon (which is basically the general approach to dynamic optimization as described in 4.1).

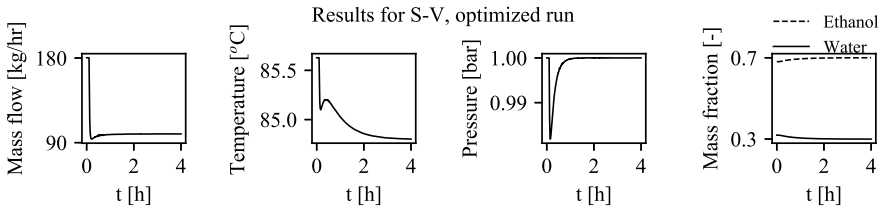



**Figure 23.** The results for the module B outbound stream S-V, for $\tau = 0.1$ h, during an optimized run.

#### *4.2.3 A note on steady-state optimization using PyMoC*

The basics of steady-state optimization have been described in this chapter, and whilst PyMoC is intended for dynamic modeling, it is possible to modify it into supporting steady-state simulations of the modules as well (and by extension steady-state optimization). This can be done by setting the simulation option called *RunMode* to 'Steady State', and making some modifications to how PyMoC handles the runtime. However, this has not been tried before, and is currently an unexplored possibility, but it is suspected that the modules may need to be designed to handle steady-state simulations as well.

## 5. Conclusions

During the project, the main goals were to develop a method for modeling complex processes, to create process flowsheet models, and to develop a simulation tool to use for model-based studies.

At the kernel of the modeling method was that it should be done in a modularized fashion. The main advantages of modularization, as described in section 2.1, includes that it allows for an easier overview, work-flow parallelization, independent modeling, and software reuse. It also leads to smaller, less complex modules that carry a significantly decreased risk of suffering from convergence issues. As such, this goal was achieved. Using the modeling method, modules for a separation process were created.

However, the developed modules were tuned to averaged plant data but were not mathematically calibrated or rigorously field validate, which may be a very good idea to do in the future.

In order to capitalize on the modeling method and use the modules to perform model-bases studies, it was necessary to employ a co-simulation strategy. For these purposes, a Python-Aspen tool-chain called PyMoC was developed. PyMoC is capable of co-simulating several modules, and thus allows for advanced model-based studies of a modularized system, e.g. optimization studies as shown in section 4.2.2.

## References

Ahmad, Z., Patle, D., and Rangaiah, G. (2016). Operator training simulator for biodiesel synthesis from waste cooking oil. *Process Safety and Environmental Protection*, 99:55 – 68.

Alobaid, F., Starkloff, R., Pfeiffer, S., Karner, K., Epple, B., and Kim, H.-G. (2015a). A comparative study of different dynamic process simulation codes for combined cycle power plants - part A: Part loads and off-design operation. *Fuel*, 153:692 – 706.

Alobaid, F., Starkloff, R., Pfeiffer, S., Karner, K., Epple, B., and Kim, H.-G. (2015b). A comparative study of different dynamic process simulation codes for combined cycle power plants - part B: Start-up procedure. *Fuel*, 153:707 – 716.

Anaconda, Inc. (2019). Website. `https://www.anaconda.com/`. Accessed 2019-08-29.

Andersson, C. (2016). *Methods and Tools for Co-Simulation of Dynamic Systems with the Functional Mock-up Interface*. PhD thesis, Lund University.

Aspen Technology, I. (2016). *EAP201 Aspen Plus: Physical Properties for Process Engineers*. Aspen Technology, Inc.

Bennett, N. and Lemoine, G. J. (2014). What a difference a word makes: Understanding threats to performance in a VUCA world. *Business Horizons*, 57(3):311 – 317.

Biegler, L. T. (2010). *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Bulian, G. and Cercos-Pita, J. L. (2018). Co-simulation of ship motions and sloshing in tanks. *Ocean Engineering*, 152:353 – 376.

CAPE-OPEN Laboratories Network (2018). Process Modeling Environment. `http://www.colan.org/category/process-modeling-environment/`. Accessed 2018-09-21.

Corbetta, M., Grossmann, I. E., and Manenti, F. (2016). Process simulator-based optimization of biorefinery downstream processes under the generalized disjunctive programming framework. *Computers & Chemical Engineering*, 88:73 – 85.

Dimian, A. C., Bildea, C. S., and Kiss, A. A. (2014). Chapter 2 - introduction in process simulation. In Dimian, A. C., Bildea, C. S., and Kiss, A. A., editors, *Integrated Design and Simulation of Chemical Processes*, volume 35 of *Computer Aided Chemical Engineering*, pages 35 – 71. Elsevier.

Felippa, C. A., Park, K., and Farhat, C. (2001). Partitioned analysis of coupled mechanical systems. *Computer Methods in Applied Mechanics and Engineering*, 190(24):3247 – 3270. Advances in Computational Methods for Fluid-Structure Interaction.

Floudas, C. (1995). *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Topics in Chemical Engineering. Oxford University Press.

Hasse, H. and Maurer, G. (1991). Vapor-liquid equilibrium of formaldehyde-containing mixtures at temperatures below 320 k. *Fluid Phase Equilibria*, 64:185 – 199.

Haus, J., Hartge, E.-U., Heinrich, S., and Werther, J. (2018). Dynamic flowsheet simulation for chemical looping combustion of methane. *International Journal of Greenhouse Gas Control*, 72:26 – 37.

Lee, J. H., Shin, J., and Realff, M. J. (2018). Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, 114:111 – 121. FOCAPO/CPC 2017.

Lin, Z., Wang, J., Nikolakis, V., and Ierapetritou, M. (2017). Process flowsheet optimization of chemicals production from biomass derived glucose solutions. *Computers & Chemical Engineering*, 102:258 – 267.

Luyben, W. (2013). *Distillation Design and Control Using Aspen Simulation*. Wiley.

Luyben, W. L. (1989). *Process Modeling, Simulation and Control for Chemical Engineers*. McGraw-Hill Higher Education, 2nd edition.

Luyben, W. L. (2002). *Plantwide dynamic simulators in chemical processing and control*. Marcel Dekker, Inc., New York.

Maciejowski, J. M. (2002). *Predictive Control with Constraints*. Prentice Hall.

Maurer, G. (1986). Vapor-liquid equilibrium of formaldehyde-and water-containing multicomponent mixtures. *AIChE Journal*, 32(6):932–948.

Microsoft (2018). Component Object Model (COM). `https://docs.microsoft.com/en-us/windows/desktop/com/component-object-model--com--portal`. Accessed 2018-07-09.

Mikkonen, H., Lappalainen, J., Pikkarainen, T., and Kuivalainen, R. (2017). Modelling and dynamic simulation of the 2nd generation oxy fired power plant âĂŞ oxidant fan failure case. *Energy Procedia*, 114:561 – 572. 13th International Conference on Greenhouse Gas Control Technologies, GHGT-13, 14-18 November 2016, Lausanne, Switzerland.

Muñoz López, C. A., Telen, D., Nimmegeers, P., Cabianca, L., Logist, F., and Impe, J. V. (2018). A process simulator interface for multiobjective optimization of chemical processes. *Computers & Chemical Engineering*, 109:119 – 137.

Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2017). Unbiased selection of decision variables for optimization. In Espuña, A., Graells, M., and Puigjaner, L., editors, *27th European Symposium on Computer Aided Process Engineering*, volume 40 of *Computer Aided Chemical Engineering*, pages 253 – 258. Elsevier.

Nolin, M., Andersson, N., Nilsson, B., Max-Hansen, M., and Pajalic, O. (2018). Analysis of an oscillating two-stage evaporator system through modelling and simulation: an industrial case study. *Chemical Engineering Transactions*, 69:481–486.

Oppelt, M., Wolf, G., and Urbas, L. (2015). Life cycle simulation for a process plant based on a two-dimensional co-simulation approach. In Gernaey, K. V., Huusom, J. K., and Gani, R., editors, *12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering*, volume 37 of *Computer Aided Chemical Engineering*, pages 935 – 940. Elsevier.

Python Software Foundation (2018a). About. `https://www.python.org/about/`. Accessed 2018-09-12.

Python Software Foundation (2018b). FAQ. `https://docs.python.org/3/faq/general.html`. Accessed 2018-09-12.

Python Software Foundation (2018c). Success Stories. `https://www.python.org/about/success`. Accessed 2018-09-12.

Sellberg, A. (2018). *Open-Loop Optimal Control of Chromatographic Separation Processes*. PhD thesis, Department of Chemical Engineering, Lund University.

Sellberg, A., Andersson, N., Holmqvist, A., and Nilsson, B. (2017a). Development and optimization of a single column analog model for a multi-column counter-current solvent gradient purification process. In Espuña, A., Graells, M., and Puigjaner, L., editors, *27th European Symposium on Computer Aided Process Engineering*, volume 40 of *Computer Aided Chemical Engineering*, pages 187 – 192. Elsevier.

Sellberg, A., Holmqvist, A., Magnusson, F., Andersson, C., and Nilsson, B. (2017b). Discretized multi-level elution trajectory: A proof-of-concept demonstration. *Journal of Chromatography A*, 1481:73 – 81.

Shukla, X. U. and Parmar, D. J. (2016). Python - a comprehensive yet free programming language for statisticians. *Journal of Statistics and Management Systems*, 19(2):277–284.

Skorych, V., Dosta, M., Hartge, E.-U., and Heinrich, S. (2017). Novel system for dynamic flowsheet simulation of solids processes. *Powder Technology*, 314:665 – 679. Special Issue on Simulation and Modelling of Particulate Systems.

Soares, R. and Secchi, A. (2004). Modifications, simplifications, and efficiency tests for the cape-open numerical open interfaces. *Computers & Chemical Engineering*, 28(9):1611 – 1621.

Steinbrink, C., Lehnhoff, S., Rohjans, S., Strasser, T. I., Widl, E., Moyo, C., Lauss, G., Lehfuss, F., Faschang, M., Palensky, P., van der Meer, A. A., Heussen, K., Gehrke, O., Sansano, E. G., Syed, M. H., Emhemed, A., Brandl, R., Nguyen, V. H., Khavari, A., Tran, Q., Kotsampopoulos, P., Hatziargyriou, N., Akroud, N., Rikos, E., and Degefa, M. (2017). Simulation-based validation of smart grids - status quo and future research trends. In *Proceedings of 8th International Conference on Industrial Applications of Holonic and Multi-Agent Systems*. IEEE, Piscataway, N.J.

Sun, W., Nagpal, K. M., Khargonekar, P. P., and Poolla, K. R. (1992). Digital control systems: $\mathscr{H}_\infty$ controller design with a zero-order hold function. In *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*, pages 475–480 vol.1.

Sundqvist, M. (2014). Modeling and simulation of a polyol process at perstorp ab using chemcad. Master's thesis, Lund University, Faculty of Engineering, Department of Chemical Engineering.

Ungerer, M. J. and Goodchild, M. F. (2002). Integrating spatial data analysis and GIS: a new implementation using the Component Object Model (COM). *International Journal of Geographical Information Science*, 16(1):41–53.

Walker, M. (2018). Enabling co-simulation of tokamak plant models and plasma control systems. *Fusion Engineering and Design*, 127:60 – 65.

Yang, X., Xu, Q., Zhao, C., Li, K., and Lou, H. H. (2009). Pressure-driven dynamic simulation for improving the performance of a multistage compression system during plant startup. *Industrial and Engineering Chemistry Research*, 48(20):9195–9203.

Zitney, S. E. (2010). Process/equipment co-simulation for design and analysis of advanced energy systems. *Computers & Chemical Engineering*, 34(9):1532 – 1542. Selected papers from the 7th International Conference on the Foundations of Computer-Aided Process Design (FOCAPD, 2009, Breckenridge, Colorado, USA.

# Paper VI

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. 
https://doi.org/10.1016/B978-0-444-64235-6.50282-5

# Multi-flowrate Optimization of the Loading Phase of a Preparative Chromatographic Separation

Anton Sellberg, Mikael Nolin, Anton Löfgren, Niklas Andersson and Bernt Nilsson*

*Department of Chemical Engineering, Lund University, P.O. Box 124, SE-221 00 Lund, Sweden*
*bernt.nilsson@chemeng.lth.se*

## Abstract

This contribution consists of a multi-flow rate optimization method to maximize the key performance indicators, yield, degree of resin utilization and productivity for a chromatographic loading step. The protein adsorption to the resin is modeled using a mechanistic column model with a steric mass action adsorption isotherm. The model was used to optimize the flow rate trajectories and the loading duration by using a simultaneous method where the controls and state variables were discretized using a direct and local collocation method on finite elements in the temporal dimension and a finite volume weighted essentially non-oscillatory scheme in the spatial dimension. The discretized nonlinear program was solved using an interior point method. The two and three dimensional pareto fronts for the multi-objective optimization are presented and the multi-flow rate strategy showed considerable improvement in all objectives at different points on the pareto front.

**Keywords:** Modelling, Optimization, Collocation, Protein, Chromatography

## 1. Introduction

Purification of proteins typically consists of a clarification step, a chromatographic capture step and one or several additional orthogonal chromatography steps. This contribution is concerned with the loading of the chromatography columns and mainly the loading of the capture step. Since the capacity of chromatographic resins are low, resin utilization is of great importance. Resin utilization is especially important in the capture step since the capture step receives the highest amount of product protein and makes use of expensive resin, e.g. protein A or mixed-mode resins. In addition to resin utilization, product yield and production rate needs to be taken into account when optimizing the loading phase. To increase the resin utilization and product yield several design and control approaches have been suggested. Capture-SMB (Angarita et al., 2015) and periodic counter current chromatography, which uses two and three columns in sequence to fully load one of the columns while collecting the product breakthrough on the following column. Both of these approaches have the drawback of needing extra columns and equipment as well as a flow rate limitations due to the increased pressure drop over the interconnected columns. Another approach has been to use dual-flowrate loading strategies on a single column to maximize the dynamic binding capacity and productivity. This approach does not need any extra equipment or columns and still outperforms single loading strategies (Ghose et al., 2004).

The aim of this study is to expand the dual-flowrate strategy to include any number of flowrates during the loading. This is achieved by mechanistic modeling of the process and discretization of the temporal domain while keeping a constant flowrate in each time horizon, i.e. piecewise constant controls. Yield, production rate and degree of resin utilization are identified as key process indicators (KPIs) which are weighed together to form a scalar objective. This approach results in

a large-scale dynamic optimization problem constrained by partial differential algebraic equations which is solved for both single and multiple flowrates and variable loading time. A case study where a 4 g/L lysozyme solution is loaded on a 1 ml prepacked Sepharose SP Fast Flow column is presented to show the advantages of the multi-flow rate strategy compared to the single flow rate strategy.

## 2. Chromatography Model

The Transport-Dispersive Model (Schmidt-Traub et al., 2006) was used to model the breakthrough curves of protein during column loading. The model equations, Eq. 1 and 2, are defined over the spatial, $z \in [z_0, z_f]$ and temporal, $t \in [t_0, t_f]$ domain. The mass-balance over the mobile and stationary phase is governed by:

$$\frac{\partial c_\alpha}{\partial t} = \frac{\partial}{\partial z}\left(D_{\text{ax}}\frac{\partial c_\alpha}{\partial z} - \frac{\dot{Q}}{\varepsilon_t A_c}c_\alpha\right) - \frac{1-\varepsilon_c}{\varepsilon_t}\frac{\partial q_\alpha}{\partial t}, \quad \alpha \in \{\text{lys}, s\} \tag{1}$$

$$\frac{\partial q_\alpha}{\partial t} = \begin{cases} -k_{\text{film}}\left(q_\alpha - q_\alpha^*\right) & \text{for } \alpha = \text{lys} \\ -\nu\dfrac{\partial q_{\text{lys}}}{\partial t} & \text{for } \alpha = s \end{cases} \tag{2}$$

where $c_\alpha$ is the concentration of component $\alpha$ in the mobile phase, $t$ is time, $z$ is the spatial coordinate, $D_{\text{ax}}$ is the axial dispersion, $\dot{Q}$ is the volumetric flow rate, $\varepsilon_t$ is the total void, $\varepsilon_c$ is the column void, $A_c$ is the column cross section area, $k_{\text{film}}$ is the film mass transfer parameter, $q_\alpha$ is the concentration of component $\alpha$ in the stationary phase, $\nu$ is the characteristic charge of lysozyme and $q_\alpha^*$ is the adsorbed concentration of component $\alpha$ at equilibrium with the mobile phase, which is governed by the steric mass action isotherm (Brooks and Cramer, 1992):

$$0 = c_\alpha - \frac{1}{K_M}\frac{q_\alpha^* c_s^\nu}{\left(\Lambda - (\sigma+\nu)q_\alpha^*\right)^\nu} \tag{3}$$

where $K_M$ is the equilibrium constant, $\Lambda$ is the total ion exchange capacity and $\sigma$ is the steric shielding parameter. The SMA isotherm was then reparametrized according to:

$$K_{\text{eq}} = K_M \Lambda^\nu, \quad q_{\max} = \Lambda(\sigma+\nu)^{-1} \tag{4}$$

where $K_{\text{eq}}$ is the new equilibrium constant for adsorption and desorption of lysozyme and $q_{\max}$ is the maximum concentration fo adsorbed lysozyme. The total film mass transfer resistance is the sum of the inner and outer film resistance. The total film mass transfer parameter, the inner film transfer parameter ($k_s$) and the outer film transfer parameter ($k_o$) depends on the stationary phase saturation and flow rate according to (Steinebach et al., 2016):

$$k_{\text{film}} = \left(\frac{1}{k_s} + \frac{1}{k_o}\right)^{-1}, \quad k_s = k_s^{'}\frac{(1-\Gamma)^{1/3}}{1-(1-\Gamma)^{1/3}}, \quad k_o = k_o^{'}\left(\frac{\dot{Q}}{A_c\varepsilon_c}\right)^{1/3} \tag{5}$$

where $k_s^{'}$ is the inner film mass transfer resistance parameter, $\Gamma$ is the saturation level of the stationary phase, i.e. $\Gamma = q_\alpha/q_{\max}$ for rectangular isotherms and $k_o^{'}$ is the outer film mass transfer resistance parameter. Danckwert boundary conditions are used for the column inlet and homoge-

neous von Neumann conditions are used at the column outlet:

$$D_{\mathrm{ax}}\frac{\partial c_\alpha(t,z_0)}{\partial z}-\frac{\dot{Q}}{A_c\varepsilon_t}c_\alpha(t,z_0)=\frac{\dot{Q}}{A_c\varepsilon_t}c_{\mathrm{load},\alpha}\Pi(t,t_0,\Delta t_{\mathrm{load}}) \tag{6}$$

$$D_{\mathrm{ax}}\frac{\partial c_s(t,z_0)}{\partial z}-\frac{\dot{Q}}{A_c\varepsilon_t}c_s(t,z_0)=\frac{\dot{Q}}{A_c\varepsilon_t}c_{\mathrm{A},s} \tag{7}$$

$$\frac{\partial c_\alpha(t,z_f)}{\partial z}=0 \tag{8}$$

$$\frac{\partial c_s(t,z_f)}{\partial z}=0 \tag{9}$$

where $c_{\mathrm{load},\alpha}$ is the load concentration of component $\alpha$, $\Pi(t,t_0,\Delta t_{\mathrm{load}})\in[0,1]$ is a smooth rectangular function in the temporal horizon $[t_0,t_0+\Delta t_{\mathrm{load}}]$, $\Delta t_{\mathrm{load}}$ is the load time and $c_{\mathrm{A},s}$ is the salt concentration in the low salt buffer. At the start of each run the column only contains low salt buffer. The column void, $\varepsilon_c$, was 0.3 and the total void, $\varepsilon_t$, was 0.9. The isotherm and mass transfer coefficients are summerized in Table 1.

Table 1: Model parameters used to model the adsorption of lysozyme to the resin.

| $q_{\max}$ [mol m$^{-3}$] | $\nu$ [−] | $K_{\mathrm{eq}}$ [(mol m$^{-3}$)$^{\nu}$] | $k_s'$ [s$^{-1}$] | $k_0'$ [m$^{-1/3}$s$^{-2/3}$] | Pe [−] |
|---|---|---|---|---|---|
| 17.6 | 5.97 | $3.40\cdot10^{14}$ | $1.11\cdot10^{-13}$ | $2.42\cdot10^{-2}$ | 0.50 |

## 3. Multi-Objective Dynamic Optimization Problem

To formulate the optimization problem, a well-defined objective function is needed. The basis of the objective functions is the three different key performance indicators (KPIs) for the process. The KPIs are weighed together to form the objective function. In this case yield, $Y(t)$, productivity, $P(t)$ and degree of stationary phase utilization, $U(t)$ are the KPIs of the process. The KPIs are defined as:

$$Y(t)=m_{\mathrm{c}}(t)m_{\mathrm{in}}(t)^{-1}, \tag{10}$$

$$P(t)=m_{\mathrm{c}}(t)\Big[\Delta t_{\mathrm{load}}+\Delta t_w\Big]^{-1}, \tag{11}$$

$$U(t)=m_{\mathrm{c}}(t)\Big((1-\varepsilon_t)V_c q_{\max}\Big)^{-1} \tag{12}$$

where $m_{\mathrm{c}}$ is the amount of lysozyme adsorbed to the stationary phase, $m_{\mathrm{in}}$ is the amount of lysozyme loaded to the column, $\Delta t_w$ is the time it takes to wash, clean and regenerate the column and $V_c$ is the column volume. The amount of lysozyme adsorbed to the stationary phase and loaded is defined as:

$$\frac{\mathrm{d}m_{\mathrm{in}}(t)}{\mathrm{d}t}=\Pi(t,t_0,\Delta t_{\mathrm{load}})\dot{Q}(t)c_{\mathrm{feed}}, \tag{13}$$

$$\frac{\mathrm{d}m_{\mathrm{c}}(t)}{\mathrm{d}t}=\Pi(t,t_0,\Delta t_{\mathrm{load}})\frac{1-\varepsilon_c}{\varepsilon_t}A_c\int_{z_0}^{z_f}\frac{\mathrm{d}q_{\mathrm{lys}}}{\mathrm{d}t}\mathrm{d}z \tag{14}$$

The cost function for the optimization can then be expressed as:

$$-\Big[\omega_Y\tilde{Y}(t_f)+\omega_P\tilde{P}(t_f)+\omega_U\tilde{U}(t_f)\Big]+\mathrm{R}_Q\sum_{i=1}^{N_Q-1}\Delta\dot{Q}_i^2, \tag{15}$$

$$\sum_{i\in\{Y,P,U\}}\omega_i=1 \tag{16}$$

where the bracketed term is the weighted normalized KPIs, the second term is penalizing the squared difference between two flow rates and $R_Q$ is the penalizing weight. The penalizing term is added to make the resulting flow rate trajectory smoother and to avoid pressure spikes caused by a large step-wise change in flow rate. The complete multi-objective dynamic optimization problem can finally be formulated as:

$$\begin{aligned}
&\text{min.} && -\left[\omega_Y \tilde{Y}(t_f) + \omega_P \tilde{P}(t_f) + \omega_U \tilde{U}(t_f)\right] + \mathrm{R}_Q \sum_{i=1}^{N_Q-1} \Delta\dot{Q}_i^2, && \text{(17a)}\\
&\text{w.r.t.} && \boldsymbol{x} : [t_0, t_f] \to \mathbb{R}^{N_x}, && \boldsymbol{y} : [t_0, t_f] \to \mathbb{R}^{N_y},\\
& && \boldsymbol{Q} \in \mathbb{R}^{N_Q}, && \Delta t_{\text{load}} \in \mathbb{R},\\
&\text{s.t.} && \boldsymbol{0} = \boldsymbol{F}(t, \dot{\boldsymbol{x}}(t), \boldsymbol{x}(t), \boldsymbol{y}(t), \dot{\boldsymbol{Q}}(t), \Delta t_{\text{load}}), && \boldsymbol{x}(t_0) = \boldsymbol{x}_0, && \text{(17b)}\\
& && \boldsymbol{g}_e(t, \dot{\boldsymbol{Q}}(t)) = 0, && && \text{(17c)}\\
& && Q_L \le \dot{Q}(t) \le Q_U, && \Delta t_L \le \Delta t_{\text{load}} \le \Delta t_U, && \text{(17d)}\\
& && \forall t \in [t_0, t_f] &&
\end{aligned}$$

The optimization variables consist of the piecewise constant flow rates, $\boldsymbol{Q}$, the free manipulated variable, $\Delta t_{\text{load}}$, the state variables, $\boldsymbol{x} = \left(c_{\text{lys}}, q_{\text{lys}}, c_s, m_c, m_{\text{in}}\right)$ and the algebraic variable $\boldsymbol{y} = q^*_{\text{lys}}$. The trajectories $\boldsymbol{x}$ are determined by the PDAE in Eq. (17b), the equality constraint on $Q$ in each time horizon in Eq. (17c) and the bounds on $Q$ and $\Delta t_{\text{load}}$ in Eq. (17d).

