Doctoral theses at NTNU, 2023:127

Joakim R. Andersen

# Contributions to optimizing control

## Inspired by challenges in oil and gas production

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of
Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Joakim Rostrup Andersen

# Contributions to optimizing control

## Inspired by challenges in oil and gas production

Thesis for the Degree of Philosophiae Doctor

Trondheim, May 2023

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



NTNU
Norwegian University of
Science and Technology

**NTNU**
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics



ISBN 978-82-326-5577-9 (printed ver.)
ISBN 978-82-326-6850-2 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

IMT-report 2023-07-W

Doctoral theses at NTNU, 2023:127

Printed by NTNU Grafisk senter

# Summary

This dissertation treats optimization and control with emphasis on oil and gas applications, and consists of an introduction followed by five main contributions.

In the first contribution, we focus on allocating a limited available resource between a set of units which is a problem that arises in several application areas. We propose an online derivative-free trust-region model-based method to tackle a fairly general version of the resource allocation problem where units may be turned on or off. The units are considered as black boxes which may only be evaluated given that all the other units are evaluated simultaneously, and no gradient information is available. This method was inspired by an industrial problem and emphasis is put on both providing feasible points during the optimization and on not incurring additional increase in cost while searching for the optimum. The latter cannot be guaranteed, but the algorithm allows for automatic or manual ranking of the different units to attempt to reduce negative impact on the cost.

In the second contribution, we treat systems with fast and slow dynamics which give rise to objectives in different time scales which may not be aligned. The existing dynamic optimal control methods might become computationally infeasible due to the fine discretization required to capture the fast dynamics. On the other hand, a Real-Time Optimization (RTO) method based on steady-state models, which is computationally efficient, can greedily drive the plant towards optimal operation. The drawback of the RTO approach is that it may yield actions that only focus on near future goals and the objectives involving the slower dynamics are neglected. We propose to extend RTO with a lookahead strategy by introducing a predictor to capture the effect of changing the current controls on the long-term objective. In this way, we introduce the long-term objectives in RTO while maintaining its computational efficiency and not losing focus of short-term objectives.

In the third contribution, we tackle the Daily Production Optimization (DPO) problem which is the task of maximizing production of hydrocarbons subject to operational constraints. Handling of uncertainty in model structure and parameters is of high importance to the usefulness of the solution. Ignoring these challenges will, most likely, render the solution either infeasible or the solution will not be an optimum of the plant. We apply a Reinforcement Learning based Economic Nonlinear Model Predictive Control (RL-based ENMPC) which uses state- and output-measurements from the plant to iteratively update the controller.

Thereafter, in the fourth contribution, we investigate similarities and differences between RL-based ENMPC and Modifier Adaptation. Both methods aim to correct for plant-model mismatch in order to improve economic performance. We suggest specific parametrizations of the RL-based ENMPC which allow for selecting, or learning of, a closed-loop steady state in addition to the local policy around it. The proposed parametrizations are theoretically justified, in addition to being illustrated through examples.

Finally, in the fifth contribution, the focus is moved away from production optimization to linear regression which is concerned about fitting a model to a set of data. The weighted least squares method is a standard tool for performing linear regression. In this paper, we focus on the case when some of the samples are given priority over others. The residuals for these samples should be given an infinite weighting compared to other samples. However, due to numerical limitations, a weight which is finite but sufficiently large must be chosen instead. We suggest an alternative approach that in practice allows infinite weighting. This is achieved by reformulating the regression optimization problem as a bilevel program.

# Preface

This thesis is submitted in partial fulfilment of the requirements for the degree of Philosophiae Doctor (Ph.D.) at the Norwegian University of Science and Technology (NTNU). The work has been carried out at the Department of Engineering Cybernetics under the supervision of Professor Lars Imsland and co-supervision of Professor Alexey Pavlov.

The project has been part of BRU21 – NTNU Research and Innovation Program on Digital and Automation Solutions for the Oil and Gas Industry (`www.ntnu.edu/bru21`).

## Acknowledgements

I would first and foremost like to thank my supervisor Professor Lars Imsland for his belief in me, his encouragement and his support over the last 4-5 years. I want to thank him for the many fruitful and interesting discussions we have had. He has given me the freedom to pursue the topics that have caught my interest, and always been positive to explore new directions. And lastly, I want to thank him for always making me believe in myself. I could not have wished for a better supervisor.

I would like to thank my co-supervisor Professor Alexey Pavlov for being a good sparring partner, and for providing interesting perspectives.

In addition, I would like to thank Professor Sebastien Gros for his engagement in my final work. The many discussions and the support have been crucial for the completion of the last paper.

At the start of the PhD, I was lucky enough to share an office with Andreas, Mathilde and Otávio. I would like to thank them for all the interesting discus-

sions we had on modeling of oil and gas things and on optimization. But most importantly: thank you for making the start of the PhD fun! A motivating coffee break was always around the corner.