## 4. Modeling and Optimization Environment

The model presented in section 2 and the dynamic optimization problem presented in section 3 were implemented in the Modelica and Optimica languages. The open-source platform JModelica.org (Åkesson et al., 2010) was used for both simulation and optimization purposes. The spatial domain was discretized using 25 finite volumes and a weighted essentially non-oscillating (Shu, 1998) scheme with 3 sub-stencil was used to approximate the convective and diffusive flux.

For simulation purposes the model was compiled to a functional mock-up unit before being interfaced with CVode from the SUNDIALS solver suite (Hindmarsh et al., 2005). The backward differentiation formula and a relative and absolute tolerance of $10^{-10}$ was used to simulate the model. The multi-objective dynamic optimization problem presented in section 3 was solved using a direct and local collocation method with 150 finite elements. Each element contained two Radau points and the solution was estimated using Lagrange polynomials (Magnusson and Åkesson, 2015). The nonlinear program resulting from the collocation was solved using the the primal-dual interior point method IPOPT (Wächter and Biegler, 2006) and the linear solver MA57 from HSL (HSL, 2013). The automatic differentiation package, CasADi (Andersson, 2013), was used to compute the first- and second-order derivatives. The starting point of the optimization was obtained by simulating the flow rate trajectory with a constant control signal of 2 ml min$^{-1}$. The optimization was solved with a tolerance of $10^{-8}$.

## 5. Results and Discussion

Two different cases were optimized to assess the performance of the suggested multiflow rate loading strategy. The conventional single flow rate i.e. $N_Q = 1$ was used as a base case and a multiflow rate strategy with $N_Q = 50$ was used to demonstrate the advantages with multiflow rate loading. The resulting breakthrough curves and optimal flow rates for two different objective weight combinations can be seen in Figure 1. Subplot a) and c) has the objective weights, $\omega_P = 0.440$,

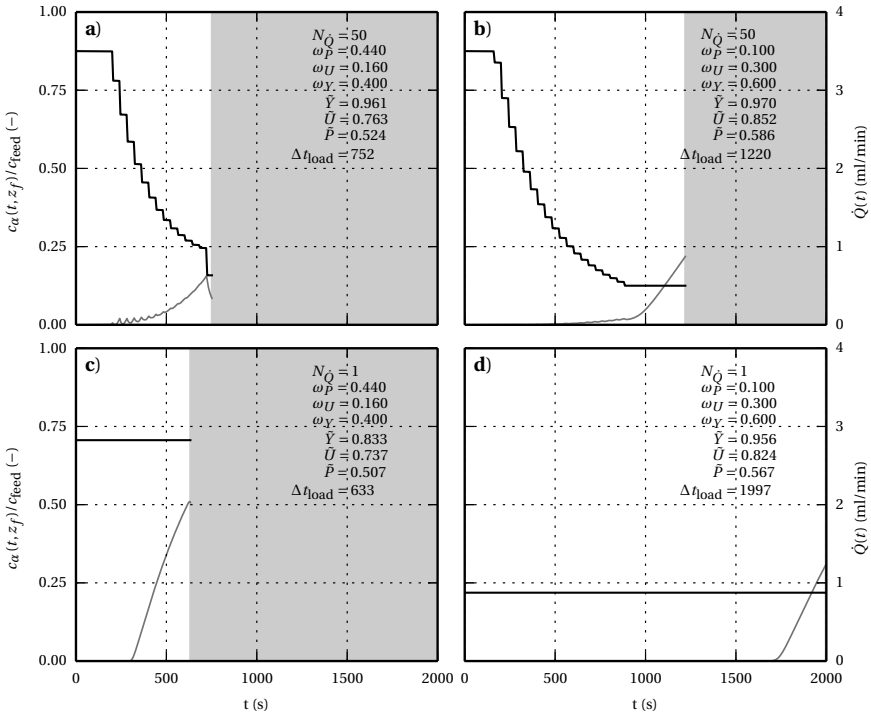



Figure 1: Optimal breakthrough trajectories for the column. The solid black step-function is the flow rates used to load the column, the solid gray line is the protein breakthrough at the column outlet and the gray area indicates the reduction in cycle time, i.e. $t_f - \Delta t_{\text{load}}$. The solution to the optimization problem using $N_Q = 50$ is shown in subplot a) and b). The solution to the optimization problem using $N_Q = 1$ is shown in subplot c) and d).

$\omega_U = 0.160$ and $\omega_Y = 0.400$, which results in comparable results with respect to utilization and productivity but the multi-flow rate strategy has a 15% increase in yield. Comparing the results in subplot b) and d), $1.5 - 3.4\%$ lower results in all objectives can be observed for the single flow rate strategy while loading 777s (63.7%) longer than the multi-flow rate strategy. Thus if only a shorter loading time can be allowed, the results of the single flow rate strategy will decrease more than the results of the multi-flow rate strategy. Furthermore any time-saving improvements to the elution, cleaning and regeneration strategy i.e. lowering of $\Delta t_w$ in Equation 11, will result in a higher relative improvement in production rate for the multi-flow rate strategy.

For each of the cases, a three-dimensional and three two-dimensional pareto fronts were sampled, presented in Figure 2. The complex trade-off between the three objective functions can be seen in the three-dimensional pareto plot. For combinations containing a high weight in yield the multi-flow rate strategy shows better performance in all objectives compared to the single flow rate strategy. This is due to the fact that for single flow rate loading having a large weight on yield means running the process at a low flow rate to avoid protein breakthrough. A constant low flow rate also means that only a small amount of protein will be loaded onto the column i.e. resin utilization and production rate will be low. For the multi-flow rate strategy the process can be run at high flow rates until breakthrough occurs and then gradually decrease the flow rate to maximize the amount of protein loaded while minimizing the breakthrough.

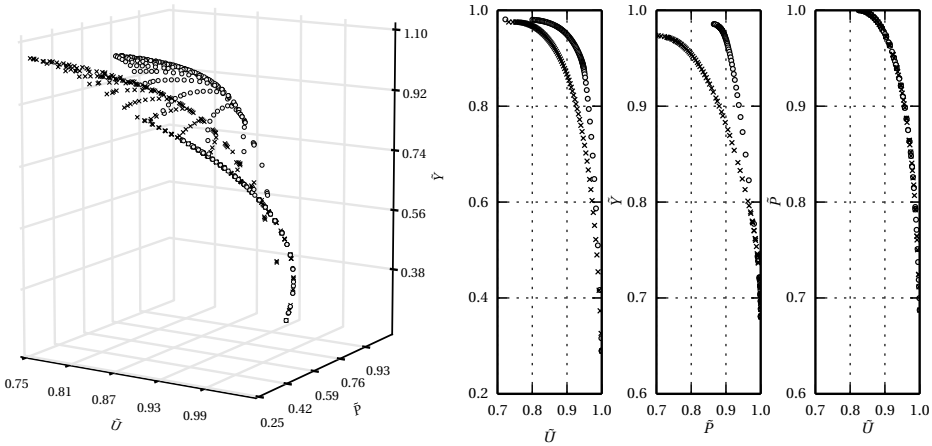



Figure 2: The multi-flow rate strategy is the *o* markers while the single flow rate strategy is *x* in all subplots. The result from the tri-objective optimization is presented in the leftmost sub-plot. The three two dimensional subplots show the pareto front with one objective fixed to zero.

## 6. Conclusions

The presented multi-flow rate approach outperforms the single flow-rate for most points on the three dimensional pareto front. The advantage is most pronounced for pareto solutions where no objective function has a weight of zero. However the multi-flow strategy still outperforms the single flow strategy for most weight combinations if $\omega_P = 0$ or $\omega_U = 0$. For $\omega_Y = 0$, the single and multi-flow rate strategies collapse to the same Pareto front because breakthrough of protein has no impact on the production rate and resin utilization, instead all the solutions uses the highest flow rate and only $\Delta t_{\text{load}}$ varies between solutions.

## References

J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl, H. Tummescheit, 2010. Modeling and optimization with optimica and JModelica.org-Languages and tools for solving large-scale dynamic optimization problems. Computers and Chemical Engineering 34 (11), 1737–1749.

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. Angarita, T. Müller-Späth, D. Baur, R. Lievrouw, G. Lissens, M. Morbidelli, 2015. Twin-column CaptureSMB: a novel cyclic process for protein a affinity chromatography. Journal of Chromatography A 1389, 85–95.

C. A. Brooks, S. M. Cramer, 1992. Steric mass-action ion exchange: displacement profiles and induced salt gradients. AIChE Journal 38 (12), 1969–1978.

S. Ghose, D. Nagrath, B. Hubbard, C. Brooks, S. M. Cramer, 2004. Use and optimization of a dual-flowrate loading strategy to maximize throughput in protein-a affinity chromatography. Biotechnology progress 20 (3), 830–840.

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 31 (3), 363–396.

HSL, 2013. A collection of fortran codes for large scale scientific computation. `http://www.hsl.rl.ac.uk/`.

F. Magnusson, J. Åkesson, 2015. Dynamic optimization in JModelica.org. Processes 3 (2), 471–496.

H. Schmidt-Traub, M. Schulte, A. Seidel-Morgenstern, 2006. Preparative chromatography: of fine chemicals and pharmaceutical agents. John Wiley & Sons.

C.-W. Shu, 1998. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In: Advanced numerical approximation of nonlinear hyperbolic equations. Springer, pp. 325–432.

F. Steinebach, M. Angarita, D. J. Karst, T. Müller-Späth, M. Morbidelli, 2016. Model based adaptive control of a continuous capture process for monoclonal antibodies production. Journal of Chromatography A 1444, 50–56.

A. Wächter, L. T. Biegler, 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming 106 (1), 25–57.

# Paper VII

Received: 11 March 2019 | Revised: 20 May 2019 | Accepted: 3 June 2019
DOI: 10.1002/btpr.2871

RESEARCH ARTICLE

BIOTECHNOLOGY PROGRESS

# Optimization of integrated chromatography sequences for purification of biopharmaceuticals

**Anton Löfgren** | **Mikael Yamanee-Nolin** | **Simon Tallvod** | **Joaquín G. Fons** | **Niklas Andersson** | **Bernt Nilsson**

Department of Chemical Engineering, Lund University, Lund, Sweden

**Correspondence**
Bernt Nilsson, Department of Chemical Engineering, Lund University, P.O. Box 124, SE211 00 Lund, Sweden.
Email: bernt.nilsson@chemeng.lth.se

**Funding information**
VINNOVA; Swedish Orphan Biovitrum; Novo Nordisk

**Abstract**

With continued development of integrated and continuous downstream purification processes, tuning and optimization become increasingly complicated with additional parameters and codependent variables over the sequence. This article offers a novel perspective of nonlinear optimization of integrated sequences with regard to individual column sizes, flow rates, and scheduling. The problem setup itself is a versatile tool to be used in downstream design which is demonstrated in two case studies: a four-column integrated sequence and a continuously loaded twin-capture setup with five columns.

**KEYWORDS**
chromatography, continuous bioprocessing, integrated column sequences, multiple column design, optimization

## 1 | INTRODUCTION

Continuous processes are embraced in several fields, such as petroleum, steel, and so in, but only in recent years biopharma has shown serious consideration into adopting these strategies.[1,2] Despite all advancements that have been made in the pharmaceutical field, there is still a lot of work to be done before implementing and using a complete continuous and integrated process in production. A main reason for this seems to be the added complexity in unit operations compared to conventional batch system variants.[2] However, the benefits seem to be worth it for several reasons. Firstly, there is cost savings with increased productivity, less unit operations, smaller scale production, which reduces equipment size and facility footprint, opportunity for disposable technologies, and reduced operational costs. Secondly, the potential benefit of the increased responsiveness that is drawn from low residence times and cycle times as well as the potential for standardized platforms to flexibly change production between different products. A third perspective is the increased product quality which should come from lower residence times and fewer hold tanks.[3]

Periodic cyclic steady state is achieved when unit operations are connected to each other, but still allow gradients of concentration, pH, and conductivity to elute products.[2] Studies[4-6] have been performed with several chromatography columns in integrated sequences that connected the columns only during elute and load phases, and then pumping through two columns in row. This was performed by redirecting the flow path only during the elute phase to the next column valve, creating a looped sequence.

Including the reactor in continuous bioprocesses is a great step toward achieving a complete end-to-end process.[7] However, this comes with new challenges and complexities where the capture step receives a continuous feed stream. This problem can be dealt with by different techniques, for example, periodic countercurrent chromatography,[1] twin-capture,[8] and capture-simulated moving bed.[9]

When constructing a purification process, several recovery and purification steps are often necessary to isolate the target molecule. The downstream process must be carefully designed due to several reasons: there is no single or general method to purify all kinds of proteins; it is the most expensive step; and it also has a significant effect

on therapeutic performance.[10] Thus, optimization and design of downstream sequences are complex and require a great understanding of all parameters involved. There are helpful holistic process development strategies called expert systems, which construct an optimal purification scheme with different column types and their sequential order by analyzing data about the target molecule.[11,12] Several heuristics also exist for finding the overall best downstream design with regard to the type of columns and their order,[13] for example, using orthogonal purification strategies that separate based on different physicochemical properties such as anion exchange followed by cation exchange instead of two separations of similar type.

The purpose of this paper is to formulate additional heuristics that are useful when designing integrated sequences that consist of direct column-to-column flows. This complements the previously mentioned expert systems and already established design heuristics with a novel perspective for integrated sequences. Two different cases will be studied which both build on the same theoretical background but have different objectives. The first case is the optimization of an integrated and connected batch sequence with regard to column volumes and time. The second case is about adjusting a sequence that handles a constant stream from a continuous reactor dealt with by twin capture followed by an integrated sequence of columns and the objective is to lower the column volumes and reduce the buffer consumption.

## 2 | MATERIALS AND METHODS

Multiple chromatography columns are often necessary for satisfying purity in downstream processes. As stated in Section 1, integrated sequences with column-to-column pooling in several steps are a way of making the downstream process more effective as opposed to manual batch protocols. However, integration and continuous processing come with added complexity. This section will make an attempt at structuring choices when designing complete integrated sequences. Bear in mind that columns are only connected during elute and load phases.

The volume of a column might be chosen so that the eluting flow rate of the previous connected column must be reduced for the column to be able to handle it or to increase the volume so that the flow rate is accommodated; an example of which is presented in Figure 1, where it follows that the higher order columns have more variable volumes and flow rates. The implication of this is that sometimes a choice must be made of either choosing high volume and thus the possibility of maximum flow rate or the opposite which is a lower volume but then a lower flow rate. So either faster processing or lower column volumes can be found. This choice must be made for every column connection in a sequence, but as will be shown, in some circumstances there is only a single optimal point at a connection and thus no choice.

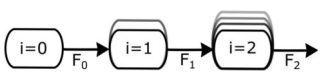



**FIGURE 1** A sequence of three columns. The number of different column sizes on each *i* depicts the variable volume selection that increases with *i*. Note that the flow is only connected between columns during elute phases

The length of the columns is assumed to remain constant during scale-up. This means that the maximum flow rate increases proportionally to the size of each column.

The first column in a sequence is denoted as $i = 0$, the second as $i = 1$, and so on, see Figure 1 for reference. The absolute capacity (mg/CV) of column $i$ must be enough so that it can bind in at least the same amount as the first column, $i = 0$. This is calculated with regard to the product loss and removed impurities on every column step. The term recovery yield, $u$, is introduced to describe this loss. $u$ is a parameter between 0 and 1 that describes how much of what was captured in a column is recovered and binds to the next. Thus, the absolute capacity of a column is only dependent on the capacity of the first column volume and the intermediate recovery yields.

As the maximum flow rate of a column is based on the pressure drop, and in the coupled eluate and load phases, it is reasonable that the pressure drop will increase because of the two-column train, and a robustness factor is introduced. The robustness factor, when >1, moves the optimal decision slightly away from the maximum flow rate in each step. Capacity robustness will not be discussed as it is a simple matter of increasing all column volumes equally.

$x_i$ is defined in Equation (1) as the ratio of resin volume between two consecutive columns. Column 0 becomes the capacity baseline that the rest of the columns must match. $y_i$ is defined in Equation (2) and the elute flow rate of the last column in a sequence is assumed to be set to maximum. The number of variables to optimize in a sequence is two times the number of columns subtracted by 1, that is, $x_i$ and $y_i$ for every connection. Definitions of other parameters can be found in the nomenclature list.

$$x_i = \frac{V_i}{V_{i-1}}, \tag{1}$$

$$y_i = \frac{F_{i-1,E}}{F_{i,\max}}. \tag{2}$$

### 2.1 | Constraints

The connection between columns must fulfill certain constraints to avoid above maximal back pressure and be designed to have enough capacity to avoid bottlenecks in the process. These requirements are expressed in mathematical form, based on $x$ and $y$ in Data S1.

There are two flow rate-limiting constraints where one of them sets a maximum flow rate to avoid maximum back pressure on the previous column, $i - 1$, and the other for the back pressure of the current column (including dilution), $i$, called *current FR limit* and *previous FR limit*, respectively. The *current FR limit* is inclined because its value depends on the current column and scales with it, while the *previous*

*FR limit* is a horizontal line that the flow rate cannot exceed because it is based on the previous column.

The capacity of all columns must accommodate at minimum the same amount as the first column in the sequence, which is used as baseline. The product loss of each unit is also accounted for. This third and final constraint is called the *capacity limit*.

Demonstrations of the general correlation between two connected columns through $x$ and $y$ are represented visually in Figure 2 by the two subplots where the grey area is outside the constraints and the white area within the constraints. Note that a chosen design value must be above $y = 0$, below the *previous FR limit*, and the *current FR limit* lines and on the right-hand side of the *capacity limit*. The most cost efficient choice is a low $x_i$ and a high $y_i$ which makes the entire *current FR limit* line optimal choices, or the single corner in the other situation.

When $x$ values are close to 1 in a sequence, the volume is a product sum of variables with little variance which makes the problem almost linear, and with completely linear cases there exists strictly either one or two superior points for each connection. These are named *corner* solutions. It is possible to calculate if there are one or multiple optimal points in a connection, the equation can be found in Data S1.

The intricate and interesting aspect is that choices over the sequence interact with each other. For instance, when a choice of $x$ and $y$ is made at $i = 1$, the choice at $i = 2$ will be a different one, as the column volume $i = 1$ is entirely different and absolute capacities change accordingly. This implies that the number of different options at $i$ are multiplied with the number of options $i - 1$. So the number of different purification sequence designs is exponential in relation to the number of columns, except for the single corner solutions.

As there are two variables for every column connection, $x$ and $y$, a sequence of four columns has a total of six variables in the optimization. The possible solutions to the optimization will be presented in $x - y$ plots wherein every subfigure a point is coordinates for $x$ and $y$, and thus represent a choice for each column connection.

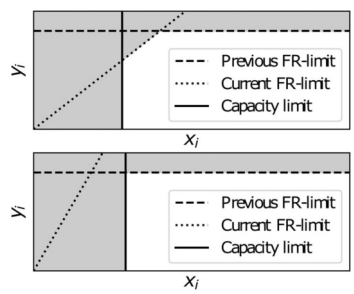



**FIGURE 2** $x - y$ plots. All points in the white region are feasible choices inside the constraints. In the top figure, all points on the $i$ maximum flow line as well as the two corners that lie inside the feasible region are optimal choices, thus multiple choices. In the bottom figure, there is only a single optimal point that is a corner where the capacity line crosses the $i - 1$ maximum flow rate line, that is, no choice

Ultrafiltration and diafiltration techniques are easily implemented features that can be integrated in the optimization. They would simplify the optimization as they require a flask that the pool from a column is loaded to or from, dividing the integrated sequence in two.

## 2.2 | Methods

### 2.2.1 | Case Study I: A general three-column downstream purification

It is very common with downstream processes based mainly on three chromatography purification steps, for capture, intermediate, and polishing. A typical case is the purification of monoclonal antibodies with Protein A-based capture step, a cation exchange intermediate step for aggregate removal, and a final polishing step, often an anion exchange. This is generalized in Case I example with three columns connected in sequence.

The first column is fixed to 24.5 mL; other details of this purification scheme can be found in Data S1. The pool is diluted by a factor of 1.5 in both connections making the *current FR limit* line more horizontal. This case study will compare the combinations of all corner solutions to a Pareto study explained further.

The results will also be compared to a disconnected column sequence, where the pool from one column is not immediately redirected to the next. Some assumptions are made in this comparison: minimal column sizes with regard to capacity are assumed, the maximum flow rate of the column is used during elution and loading, and the same dilution factors as in the connected sequence are used.

### 2.2.2 | Case Study II: Continuous twin-capture

The second case is an integrated purification platform with five columns that is an extension to Case Study I, demonstrating the possibility of the equation system to schedule a sequence to handle a constant feed stream. The feed stream comes from a continuous bioreactor and has a feed flow rate of 8 L a day and a concentration of 10 mg protein/L. There are two capture columns that alternate receiving the feed. During the loading time of one of the capture columns the rest of the sequence will be running, see the Gantt chart in Figure 3. There are four phases in every column: load, wash, elute, and regeneration, and the elute phase happens simultaneously as the load phase of the following column. The columns should be as small as possible to reduce resin costs while being within the time frame that allows a cycle to be fully purified while the other capture column is loaded.

A new constraint is introduced in this case study, which is that the entire sequence starting from washing the capture column to regenerating the last column must be shorter than the load time of the capture column. To conclude, the objective is still to minimize the total volume of all the columns, but with the added task of handling the continuous feed stream. The volume of the capture column, $V_0$, is

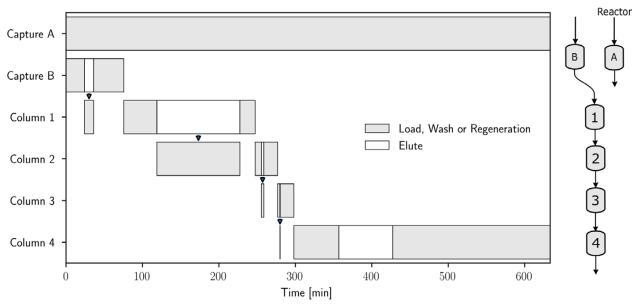



**FIGURE 3** Case II. A Gantt chart representation of all steps in the continuous twin-capture sequence. Column A or B is loaded while the rest of the sequence is running. After a complete sequence, connections to A and B are switched and B is loaded instead while a new sequence starts with a Column A wash step. Note that on every column step the load phase is active simultaneously as the elute step of the previous column, apart from the capture column

therefore added as an additional variable while it was fixed in the previous case. All necessary data for the sequence are presented in Table S2 found in Data S1.

### 2.2.3 | Multiobjective optimization

In both case studies, there are two competing objectives to be minimized. In the first case, the objective is to reduce the total column volume and the total process time. As these two tasks are competing and are not compatible with each other, there will instead of a single optimal point be several. This is why a new objective function is introduced which tests for both objectives but at different weights to the individual objectives and finds the most efficient choice for each weight. This is called a Pareto optimization.

The overall objective for the second case is to find a solution that handles the constant feed stream. As buffer consumption also is a nonnegligible cost when producing pharmaceuticals, it is worthwhile to investigate how a set of different column volumes impacts it. Therefore, a Pareto optimization described in Data S1 is performed also in the second case study to show this relationship.

## 2.3 | Solver

The optimization is a nonlinear problem as the solution depends on a product sum of $x$ values, see Equations (4) and (7) in Data S1. This generally means that the optimal point does not have to be at the constraint boundary or in any of the corners.

The optimization was carried out using the Scientific Computing Tools for Python, SciPy, version 1.1.0, https://scipy.org/. The function *optimize.minimize* from *SciPy* was used, with the method sequential least squares programming, which is an iterative method for solving nonlinear optimization problems consisting of both constraints and bounds.

# 3 | RESULTS AND DISCUSSION

The robustness factor moves the flow rate down slightly in all connections. The grey lines in Figures 5 and 7 are plotted to show potential choices without any robustness factor. Throughout this section, solutions are referred to as complete sequences and choices as the individual steps within the solutions.

## 3.1 | Case I

All optimized results overlap the corner solutions, although one of the solutions was never selected by the algorithm as an optimal solution, see Figure 4. Solution *vv*, volume-then-volume, and *tt*, time-then-time, are minimizing the volume and the process time, respectively, while solution *vt* and *tv* are mixes of the two, where *tv* was exempted in the Pareto optimization.

The choices of solution *tv* are shown in Figure 5 with the dots, first choosing a low volume and flow rate and then the second choice is higher volume and flow rate. Solution *vt* is the opposite, where the greater volume is initially selected and then the lower volume with the result that the last volume is still higher overall, since $x$ is the relation between volumes of column $i$ and $i-1$. So even if it would be

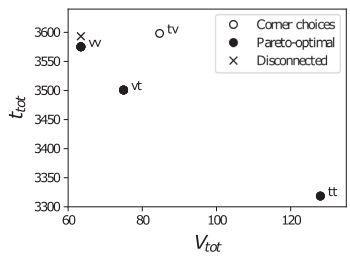



**FIGURE 4** Case I Pareto front of solutions. All circles were obtained when selecting corner choices in the $x-y$ plots. The symbols represent the choice in each solution: *vv* is volume-then-volume, *vt* is volume-then-time, *tv* is time-then-volume, and *tt* is time-then-time. The full circles are choices selected by the Pareto optimization. As *tv* was the only corner choice not selected, it is not Pareto-optimal. The disconnected solution is where the columns are not coupled and the pool is not directly applied to the next column

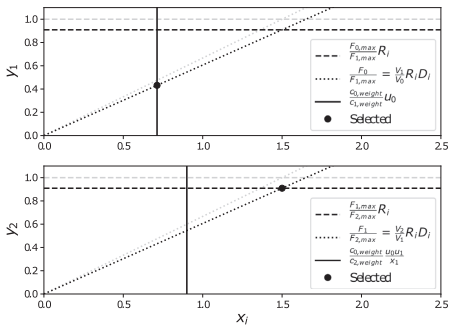



**FIGURE 5** Case I. The choices in solution *tv* from Figure 4. The choice in the upper part is the lowered flow rate, allowing the column volume to be lower. The second choice is the higher flow rate resulting in a slightly higher column size in the last step. Note that the grey lines are the nonrobust constraints shown to visualize the effect of the robustness factor, in this case $R$ = 1.1, which gives a lowered flow rate of 10%

the same $x$ value, the absolute volume is still greater. This result is a demonstration that early choices have an impact on later choices, as discussed in the section 2.1.

A comparison of connected and disconnected column connections was made and is represented in Figure 4. The disconnected solution for this case is close to the connected solution. However, this depends greatly on the dilution factor. Small values of the dilution factor make the connected solution more efficient and vice versa. This is explained by that the incline of the *current FR limit* forces down $y$ which increases $t_{tot}$ and thus defeats the purpose of connecting the columns. With dilution factors equal to 1 and comparing the lowest volumes, $t_{tot}$ for disconnected is 3,493 min and for connected 3,396 min (not shown in any figure). Our conclusion is that connected sequences are generally more beneficial if no or low dilution is required between steps, otherwise the flow rate becomes lowered so much that a disconnected sequence might be faster, assuming a time efficient dilution.

The results from this case study can help to design a variety of integrated sequences as the case is quite general with its three connected bind-and-elute columns.

The results from this case study can be directly applied to complete downstream purification units as was discussed in Section 1.[4-6] For research purposes, integrated and automated processes that can be sped up can increase the research output when testing a variety of candidates. At large manufacturing facilities it might be worthwhile to keep volumes down if steps are connected as resin material is a great cost at full-scale.

## 3.2 | Case II

The solution with the highest total volume and lowest buffer consumption ($w$ = 0) had a capture column size of 304 mL which decreased to 213 mL ($w$ = 1) while traversing the Pareto in Figure 6. The cycle time for Case Study II changes with the capture column size from 820 min ($w$ = 0) to 575 min ($w$ = 1). It is reasonable that lower volumes also had lower cycle times, and also that the buffer consumption increased, per mg product, as the buffer exchangers were larger to be able to handle the flow rate making it possible to stay within the cycle time constraint.

Comparing $y_2$ in Figure 7, the optimal solution is on the flow rate limit constraint for lower $w$ values and not in a corner. This special balance might be because the connection between a buffer exchanger and a bind-and-elute column, that is, large to small column, gives a very small *capacity limit*. With $w$ = 0.2, the corner with the greater $x$ value corrects the total volume while a low flow rate limits the cycle time. Apparently, these corners are worse with these special circumstances, which influence the algorithm to select a point on the *current FR limit* line.

Lower capture column size corresponds to lower total volume of the sequence. Even though $x_2$ is higher at $w$ = 0.8, the initial capture column size, and $x_3$ are low enough so that the total column size is lower than at $w$ = 0.2, see Figures 7 and 6.

As there were big size differences of columns in this case, all connections were not simple choices and instead there were gliding optimal points over the Pareto front. The implication is that choices are more complex in a connected step from a large to a small column.

This case study was more specialized but allowed the reader to see how a downstream section, with a constantly received volumetric flow rate as a constraint, might be scheduled with the help of the nonlinear optimization described in this paper. The practical significance of this is that integrated sequences that have a large variety of inherent choices that must be made can be optimized to function within a constraint by tweaking volumes, flow rates, and cycle times.

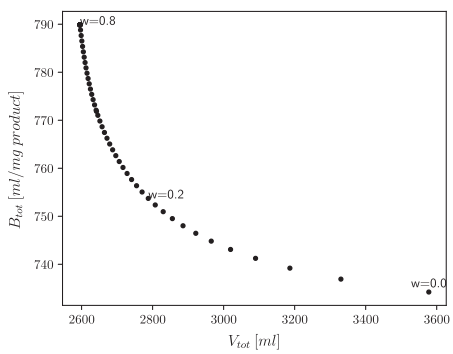



**FIGURE 6** Case II Pareto front over the total volume of columns and the total buffer consumption. The location in the Pareto weights are shown at 0 and the two locations shown in Figure 7 at 0.2 and 0.8

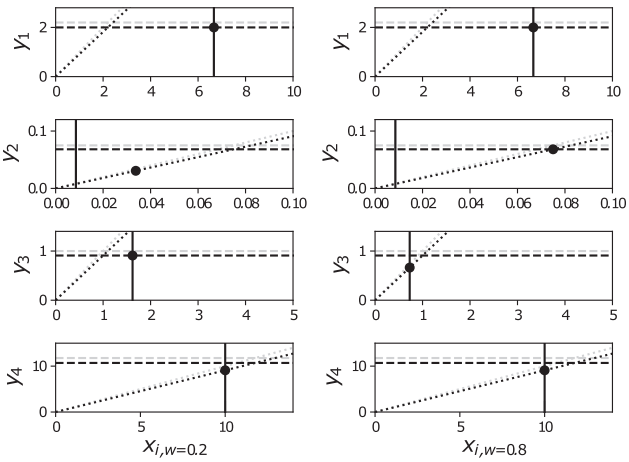



**FIGURE 7** Case II. $x - y$ plots over the four-column connections with $w = 0.2$ in the left subfigures and $w = 0.8$ in the right subfigures. For $w = 0.2$, the capture column size is 235 mL and the total volume is 2,795 mL. For $w = 0.8$, the capture column size is 213 mL and the total volume is 2,593 mL

### 3.3 | General remarks

Interestingly, bind-and-elute columns act similarly to buffer exchange columns. The only principal difference is the magnitude of the $x$ values and therefore the volumetric flow rate in the elute phase must be set carefully to not exceed the maximum flow rate limit of following column.

When the capacity is high enough in column $i - 1$ compared to column $i$, there will only be a single optimal choice. Newer versions of affinity columns usually have a very high capacity, compared to maximum flow rate, which reduces the overall problem size as the following column must be relatively large to handle the load which makes it likely that it is able to handle the maximum eluting flow rate.

## 4 | CONCLUSIONS

For designers of downstream purification processes, this study can be seen as a tool to use for integrated and continuous sequences and a couple of different heuristics can be drawn from the exemplified case studies.

Utilizing the demonstrated algorithm, a sequence of choices can tune a purification process toward either speed or column size. Both are important parameters during optimization and scheduling of a downstream process to keep resin material costs down, increase the speed of a system, and calibrate a complex system to handle a constant feed stream from a perfusion reactor.

Assumed optimal choices, corner choices in the article, reflected accurately the optimization of Case Study I when there were relatively equal column capacities. Even though the problem was nonlinear, assumed optimal choices were the same as those chosen by the optimization algorithm, except for one solution. Our advice from that result is that if speed is necessary, keep volumes down early in the sequence and increase only in the final steps.

Connected sequences are faster if there is no or low dilution factors between column steps as the elute and load phases are combined. With dilution, however, the flow rate must be lowered to avoid transcending the maximum back pressure.

With the given reactor feed, the optimization, combined with constraints and column data, was shown to be able to produce an optimal operating sequence. The amount of buffer used can be traded with some reduction of total resin volume.

Manufactured columns come in a selection of column sizes and radii. Instead of looking at a continuous range of column radius, it could be more realistic to study discrete values of column sizes. Another expansion to this study could be to include packing material costs as an additional parameter for a more detailed cost analysis.

## NOMENCLATURE

| | |
|---|---|
| $B$ | Buffer consumption (mL/mg product) |
| BE | Buffer exchanger (–) |
| $c$ | Capacity by weight or volume (mg product/mL packing) or (mL load/mL packing) |
| CV | Column volumes (–) |
| $D$ | Dilution factor (–) |
| $F$ | Volumetric flow rate (CV/min) |
| $n$ | Number of connections (–) |
| $R$ | Robustness factor (–) |
| $t$ | Time (min) |
| $u$ | Recovery yield (–) |
| $V$ | Column volume (mL/column) |
| $w$ | Weight factor (–) |
| $x$ | Column volume quotient (–) |

$y$ Flow rate factor (–)

## Greek Symbols

$\varphi$ Phase volumes CV


## ACKNOWLEDGMENTS

Novo Nordisk A/S, Swedish Orphan Biovitrum, and the VINNOVA are gratefully acknowledged for financial support.



## ORCID

*Anton Löfgren* https://orcid.org/0000-0002-8182-8816
*Niklas Andersson* https://orcid.org/0000-0002-6032-5494


## REFERENCES


1. Warikoo V, Godawat R, Brower K, et al. Integrated continuous production of recombinant therapeutic proteins. *Biotechnol Bioeng*. 2012;109(12):3018-3029.
2. Rathore AS, Kateja N, Agarwal H. Continuous downstream processing for production of biotech therapeutics. In: Subramanian G, ed. *Continuous Biomanufacturing: Innovative Technologies and Methods*. Weinheim, Germany: John Wiley & Sons; 2017:261-288.
3. Patil R, Walther J. Continuous manufacturing of recombinant therapeutic proteins: upstream and downstream technologies. *New Bioprocessing Strategies: Development and Manufacturing of Recombinant Antibodies and Proteins*. Framingham, MA: Springer; 2017:277-322.
4. Andersson N, Lfgren A, Olofsson M, Sellberg A, Nilsson B, Tiainen P. Design and control of integrated chromatography column sequences. *Biotechnol Prog*. 2017;33(4):923-930.
5. Lofgren A, Andersson N, Sellberg A, Nilsson B, Lofgren M, Wood S. Designing an autonomous integrated downstream sequence from a batch separation process—an industrial case study. *Biotechnol J*. 2018;13(4):e1700691.
6. Steinebach F, Ulmer N, Wolf M, et al. Design and operation of a continuous integrated monoclonal antibody production process. *Biotechnol Prog*. 2017;33(5):1303-1313.
7. Godawat R, Konstantinov K, Rohani M, Warikoo V. End-to-end integrated fully continuous production of recombinant monoclonal antibodies. *J Biotechnol*. 2015;213:13-19.
8. Angarita M, Müller-Spth T, Baur D, Lievrouw R, Lissens G, Morbidelli M. Twin-column CaptureSMB: a novel cyclic process for protein A affinity chromatography. *J Chromatogr A*. 2015;1389:85-95.
9. Baur D, Angarita M, MüllerSpth T, Steinebach F, Morbidelli M. Comparison of batch and continuous multicolumn protein A capture processes by optimal design. *Biotechnol J*. 2016;11(7):920-931.
10. Costa S, Almeida A, Castro A, Domingues L. Fusion tags for protein solubility, purification and immunogenicity in *Escherichia coli*: the novel Fh8 system. *Front Microbiol*. 2014;5:63.
11. Nfor BK, Verhaert PD, van der Wielen LA, Hubbuch J, Ottens M. Rational and systematic protein purification process development: the next generation. *Trends Biotechnol*. 2009;27(12):673-679.
12. Vasquez-Alvarez E, Pinto J. Efficient MILP formulations for the optimal synthesis of chromatographic protein purification processes. *J Biotechnol*. 2004;110(3):295-311.
13. Baumann P, Hubbuch J. Downstream process development strategies for effective bioprocesses: trends, progress, and combinatorial approaches. *Eng Life Sci*. 2017;17(11):1142-1158.


## SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.


**How to cite this article:** Löfgren A, Yamanee-Nolin M, Tallvod S, Fons JG, Andersson N, Nilsson B. Optimization of integrated chromatography sequences for purification of biopharmaceuticals. *Biotechnol Progress*. 2019;e2871. https://doi.org/10.1002/btpr.2871

# Paper VIII

Journal of Environmental Management 240 (2019) 527–536

Contents lists available at ScienceDirect

Journal of Environmental Management

journal homepage: www.elsevier.com/locate/jenvman

Research article

# A physically based model for mesoscale SuDS – an alternative to large-scale urban drainage simulations

Salar Haghighatafshar[a,*], Mikael Yamanee-Nolin[b], Magnus Larson[c]

[a] Water and Environmental Engineering, Department of Chemical Engineering, Lund University, P.O. Box 124, SE-221 00 Lund, Sweden
[b] Department of Chemical Engineering, Lund University, P.O. Box 124, SE-221 00 Lund, Sweden
[c] Department of Water Resources Engineering, Lund University, P.O. Box 118, SE-221 00 Lund, Sweden

ARTICLE INFO

*Keywords:*
SuDS
SCM
Modelling
Hydraulics
Rainfall-runoff
Urban hydrology

ABSTRACT

This study presents a deterministic, lumped model to simulate mesoscale sustainable drainage systems (SuDS) based on a conceptualization of the stormwater control measures (SCMs) making up the system and their influence on the runoff process. The conceptualization mainly relies on parameters that are easily quantifiable based on the physical characteristics of the SCMs. Introducing a nonlinear reservoir model at the downstream end of the SuDS results in a fast model that can realistically describe the runoff process at low computational cost. Modelled hydrographs for the study area in Malmö, Sweden, matched data with regard to the overall shape of the hydrograph as well as the peak discharge and lag time. These output parameters are critical factors to be considered in the design of large systems consisting of mesoscale SuDS. The algebraic foundation of the developed model makes it suitable for large-scale applications (e.g., macroscale), where the simulation time is a decisive factor. In this respect, city-wide optimization studies for the most efficient location and implementation of SuDS are substantially accelerated due to fast and easy model setup. Moreover, the simplicity of the model facilitates more effective communication between all the actors engaged in the urban planning process, including political decision makers, urban planners, and urban water engineers.

## 1. Introduction

Sustainable drainage systems (SuDS) are considered as a viable alternative for urban drainage in a changed climate. SuDS as a concept has been around since the 1970s, but has gained increased attention in the research community during the recent decades due to full-scale implementations. So far, numerous research and investigations have focused on the effects and consequences of stormwater control measures (SCMs) at local scale, whereas a clear shortage has been identified regarding the knowledge on upscaling SCMs (Golden and Hoghooghi, 2018). Thus, determining the hydraulic behavior of SuDS is of great significance and different modelling approaches have been suggested for describing individual SCMs as constituents of SuDS (García-Serrana et al., 2017; Locatelli et al., 2014; Roldín et al., 2013). However, examples of models covering both the mesoscale (neighbourhood-scale) and macroscale (city-scale) processes are relatively rare. Two major factors are known to be the main reasons for the lack of appropriate modelling approaches at these scales: (1) scarce large-scale implementations of SuDS in cities (Loperfido et al., 2014; Zhang et al., 2012) and (2) complexity of the existing modelling methodologies leading to extensive parameter estimation and calibration/validation procedures, as well as computationally costly models (Elliott et al., 2009; Freni et al., 2010; Haghighatafshar et al., 2018b; Jayasooriya and Ng, 2014; Krebs et al., 2014; Locatelli et al., 2014). Therefore, studies regarding upscaling of SuDS to a city-wide level, which demand numerous simulations of SuDS-drainage network interaction, are presently not feasible using the available distributed hydrodynamic models (e.g., MUSIC, SWMM, MIKE 21 and MIKE FLOOD). Even models based on artificial intelligence (AI), such as Artificial Neural Networks (ANN) using machine learning techniques that are widely used for rainfall-runoff simulations in regional scale watershed studies – with rare applications in exclusively urban studies – (Adamowski and Prasher, 2012; Hu et al., 2018; Mosavi et al., 2018), would not serve the urban drainage requirements with regard to upscaling of SuDS to city level. This is mainly because such black-box models do not provide substantial information on how the characteristics and the configuration of the system would affect the runoff outcome, ignoring any interaction between SuDS and the existing pipe-network. More importantly,

* Corresponding author.
*E-mail addresses:* Salar.Haghighatafshar@chemeng.lth.se (S. Haghighatafshar), Mikael.Yamanee-Nolin@chemeng.lth.se (M. Yamanee-Nolin), Magnus.Larson@tvrl.lth.se (M. Larson).

https://doi.org/10.1016/j.jenvman.2019.03.037
Received 5 November 2018; Received in revised form 28 January 2019; Accepted 7 March 2019

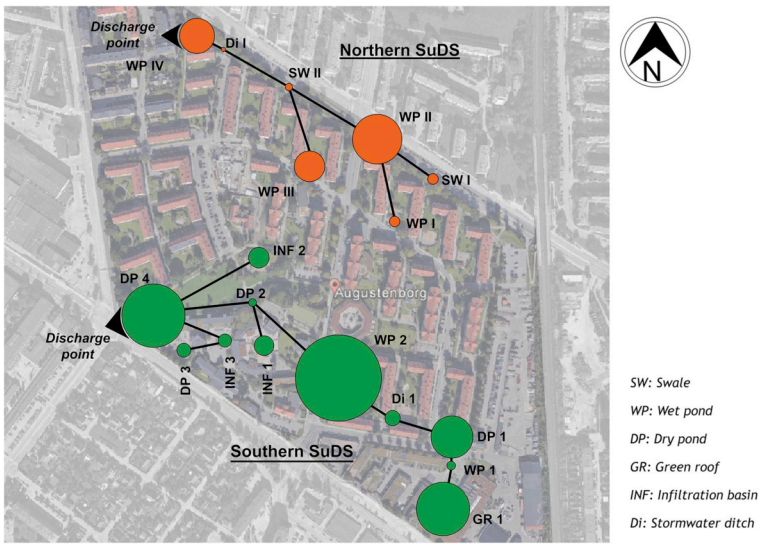



**Fig. 1.** The configuration the Northern and Southern SuDS, Augustenborg with different types and setups of SCMs, adopted from Haghighatafshar et al. (2018a).

enormous amount of monitored rainfall-runoff data with large diversity are required for training the model to deliver reliable results for any likely scenario (Géron, 2017; Halevy et al., 2009). Such datasets are rarely available within the field of urban drainage modelling.