I would also like to thank Mikkel for all the fun and interesting talks we shared during walks under the corona lockdown.

I would also like to thank my colleagues at the Department of Chemical Engineering for welcoming me into their social group: Dinesh, Adriaen, Julian, Andrea, Allyne, Lucas, and the many more.

A special thanks goes to Dinesh Krishnamoorty and to my two encouraging cosupervisors during my master's: Mathias Bellout and Andrés Codas. Without you I would probably never have applied for a PhD. You inspired me.

Finally, I want to thank my family for their endless support during my education. Always having my family there whenever needed has been priceless.

# Contents

# 1 | Introduction

This thesis is concerned with optimizing control for processes with an emphasis on the oil and gas industry. The term *production optimization* is commonly used in the oil and gas industry for referring to the concept of optimizing the production of oil and gas. Short-term production optimization, with a time horizon of days, is known as Daily Production Optimization in the petroleum industry.

The DPO activity is concerned with maximizing the immediate revenue arising from extracting hydrocarbons which are trapped in the subsurface reservoirs. While doing so, several constraints must be satisfied. Gas and water are typically extracted alongside oil, and the topside production facility may have a limitation on how much gas and water it may process. If the downhole pressure in a well is not sufficient to provide an economically viable production, gaslift may be injected into the wellbore to lower the density of the fluid mixture. The distribution of available gaslift between the set of wells is another constraint that must be handled when performing DPO. In DPO, the control inputs are typically the chokes openings, and the settings of compressors and pumps.

DPO, or more generally Real-Time Optimization (RTO)[1], can be seen as a part of the overall process control of a plant. Process control is a complex problem. To ease the task, a decision hierarchy is often used. Decisions are split into different layers depending on their time scale, see Fig. 1.1. This type of control hierarchy is widely accepted, see *e.g.* [2] and [3].

The control horizons considered in each layer gets shorter the lower in the hierarchy you go. The different time ranges are shown in the figure for the different layers. However, these may vary depending on the considered process. Each layer typically receives set-points from the above layer and then report back the results.

The focus of this thesis has been on the Real-Time Optimization (RTO) layer,

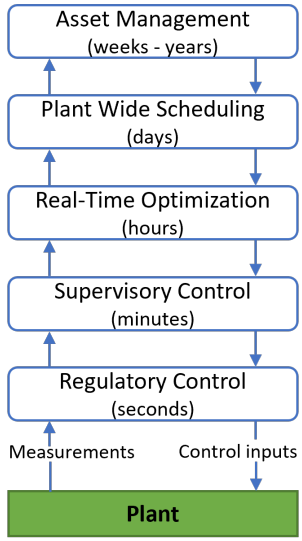


Figure 1.1: Control hierarchy in process control[4].

and the Supervisory Control layer. The RTO layer typically considers complex nonlinear steady-states models with an economic cost. The Supervisory Control layer has been assumed to be a Model Predictive Control (MPC). The MPC typically uses simple linear models and a quadratic cost to keep the computational cost manageable for practical implementation.

## 1.1 Motivation

The time scale separation done in Fig. 1.1 does not necessarily imply (economic) optimality of the overall process control[3]. This non-optimality has been one of the motivations for this thesis. This thesis has been looking into different methods for and aspects on considering more than one time horizon simultaneously in order to achieve better economic performance.

In addition, this thesis has been motivated by the issue of *plant-model mismatch*. This issue arises when a model is used for controlling or optimizing a system, but the model is incorrect. This may either be due to that the model parameters are incorrectly chosen, or that some of the underlying dynamics are not modelled resulting in a structural mismatch between the model and the true dynamics. Such plant-model mismatch may lead to both non-optimal

control and to potential infeasible control if output constraints are present.

## 1.2 Publications

Five publications have been written during the work leading up to this thesis. The publications are listed below. The three conference papers have been accepted, presented and published. The two journal articles are submitted. Paper **A** has undergone major revision and has been resubmitted.

### Journal publications

Paper **A Andersen, J. R.**, Imsland, L., "Data-driven derivative-free trust-region model-based method for resource allocation problems". In: *Computers & Chemical Engineering* (2023). (Submitted)

Paper **D Andersen, J. R.**, Gros, S., Imsland, L., "Parametrization of learning based MPC for process control". In: *Journal of Process Control* (2023). (Submitted)

### Conference publications

Paper **B Andersen, J. R.**, Lima Silva, T., Imsland, L., Pavlov, A., "Real time optimization of systems with fast and slow dynamics using a lookahead strategy". In: *2020 59th IEEE Conference on Decision and Control (CDC)* (2020), pp. 2342–2349. DOI: 10.1109/CDC42340.2020.9304460

Paper **C Andersen, J. R.**, Imsland, L., "Application of Data-Driven Economic NMPC on a Gas Lifted Well Network". In: *IFAC-PapersOnLine* 54.3 (2021), pp. 275–280. DOI: https://doi.org/10.1016/j.ifacol.2021.08.254

Paper **E Andersen, J. R.**, Imsland, L., "Bilevel programming as a means of infinite weighting in regression problems". In: *IFAC-PapersOnLine* 55.7 (2022), pp. 851–856. DOI: https://doi.org/10.1016/j.ifacol.2022.07.551

## 1.3 Outline

The remaining part of the dissertation is outlined as follows. In Chapter 2, the different problems we have studied are presented. In Chapter 3, we elaborate on the contributions of each paper. Conclusion and future work are given in Chapter 4. The papers written as a part of the dissertation are available in Chapter 5.

# 2 | Background

In this chapter, the different problems that have been studied in the thesis will be presented. The problems are all related to optimization and control, and were inspired by different aspects on production optimization within the oil and gas industry.

## 2.1 Resource allocation problem

The Resource Allocation Problem (RAP) is concerned with allocation a shared resource between a set of units to optimize a cost function, and may be seen as a version of the RTO in Fig. 1.1. The formulation studied in this thesis reads as

$$\min_{\mathbf{u},\mathbf{z}} \quad f(\mathbf{u}) = \sum_{i=1}^{n_u} f_i(u_i) \tag{2.1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_u} u_i = u_{\max} \tag{2.1b}$$

$$z_i \underline{u}_i \leq u_i \leq z_i \overline{u}_i \tag{2.1c}$$

$$u_i \in \mathbb{R} \tag{2.1d}$$

$$z_i \in \{0, 1\} \tag{2.1e}$$

where $n_i$ is the number of units, $f(\mathbf{u})$ is the cost to minimize, $u_i$ is the resource allocated to unit $i$, $f_i(u_i)$ is the cost of allocation $u_i$ resource to unit $i$, and $u_{\max}$ is the total available resource that must be shared between the units. Each unit must either be allocated a resource amount between $\underline{u}_i$ and $\overline{u}_i$, or the unit must be turned off by setting the associated binary variable $z_i$ to zero. It is assumed that each $f_i$ is convex. Furthermore, these relationships may

only be sampled, and thus are seen as black-boxes. In addition, $f_j$ can only be evaluated if all the other $f_i$'s are evaluated simultaneously.

The specific formulation above was inspired by the gaslift allocation challenge in the oil and gas industry. The available gaslift ($u_{\max}$) must be allocated between a set of $n_u$ wells (or units) to maximize the oil production. Each well may have a lower and upper bound on how much gaslift it may handle in order to operate in a safe and stable region. The binary variables represent the possibility of having a well open ($z_i = 1$) or closed ($z_i = 0$). The relationship between injected gaslift and produced oil is typically concave which justify the assumption of convexity. The optimization is assumed to be done while the production is running. This is the reason why it is required that all units must be evaluated simultaneously. At this point, we may assume that the oil rate for each well can be measured during production.

Resource Allocation Problems are not a new challenge, and if no binary variables and the $f_i$'s were known (and not black-boxes), existing methods could be used, see *e.g.*, [10]. If the binary variables were disregarded, then a set of different derivative-free optimization methods could be used, see *e.g.*, [11–13].

The gaslift allocation challenge have previously been tackled by derivative-free trust-region model-based methods [14–16]. These papers consider a similar setup to (2.1), *i.e.*, the minimization of a function subject to linear constraints, except for [16] which in addition handles nonlinear constraints. However, they do not include the decisions on opening or closing the wells. Furthermore, suggested evaluation points may be infeasible with respect to constraints.

In Paper **A**, we propose a tailored derivative-free trust-region model-based optimization method for (2.1) which exploits problem structure . Furthermore, it only ask to evaluate points which are feasible with respect to all constraints during the optimization procedure.

## 2.2 Optimal control of systems with fast and slow dynamics

The optimization and control of an oil reservoir asset may be structured in a similar way as the decision hierarchy in Fig. 1.1, see [1]. Optimization of the reservoir happens in several different time-scales. We will focus on the perspective of two different groups of engineers. The first group, the reservoir engineers, make plans on the drainage strategy of the reservoir. The second

group, the production engineers, are concerned with the daily production optimization. Naturally, these two groups of engineers consider time scales which differs in many orders of magnitude (months or years versus hours or days).

The optimization objectives when different time horizons are considered may not be aligned. However, some of the control inputs are shared by the two groups of engineers, but of course only one control action may be applied to the real system.