In order to tackle the abovementioned challenges regarding large-scale simulations, new appropriate and efficient model concepts have to be developed for SuDS. Since urban drainage infrastructure performs as a dynamic and interconnected complex system of systems – i.e. reservoirs, SCMs, SuDS, pipe-networks, combined sewer overflows (CSO), wastewater treatment plants, receiving waters, etc. – the required simulation tool for SuDS should be capable of taking hydraulic interactions with the other elements of urban drainage network into consideration, in addition to being fast and reliable. The compromise would be to adopt simplified conceptual models for SuDS with focus on estimation of key parameters of the generated hydrographs, a catchment response time parameter and the peak discharge (Gericke and Smithers, 2018).

Thus, the aim of this study is to develop a fast, robust and flexible physically based model for inexpensive simulation of existing SuDS at meso- and macroscale with focus on estimation of lag time (as the selected catchment response time parameter) and peak discharge. The model has to be capable of being coupled with 1-dimensional sewer models for investigation of interactions between SuDS and sewer networks. This can be used in preliminary screening studies regarding the required retention capacity and the location of SuDS with respect to the entire sewer network (Zoppou, 2001), while in the later stages of the study, more complex models can be employed for detailed investigation of the selected locations.

## 2. Methodology

Haghighatafshar et al. (2018a) introduced a conceptual model for SuDS, developed using observed discharge patterns from a full-scale implementation of unique mesoscale SuDS in downtown Malmö, Sweden. The model schematized mesoscale SuDS as a 1-dimensional series of interconnected retention basins (SCMs) of different sizes and types. It was shown that the order and placement of the constituent SCMs determine the overall performance of the SuDS. In the present study, the conceptual model is implemented mathematically to estimate the total discharge volumes using easily quantifiable physical parameters. Moreover, the model is further developed and enhanced to simulate the entire discharge hydrograph from mesoscale SuDS with the objective to predict the peak discharge and lag time, which are important parameters in the design of individual SCMs as well as meso- and macroscale SuDS. The algebraic formulation of the developed model makes it suitable for large-scale applications (e.g., macroscale), where the simulation time is a decisive factor. The development of the model is based upon studies made in Augustenborg in Malmö, Sweden. Flow measurements at the SuDS in Augustenborg were carried out during a 2-year period and were used for calibration and validation of the model. This section provides information about the case study area and the discharge measurements.

### 2.1. Augustenborg, Malmö

The Augustenborg area in the centre of Malmö, Sweden, contains unique examples of mesoscale SuDS. Augustenborg was originally drained via the municipal combined sewer network that was built during the 1950s. However, in the late 1990s, the runoff from the area was disconnected from the combined sewer system and it was led through the SuDS-retrofits constructed on the surface. The area consists of two separate SuDS-retrofits (denoted as the Northern and Southern SuDS), in total encompassing a drainage area of about 16 ha. The implementation of SuDS-retrofits in Augustenborg was done as part of a larger project for upgrading the social status of the neighbourhood. The Northern SuDS serves a catchment of about 6.3 ha and mainly consists of swales and open channel/ditch systems with a few wet ponds. The Southern SuDS receives runoff from an area of about 9.6 ha and is more

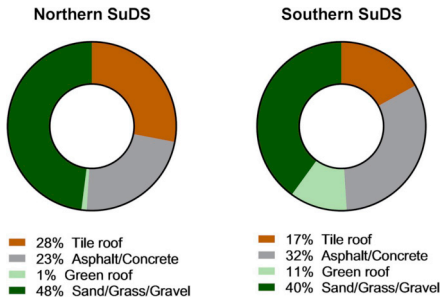



**Fig. 2.** The distribution of different types of surfaces in the Northern and Southern SuDS, Augustenborg.

diverse in terms of the employed SCMs. Green roofs, infiltration basins, wet/dry ponds, swales, and designated flood area are all present in the Southern SuDS.

The outflow from both systems is eventually discharged into the underground pipe network of the city, where the flowmeters were installed. The configuration and setup of the Northern and the Southern SuDS in Augustenborg are presented and discussed in detail in Haghighatafshar et al. (2018a). The layout of the implemented SCMs in each system and their connection order are presented in Fig. 1.

Fig. 2 shows the distribution of different types of surfaces in the Northern and Southern SuDS in Augustenborg. As seen, almost 50% of the surfaces in both the Northern and Southern SuDS consist of pervious surfaces of different kinds. Green roofs encompass a substantial share of the catchment for the Southern SuDS (approximately 11%), whereas the share of green roofs in the Northern SuDS is negligible (1%).

### 2.2. Rainfall-runoff measurements

The discharge from the Northern and the Southern SuDS in Augustenborg was measured at the most downstream points of the catchments where the overflow from the systems is diverted into the municipal pipe-network. The flow gauges used in this study were Mainstream Portable Area-Velocity (AV)-flowmeters. A single tipping bucket rain gauge with 0.2 mm resolution, Casella CEL, was installed in the area for monitoring the rainfall. The rainfall and runoff was monitored and logged for about 2 years recording 10 larger rainfall events that generated significant discharge from the systems; these events were selected for model calibration and validation in this study. Table 1 presents the characteristics of the recorded rainfall events denoted with the identifiers (IDs) *A* to *J*. The four recorded rainfall events with the median depths (i.e. *D*, *E*, *F* and *G*) were selected to be used for the calibration, while the remaining six events (both smaller and larger than the calibration events) were kept for the validation.

**Table 1**
The characteristics of the rainfall events used in this study.

| Status | Rainfall ID | Rainfall depth (mm) | Rainfall duration (h) | Rainfall peak (h) |
|---|---|---|---|---|
| *Validation* | A | 7.8 | 1.34 | 0.07 |
| *Validation* | B | 10.6 | 3.41 | 3.05 |
| *Validation* | C | 13.4 | 2.25 | 0.03 |
| **Calibration** | **D** | **13.8** | **9.34** | **8.75** |
| **Calibration** | **E**[a] | **15.6** | **4.84** | **0.23** |
| **Calibration** | **F**[b] | **17.4** | **3.92** | **0.07** |
| **Calibration** | **G** | **17.8** | **15.58** | **11.77** |
| *Validation* | H | 19.0 | 22.70 | 2.64 |
| *Validation* | I | 22.6 | 4.17 | 1.42 |
| *Validation* | J | 28.4 | 9.17 | 6.58 |

[a] Rain E was only included in the calibration of the *Northern SuDS*.
[b] Rain F was only included in the calibration of the *Southern SuDS*.

Lag time in hydrological studies may have different definitions depending on the compared reference points on the hectograph and the corresponding hydrograph (Gericke and Smithers, 2014; Schulz and Lopez, 1974). However, in the context of urban hydrology, lag time can also be calculated as the time interval between the rainfall peak and the peak discharge (Grayson et al., 2010; Mansell, 2003), which is also employed in this study. In case of rainfall events with double peaks, the later peak and its corresponding discharge are employed for the lag time calculations. The reason behind this specific criterion is the hypothesis that the first peak is possibly consumed for filling the retention capacity in the system, whereas the second peak has a direct connection to the peak discharge.

#### 2.2.1. Possible error sources

It is important to note that the on-site measurements of flow and rainfall are subjected to some degree of uncertainty. The flowmeters tend to miss the low flows when the water depth in the pipe is less than the thickness of the sensor. The effect of this uncertainty on the total runoff volume depends on the characteristics of the hydrograph with respect to the distribution of volume against flow. Moreover, it is assumed that the rainfall recorded by the single tipping bucket rain gauge is homogeneously distributed and thus is representative for the entire study area. However, studies show that the spatial variation in the rainfall pattern (with respect to depth, intensity, and duration) can be substantial, even across sub-kilometer catchment scale (Fiener and Auerswald, 2009). It is also found that the spatial variability resulting from the rainfall dynamics, i.e., movement of rainfall over a catchment, affects the resulting hydrograph (Singh, 1997). For example, in case of Augustenborg, the Southern SuDS with a catchment area of 9.6 ha is more than 50% larger than the Northern SuDS and this could negatively affect the reliability of the runoff measurements in the Southern SuDS compared to the Northern SuDS.

## 3. Model development

### 3.1. Schematization of SCMs and mesoscale SuDS

In the specific schematization employed in this study, a single SCM – regardless of its type – is defined by three fundamental parameters, being the surface area of the SCM ($A_{SCM}$), retention depth in the freeboard ($S_{fb}$) and rapid infiltration depth ($S_{inf}$) (Haghighatafshar et al., 2018a). Rapid infiltration depth in this context is the infiltration capacity primarily provided by the unsaturated zone of the filter medium. For simplicity, it is assumed that the infiltration process stops as soon as the available capacity in the unsaturated zone is consumed, i.e., no seepage of infiltrated water to the groundwater/deeper soil layers is considered. The values for the rapid infiltration depths are adopted from (Haghighatafshar et al., 2018b; Nordlöf, 2016).

Fig. 3 shows how these three parameters are defined to characterize the principal retention-based functions of a SCM.

The SCMs in Augustenborg are characterized based on the scheme presented in Fig. 3 and their characteristic values are presented in Table 2.

Mesoscale SuDS consist of multiple interconnected SCMs, as shown in Fig. 4, through which proportions of the rainfall are intercepted in upstream SCMs (*effective retention*, $R_e^i$), and the rest of the runoff flows to the immediate downstream SCM (*i.e.*, $V_{out}^i$). Consequently, a discharge from the entire system is initiated when the retention capacity of the most downstream SCM is exceeded. Thus, the model operates by employing a volume transfer approach, where the storage volumes and the volume transfers are derived from the properties of the SCMs and the general characteristics of the catchment. The governing equations for effective retention and discharge volume from each SCM are given

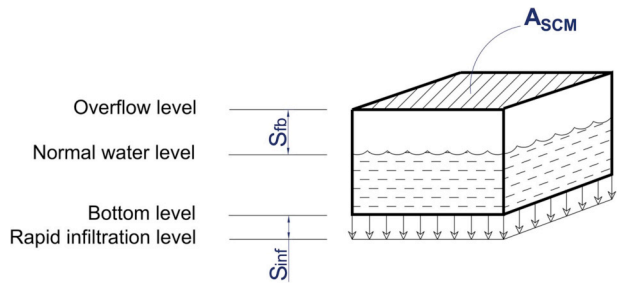



**Fig. 3.** The employed scheme for a single SCM within the framework of the developed model. $A_{SCM}$: surface area of the SCM, $S_{fb}$: retention depth in the freeboard and $S_{inf}$: rapid infiltration depth.

by equations (1) and (2).

$$R_e^i = \frac{(S_{fb}^i + S_{inf}^i) \times A_{SCM}^i}{DCIA^i} \quad (1)$$

$$V_{out}^i = \frac{(R - R_e^i) \times DCIA^i}{1000} + \sum_{j=1}^{a} V_{out}^{i-j} \quad (2)$$

In equations (1) and (2), $R_e^i$ is the effective retention capacity of the SCM i (mm), $S_{fb}^i$ is the storage depth in the freeboard of the SCM (mm), $S_{inf}^i$ is the storage depth in the infiltration layer, $A_{SCM}^i$ is the area occupied by the SCM ($m^2$), $DCIA^i$ is the directly connected impervious area to the SCM ($m^2$), $V_{out}^i$ is the discharged volume from the SCM ($m^3$), $R$ is the rainfall depth (mm), and $\sum_{j=1}^{a} V_{out}^{i-j}$ is the sum of the inflow to the SCM from the adjacent upstream SCM(s) ($m^3$), where $a$ is the number of the immediate upstream SCMs connected to component $i$. Notice that if $V_{out}^i < 0$, it should be set equal to zero. Depending on the depth of rainfall with respect to the effective retention capacity ($R - R_e^i$), Eq. (2) represents different possible conditions with regard to the SCM. These conditions can be categorized as follows:

$\boldsymbol{R > R_e^i} \rightarrow$ The local retention capacity of the SCM, i.e., $S_{fb}^i + S_{inf}^i$, is already exceeded and a discharge is initiated from the SCM. Any incoming volumes from the immediate upstream SCM(s), i.e., $\sum_{j=1}^{a} V_{out}^{i-j}$, will also pass through the system, without any retention.

$\boldsymbol{R = R_e^i} \rightarrow$ The local retention capacity of the SCM, i.e. $S_{fb}^i + S_{inf}^i$, is already achieved so depending on whether there is an input from the immediate upstream SCM(s), there might be a discharge from the system: $V_{out}^i = \sum_{j=1}^{a} V_{out}^{i-j}$.

$\boldsymbol{R < R_e^i} \rightarrow$ The local retention capacity is partly used up. This means that the discharge from the system is initiated if and only if $\sum_{j=1}^{a} V_{out}^{i-j} > \frac{|R - R_e^i| \times DCIA^i}{1000}$, else $V_{out}^i = 0$.

In order to develop the volume-based equations (1) and (2) to describe flow from the SuDS, the rainfall data should be introduced as an *equally spaced accumulated rain depth time series*. In this study, the constant time-step is selected to be 5 min (*i.e.*, $\Delta t = 5$ min). Thus, for any given time ($t$), equation (2) can be transformed to equation (3). No flow distribution is considered for the discharge from the components; the outflow from a component is obtained as an average flow for the entire time step.

$$q^{i,t} = \frac{V_{out}^{i,t} - V_{out}^{i,t-1}}{\Delta t} = \frac{(R^{acc,t} - R_e^i) \times DCIA^i}{\Delta t \times 1000} + \sum_{j=1}^{a} \frac{V_{out}^{i-j,t}}{\Delta t} - q^{i,t-1}, \quad for\ i = 1 \ldots n \quad (3)$$

In which, $q^{i,t}$ is the discharge from component $i$ at time $t$ and $R^{acc,t}$ is the accumulated rainfall (mm) at time $t$. This equation is used to describe the discharge from components i= 1...n. The flow characteristics of the discharge at the most downstream point in the SuDS (n) is especially important when simulating the influence of the entire urban drainage network on the runoff process. In the schematization used in this paper, the discharge from component (n) is assumed to be the inflow to a virtual component ($Q_{in}$), for which the discharge ($Q_{out}$) is simulated using a non-linear reservoir model. Fig. 4b shows how the final

**Table 2**
The characteristics of the implemented SCMs in the Augustenborg's SuDS, adopted from Haghighatafshar et al. (2018a,b).

| *System* | *SCM ID* | *Storage, $S_{fb}^i$ (mm)* | *Infiltration $S_{inf}^i$ (mm)* | *SCM area, $A_{SCM}$ ($m^2$)* | *DCIA ($m^2$)* | *Fed by* |
|---|---|---|---|---|---|---|
| *Northern SuDS* | ***SW I*** | 5 | 15 | 740 | 2780 | – |
| | ***WP I*** | 250 | 0 | 90 | 3920 | – |
| | ***WP II*** | 250 | 0 | 200 | 1120 | SW I, WP I |
| | ***WP III*** | 250 | 0 | 90 | 1620 | – |
| | ***SW II*** | 5 | 15 | 240 | 3400 | WP II, WP III |
| | ***Di I*** | 5 | 0 | 80 | 2100 | SW II |
| | ***WP IV*** | 350 | 0 | 160 | 3500 | Di I |
| *Southern SuDS* | ***GR 1*** | 0 | 45 | 10000 | 10000 | – |
| | ***WP 1*** | 200 | 0 | 140 | 8500 | GR 1 |
| | ***DP 1*** | 105 | 45 | 100 | 560 | WP 1 |
| | ***Di 1*** | 200 | 0 | 98 | 1685 | DP 1 |
| | ***WP 2*** | 150 | 0 | 700 | 560 | Di 1 |
| | ***INF. 1*** | 35 | 25 | 800 | 3150 | – |
| | ***DP 3*** | 105 | 45 | 170 | 2760 | – |
| | ***INF. 2*** | 0 | 25 | 800 | 1300 | – |
| | ***DP 2*** | 0 | 15 | 200 | 1180 | WP 2, INF 1 |
| | ***INF. 3*** | 500 | 25 | 115 | 5460 | DP 3 |
| | ***DP 4*** | 25 | 25 | 900 | 300 | DP 2, INF 2, INF 3 |

SW: Swale, WP: Wet pond, DP: Dry pond, Di: Ditch (stormwater ditch), GR: Green roof, INF: Infiltration basin.

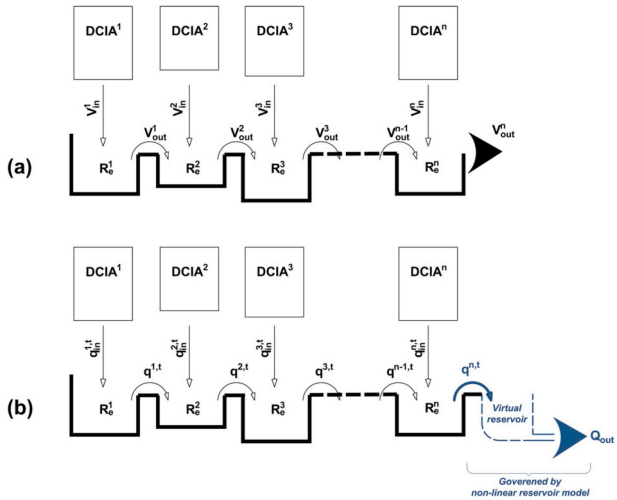



**Fig. 4.** (a) Schematization of mesoscale SuDS based on the conceptual model introduced by Haghighatafshar et al. (2018a). (b) Enhanced version of the volume-transfer model resulting in discharge (flow) simulation.

discharge is treated in the applied scheme, leading to the following equation:

$$\frac{dS}{dt} = Q_{in} - Q_{out} \overset{Q_{out} = kS^m}{\Rightarrow} \frac{\boldsymbol{dS}}{\boldsymbol{dt}} = \boldsymbol{Q_{in} - kS^m} \tag{4}$$

where $S$ ($m^3$) is the dynamic storage volume in the virtual reservoir, $k$ ($min^{-1}$) and $m$ (no units) are the reservoir coefficients.

As seen, the model utilizes very few (three) and consistent parameters to describe different types of SCMs, namely $S_{fb}^i$, $S_{inf}^i$ and $A_{SCM}^i$. This consistency and small number of parameters make the process of the parameter estimation relatively easy and straightforward compared to other models like SWMM, which requires approximately 12 parameters for simulation of green roofs or 7 parameters for pervious pavement structures (Jato-Espino et al., 2016).

The open-source programming language Python™ was employed for implementation of the model, as well as parameter estimation (calibration), validation, and sensitivity analysis. The Python-code generated for the model is able to communicate with Microsoft Excel in which an easy-to-use method is employed to describe the configuration of the mesoscale SuDS, similar to the setup presented in Table 2.

## 4. Results and discussion

### 4.1. Total flow volume

The ten rainfall events, monitored during the study, were simulated to estimate the total discharge volume using equations (1) and (2). Fig. 5 shows the observed discharge volume versus the results from the model. As seen, the model can reproduce the observed discharged volumes for the ten rainfall depths recorded during this study without a complex calibration process. In other words, only the physical description of the system, through equations (1) and (2), suffices to yield a satisfactory estimation of the total discharge volume. It is also shown that using DCIA as the sole contributor to the runoff can be an efficient alternative to simulate rainfalls of up to 2.5 mm/min with a maximum depth of about 30 mm.

### 4.2. Model calibration

Since not all of the registered rainfalls generated runoff/discharge, especially in the Southern SuDS, only three rainfall events with different volumes and durations (out of the total 10) were selected for calibrating the model for the Northern SuDS (rainfalls D, E, and G) and the Southern SuDS (rainfalls D, F, and G) as shown in Tables 2 and 3.

The *Nash-Sutcliffe Efficiency Index* (NSE) (Nash and Sutcliffe, 1970), was employed to objectively estimate the optimum $k$ and $m$ parameters for the nonlinear reservoir model, equation (4), and is presented in equation (5) below:

$$NSE = 1 - \frac{\sum_{i=1}^{N} (\hat{Q}_i - Q_i)^2}{\sum_{i=1}^{N} (Q_i - \bar{Q}_i)^2} \quad , \quad -\infty < NSE < 1.0 \tag{5}$$

where $\hat{Q}_i$ and $Q_i$ are the modelled and observed discharge values, respectively. $\bar{Q}_i$ is the mean of the observed values and $N$ is the number of data points in the sample (sample size).

The NSE has been widely employed in hydraulic and hydrological studies in order to assess the accuracy of the models (Brunetti et al., 2017; Palla and Gnecco, 2015; Rujner et al., 2018; Soulis et al., 2017). An NSE of 1 indicates perfect model fit with observations, whilst an NSE of 0 implies that the model is as good a predictor as the mean value of the observations (Jain and Sudheer, 2008).