A mathematical formulation of the considered problem reads as

$$\min_{\mathbf{u}} \sum_{\forall k \in \mathcal{K}} t_k \left[ \alpha L^s(\mathbf{x}_{k+1}, \mathbf{u}_k) + (1-\alpha) L^l(\mathbf{x}_{k+1}, \mathbf{u}_k) \right] \tag{2.2a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad \forall k \in \mathcal{K} \tag{2.2b}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k), \quad \forall k \in \mathcal{K} \tag{2.2c}$$

$$\mathbf{0} \leq \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k), \quad \forall k \in \mathcal{K} \tag{2.2d}$$

where $\mathcal{K} := [0, 1, \ldots, N-1]$ is the set of time steps, $t_k$ is the $k$th discretization step size, $\mathbf{f}$ and $\mathbf{x}_k$ are the dynamic model and state vector of the slow dynamics, $\mathbf{h}$ is the constraint vector, $\mathbf{g}$ represents the dynamics arising from the fast dynamics and is here imposed as steady-state equations. Finally, $L^s$ and $L^l$ are the short-term and long-term objectives, respectively, and $\alpha$ is a weighting factor between these two goals. In the oil reservoir setting, the $\mathbf{x}$ would represent the state of the reservoir, and the fast dynamics from the gathering network are represented through the algebraic equations in $\mathbf{g}$.

A standard procedure of solving the optimal control problem in (2.2) is through direct transcription methods such as multiple or single shooting, see *e.g.* [17]. A framework for performing multiple shooting on oil reservoirs (without the fast dynamics) are were developed in [18]. This framework was extended to handle output constraints from the gathering network in [19]. However, these may run into computational limitations as the complexity is typically cubic in state- and input dimension, and the time horizon. If we want the fast dynamics and the constraints to be satisfied at all times, a fine discretization would be needed, which could potentially make the computationally burden of the framework in [19] too large.

In Paper **B**, we propose a method which is based on solving a series of RTO steps, but where the RTO is extended with a "lookahead" functionality to (try to) improve the long-term performance.

## 2.3 Economic Nonlinear Model Predictive Control under plant-model mismatch

Economic Nonlinear Model Predictive Control (ENMPC) is a control method which combines the RTO and the MPC layer in Fig. 1.1[20]. The ENMPC has an economic cost and nonlinear models, as in RTO, but uses dynamic models, as in MPC. The goal of combining these two layers is to increase the economic performance. The additional computational burden of considering nonlinear models and an economic cost is a disadvantage of ENMPC.

When using RTO, MPC or ENMPC, the quality and correctness of the models are of high importance regarding the usefulness of the solutions. Bad models may lead to sub-optimal and potentially infeasible control if output constraints are considered. The incorrect or bad models are sometimes referred to as plant-model mismatch. To somehow handle the plant-model mismatch is of high importance to improve the control.

A general ENMPC formulation reads as

$$\min_{\mathbf{x},\mathbf{u}} \quad T(\mathbf{x}_N) + \sum_{k=0}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) \tag{2.3a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{2.3b}$$

$$\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \leq 0 \tag{2.3c}$$

$$\mathbf{x}_0 = \mathbf{s} \tag{2.3d}$$

where $T$ is the terminal cost, $l$ is the stage cost, $N$ is the number of steps in the control horizon, $\mathbf{f}$ is the dynamic model, $\mathbf{h}$ contains constraints, $\mathbf{s}$ is the initial state, the $\mathbf{x}_k$'s and $\mathbf{u}_k$'s are the states and inputs. $T$, $l$, $\mathbf{f}$, and $\mathbf{h}$ may all be incorrectly modelled. Importantly, they may be incorrect no matter which parameters are in use, i.e., they may include structural mismatch between the modelled and real equations.

Two different methods of handling the plant-model mismatch have been investigated: Reinforcement Learning based ENMPC (RL-based ENMPC) and Modifier Adaptation (MA).

Both of these methods have the perspective that the goal is to modify the optimization formulation directly in order to achieve optimal performance. That means that they do not follow the more classical approach of trying to fit the model to the observed data.

Modifier Adaptation[21–23] was originally a steady-state method made for

achieving optimal steady state of the plant seen from the RTO perspective. Lately, it has also been extended to offset-free ENMPC, see [24, 25]. This new offset-free ENMPC with MA aims to achieve the optimal steady state despite plant-model mismatch, and uses the dynamic controller in order to also handle transients.

The RL-based ENMPC[26–28] approaches the same problem from a different perspective. In contrast to the MA method, this method is based on collecting data from transients and not steady states. Furthermore, it can be used to optimize the transients in addition to the steady state. In order for the RL-based ENMPC to be able to learn the optimal policy, the parametrization must be *rich enough*.

In Paper **D**, we propose parametrizations of a generic ENMPC which will allow for selecting, or learning of, a closed-loop steady state and the local policy around it. Furthermore, we compare MA to the RL-based ENMPC.

## 2.4 Linear regression with sample prioritization

If one would like to perform linear regression with sample prioritization one could use weighted least squares and assign a *sufficiently large* weight to the prioritized samples. A general weighted linear least squares regression problem reads as[29]

$$\min_{\mathbf{p}} \quad \sum_{j=1}^{n} w_j (y_j - \hat{y}(x_j; \mathbf{p}))^2 \tag{2.4}$$

where $\mathbf{p}$ are the parameters, $n$ is the number is samples, $x_j$'s are the inputs, $y_j$'s are the outputs and $\hat{y}$ are the predicted outputs by the model. Finally, $w_j$'s are the weights.

Conceptually, to perform sample prioritization, one would set $w_j = 1$ for the non-priority samples and $w_j = \infty$ for the prioritized samples. Obviously, practical implementations are not able to handle $\infty$ and a *sufficiently large* weight must be chosen instead. However, deciding this weight may be troublesome. The samples with $w_j = 1$ should only be used if there are more degrees of freedom in the optimization after the prioritized samples are considered.

In Paper **E**, we propose a bilevel optimization formulation for this problem. Bilevel programming has previously been used for parameter estimation. *E.g.*, it has been used to find hyper-parameters simultaneously as learning the model

parameters in a machine-learning setting [30], and it has been exploited to improve the realism of the resulting model [31].

# 3 | Contributions

In this chapter, we elaborate on the contributions of each paper.

### Paper A - Data-driven derivative-free trust-region model-based method for resource allocation problems

This paper is concerned about the resource allocation problem given in Section 2.1.

We propose a data-driven derivative-free trust-region model-based optimization method. The proposed method is data-driven in the sense that it only relies on measured data, and no mechanistic models are used. In this way, we overcome the issue of sub-optimality due to plant-model mismatch. It is derivative-free in the sense that no gradient information from the underlying system is available. And it is a trust-region model-based method in the sense that the input to output relationships are modelled by linear models, and these are trusted within some trust-region. Problem structure is exploited in order to be more sample efficient when creating the linear models. In addition, the method includes the decision of turning the units on or off.

We also propose a decomposition of what is known as the Gas Lift Performance Curve (GLPC). This is a curve that is typically used in the allocation of available gaslift between a set of wells. The decomposition allows for better utilization of live production data. This decomposition removes the need of the assumption of needing to be able to measure the oil flow rate for each well during production.

### Paper B - Real time optimization of systems with fast and slow dynamics using a lookahead strategy

This paper is concerned about the optimization of systems with fast and slow dynamics as discussed in Section 2.2.

The method suggested in the paper takes the production engineers perspective as a starting point, and finds the set of controls that will maximize the current steady-state production. Then the proposed method takes one more step where also the (predicted) effect of the current control actions on the long-term economic return are taken into account.

The suggested method is generalized to a framework which may be used to optimize systems consisting of both fast and slow dynamics using a lookahead strategy.

### Paper C - Application of Data-Driven Economic NMPC on a Gas Lifted Well Network

Model uncertainty when doing any model-based production optimizing is of high importance. An incorrect transition model may lead to suboptimal production, and if constraints are present, the control action may even be infeasible for the real system.

This paper contains an application of Reinforcement Learning based ENMPC. A setup of two gas lifted wells were considered, and (close to) optimal control was achieved.

### Paper D - Parametrization of learning based MPC for process control

Reinforcement Learning based Economic Nonlinear Model Predictive Control (RL-based ENMPC) is a method where RL is applied to update the ENMPC in order to achieve an optimal policy. The RL updates the parameters in order to improve the closed-loop performance, and not to improve the correctness of any model predictions.

The concept of directly modifying the optimization formulation in order to achieve an optimal steady state is known as “Modifier Adaptation” in the RTO literature. MA have been merged with ENMPC in order to create offset-free

ENMPCs. However, as the MA is a steady-state method, it requires steady-state data (or prediction of such data) to update the modifiers.

This paper suggests parametrizations of a general ENMPC in order to achieve rich enough parametrizations for the ENMPC to be able to deliver (i) an optimal closed-loop steady state, and (ii) locally optimal transients to the optimal steady state. Theoretical justifications and illustrative examples are provided to support the suggested parametrizations.

The proposed parametrizations are applied to a wind turbine example, where it is compared to a state-of-the-art method which combines MA with offset-free ENMPC. The proposed method outperforms the ENMPC with MA. The parametrization which also allows for learning of a locally optimal policy outperforms the one where only the optimal closed-loop steady state can be learnt.

This paper contains theoretical contributions to the method applied in Paper **C**.

### Paper E - Bilevel programming as a means of infinite weighting in regression problems

This paper proposes a method which allows for sample prioritization as discussed in Section 2.4 without needing to decide a *sufficiently large* weight for the prioritized samples. This is achieved by using what is known as bilevel programming.