The results of a preliminary investigation showed that the linear reservoir model showed as good agreement with observations as the nonlinear reservoir model (i.e. $m$-values for the Northern and Southern SuDS were about 1.05 and 1.10, respectively). Thus, the hydrographs behave according to a linear reservoir model. Consequently, to further simplify the model, a linear reservoir model was employed instead, i.e., $m = 1$. The model was then calibrated with respect to only the $k$ parameter, which furthermore reduces the search space of the calibration problem, thereby decreasing the difficulty in the parameter estimation. In order to avoid negative and zero discharges, $k$ is restricted to have positive/nonzero values during the calibration process. The final calibration procedure using the NSE was formulated as the

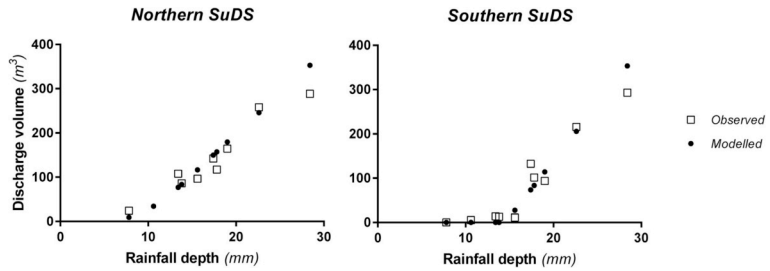



**Fig. 5.** Modelled and observed discharge volumes from the Northern and the Southern SuDS in Augustenborg for the studied rainfall events. Note that no flow measurement was available for Rainfall B (10.6 mm) in the Northern SuDS.

following optimization problem:

max *NSE*
w.r.t. *k*
s.t. *Equations* (1) – (4)
$k > 0$

The results of the calibration are presented in Table 3. With a constant value of 1.00 for *m*, the parameter *k* was calibrated to 0.017 $min^{-1}$ and 0.014 $min^{-1}$ for the Northern and Southern SuDS, respectively. The obtained maximized NSE values are also presented in Table 3. It is noted that the NSE values for the Northern SuDS are considerably larger than those of the Southern SuDS, implying that the selected calibration rainfall events deliver more reliable values for the calibration parameters in the former case. This can also be seen in the calibrated mean *k* where the corresponding standard deviation is smaller for the Northern SuDS ( ± *0.005*) than that of the Southern SuDS ( ± *0.009*). Generally, with respect to the achieved maximum NSE during the calibration process, most values are classified either *Satisfactory* (0.50 < NSE < 0.65), *Good* (0.65 < NSE < 0.75) or *Very good* (NSE > 0.75) based on a performance rating suggested by Moriasi et al. (2007).

### 4.3. Model validation

Fig. 6 shows examples of modelled versus observed hydrographs for the SuDS in Augustenborg. The model is able to satisfactorily predict the discharge rates from both the Northern and Southern SuDS for all events using similar parameter values, which indicate model reliability and robustness. NSE was also employed to evaluate the goodness of fit between measured and modelled values. NSE values, as shown in Fig. 6, are mostly high and close to 1.0 which is an indication for good agreement between observed and modelled discharge rates. The observed and modelled hydrographs for the Northern SuDS generally show a better agreement that those of the Southern SuDS. For Rainfall *J* in the Southern SuDS, NSE is found to be *-0.18*. This is the only relatively low value among the NSEs found in this study, which can be due to in-built model characteristics that does not consider the reduction in retention capacity due to antecedent rainfalls. As seen in Fig. 6f, the model underestimates the discharge rates during the first rainfall peak, while the discharges for the second peak in the rainfall are better estimated. This means that most probably, there has been an antecedent rainfall prior to the first peak, which has already filled up the SCMs. An already filled system, thereby, leads to higher discharge rates at the first peak in the rainfall, while the model assumes that the retention capacity of the system is vacant, generating smaller discharge rates. This effect decreases as the second rainfall peak strikes, leading to better modelled results.

Fig. 7 illustrates the performance of the model with regard to peak discharge for all rainfall-runoff events recorded in this study for the Northern and the Southern SuDS, respectively. The model produces comparable results for the Northern SuDS (Fig. 7a), whereas the simulations for the Southern SuDS show less good agreement with the observed data (Fig. 7b).

Fig. 8 shows how the modelled lag time deviates from the observed values after normalization with the rainfall duration. It is important to note that the accuracy of the estimated lag time shall be considered with regard to the duration of the rainfall. A longer rainfall duration implies that it is more likely to have a larger error between the modelled and the observed lag time. In order to compensate for the rainfall duration, the error between observed and modelled lag time is normalized with the rainfall duration. Results of the modelled lag times in the systems of Augustenborg also show that the Northern SuDS (Fig. 8a) is better simulated than the Southern SuDS (Fig. 8b). In addition to the error sources explained in section 2.2., the lower agreement between the observed and the modelled parameters in the Southern SuDS can partly be explained by the more complex discharge conditions and diverse types of SCMs that prevail in the Southern SuDS. Overall, the model is found to produce satisfactory results considering the simplifications made to achieve fast and easy simulations at low computational costs.

Such a fast and easily applicable model is essential for demanding simulations (*e.g.*, city-wide modelling, application of long-term rainfall time series, Monte-Carlo techniques), but will also play an important

**Table 3**
The results of the calibration for *m* and *k* values as well as their corresponding $NSE_{max}$ obtained in Northern and Southern SuDS in Augustenborg.

| Rainfall ID | Northern SuDS | | | Rainfall ID | Southern SuDS | | |
|---|---|---|---|---|---|---|---|
| | *k* | *m* | $NSE_{max}$ | | *k* | *m* | $NSE_{max}$ |
| **D** | 0.011 | 1.00 | 0.70 | **D** | 0.006 | 1.00 | 0.12 |
| **E** | 0.023 | 1.00 | 0.78 | **F** | 0.026 | 1.00 | 0.70 |
| **G** | 0.017 | 1.00 | 0.85 | **G** | 0.010 | 1.00 | 0.59 |
| **Mean (deviation)** | **0.017** ( ± *0.005*) | **1.00** ( ± *0.00*) | **0.78** ( ± *0.06*) | **Mean (deviation)** | **0.014** ( ± *0.009*) | **1.00** ( ± *0.00*) | **0.47** ( ± *0.25*) |

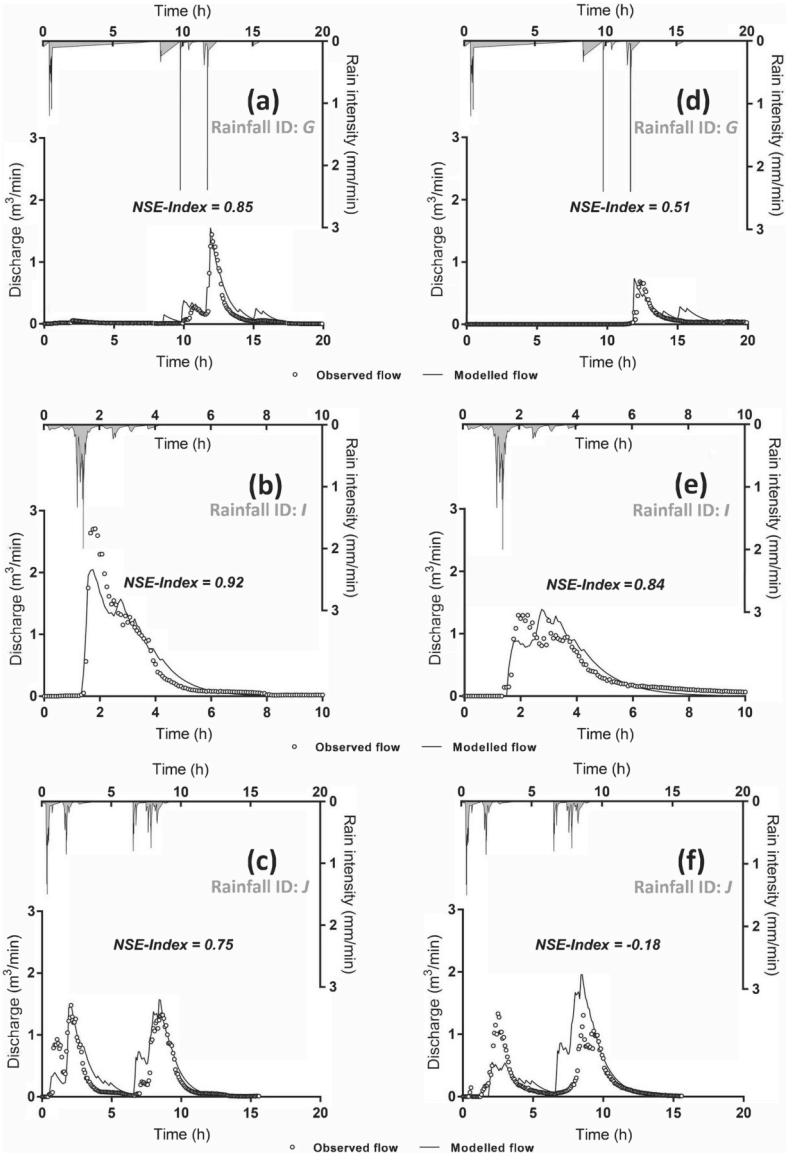



**Fig. 6.** Examples of the observed versus modelled discharges from the Northern (a, b, c: k = 0.017 $min^{-1}$; m = 1.00) and Southern SuDS (d, e, f: k = 0.014 $min^{-1}$, m = 1.00) in Augustenborg.

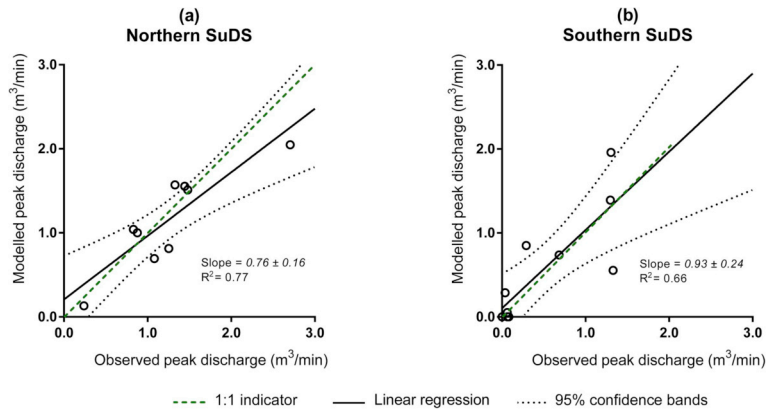



**Fig. 7.** Model performance with regard to peak discharge for (a) Northern SuDS (b) Southern SuDS in Augustenborg.

role in facilitating the communication between urban water engineers and urban planners who are supposed to design and implement SuDS through a mutual perspective and collaboration.

However, the current version of the model is designed and developed to simulate single rain events, since it does not take the relief in retention capacity through infiltration and evapotranspiration into account. Moreover, initial soil moisture and evapotranspiration processes for the permeable surfaces of the catchment are also excluded for simplicity as the model employs only DCIA as the contributing surface to runoff. It is possible, that in the future developments, such functions could be added to the description of the SCMs through employment of physically-based infiltration equations, e.g., nonlinear/linear reservoir models.

In order to extend the applicability of the model to nonexistent systems/unmonitored catchments with different configurations and outlet setups, it may be possible to relate the calibration parameters of the nonlinear reservoir model, i.e., $k$ and $m$, to some physical characteristics of the system. For instance, the model parameters $k$ and $m$ could tentatively be estimated if the nonlinear reservoir model is interpreted as a weir equation (or bottom outlet equation) governing the discharge. The fact that the weir discharge from SuDS, as demonstrated for Augustenborg, in principle follows a linear reservoir dynamics (i.e., $m = 1$) makes the described process more convenient (only $k$ needs to be estimated). In addition, the fact that the calibrated values of $k$ for both systems (0.017 and 0.014 $min^{-1}$ for Northern and Southern SuDS, respectively) are very close and have similar order of magnitude can be interpreted as an indication for the physical similarities in outlet setups of the systems. In this way, the number of coefficients to be estimated can be reduced, substituted with recommended values instead, while yielding a more robust and easy-to-use model.

### 4.4. Sensitivity analysis

To gain insights to how the different parts of the model affect its response, sensitivity analyses were performed for both the Northern and the Southern systems. These were carried out using the One-Factor-at-a-Time (OFAT) method, which entails perturbing one parameter (ceteris paribus) and studying the effect on the model output (Nolin et al., 2018). The OFAT method is simple to implement, and is helpful in understanding the dependency of the model on the investigated parameters (Delgarm et al., 2018). In this case, SCM properties (such as retention, SCM area, DCIA) as well as the nonlinear reservoir model parameters ($k$ and $m$) were individually perturbed at three different levels, i.e., ± 10%, ± 25% and ± 50%, representing reasonable,

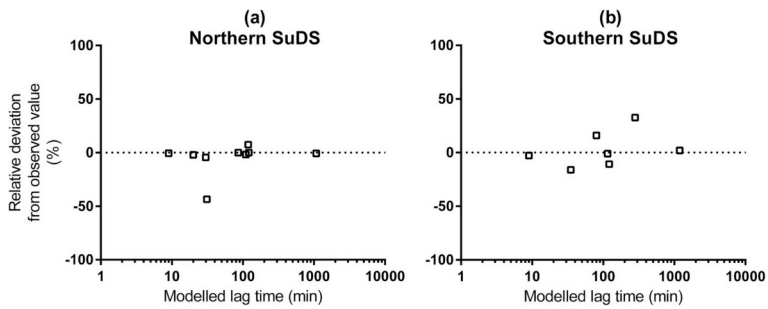



**Fig. 8.** Relative deviation of modelled lag times from observed values, normalized against the total duration of the rainfall.

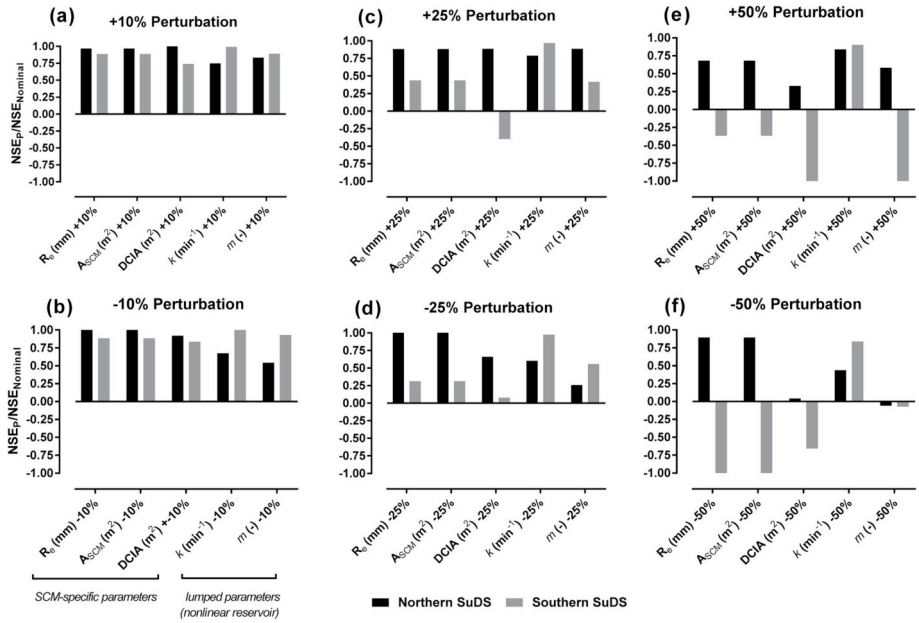



**Fig. 9.** Sensitivity analysis based on OFAT method showing the obtained NSE for the Northern and Southern SuDS in case of ± 50% alteration in the model parameters.

moderate, and extreme perturbation scenarios, respectively. Note that no recalibration of the model was performed during this investigation, and the $k$ values of 0.017 $min^{-1}$ and 0.014 $min^{-1}$ for the Northern and Southern SuDS, respectively, were used when not perturbed themselves. The results from the OFAT sensitivity studies for the Northern and Southern SuDS when perturbing the aforementioned parameters are presented in Fig. 9, in which the relative variation of the achieved NSE after perturbation ($NSE_P$) against the calibrated/nominal NSE ($NSE_{Nominal}$ = 0.92 [Northern SuDS]; 0.84 [Southern SuDS]) is used as an indicator for model sensitivity.

The results show impact from all perturbations, making the NSE lower than or equal to the nominal case; this is expected since the nominal case is calibrated to maximize the NSE. However, ± 10% perturbations (Fig. 9a and b) for all parameters result in acceptable NSE values, retaining 75%–100% of the nominal NSE for +10% perturbation for both Northern and Southern SuDS, whereas in case of −10% perturbation, model performance slightly deteriorates for $k$ and $m$ for the Northern SuDS. In general, regarding the lumped parameters, the model is found to be more sensitive to perturbations of $m$ compared to those of $k$, but this has a negligible significance since the model behavior is rather linear, which means $m$ is in general equal to 1.0.

In case of ± 25% perturbations, the model performance deteriorates considerably for the Southern SuDS, while the Northern SuDS tends to retain acceptable results. It is also important to note that the SCM-specific parameters ($R_e$, $A_{SCM}$ and DCIA) are relatively easily quantifiable; thus, ± 10% perturbations is assessed to be more realistic for them than ± 25% and ± 50% perturbations.

There is a clear difference in the effect of the perturbations on the two systems, where the response of the Southern model is impacted to a higher degree for all perturbations except for those on $k$. Furthermore, varying $k$ by ± 50% has a relatively small effect for the Southern SuDS, whereas it has a similar effect as the other parameters for the Northern SuDS. Combined, the low effect of $k$ and the difference in effects between the two systems indicate that the physically based model is robust and captures system behavior quite well, since the lumped parameters do not change the solution substantially. It also highlights the importance of proper quantification of the SCM-specific parameters – tolerating an error of about ± 10% – as these can have a large influence on the model output.

## 5. Concluding remarks

The event-based model developed in this study fulfills the overall objective, which was to predict the discharge pattern (hydrograph) from mesoscale SuDS at low computational costs and with acceptable accuracy. The complete hydrograph from a mesoscale SuDS for a 10-h rainfall event is generated in a matter of seconds. This model property facilitates large-scale/city-wide optimization studies that are essential for long-term, sustainable performance of urban drainage networks. Such city-wide optimization using the developed rapid simulation tool could be regarded as a preliminary screening stage after which more complex models can be employed for further investigation at selected sites and locations for prospective SuDS retrofits. Presently, the model uses the nonlinear reservoir equation only to characterize the last compartment in the SuDS. A more detailed and representative approach would be to apply the nonlinear reservoir model to the discharges from all the upstream SCMs as well, *i.e.*, $q^{1,t}$, $q^{2,t}$, …, $q^{n-1,t}$. It might also be desirable to introduce infiltration and evapotranspiration rates – as sink

functions – to the components (SCMs) to facilitate long-term continuous rainfall-runoff simulations. However, for validation this would require more extensive data collection at many discharge points from the SCMs. Furthermore, it means that the *k* and *m* constants have to be quantified for each compartment (SCM), which will make the model a more complicated tool to use. On the other hand, depending on the type of SCM, analytical or empirical expression may be developed for *k* and *m*.

## Declarations of interest

None

## Acknowledgements

This work was financially supported by J. Gustaf Richert Foundation at SWECO (grant number 2015-00181), Sweden's Innovation Agency (VINNOVA) via *Future City Flow* project (grant number 2017-01046) at Sweden Water Research as well as the Swedish Water and Wastewater Association (*Svenskt Vatten*) via *VA-teknik Södra*. The authors also acknowledge the personnel at the regional water and wastewater utility company (VA SYD) for their support with respect to measuring instrument and data collection. *Tomas Wolf* and *John Hägg* at VA SYD are especially thanked for their contribution to setup and maintenance of the instruments. Our colleagues *Miguel Sanchis Sebastia* and *Niklas Andersson* at the Department of Chemical Engineering, Lund University are also acknowledged for their input and advice during the course of code-development.

## Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.jenvman.2019.03.037.

## References

Adamowski, J., Prasher, S.O., 2012. Comparison of machine learning methods for runoff forecasting in mountainous watersheds with limited data. J. Water Land Dev. 17. https://doi.org/10.2478/v10025-012-0038-4.

Brunetti, G., Šimů Nek, J., Turco, M., Piro, P., 2017. On the Use of Surrogate-Based Modeling for the Numerical Analysis of Low Impact Development Techniques. https://doi.org/10.1016/j.jhydrol.2017.03.013.

Delgarm, N., Sajadi, B., Azarbad, K., Delgarm, S., 2018. Sensitivity analysis of building energy performance: a simulation-based approach using OFAT and variance-based sensitivity analysis methods. J. Build. Eng. 15, 181–193. https://doi.org/10.1016/J.JOBE.2017.11.020.

Elliott, A.H., Trowsdale, S.A., Wadhwa, S., 2009. Effect of aggregation of on-site stormwater control devices in an urban catchment model. J. Hydrol. Eng. 14, 975–983. https://doi.org/10.1061/(ASCE)HE.1943-5584.0000064.

Fiener, P., Auerswald, K., 2009. Spatial variability of rainfall on a sub-kilometre scale. Earth Surf. Process. Landforms 34, 848–859. https://doi.org/10.1002/esp.1779.

Freni, G., La Loggia, G., Notaro, V., 2010. Uncertainty in urban flood damage assessment due to urban drainage modelling and depth-damage curve estimation. Water Sci. Technol. 61, 2979–2993. https://doi.org/10.2166/wst.2010.177.

García-Serrana, M., Gulliver, J.S., Nieber, J.L., 2017. Non-uniform overland flow-infiltration model for roadside swales. J. Hydrol. 552, 586–599. https://doi.org/10.1016/J.JHYDROL.2017.07.014.

Gericke, O.J., Smithers, J.C., 2018. An improved and consistent approach to estimate catchment response time parameters: case study in the C5 drainage region, South Africa. J. Flood Risk Manag. 11, S284–S301. https://doi.org/10.1111/jfr3.12206.

Gericke, O.J., Smithers, J.C., 2014. Review of methods used to estimate catchment response time for the purpose of peak discharge estimation. Hydrol. Sci. J. 59, 1935–1971. https://doi.org/10.1080/02626667.2013.866712.

Géron, A., 2017. Hands-On Machine Learning with Scikit-Learn and TensorFlow - Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.

Golden, H.E., Hoghooghi, N., 2018. Green infrastructure and its catchment-scale effects: an emerging science. Wiley Interdiscip. Rev. Water 5, e1254. https://doi.org/10.1002/wat2.1254.

Grayson, R., Holden, J., Rose, R., 2010. Long-term change in storm hydrographs in response to peatland vegetation change. J. Hydrol. 389, 336–343. https://doi.org/10.1016/j.jhydrol.2010.06.012.

Haghighatafshar, S., la Cour Jansen, J., Aspegren, H., Jönsson, K., 2018a. Conceptualization and schematization of mesoscale sustainable drainage systems: a full-scale study. Water 10, 1041. https://doi.org/10.3390/w10081041.

Haghighatafshar, S., Nordlöf, B., Roldin, M., Gustafsson, L.-G., la Cour Jansen, J., Jönsson, K., 2018b. Efficiency of blue-green stormwater retrofits for flood mitigation – conclusions drawn from a case study in Malmö, Sweden. J. Environ. Manag. 207, 60–69. https://doi.org/10.1016/j.jenvman.2017.11.018.

Halevy, A., Norvig, P., Pereira, F., 2009. The unreasonable effectiveness of data. IEEE Intell. Syst. 24, 8–12. https://doi.org/10.1109/MIS.2009.36.