# 4 | Concluding remarks

## 4.1 Conclusion

This dissertation consists of five papers. In the following, a conclusion for each of the papers are given.

In Paper **A**, we proposed a derivative-free trust-region model-based algorithm to tackle a resource allocation problem with some specific characteristics that were inspired by a real world problem. The algorithm converged to local optima in all the compared cases. In addition, we proposed a decomposition of the gas lift performance curve to exploit available sensor data. Furthermore, we illustrated how this decomposition can be combined with the suggested algorithm.

The simulation study in Paper **B** showed that including the long-term effects into the short-term decision making process was valuable. The existing methods in literature to optimize systems with both fast and slow dynamics, may suffer from the computational requirements due to the dimensionality. Disregarding the long-term effects gives rise to a method that is fast but non-optimal in the long run. The suggested methodology combines the advantages of the traditional RTO with a lookahead strategy to improve the long-term goal. The simulation study shows promising results for this novel methodology.

In Paper **C**, we applied a data-driven ENMPC method to a simulation study of two gas lifted oil wells. The results were promising in the sense that the data-driven ENMPC was able to tune the parameters to a set of values that gave a near-optimal operation of the process even though there were more incorrectly guessed parameters than the number of parameters that the learning method was allowed to change.

In Paper **D**, two different parametrizations of a generic ENMPC formulation

has been proposed. The parametrizations allows for selecting the closed-loop steady state and the local policy around the steady state. Each parametrization is justified theoretically and through examples. The complete parametrization, joined by RL to learn parameters, is shown to be superior compared to an state-of-the-art offset-free ENMPC with Modifier Adaptation in the considered simulation study of a wind turbine.

Paper **E** presented a novel application of bilevel programming where the formulation was used to introduce infinite weighting. In the studied example, we saw that the proposed method allows for prioritization without having to determine a weighting factor. A disadvantage of the proposed method, compared to the weighting approach, is that integer decision variables will be introduced if there are constraints on the lower-level problem and a Mixed Integer Nonlinear Program (MINLP) solver is required instead of an NLP solver. MINLP solvers are typically slower than its integer-free counterpart. Nonetheless, the method removes issues with ill-conditioned formulations due to a large weight. Further, as no weight is used, the tuning process is eliminated.

## 4.2 Future work

Each of the two journal papers contain interesting research directions. The method in Paper **A** should be tested on more problems. Ideally, on an advanced simulator with realistic measurement noise to evaluate how it will work in more realistic setups. Furthermore, it could be of interest to look into how to efficiently handle output constraints that needs to be modelled. To make the algorithm more applicable, it could be interesting to adapt it to handle the case when an imposed control action is unobtainable by the system.

Future research directions for the parametrizations on RL-based ENMPC, Paper **D**, could be to investigate different RL methods for different processes. An interested question is: How can we exploit the collected data most efficiently? Furthermore, it could be interesting to applying the RL-based ENMPC on a setup with advanced simulators in the DPO setting. This would most likely imply that binary variables must be included in order to account for opening/closing of wells in addition to routing.

The quality of the solution of using the proposed lookahead RTO method in Paper **B** is highly dependent on the predictor. It could be interesting to create the predictor based on several realizations of the considered reservoir, and see how that impacts the result if uncertainty in the models are present.

Moreover, if the predictor is calculated using an initial trajectory ($u_{\text{init}}$), it should be investigated how to keep the path feasible as we progress in time. This could potentially also include the decision of opening/closing wells.

# 5 | Publications

# A Data-driven derivative-free trust-region model-based method for resource allocation problems

**Andersen, J. R.**, Imsland, L., "Data-driven derivative-free trust-region model-based method for resource allocation problems". In: *Computers & Chemical Engineering* (2023). (Submitted)

## CRediT authorship contribution statement

**Joakim Rostrup Andersen:** Conceptualization, Methodology, Software, Validation, Writing - Original Draft, Visualization.
**Lars Imsland:** Conceptualization, Writing - Review & Editing, Supervision.

# Data-driven derivative-free trust-region model-based method for resource allocation problems

Joakim Rostrup Andersen[1] and Lars Imsland[1]

[1]Department of Engineering Cybernetics, NTNU, Trondheim, Norway

**Abstract:** Allocating a limited available resource between a set of units is a problem that arises in several application areas. We propose an online derivative-free trust-region model-based method to tackle a fairly general version of the resource allocation problem where units may be turned on or off. The units are considered as black boxes which may only be evaluated given that all the other units are evaluated simultaneously, and no gradient information is available. This method was inspired by an industrial problem and emphasis is put on both providing feasible points during the optimization and on not incurring additional increase in cost while searching for the optimum. The latter cannot be guaranteed, but the algorithm allows for automatic or manual ranking of the different units to attempt to reduce negative impact on the cost. The algorithm was applied to a case study from the petroleum industry where fast convergence was observed.