Hu, C., Wu, Q., Li, H., Jian, S., Li, N., Lou, Z., Hu, C., Wu, Q., Li, H., Jian, S., Li, N., Lou, Z., 2018. Deep learning with a long short-term memory networks approach for rainfall-runoff simulation. Water 10, 1543. https://doi.org/10.3390/w10111543.

Jain, S.K., Sudheer, K.P., 2008. Fitting of hydrologic models: a close look at the nash–sutcliffe Index. J. Hydrol. Eng. 13, 981–986. https://doi.org/10.1061/ASCE1084-0699200813:10981.

Jato-Espino, D., Charlesworth, S., Bayon, J., Warwick, F., 2016. Rainfall–runoff simulations to assess the potential of SuDS for mitigating flooding in highly urbanized catchments. Int. J. Environ. Res. Public Health 13, 149. https://doi.org/10.3390/ijerph13010149.

Jayasooriya, V.M., Ng, A.W.M., 2014. Tools for modeling of stormwater management and economics of green infrastructure practices: a review. Water, Air, Soil Pollut. 225, 2055. https://doi.org/10.1007/s11270-014-2055-1.

Krebs, G., Kokkonen, T., Valtanen, M., Setälä, H., Koivusalo, H., 2014. Spatial resolution considerations for urban hydrological modelling. J. Hydrol. 512, 482–497. https://doi.org/10.1016/j.jhydrol.2014.03.013.

Locatelli, L., Mark, O., Mikkelsen, P.S., Arnbjerg-Nielsen, K., Bergen Jensen, M., Binning, P.J., 2014. Modelling of green roof hydrological performance for urban drainage applications. J. Hydrol. 519, 3237–3248. https://doi.org/10.1016/j.jhydrol.2014.10.030.

Loperfido, J.V., Noe, G.B., Jarnagin, S.T., Hogan, D.M., 2014. Effects of distributed and centralized stormwater best management practices and land cover on urban stream hydrology at the catchment scale. J. Hydrol. 519, 2584–2595. https://doi.org/10.1016/j.jhydrol.2014.07.007.

Mansell, M.G., 2003. Rural and Urban Hydrology. Thomas Telford Publishinghttps://doi.org/10.1680/rauh.32309.

Moriasi, D.N., Arnold, J.G., Liew, M.W. Van, Bingner, R.L., Harmel, R.D., Veith, T.L., 2007. Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. Trans. ASABE 50, 885–900.

Mosavi, A., Ozturk, P., Chau, K., 2018. Flood prediction using machine learning models: literature review. Water 10, 1536. https://doi.org/10.3390/w10111536.

Nash, J.E., Sutcliffe, J.V., 1970. River flow forecasting through conceptual models part I — a discussion of principles. J. Hydrol. 10, 282–290.

Nolin, M., Andersson, N., Nilsson, B., Max, M., Pajalic, O., 2018. Analysis of an oscillating two-stage evaporator system through modelling and Simulation : an industrial case study. Chem. Eng. Trans. 69.

Nordlöf, B., 2016. 1D/2D Modeling of the Open Stormwater System of Augustenborg Using MIKE FLOOD by DHI. Project report available at: Water and Environmental Engineering at the Department of Chemical Engineering. Lund University, Lund, Sweden.

Palla, A., Gnecco, I., 2015. Hydrologic modeling of Low Impact Development systems at the urban catchment scale. J. Hydrol. 528, 361–368. https://doi.org/10.1016/j.jhydrol.2015.06.050.

Roldin, M., Locatelli, L., Mark, O., Mikkelsen, P.S., Binning, P.J., 2013. A simplified model of soakaway infiltration interaction with a shallow groundwater table. J. Hydrol. 497. https://doi.org/10.1016/j.jhydrol.2013.06.005.

Rujner, H., Leonhardt, G., Marsalek, J., Viklander, M., 2018. High-resolution modelling of the grass swale response to runoff inflows with Mike SHE. J. Hydrol. 562, 411–422. https://doi.org/10.1016/J.JHYDROL.2018.05.024.

Schulz, E.F., Lopez, G., 1974. Determination of Urban Watershed Response Time.

Singh, V.P., 1997. Effect of spatial and temporal variability in rainfall and watershed characteristics on stream flow hydrograph. Hydrol. Process. 11, 1649–1669. https://doi.org/10.1002/(SICI)1099-1085(19971015)11:12<1649::AID-HYP495>3.0.CO;2-1.

Soulis, K.X., Valiantzas, J.D., Ntoulas, N., Kargas, G., Nektarios, P.A., 2017. Simulation of Green Roof Runoff under Different Substrate Depths and Vegetation Covers by Coupling a Simple Conceptual and a Physically Based Hydrological Model. https://doi.org/10.1016/j.jenvman.2017.06.012.

Zhang, X., Shen, L., Tam, V.W.Y., Lee, W.W.Y., 2012. Barriers to implement extensive green roof systems: a Hong Kong study. Renew. Sustain. Energy Rev. 16, 314–319. https://doi.org/10.1016/J.RSER.2011.07.157.

Zoppou, C., 2001. Review of urban storm water models. Environ. Model. Softw 16, 195–231. https://doi.org/10.1016/S1364-8152(00)00084-0.

# Paper IX

Journal of Hydrology 578 (2019) 124125

Contents lists available at ScienceDirect

Journal of Hydrology

journal homepage: www.elsevier.com/locate/jhydrol

Research papers

# Hydroeconomic optimization of mesoscale blue-green stormwater systems at the city level

Salar Haghighatafshar[a,*], Mikael Yamanee-Nolin[b], Anders Klinting[c], Maria Roldin[d], Lars-Göran Gustafsson[d], Henrik Aspegren[a,e], Karin Jönsson[a]

[a] Water and Environmental Engineering, Department of Chemical Engineering, Lund University, P.O. Box 124, SE-22100 Lund, Sweden
[b] Department of Chemical Engineering, Lund University, P.O. Box 124, SE-22100 Lund, Sweden
[c] DHI Denmark (Head Office), DHI A/S – DHI Water Environment Health, Agern Allé 5, DK-2970 Hørsholm, Denmark
[d] DHI Sweden, Södra Tullgatan 3, SE-21140 Malmö, Sweden
[e] Sweden Water Research AB, Ideon Science Park, Scheelevägen 15, SE-22370 Lund, Sweden

ARTICLE INFO

This manuscript was handled by G. Syme, Editor-in-Chief

*Keywords:*
Blue-green stormwater
Mesoscale
Macroscale
Urban drainage
Urban infrastructure
Optimization
Hybrid modeling
Cosimulation

ABSTRACT

The development of tools to help cities and water utility authorities communicate and plan for long-term sustainable solutions is of utmost importance in the era of a changing and uncertain climate. This study introduces a hybrid modeling concept for the cosimulation of mesoscale blue-green stormwater systems and conventional urban sewer networks. The hybrid model successfully introduces the retention/detention effects of mesoscale blue-green stormwater systems to the hydraulic dynamics of the sewer network. The cosimulation package was further facilitated with a cost-oriented multiobjective optimization algorithm. The aim of the scalar multi-objective optimization was to minimize the total cost comprising both *flooding costs* and *action costs* – both parameters solely representing the financial components of cost – through optimal placement of mesoscale blue-green systems of optimal size. The suggested methodology provides a useful platform for sustainable management of the existing sewer networks in cities from a hydroeconomic perspective.

## 1. Introduction

The limitation in drainage capacity and its consequent outcomes have been discussed among urban engineers since the late 19th century (Kuichling, 1889; Lloyd-Davies, 1906). The focus on urban flooding as a serious challenge has been intensified due to the observed increase in the frequency of rainfall events/pluvial floods as a result of climate change. Cities have struggled with the mitigation of urban flooding ever since and have recently become interested in the notion of blue-green stormwater solutions, also known as stormwater control measures (SCM). However, there is still no consensus in the scientific community or among the city authorities on how, where and to what extent these solutions shall be implemented.

Many large and old cities have inherited a larger proportion of their infrastructure from the far past. In a changing climate, it is highly desirable to sustain the functionality of the existing infrastructure and to maintain resilience in the case of natural catastrophes. This is especially desirable from an economic point of view, as the intensity and frequency of extreme rain events are increasing. This can be done by introducing flexibility to urban infrastructure, avoiding a technical lock-in to solely pipe-based drainage systems. In addition, further urbanization in terms of altered land use and further densification leads to increased impermeable surfaces compared to the situation for which the drainage system was designed. This means that even under a constant climate scenario, elevated flooding will still be a serious challenge considering the current urban planning and engineering practices (Berndtsson et al., 2019). Cities are trying to enhance the capacity of the drainage system by increasing the safety margins in the design criteria as well as by replacing the existing pipes with larger culverts. This is done, for instance, by introducing climate factors to design criteria (Arnbjerg-Nielsen, 2012; SWWA, 2016; Watt et al., 2003), updating intensity-durationfrequency (IDF) curves (Guo, 2006; Hailegeorgis et al., 2013; Lima et al., 2018; Mailhot et al., 2007) and construction of large stormwater tunnels in cities (Dolowitz et al., 2018; VA SYD, 2019). Such measures might be effective to some extent, but in the era of a nonstationary climate – as demonstrated and argued by Milly et al. (2008), Vogel et al. (2011), and Liu et al. (2017b) – manipulation of drainage capacity might not be an optimal solution. A parallel solution would be to manipulate the contributing catchments – by introducing retention/detention capacities, e.g., via blue-green

* Corresponding author.
*E-mail address:* Salar.Haghighatafshar@chemeng.lth.se (S. Haghighatafshar).

https://doi.org/10.1016/j.jhydrol.2019.124125
Received 23 July 2019; Received in revised form 29 August 2019; Accepted 6 September 2019
Available online 09 September 2019

stormwater systems – to modify the volume and the flow of the generated runoff (Azzout et al., 1995; Fletcher et al., 2015; Stahre, 2006, 1993).

Blue-green stormwater systems aim to mimic a naturally-oriented water cycle as well as introducing amenity by juxtaposing water and greenery in urban environments (Everett et al., 2015). These systems combine natural hydrological and ecological values and are shown to have considerable contributions to flood mitigation (BlueGreenCities, 2019). This is done by slowing down runoff and by improving infiltration, evapotranspiration, detention, and surface storage. Green roofs, wet and dry ponds, swales, biofilters (raingardens), and infiltration basins are all individual solutions (SCMs) within the context of blue-green stormwater management. In this paper, however, a *blue-green system* is defined as an interconnected group of blue-green stormwater solutions or SCMs, which can be implemented on different scales depending on the effect they are expected to deliver (Demuzere et al., 2014). The presented definition for the blue-green stormwater systems – as used within the context of this paper – is also regarded as sustainable drainage systems (SuDS) (Fletcher et al., 2015). According to a classification by Haghighatafshar et al. (2018b), blue-green stormwater systems can be implemented at:

- *microscale*, where single and discrete blue-green stormwater solutions or SCMs are implemented locally and the discharge from each SCM is directly connected to the urban drainage network;
- *mesoscale*, where the blue-green stormwater system is implemented as a group of tree-structured SCMs, in which the discharge from one (or more) SCM(s) flows into the next immediate SCM lying downstream and eventually to the recipient/urban drainage network. This configuration of SCMs is frequently referred to as "SuDS management train" (Kirby, 2005). A detailed conceptualization of mesoscale systems is presented and discussed by Haghighatafshar et al. (2018a) as a definition for blue-green stormwater systems;
- *macroscale*, where blue-green stormwater systems are upscaled to the city level and in a hydraulic context, might have considerable effects on the functionality of the existing urban drainage network.

Whilst acknowledging all legal and institutional obstacles (Berndtsson et al., 2019; Wihlborg et al., 2019), one of the possible scenarios with respect to the realization of the macroscale blue-green stormwater systems is to upscale the mesoscale blue-green stormwater system, i.e., to replicate the mesoscale blue-green stormwater system in multiple locations of the city (Haghighatafshar et al., 2018a). The replication of mesoscale blue-green stormwater systems in multiple locations in the city could result in the concept of *sponge cities* – a concept and practice developed in China (Liu et al., 2017a; Ren et al., 2017; Zhang et al., 2018) – or a macroscale blue-green stormwater system (Haghighatafshar et al., 2018a) in which city surfaces are designed to contain enough infiltration, retention or detention volume to overcome the consequences of flooding along with stormwater management, rainwater harvesting and stormwater quality improvement (Zhang et al., 2018). The collection of these services in the cities could potentially help survival of the highly contrasted wet and dry seasons (Trenberth et al., 2014) that lead to floods and droughts, respectively. The literature available concerning the macroscale implementation of blue-green stormwater systems within the context of sponge cities is increasing, and different aspects of the challenge are addressed (Fenner and Richard, 2017; Li et al., 2019; Ren et al., 2017; Zhang et al., 2018). For instance, land availability, land use, population density, topology and geological characteristics have been addressed for the implementation of stormwater control measures by investigating the urban morphology (Bach et al., 2013; Romnée et al., 2015). There are also studies that have looked into biophysical factors as well as the sociodemographic status of urban districts for the likely implementation of blue-green stormwater systems (Kuller et al., 2016). Zischg et al. (2018) and Cunha et al. (2016) have studied how microscale SCMs (single blue-green stormwater measures) and underground storage units along the pipes affect the performance of the local pipe network under design storm scenarios, and have consequently recommended a placement strategy. A similar study was carried out by Wang et al. (2017), who have optimized placement of storage tanks using a two-stage approach for flood mitigation. Zischg et al. (2019) take a step further and develop a methodology for assessing the influence of different sociotechnical pathways on the future transitions of urban drainage systems. There are also studies through which microscale implementations of blue-green measures in smaller urban districts are optimized (Eckart et al., 2018; Huang et al., 2018; Leimgruber et al., 2019). Furthermore, investigations regarding the macroscale hydro-economic optimization of multiple mesoscale blue-green stormwater systems in interaction with the existing and mostly underground urban drainage infrastructure have recently been drawing attention in the scientific community (e.g., Bakhshipour et al., 2019; Glenis et al., 2018; Zhou et al., 2018). Sustaining the functionality of the existing drainage network, especially the combined sewer network, which is associated with basement flooding and higher health risks, is a crucial subject in cities worldwide.

Such upscaling of mesoscale blue-green stormwater systems has hypothetically the potential to prevent the hydraulic overloading of the drainage network. However, economic consequences should also be taken into consideration since economy – along with ecology and society – is one of the fundamental pillars of sustainability (Wilkins, 2008). This means that both the quantity (retention volume) and location of the mesoscale blue-green stormwater systems (i.e., distribution of retention volumes) with respect to the existing drainage network must be investigated and optimized with regard to costs and benefits. Costs and benefits may, in a broad context, include a range of different socio-ecological factors in addition to monetary/financial aspects. However, in order to delimit the study boundaries in this paper, we focus on strictly financial costs related to the implementation of blue-green systems and flood hydraulics.

Therefore, the aim of this study is to develop and introduce a methodology to systematically locate mesoscale blue-green stormwater systems throughout the city to achieve the lowest possible flooding-damage related costs through a cost-effective implementation scenario. Having the methodology development in focus at a large-scale view, a generic transfer of mesoscale blue-green systems is applied in order to facilitate the theoretical siting and sizing of these systems throughout cities. It should be noted that practical implementation with regard to detailed design and choice of SCMs according to local circumstances in each catchment – e.g. land availability, land ownership, and physical characteristics of the area – is left for complementing studies.

## 2. Software architecture

To perform the optimization study, cosimulation of physical rainfall-runoff processes encompassing three different domains was structured. These domains were:

$D_1$ Hydrological simulation of conventional catchments
$D_2$ Hydrological and hydraulic simulation of blue-green catchments
$D_3$ Hydrodynamic simulation of the sewer system loaded by the results from domains $D_1$ and $D_2$.

To manage the cosimulation of these three domains, a hybrid model consisting of two modules, *hydrological* and *hydraulic*, was constructed. Domains $D_1$ and $D_2$ are included in the *hydrological module,* whereas domain $D_3$ is simulated under the *hydraulic module*.

The model concerning the simulation of the mesoscale blue-green system (Domain $D_2$) is the model developed and presented in detail by Haghighatafshar et al. (2019), while the rainfall-runoff engine of MIKE Urban by DHI was employed for the hydrological simulation of conventional catchments (Domain $D_1$). The mesoscale blue-green model

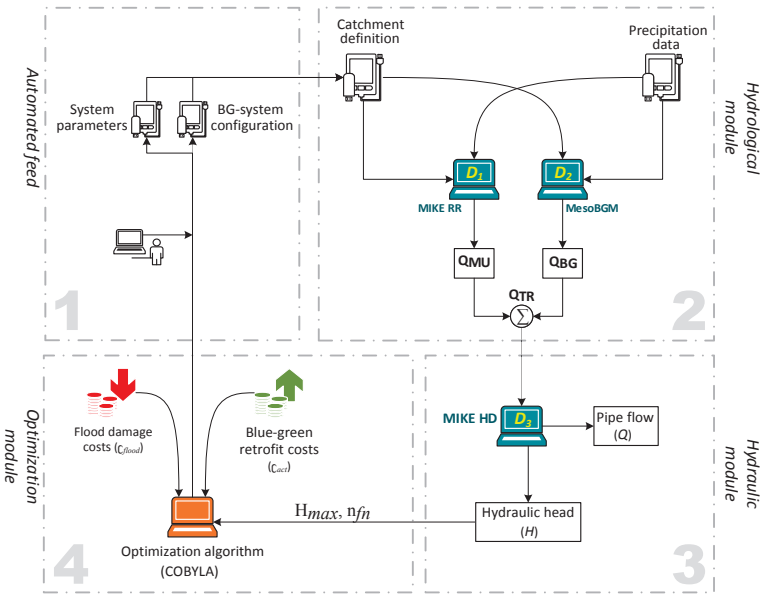



**Fig. 1.** The modeling and optimization logics (encompassing four major modules) were implemented in MIKE Operations to couple MIKE Urban and MesoBGM.

(*MesoBGM*) – introduced by Haghighatafshar et al. (2019) – simulates the hydrograph of the discharge from a mesoscale blue-green stormwater system. This discharge is then accumulated with MIKE Urban's rainfall-runoff (MIKE RR) output and is introduced as the input to the 1D MIKE Urban hydrodynamic simulation model (MIKE HD). MIKE Operations by DHI were employed as the cosimulation platform to facilitate communication between the three models, i.e., *MesoBGM*, MIKE RR, and MIKE HD. The suggested algorithm implemented in MIKE Operations is presented in Fig. 1, encompassing four major modules:

*User-defined/automated feed* where the parameter sets for the definition of catchments as well as the system parameters are managed.
*Hydrological module* for calculation of the hydrological load ($D_1$ and $D_2$),
*Hydraulic module* for assessing the performance of the drainage network ($D_3$),
*Optimization module* for performing the optimization based on the cost-benefit assessment,

The characteristics of the urban catchment in the joint model are defined in the user-defined/automated feed block, which is in turn fed into the hydrological module. The estimation of model parameters for $D_1$ is performed according to the standard procedure recommended by the Swedish Water and Wastewater Association (SWWA, 2016), whereas the characteristics of the mesoscale blue-green stormwater system ($D_2$) are defined according to Haghighatafshar et al. (2019). In addition to network setup, runoff coefficients, time-area curves, retention volumes, etc., the introduced feed also consists of the blue-green retrofit area ($A_{BG,i}$) in the $i^{th}$ subcatchment with a total area of $A_{SC,i}$. Based on the introduced feed, $D_1$ and $D_2$ generate hydrological load schemes, denoted as $Q_{MU,i}$ and $Q_{BG,i}$ (time series), respectively, in which the contributing area to $Q_{MU,i}$ is $A_{SC,i} - A_{BG,i}$.

In the next stage, the overall hydrologic load (total runoff, $Q_{TR}$) – as a cumulated time series – is calculated according to Eq. (1).

$$Q_{TR} = \sum_{i=1}^{n} (Q_{MU,i} + Q_{BG,i}) \tag{1}$$

This means that the discharge from the mesoscale blue-green system is still connected to the sewer network. The cumulative total flow, $Q_{TR}$, is then introduced as the network load to the MIKE HD model ($D_3$), through which 1D Saint-Venant's mass and momentum equations are applied for the computation of flow ($Q$) and piezometric pressure (hydraulic head, $H$) in the pipes (DHI, 2017). The ground level – in the context of a separate sewer network – and the basement level – in the context of a combined sewer network – for each manhole can be used to define a critical hydraulic head, , at which exceedance can be interpreted as flooding. It should also be noted that the 0D/1D nature of the hybrid model made it appropriate for performing numerous large-scale simulations in terms of simulation time and computational costs.

### 2.1. The optimization module

As shown in Fig. 1, optimization is performed with regard to flood economics. The basic idea is that the investment in flood mitigation measures would lead to lower flood damage costs, hence these two parameters, i.e., investment in measures and maintenance (cost of action, $\mathbb{C}_{act}$) and flood damage cost ($\mathbb{C}_{flood}$) can be incorporated into a scalar cost function. However, it is necessary that these costs are quantified according to the specific perspective of the stakeholder who performs the optimization, e.g., insurance companies, water utility companies, municipalities, or government. Note that additional socio-ecological


**Fig. 2.** The catchment area, encompassing 954.35 ha, connected to the combined sewer network in Malmö, Sweden, drained through the Turbinen pump station. Background picture: GSD-Orthophoto, courtesy of The Swedish Mapping, Cadastral and Land Registration Authority, ©*Lantmäteriet*.

benefits associated with blue-green stormwater systems are not included in this study. Incorporation of monetized and quantified added socio-ecological values would presumably have strong impacts on optimization results.