## 1 Introduction

The resource allocation problem is concerned with optimally allocating a limited available resource between a set of units to minimize (or maximize) a cost (or reward) function. This type of problem arises in different application areas such as queuing control, computer resource allocation, apportionment, load distribution, portfolio selection and production planning (Katoh, Shioura, and Ibaraki, 2013). Several different variations of the resource allocation problem exists. The formulation studied in this work is mathematically formulated as:

$$\min_{\mathbf{u},\mathbf{z}} \quad f(\mathbf{u}) = \sum_{i=1}^{n_u} f_i(u_i) \tag{1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_u} u_i = u_{\max} \tag{1b}$$

$$z_i \underline{u}_i \leq u_i \leq z_i \overline{u}_i \tag{1c}$$

$$u_i \in \mathbb{R} \tag{1d}$$

$$z_i \in \{0, 1\} \tag{1e}$$

where $u_{\max}$ is the available resource, $u_i$ is the amount of resource allocated to unit $i$, and $f_i(u_i)$ is the cost of allocating $u_i$ to unit $i$. Bold notation is used for vectors: $\mathbf{u} = [u_1, u_2, \ldots, u_{n_u}]^\top$ and $\mathbf{z} = [z_1, z_2, \ldots, z_{n_u}]^\top$. If all $z_i = 1$, then this would be a standard problem formulation, as found in Katoh, Shioura, and Ibaraki (2013). The binary variables allow for turning on and off units if necessary to achieve a better cost. Throughout this work, we assume that the $f_i$'s are convex, and thus also $f$. Should the $f_i$'s be non-convex, we rely on the generally accepted, though not proved, statement that derivative-free methods typically have the ability to find good local minima due to their relative crudeness, see Conn, Scheinberg, and Vicente (2009). Moreover, we assume the following characterizes $f_i$:

i) the $f_i$'s are unknown functions, and can only be evaluated and no gradient information is available,

ii) $f_j$ can only be evaluated if all the other $f_i$'s are evaluated simultaneously,

iii) all the constraints must be satisfied to perform an evaluation, and

iv) evaluating $f$ is costly.

The reason for the latter, in addition to the actual cost of measurement, could be that evaluating $f$ involves a change of operation point with potential loss of revenue, and also, if there are underlying dynamics, that reaching a new steady-state after changing the allocation may be a slow process. The term *true function* is used to refer to the unknown cost function $f(\mathbf{u})$. In this work, the resource allocation problem is formulated as a minimization problem where the objective function is a metric of the cost of allocating the resource. If the true function is the revenue of allocating the resources instead of the cost, the objective function should simply be multiplied by $-1$ to achieve a maximization problem.

We propose a customized derivative-free trust-region model-based method to solve the problem (1) taking the characteristics i)-iii) into account. In addition, consideration iv) is dealt with by introducing logic to try to reduce the negative impact on the cost.

In derivative-free trust-region model-based optimization it is, typically, assumed that the to-be-optimized function, or *true function*, is unknown. This function can be seen as a black box as it may only be evaluated, and no gradient information is available. A standard approach in these methods is to build a first or second order polynomial model of the true function using the gathered evaluations. This *surrogate model* is trusted within some *trust-region* typically defined by a *center point* and a *trust-region radius*. The surrogate model of the cost is minimized within the trust-region, and the true function is evaluated at this point. Assuming that the prediction of the model is good, the point is accepted as the new center point and the process is repeated. If the model prediction is bad, steps must be taken to improve the surrogate model.

The most distinct feature of the proposed algorithm is that it has as many trust-region radii and models as there are units. Each unit is treated individually when it comes

to modeling. Furthermore, the algorithm only provides feasible point with respect to the constraints of the problem (1).

To the best of the authors' knowledge, this is the first work to explicitly tackle the resource allocation problem with a (tailored) derivative-free trust-region model-based method. In addition, no previous literature on applying a derivative-free trust-region method on resource allocation problems, nor on such problems with units being considered black boxes, were found. Of course, there is a significant body of literature on more general black box optimization formulations, including Conn, Scheinberg, and Vicente (2009), Powell (1994), Powell (2009), Bajaj, Iyer, and Faruque Hasan (2018), and Bajaj and Hasan (2019).

The proposed algorithm is applied to an example coming from the oil and gas industry. The goal is to distribute the available lift gas (the resource) between different wells (the units) to maximize the oil production (the revenue). The example is detailed in Section 4. This problem has been studied in several papers, see *e.g.*, Krishnamoorthy, Foss, and Skogestad (2016), Krishnamoorthy, Pavlov, and Li (2016), Peixoto et al. (2015), Rashid (2010), and Rashid, Bailey, and Couet (2012) and the references therein. Differently from these, we focus on the setup where there is no available model of the relationship between the allocated resources and the resulting costs. *I.e.*, the wells, or units, may be viewed as black boxes. Further, only steady-states are considered and no interactions between the wells (through the reservoir) are considered. Moreover, the decision of closing and opening wells are included. The resulting algorithm is a data-driven optimization method.