The annual cost of action and flood damage cost are calculated according to Eqs. (2) and (3), respectively as follows:

$$C_{act} = f\left(\sum A_{BG}(u), c_{BG}, i, L\right) = \sum A_{BG}\left(c_{BG}\cdot\left[\frac{i(1+i)^L}{(1+i)^L-1}\right] + \frac{M}{L}\cdot\sum_{l=0}^{L-1}(1+\alpha)^l\right) \quad (2)$$

where $\sum A_{BG(u)}$ is the total area of the retrofitted blue-green stormwater system (ha) according to scenario $u$, $c_{BG}$ is the average cost of blue-green stormwater retrofits per hectare (SEK/ha), $L$ is the technical lifespan of the constructed system (50 years, which is the local standard assumption in infrastructure investments), $M$ is the annual maintenance cost, $\alpha$ is the average annual pay raise (~2%), and $i$ is the average interest rate (~3%). A similar approach for computing the annual cost of action was also adopted by Huang et al. (2018).

$$C_{flood} = f\left(n_{fn}(u), c_{fn}, i, T\right) = (c_{fn}\cdot n_{fn}(u))\left[\frac{i}{(1+i)^T-1}\right] \quad (3)$$

where $n_{fn}(u)$ is the number of flooded manholes in scenario $u$, $c_{fn}$ is the average damage cost per flooded manhole (SEK/manhole), $T$ is the recurrence interval of the optimization storm (years), and $i$ is the average interest rate (~3%). Eq. (2) represents the uniform annual worth of the investment in year 0 through the lifespan of the blue-green stormwater system by using a capital recovery factor, and Eq. (3) represents the uniform annual cost of a total future fund (i.e., future cost of flood damage) during $T$ years (assuming that the flood recurrence follows the recurrence period of the rainfall event) using a sinking fund factor (Blank and Tarquin, 2012). Subsequently, a scalar total cost function can be introduced (Eq. (4)). This total cost function (Eq. (4)) is later incorporated in the scalar multiobjective to determine a scenario matrix (u) resulting in the minimum possible cost.

$$\Phi(u) = C_{act}(u) + C_{flood}(u) \quad (4)$$

where $u$ is the scenario matrix representing the implementation extent of the blue-green stormwater system in different sub-catchments. For instance, in a drainage model with $k$ subcatchments, $u$ can be specified according to Eq. (5).

$$u = [\mathscr{F}_{BG,1}, \mathscr{F}_{BG,2}, ..., \mathscr{F}_{BG,k}] \quad (5)$$

where $\mathscr{F}_{BG,i}$ is the implementation extent, i.e., the ratio of the blue-green retrofitted area in subcatchment $i$ ($A_{BG,i}$) to the total subcatchment area ($A_{SC,i}$), defined according to Eq. (6).

$$\mathscr{F}_{BG,i} = \frac{A_{BG,i}}{A_{SC,i}} \in [0, 1] \quad (6)$$

## 3. Case study

The general methodology adopted in this study is to employ hydrodynamic modeling to systematically optimize the distribution of blue-green stormwater systems in an urban catchment drained through a combined sewer network. This case study presents an example of how the developed methodology can be employed to enhance and understand the functionality of urban drainage systems.

### 3.1. Study area: Malmö, Sweden

Malmö is the 3rd largest city in Sweden located in the southern part of the country and is populated by over 330,000 people (Malmö stad, 2018). The drainage network in the city is mainly dominated by a combined sewer system in densely built central areas, while in the suburbs, mainly separate sewer networks are functional. This study focuses specifically on the combined sewer network and its corresponding catchment. The reason is that the city has been struck by extreme rainfall events several times and has undergone a relatively large burden of expenses due to basement flooding in the combined sewer network catchments (Haghighatafshar et al., 2014; Sörensen and

Mobini, 2017).

For this study, part of the combined sewer network and its corresponding catchment, which is drained into the *Turbinen* pump station in Malmö, was selected as shown in Fig. 2. The pump station receives combined sewer flows from two separate catchments, namely, the West catchment and the East catchment, drained from the south and southeast, respectively, toward the pump station. More information about the distribution of the combined and separate sewer networks in Malmö can be found in Haghighatafshar et al. (2018b) and Sörensen and Mobini (2017).

The fundamental approach in this study is to replicate the mesoscale blue-green system in Augustenborg – as the reference mesoscale retrofit described in Haghighatafshar et al. (2019, 2018a) – to achieve an optimized upscaling scenario. Augustenborg comprises two major mesoscale blue-green stormwater systems: one each in the south (Southern system, implemented 1999–2001), with a drainage area of roughly 9.6 ha, and north (Northern system, implemented 2002–2003), with a drainage area of approximately 6.3 ha. In addition, a separate local stormwater pipe system was constructed in 2003, covering a drainage area of about 3.5 ha. These three subsystems of Augustenborg are illustrated in Fig. 3. For more information about the configuration of the mesoscale blue-green stormwater systems in Augustenborg refer to Haghighatafshar et al. (2019, 2018a). In this study, it is assumed that the Southern system in Augustenborg, accounting for a total drainage area of 9.6 ha (= $A_{Aug}$) can be replicated as a scaled retrofit depending on the area of the target catchment. In other words, the configuration in Augustenborg with regard to the drainage area, retention/detention volumes, and infiltration capacities can be linearly upsized/downsized to fit in the target catchment at various implementation extents ($\mathscr{F}_{BG,i}$). In this way, two major simplifications are introduced into the modeling process. First, there is no need to design and simulate specific systems based on local circumstances for each and every target catchment. Second, the original shape and form of the output hydrograph from Augustenborg ($Q_{Aug}$) is retained intact while values are manipulated by factor $\xi_i$. Upsizing and downsizing the Augustenborg system is performed according to the equations below:

$$\xi_i = \frac{A_{BG,i}}{A_{Aug}} \tag{7}$$

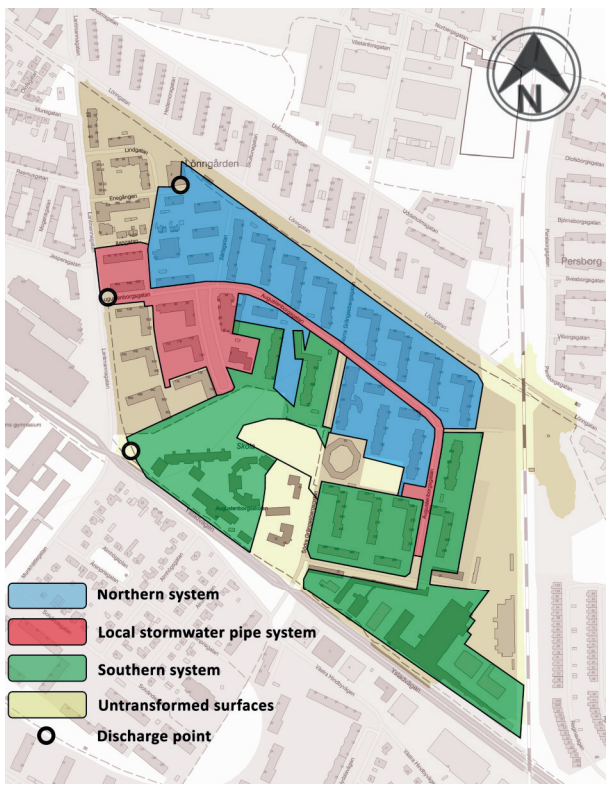



**Fig. 3.** Subsystems of Augustenborg, including two blue-green stormwater retrofits and the local pipe system. The figure also shows the discharge points from the subsystems to the sewer network.

**Table 1**
The subcatchments in the optimization study, totaling 954.35 ha (see Fig. 4 for the graphic illustration).

| ID | District name | Area (ha) | ID | District name | Area (ha) |
|---|---|---|---|---|---|
| 1 | Fridhem N | 18.09 | 14 | Djupadal NV | 48.93 |
| 2 | Ribersborg | 18.52 | 15 | Kroksbäck | 63.97 |
| 3 | Rönneholm | 58.85 | 16 | Djupadal S | 91.59 |
| 4 | Fågelbacken | 17.20 | 17 | Möllevången | 21.02 |
| 5 | Fridhem Ö | 18.88 | 18 | Södervärn | 26.61 |
| 6 | Rönneholm SV | 12.33 | 19 | Södra Sofielund | 13.28 |
| 7 | Dammfri | 58.16 | 20 | Gröndal | 61.21 |
| 8 | Lorensborg | 33.28 | 21 | Kulladal | 56.91 |
| 9 | Västervång | 10.22 | 22 | Heleneholm | 24.35 |
| 10 | Fridhem SÖ | 8.31 | 23a | Eriksfält V | 7.97 |
| 11a | Mellanheden V | 24.57 | 23b | Eriksfält Ö | 21.48 |
| 11b | Mellanheden Ö | 25.93 | 24 | Almhög | 32.30 |
| 12a | Nya Bellevue V | 15.39 | 25 | Hindby | 27.38 |
| 12b | Nya Bellevue Ö | 46.11 | 26 | Gullvik N | 11.50 |
| 13 | Rosenvång | 37.29 | 27 | Gullvik S | 42.72 |

$$Q_{TR,i} = Q_{MU} + Q_{BG} = Q_{FRC,i}\cdot(1 - \mathscr{F}_{BG,i}) + Q_{SRC,i} + Q_{ww,i} + \xi_i \cdot Q_{Aug} \quad (8)$$

where $A_{BG}$ is the area occupied by the blue-green retrofit (ha), $A_{SC}$: target subcatchment area (ha), $\mathscr{F}_{BG}$: implementation extent in the target catchment ($0 \leqslant \mathscr{F}_{BG} \leqslant 1$), $A_{Aug} = 9.6$ ha = area of the existing retrofit in Augustenborg in Malmö (the Southern retrofit), $\xi$: retrofit scale compared to the reference, i.e., the Southern retrofit in Augustenborg, $Q_{MU}$: runoff from the conventional catchment simulated in MIKE Urban, $Q_{BG}$: runoff (discharge) from the blue-green system, $Q_{FRC}$: fast runoff component of the flow (*stormwater*), $Q_{Aug}$: discharge from the Southern retrofit in Augustenborg, $Q_{ww}$: domestic wastewater flow, $Q_{SRC}$: slow runoff component of the flow (*groundwater infiltration*), and $Q_{TR}$: total runoff to be handled by the pipe network.

### 3.2. The optimization problem

The goal of the unconstrained, bounded optimization problem was to minimize the total cost of flooded nodes, $\mathbb{C}_{flood,}$ and the cost of implementation of the blue-green systems based on the Augustenborg system, $\mathbb{C}_{act}$. The cost of flooded nodes was expressed using the average cost per flooded node, $c_{fn}$, based on a rough assessment of the available data from the extensive flooding in Malmö in 2014. Thus, $\mathbb{C}_{flood}$, in the case of a 10-year rainfall event (as employed in this study), was estimated at 47,000 SEK/(year·flooded node), and $\mathbb{C}_{act}$ (including maintenance) was estimated to be 110,000 SEK/(year·retrofitted area) during the 50-year lifespan of the built system. It should be noted that these values are preliminary assessments and cannot be considered as verified template costs in other contexts.

The cost of action was assumed to be linear with respect to the area of the constructed blue-green systems, as described in Eqs. (2) and (3), and the costs were then combined according to Eq. (4). A performance indicator (PI) was then constructed and used as the objective for the optimization to more easily discern whether a tested solution is cheaper than the reference case (PI < 0) or worse (PI > 0).

The 1D model area was divided into 30 different subcatchments, as presented in Table 1 and Fig. 4. In other words, the optimization space to be investigated was a high-dimensional space with 30 dimensions corresponding to the 30 decision variables that the subcatchments constitute.

The implementation extent of each catchment ($\mathscr{F}_{BG,i}$, defined in Eqs. (5) and (6)) was used as decision variables. The decision variables were bounded by the characteristics of the catchments by setting the bounds to [0, 1], representing catchments that are empty or full (or in between) with regards to the blue-green systems. With the reference case of no implemented blue-green solutions, i.e., $\Phi_{Ref} = C_{flood}(0)$, the full

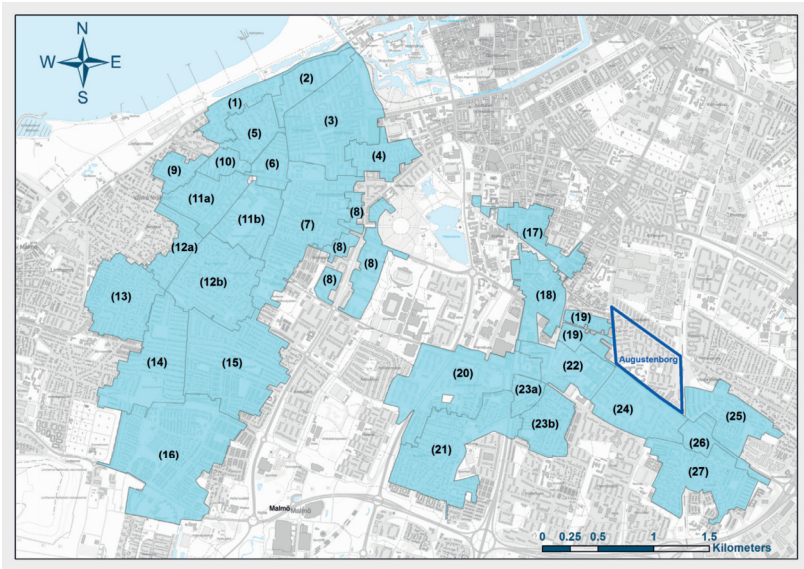



**Fig. 4.** The spatial distribution of model subcatchments in the city of Malmö, Sweden. Background picture: dimmed topographic web map, courtesy of The Swedish Mapping, Cadastral and Land Registration Authority, ©*Lantmäteriet*.

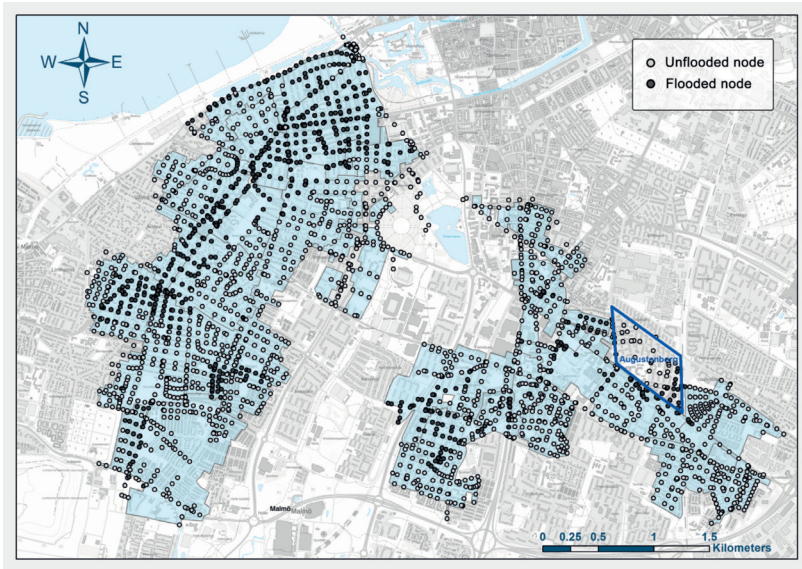



**Fig. 5.** Flooded nodes (filled markers, ●) and unflooded nodes (unfilled markers, ○) under the reference scenario, i.e., *10y-CDS*; $\mathscr{F}_{BG,i} = 0$. Background picture: dimmed topographic web map, courtesy of The Swedish Mapping, Cadastral and Land Registration Authority, *©Lantmäteriet*.

optimization problem was expressed as:

$$\min \quad PI = \frac{\Phi(u) - \Phi_{Ref}}{\Phi_{Ref}}$$

$$\text{w.r. t.} \quad u = [\mathscr{F}_{BG,1}, ..., \mathscr{F}_{BG,k}]$$

$$\text{s.t.} \quad \mathscr{F}_{BG,i} \in [0, 1] \quad \text{for } i \in [1, 2, ..., k]$$

## 4. Results and discussion

The optimization problem was solved using the SciPy implementation of the algorithm known as constrained optimization by linear approximation (COBYLA), which is a derivative free method (Conn et al., 1997; Powell, 1994). The optimization was performed for a Chicago Design Storm (CDS) (Keifer and Chu, 1957) with a recurrence interval of 10 years, denoted as *10y-CDS*. The initial values for the decision variables were the reference case, i.e., $\mathscr{F}_{BG,i} = 0$. In the reference case, the water level in 779 manholes was over the average basement level (assumed to be 1.3 m below the ground level), leading to basement flooding. The spatial distribution of the flooded manholes (model nodes) is illustrated in Fig. 5.

After the simulation of the reference case scenario, the optimization process under the *10y-CDS* continues to locate an optimum (local minimum for Φ(u)) according to the design of the COBYLA-algorithm. Fig. 6 is a two-dimensional illustration of approximately 100 iterations.

As shown in Fig. 6, the optimization started with the reference scenario, and all subsequent iterations are presented in the clockwise order. The first 30 iterations have an average PI value of approximately + 0.04 (increased costs). The optimization continues with systematic alterations in the implementation extents and moves toward an average PI of approximately −0.01 (decreased costs) and ultimately finds an optimal solution with a PI of almost −0.03 and 714 flooded nodes. The results of the optimization process suggested that subcatchments 1 and 19 should be transformed to mesoscale blue-green stormwater systems at 56.6% (≈10 ha) and 63.6% (8.5 ha), respectively. Based on this optimization, the total cost of a *10y-CDS* event will be 3% lower than the cost of flooding under the reference case scenario. This shows that there is at least one scenario through which implementation of blue-green stormwater systems would lead to less overall costs, i.e., the sum of action and flood costs, than the reference case scenario.

Although the financial benefit is limited in the optimized scenario, the total number of the flooded nodes is decreased by approximately 8.5%. The green nodes in Fig. 7 show the spatial distribution of 65 nodes that do not flood in the optimized scenario compared to the reference. It is also interesting to note that the majority of the saved nodes (44 nodes) lie within the boundaries of the optimally retrofitted subcatchments, i.e., 1 and 19, while the rest (21 nodes) are in other subcatchments with $\mathscr{F}_{BG,i=0}$. This shows that the benefits of mesoscale blue-green retrofits are not limited to the retrofit boundaries.

Although economic savings (~3%) are modest, it should be noted that this value exclusively represents the monetary and financial aspects of the action. However, from a cost-benefit perspective, the value of the blue-green systems is much more comprehensive and may include aesthetics, societal add-ons in terms of human well-being, amenity, and promoted biodiversity. In this respect, tools for monetization and quantification of multiple benefits of blue-green stormwater systems, such as BeST (Ashley et al., 2016, 2018), are becoming widely available. These tools can be employed and adapted to local contexts in order to enhance the decision-making process.

As seen in Fig. 6, there is one iteration – marked with (D) – that despite introduced blue-green retrofits – leading to elevated costs by

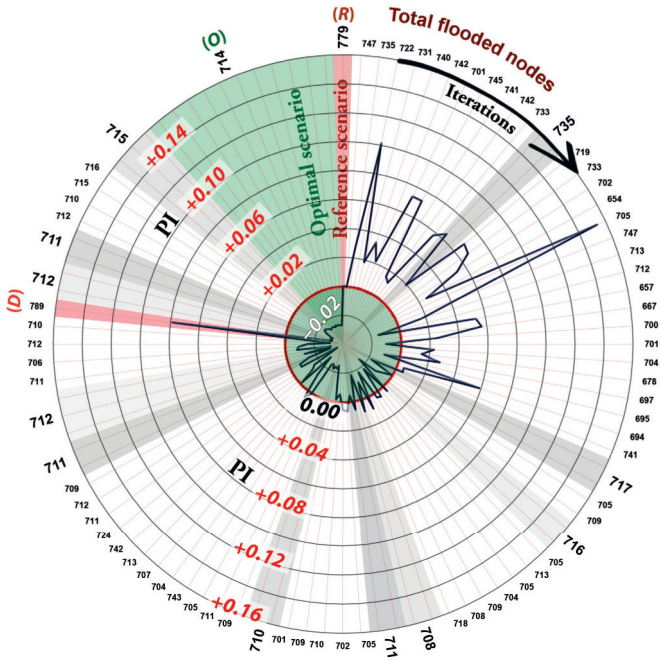



**Fig. 6.** Convergence map of 100 iterations (out of 325) using COBYLA. *Angular axes* show the total number of flooded nodes, and the *radial axes* show the performance indicator (PI) for each iteration. (R): Reference scenario, (D): Deteriorated scenario, and (O): Optimal scenario.

approximately 8% – the number of flooded nodes has increased to 789 compared to the reference scenario (779 flooded nodes). An explanation for this deteriorated behavior is that in the complex system of catchment-sewer interactions, even detained and retained discharges from catchments can possibly coincide with some peak discharge along the way causing elevated hydraulic head and, hence, flooding. This agrees with the recommendation by Haghighatafshar et al. (2018b), who advise the implementation of blue-green stormwater retrofits in the upstream areas of a city catchment to avoid (or decrease the risk of) deteriorative flow coincidences. The deteriorated flood situation, despite the introduction of blue-green stormwater systems, is an indication that unoptimized retrofits of blue-green systems in which the discharge is connected to the same sewer network do not necessarily improve the hydraulic performance of the network. The same phenomenon could also occur when rainfalls other than the optimization rainfall, with different hyetograph characteristics, strike. Thus, there might be situations through which a different rainfall would lead to different times of concentration and lag times so that the aforementioned unprecedented flow coincidences and flooding occur. A safe retrofitting strategy would thus be the total separation of the blue-green stormwater retrofits through which the overflow from the system does not enter the original sewer network but is diverted into a separate stormwater pipe network, as is the case for Augustenborg (see Fig. 4). Further research must be performed to achieve a better understanding of the complex behavior of sewer networks.

A sensitivity analysis was conducted according to the one-factor-at-a-time (OFAT) method with respect to $C_{act}$ and $C_{flood}$ in order to demonstrate the sensitivity of the PI to the quality of the cost estimation. It should be noted that different perturbations to $c_{fn}$, $c_{BG}$ and $M$ were performed as $C_{flood}$ is directly proportional to $c_{fn}$ and $C_{act}$ is directly proportional to $c_{BG} + M$ (provided $c_{BG}$ and $M$ are simultaneously and identically perturbed, all else unchanged). In effect, perturbations to the action cost and flooding cost, as well as their simultaneous perturbation (i.e. $\Phi(u)$), were performed ranging from $-50\%$ to $+50\%$. All other parameters, such as the hydrological characteristics of the toolchain, total retrofitted blue-green area, and number of flooded manholes, were kept unchanged. The resulting outcomes of these perturbations with regard to the delivered PI are shown in Fig. 8.

As shown in Fig. 8, the PI is extremely sensitive to the estimation of $C_{flood}$ rather than $C_{act}$. For instance, it is observed that a reduced cost of flooding by 30% would enhance the PI by a factor of 10 (from $-0.03$ to $-0.3$), while the corresponding PI change in the case of perturbations in $C_{act}$ is negligible. This is also reflected in the simultaneous perturbation of the costs (both cost of action and cost of flood), as it is in strong agreement with the sensitivity to $C_{flood}$ only. However, the provided results do not necessarily imply that the quality of the estimated cost of action (implementation and maintenance of the blue-green systems) has a trivial impact on the optimization outcome. The small impact of $C_{act}$ on the PI is mainly due to the fixed blue-green scenario (the scenario presented in Fig. 7) under which all perturbations are tested. As noted in Fig. 7, the retrofitted area is relatively small in comparison to the reference case (approximately 2% of the total modeled catchment area), while the proportion of the flooded manholes in the optimized scenario compared to the reference scenario is

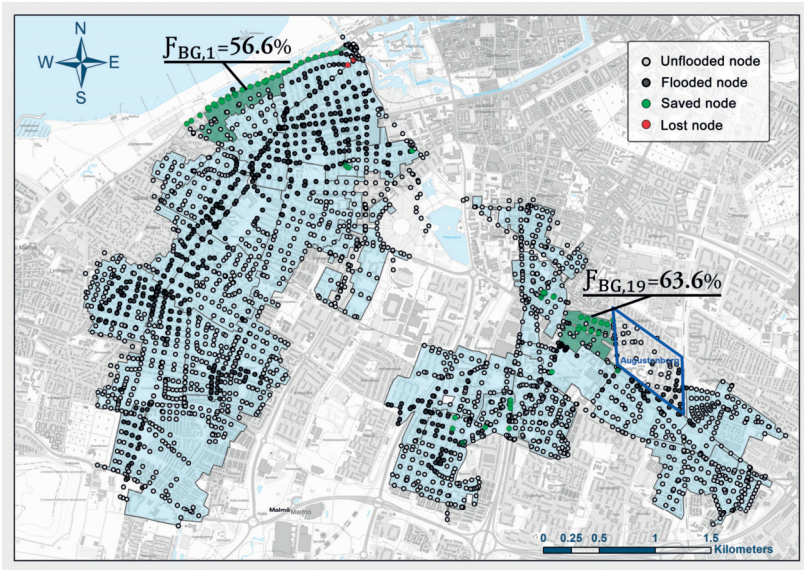



**Fig. 7.** Optimized scenario with respect to implementation extents, flooded (filled markers, ●), unflooded (unfilled markers, ○), saved (green-filled markers, ●), and lost (red-filled markers, ●) nodes. Background picture: dimmed topographic web map, courtesy of The Swedish Mapping, Cadastral and Land Registration Authority, ©*Lantmäteriet*.

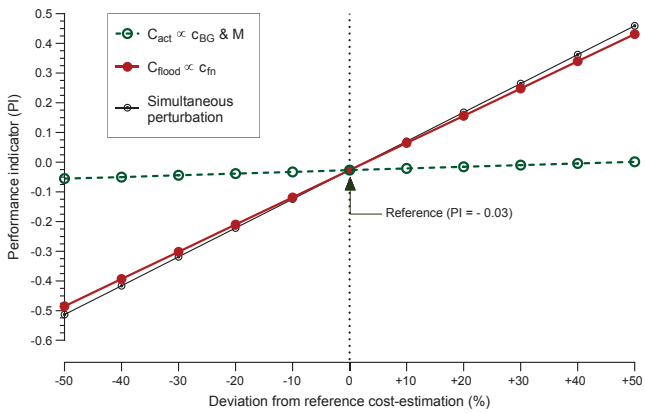



**Fig. 8.** The sensitivity of PI to the perturbations in the cost estimations associated with the optimization study.

approximately 92%. This specific optimized setup thus underlines the importance of estimation of $\mathbb{C}_{flood}$ opposed to $\mathbb{C}_{act}$. However, it is highly anticipated that perturbations in either of the associated costs (or both) would result in substantial changes in the outcome of the optimization process regarding the returned optimal $\mathbb{F}_{BG,i}$-matrix as well as the PI. Further investigations must be carried out to study the direct sensitivity of the optimization process to deviations and errors in the estimated cost components.

## 5. Concluding remarks

A hybrid model was structured to run cosimulations of mesoscale

blue-green stormwater systems and the pipe-bound sewerage network. The model was subsequently facilitated with an optimization algorithm to suggest hydroeconomically optimal solutions for citywide siting and sizing of mesoscale blue-green stormwater systems. The developed software package was tested on a city-scale catchment drained through a combined sewer system. The model delivered satisfactory results concerning promoted hydraulic performance of the sewer network (in terms of decreased flooded nodes) and financial gains. However, locating the global optimum remains a challenge due to the extremely high dimensionality of the optimization space. Given these circumstances, the model performance was shown to fulfill the general aim of the study. Further research is required to improve the optimization process and move the solution toward a global optimum. This can be done by employment of different gradient-free optimization techniques, such as evolutionary or simulated annealing algorithms, or by employing algorithms that make use of formulated analytical Jacobians. There is also the possibility to implement decision variable selection techniques to decrease the dimensionality of the decision space.

Moreover, there are indications that the performance of the optimized scenario might be specifically valid for the exact rainfall hyetograph for which the model is optimized. This also remains a serious challenge considering the stochastic nature of rainfalls and the complexity of sewer networks. It is also implied that random sizing and siting of blue-green retrofits without investigating the hydraulic consequences on the existing sewer network could lead to deterioration in the performance of the sewer network in terms of the number of flooded nodes.

## CRediT authorship contribution statement

**Salar Haghighatafshar:** Conceptualization, Investigation, Funding acquisition, Methodology, Project administration, Formal analysis, Visualization. **Mikael Yamanee-Nolin:** Methodology, Software, Visualization. **Anders Klinting:** Methodology, Software. **Maria Roldin:** Supervision, Methodology, Software. **Lars-Göran Gustafsson:** Methodology, Software. **Henrik Aspegren:** Supervision, Investigation, Resources, Funding acquisition. **Karin Jönsson:** Supervision, Investigation, Resources, Funding acquisition.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This project was financially supported by the J. Gustaf Richert Foundation at SWECO (grant number 2018-00430), Sweden Water Research and the Swedish Water and Wastewater Association via *VA-teknik Södra*. The authors are thankful to *Susanne Steen Kronborg* and *Emma Falk* at VA SYD for their contributions to the setup of the MIKE model for Malmö. All background maps (either orthophoto or topographic web map) used in the figures of this article are courtesy of the Swedish Mapping, Cadastral and Land Registration Authority, ©*Lantmäteriet*.

## References

Arnbjerg-Nielsen, K., 2012. Quantification of climate change effects on extreme precipitation used for high resolution hydrologic design. Urban Water J. 9, 57–65. https://doi.org/10.1080/1573062X.2011.630091.

Ashley, R., Digman, C., Horton, B., Gersonius, B., Smith, B., Baylis, A., 2016. Using the multiple benefits of SuDS tool (BeST) to deliver long-term benefits. In: The 9th International Conference on Planning and Technologies for Sustainable Management of Water in the City, 28 June-1 July 2016. NOVATECH, Lyon, France.

Ashley, R.M., Gersonius, B., Digman, C., Horton, B., Bacchin, T., Smith, B., Shaffer, P., Baylis, A., 2018. Demonstrating and monetizing the multiple benefits from using SuDS. J. Sustain. Water Built Environ. 4. https://doi.org/10.1061/JSWBAY.0000848.

Azzout, Y., Barraud, S., Cres, F.N., Alfakih, E., 1995. Decision aids for alternative techniques in urban storm management. Water Sci. Technol. 32, 41–48. https://doi.org/10.2166/wst.1995.0011.

Bach, P.M., McCarthy, D.T., Urich, C., Sitzenfrei, R., Kleidorfer, M., Rauch, W., Deletic, A., 2013. A planning algorithm for quantifying decentralised water management opportunities in urban environments. Water Sci. Technol. 68, 1857–1865. https://doi.org/10.2166/wst.2013.437.

Bakhshipour, A.E., Dittmer, U., Haghighi, A., Nowak, W., 2019. Hybrid green-blue-gray decentralized urban drainage systems design, a simulation-optimization framework. J. Environ. Manage. 249, 109364. https://doi.org/10.1016/j.jenvman.2019.109364.

Bernd tsson, R., Becker, P., Persson, A., Aspegren, H., Haghighatafshar, S., Jönsson, K., Larsson, R., Mobini, S., Mottaghi, M., Nilsson, J., Nordström, J., Pilesjö, P., Scholz, M., Sternudd, C., Sörensen, J., Tussupova, K., 2019. Drivers of changing urban flood risk: a framework for action. J. Environ. Manage. 240, 47–56. https://doi.org/10.1016/j.jenvman.2019.03.094.

Blank, L., Tarquin, A., 2012. Engineering Economy, seventh ed. McGraw-Hill, New York, USA.

BlueGreenCities, 2019. What is a Blue-Green City? [WWW Document]. URL http://www.bluegreencities.ac.uk/about/blue-greencitiesdefinition.aspx (accessed 8.23.19).

Conn, A.R., Scheinberg, K., Toint, P., 1997. On the convergence of derivative-free methods for unconstrained optimization. In: Iserles, A., Buhmann, M. (Eds.), Approximation Theory and Optimization: Tributes to M. Cambridge University Press, Cambridge, UK J. D. Powell, pp. 83–108.

Cunha, M.C., Zeferino, J.A., Simões, N.E., Saldarriaga, J.G., 2016. Optimal location and sizing of storage units in a drainage system. Environ. Model. Softw. 83, 155–166. https://doi.org/10.1016/J.ENVSOFT.2016.05.015.

Demuzere, M., Orru, K., Heidrich, O., Olazabal, E., Geneletti, D., Orru, H., Bhave, A.G., Mittal, N., Feliu, E., Faehnle, M., 2014. Mitigating and adapting to climate change: multi-functional and multi-scale assessment of green urban infrastructure. J. Environ. Manage. 146. https://doi.org/10.1016/j.jenvman.2014.07.025.

DHI, 2017. MIKE Urban Collection System – Modelling of storm water drainage networks and sewer collection systems (User guide). MIKE Powered by DHI.

Dolowitz, D.P., Bell, S., Keeley, M., 2018. Retrofitting urban drainage infrastructure: green or grey? Urban Water J. 15, 83–91. https://doi.org/10.1080/1573062X.2017.1396352.

Eckart, K., McPhee, Z., Bolisetti, T., 2018. Multiobjective optimization of low impact development stormwater controls. J. Hydrol. 562, 564–576. https://doi.org/10.1016/j.jhydrol.2018.04.068.

Everett, G., Lawson, E., Lamond, J., 2015. Green infrastructure and urban water management. In: Sinnett, D., Smith, N., Burgess, S. (Eds.), Handbook on Green Infrastructure: Planning Design and Implementation. Edward Elgar Publishing.

Fenner Richard, R., 2017. Spatial evaluation of multiple benefits to encourage multifunctional design of sustainable drainage in blue-green cities. Water 9, 953. https://doi.org/10.3390/w9120953.

Fletcher, T.D., Shuster, W., Hunt, W.F., Ashley, R., Butler, D., Arthur, S., Trowsdale, S., Barraud, S., Semadeni-Davies, A., Bertrand-Krajewski, J.-L., Mikkelsen, P.S., Rivard, G., Uhl, M., Dagenais, D., Viklander, M., 2015. SUDS, LID, BMPs, WSUD and more – The evolution and application of terminology surrounding urban drainage. Urban Water J. 12, 525–542. https://doi.org/10.1080/1573062X.2014.916314.

Glenis, V., Kutija, V., Kilsby, C.G., 2018. A fully hydrodynamic urban flood modelling system representing buildings, green space and interventions. Environ. Model. Softw. 109, 272–292. https://doi.org/10.1016/j.envsoft.2018.07.018.

Guo, Y., 2006. Updating rainfall IDF relationships to maintain urban drainage design standards. J. Hydrol. Eng. 11, 506–509. https://doi.org/10.1061/(ASCE)1084-0699(2006) 11:5(506).

Haghighatafshar, S., la Cour Jansen, J., Aspegren, H., Jönsson, K., 2018a. Conceptualization and schematization of mesoscale sustainable drainage systems: a full-scale study. Water 10, 1041. https://doi.org/10.3390/w10081041.

Haghighatafshar, S., la Cour Jansen, J., Aspegren, H., Lidström, V., Mattsson, A., Jönsson, K., 2014. Storm-water management in Malmö and copenhagen with regard to climate change scenarios. VATTEN. J. Water Manage. Res. 70, 159–168.

Haghighatafshar, S., Nordlöf, B., Roldin, M., Gustafsson, L.-G., la Cour Jansen, J., Jönsson, K., 2018b. Efficiency of blue-green stormwater retrofits for flood mitigation – Conclusions drawn from a case study in Malmö, Sweden. J. Environ. Manage. 207, 60–69. https://doi.org/10.1016/j.jenvman.2017.11.018.

Haghighatafshar, S., Yamanee-Nolin, M., Larson, M., 2019. A physically based model for mesoscale SuDS – an alternative to large-scale urban drainage simulations. J. Environ. Manage. 240, 527–536. https://doi.org/10.1016/j.jenvman.2019.03.037.

Hailegeorgis, T.T., Thorolfsson, S.T., Alfredsen, K., 2013. Regional frequency analysis of extreme precipitation with consideration of uncertainties to update IDF curves for the city of Trondheim. J. Hydrol. 498, 305–318. https://doi.org/10.1016/J.JHYDROL.2013.06.019.

Huang, C.L., Hsu, N.S., Liu, H.J., Huang, Y.H., 2018. Optimization of low impact development layout designs for megacity flood mitigation. J. Hydrol. 564, 542–558. https://doi.org/10.1016/j.jhydrol.2018.07.044.

Keifer, C.J., Chu, H.H., 1957. Synthetic storm pattern for drainage design. J. Hydraul. Div. 83, 1–25.

Kirby, A., 2005. SuDS – Innovation or a tried and tested practice?, In Proceedings of the Institution of Civil Engineers: Municipal Engineer. pp. 115–122. doi: 10.1680/muen.2005.158.2.115.

Kuichling, E., 1889. The relation between the rainfall and the discharge of sewers in populous districts. Trans. Am. Soc. Civ. Eng. 20, 1–56.

Kuller, M., Bach, P., Ramirez-Lovering, D., Deletic, A., 2016. The Location Choice of Water Sensitive Urban Design Within a City: A Case Study of Melbourne. IWA World

Water Congress and Exhibition, Brisbane, Australia.
Leimgruber, J., Krebs, G., Camhy, D., Muschalla, D., 2019. Model-based selection of cost-effective low impact development strategies to control water balance. Sustainability 11, 20. https://doi.org/10.3390/su11082440.
Li, C., Peng, C., Chiang, P.C., Cai, Y., Wang, X., Yang, Z., 2019. Mechanisms and applications of green infrastructure practices for stormwater control: a review. J. Hydrol. https://doi.org/10.1016/j.jhydrol.2018.10.074.
Lima, C.H.R., Kwon, H.-H., Kim, Y.-T., 2018. A local-regional scaling-invariant Bayesian GEV model for estimating rainfall IDF curves in a future climate. J. Hydrol. 566, 73–88. https://doi.org/10.1016/J.JHYDROL.2018.08.075.
Liu, H., Jia, Y., Niu, C., 2017a. "Sponge city" concept helps solve China's urban water problems. Environ. Earth Sci. 76, 473. https://doi.org/10.1007/s12665-017-6652-3.
Liu, S., Huang, S., Huang, Q., Xie, Y., Leng, G., Luan, J., Song, X., Wei, X., Li, X., 2017b. Identification of the non-stationarity of extreme precipitation events and correlations with large-scale ocean-atmospheric circulation patterns: a case study in the Wei River Basin, China. J. Hydrol. 548, 184–195. https://doi.org/10.1016/j.jhydrol.2017.03.012.
Lloyd-Davies, D.E., 1906. The elimination of storm-water from sewerage systems. Minutes Proc. Inst. Civ. Eng. 164, 41–67. https://doi.org/10.1680/imotp.1906.16637.
Mailhot, A., Duchesne, S., Caya, D., Talbot, G., 2007. Assessment of future change in intensity-duration-frequency (IDF) curves for Southern Quebec using the Canadian Regional Climate Model (CRCM). J. Hydrol. 347, 197–210. https://doi.org/10.1016/J.JHYDROL.2007.09.019.
Malmö stad, 2018. Malmö – Sweden's fastest-growing city [WWW Document]. URL https://malmo.se/Kommun–politik/Fakta-och-statistik/In-english/Demographics/Population-growth.html (accessed 12.04.18).
Milly, P.C.D., Betancourt, J., Falkenmark, M., Hirsch, R.M., Kundzewicz, Z.W., Lettenmaier, D.P., Stouffer, R.J., 2008. Climate Change. Stationarity is dead: whither water management? Science 319, 573–574. https://doi.org/10.1126/science.1151915.
Powell, M.J.D., 1994. A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Advances in Optimization and Numerical Analysis. Springer, Netherlands, Dordrecht, pp. 51–67 https://doi.org/10.1007/978-94-015-8330-5_4.
Ren, N., Wang, Qian, Wang, Qiuru, Huang, H., Wang, X., 2017. Upgrading to urban water system 3.0 through sponge city construction. Front. Environ. Sci. Eng. 11, 9. https://doi.org/10.1007/s11783-017-0960-4.
Romnée, A., Evrard, A., Trachte, S., 2015. Methodology for a stormwater sensitive urban watershed design. J. Hydrol. 530, 87–102. https://doi.org/10.1016/j.jhydrol.2015.09.054.
Sörensen, J., Mobini, S., 2017. Pluvial, urban flood mechanisms and characteristics – Assessment based on insurance claims. J. Hydrol. 555, 51–67. https://doi.org/10.1016/J.JHYDROL.2017.09.039.
Stahre, P., 2006. Sustainability in Urban Storm Drainage – Planning and Examples, first ed. Svenskt Vatten, Stockholm, Sweden.
Stahre, P., 1993. Assessment of BMPs Being Used in Scandinavia. Proceedings of the Sixth International Conference on Urban Storm Drainage. Seapoint Publishers, Niagara Falls, Ontario, Canada.
SWWA, 2016. Draiange of Runoff and Wastewater – Functional Requirements, Hydraulic Dimensioning and Design of Public Sewer Systems (in Swedish). Publication P110, Swedish Water and Wastewater Association (Svenskt Vatten), Bromma, Sweden.
Trenberth, K.E., Dai, A., van der Schrier, G., Jones, P.D., Barichivich, J., Briffa, K.R., Sheffield, J., 2014. Global warming and changes in drought. Nat. Clim. Change 4, 17–22. https://doi.org/10.1038/nclimate2067.
VA SYD, 2019. Ett säkrare och effektivare avloppssystem [A safer and more efficient wastewater system] [WWW Document]. URL https://www.vasyd.se/Artiklar/Tunnel/Malmo-avloppstunnel (accessed 08.27.19).
Vogel, R.M., Yaindl, C., Walter, M., 2011. Nonstationarity: flood magnification and recurrence reduction factors in the United States1. JAWRA J. Am. Water Resour. Assoc. 47, 464–474. https://doi.org/10.1111/j.1752-1688.2011.00541.x.
Wang, M., Sun, Y., Sweetapple, C., 2017. Optimization of storage tank locations in an urban stormwater drainage system using a two-stage approach. J. Environ. Manage. 204, 31–38. https://doi.org/10.1016/j.jenvman.2017.08.024.
Watt, E.W., Waters, D., McLean, R., 2003. Climate Change and Urban Stormwater Infrastructure in Canada: Context and Case Studies Climate Change and Urban Stormwater Infrastructure in Canada: Context and Case Studies CONTENTS PAGE. Toronto-Niagara Reg. Study Rep. Work. Pap. Ser. Rep. No. 2003-1, Meteorol. Serv. Canada, Waterloo, Ontario. 27.
Wihlborg, M., Sörensen, J., Alkan Olsson, J., 2019. Assessment of barriers and drivers for implementation of blue-green solutions in Swedish municipalities. J. Environ. Manage. 233, 706–718. https://doi.org/10.1016/j.jenvman.2018.12.018.
Wilkins, H., 2008. The integration of the pillars of sustainable development: a work in progress. McGill Int J. Sustain. Dev. Law Policy/Rev. Int. droit Polit. du développement durable McGill. https://doi.org/10.2307/24352601.
Zhang, S., Li, Y., Ma, M., Song, T., Song, R., Zhang, S., Li, Y., Ma, M., Song, T., Song, R., 2018. Storm water management and flood control in sponge city construction of Beijing. Water 10, 1040. https://doi.org/10.3390/w10081040.
Zhou, Q., Lai, Z., Blohm, A., 2018. Optimising the combination strategies for pipe and infiltration-based low impact development measures using a multiobjective evolution approach. J. Flood Risk Manage. 12. https://doi.org/10.1111/jfr3.12457.
Zischg, J., Rogers, B., Gunn, A., Rauch, W., Sitzenfrei, R., 2019. Future trajectories of urban drainage systems: a simple exploratory modeling approach for assessing socio-technical transitions. Sci. Total Environ. 651, 1709–1719. https://doi.org/10.1016/j.scitotenv.2018.10.061.
Zischg, J., Zeisl, P., Winkler, D., Rauch, W., Sitzenfrei, R., 2018. On the sensitivity of geospatial low impact development locations to the centralized sewer network. Water Sci. Technol. 77, 1851–1860. https://doi.org/10.2166/wst.2018.060.

**Stable chemical production, cheaper pharmaceuticals, and well-placed flooding countermeasures – using the same tool!**

We can surely agree that it would be nice if we could produce necessary chemicals more efficiently, provide everyone with the medicines they need at a lower cost than what is possible today, and prevent flooding in the urban environment. But what do these three things have in common, really? Well, the fact that we can *optimize* the different *processes* behind all three! There are processes everywhere in the world, complex as well as simple, and optimizing a process means improving it, making it as good as it possibly can be. However, doing this experimentally can be very time-consuming and expensive – and sometimes even practically impossible. Thus, a viable alternative is to take advantage of modern computational power and instead perform computer-aided optimization studies. For this purpose, a mathematical *model* is required, which needs to be *simulated* in order to obtain useful information about the real process on which we can base our optimization.

This thesis provides a procedure and a framework together with a number of optimization methods, which are general and generalizable tools that can be used for efficiently performing computer-aided optimization studies. These tools are based on previously published literature combined with the work that is collected and presented in this thesis, and the goal is that they can be used as an aid to improve – in principle – any real-life situation or process. The presented work also gives an indication as to what can be achieved through computer-aided optimization studies of complex processes in the different domains of production systems and urban environments, shown via case studies of industrial polyalcohol purification, chromatographic separation of pharmaceuticals, and blue-green systems for mitigation of urban flooding.

Printed by Media-Tryck, Lund 2020 NORDIC SWAN ECOLABEL 3041 0903


ISBN: 978-91-7422-732-1

Department of Chemical Engineering
Faculty of Engineering, LTH
Lund University