The task of distributing available lift gas to optimize oil production have been previously tackled by derivative-free trust-region model-based methods (Giuliani, Camponogara, and Plucenio, 2013; Giuliani and Camponogara, 2015b; Giuliani and Camponogara, 2015a). These papers consider a similar setup to (1), *i.e.*, the minimization of a function subject to linear constraints, except for Giuliani and Camponogara (2015a) which in addition handles nonlinear constraints. Further, they are not considering the option of measuring the output of each unit independently. Problem structure is exploited in Giuliani and Camponogara (2015b) to reduce the amount of required points to build a quadratic model. Our proposed method differs from the previous approaches (Giuliani, Camponogara, and Plucenio, 2013; Giuliani and Camponogara, 2015b; Giuliani and Camponogara, 2015a) in several aspects. First, we include the decision of turning on and off units. Second, the method will only provide feasible points to evaluate. The constraints are satisfied both when finding a new potential best point, and when points are found to improve the geometry to construct new models. Third, the assumed availability of the individual output of each unit allows for further exploitation of problem structure. The latter is the reason why the suggested approach has as many trust-region radii and models as there are units.

In the context of the gas lift optimization problem, we propose a decomposition of the gas lift performance curve commonly used in the distribution of the available lift gas. The decomposition allows for efficient exploitation of the available data. To the

best of the authors' knowledge, this decomposition has not previously been used to exploit data in such a manner.

This paper is structured as follows. In Section 2, the algorithm is developed and detailed. In Section 3, an extension to the algorithm is presented that exploits unused degrees of freedom to attempt reducing negative impact on the cost. In Section 4, the motivational example is detailed, and the algorithm is applied to a small example. Furthermore, the section contains details on the proposed decomposition. In Section 5, several aspects of the algorithm is investigated by using different parameters of the algorithm and different instantiations of the motivational example. In Section 6, a discussion on the algorithm is provided. Finally, in Section 7 a conclusion is provided.

# 2 Theory

In this section, the proposed algorithm will be presented. It starts by introducing all the different parts of the algorithm before they are assembled into a complete algorithm at the end. The different buildings blocks of the proposed derivative-free trust-region model-based method are inspired by the framework presented in Conn, Scheinberg, and Vicente (2009). The method differs from this framework in several ways, which will be highlighted and discussed as each building block is presented.

## 2.1 Surrogate modeling

The suggested algorithm was tailored for a setup where the quality of the input-output relationships of the units were prone to measurement errors and measurements are few. In a noise free environment, quadratic surrogate models would typically allow for faster convergence. However, we use linear models to mitigate the impact of the noise. Furthermore, the typical samples would be rather close to each other making the extrapolation capabilities of a higher order polynomial inferior. Each unit $i$ will have its own linear surrogate model $F_i$ of the true input-to-cost relationship $f_i$

$$F_i(u_i) = a_i u_i + b_i \tag{2}$$

where the parameters $a_i$ and $b_i$ are chosen such that $F_i$ either match the collected evaluated points, or minimizes the squared prediction error at these points if regression is used.

It is important to take care of the spread, or geometry, of the points that are used for finding the parameters $a_i$ and $b_i$. For example, say there are two points used for creating the linear model. First, if the two points lie on top of each other, any line through the point would be a perfect interpolation, but it is clearly of no use. Second, assume there is a trust-region radius of $10^{10}$ centered at the origin. Two points located at $u_i = 0$ and $u_i = 1$ will have a very small chance of capturing any valuable information of the true function over the entire trust-region, unless it

happens to be linear and noise free. With these two examples in mind, the following geometry requirement rules are proposed.

### 2.1.1 Geometry requirement - Interpolation

Let $u_{i,\text{cp}}$ be the current center point of the model for unit $i$, and let $\Delta_i$ denote its trust-region radius. The set $\mathbf{U}_i = \{u_{i,\text{cp}}, u_{i,1}\}$, of length $n_i$, is deemed poised (or suitable) for interpolation if:

i) At least one $j \in [1, \ldots, n_i - 1]$ satisfies

$$|u_{i,\text{cp}} - u_{i,j}| \geq \frac{1}{2}\Delta_i \tag{3}$$

ii) All $j \in [1, \ldots, n_i - 1]$ satisfies

$$|u_{i,\text{cp}} - u_{i,j}| \leq r\Delta_i \tag{4}$$

iii) All $j \in [0, \ldots, n_i - 1]$ satisfies

$$\underline{u}_i \leq u_{i,j} \leq \bar{u}_i \tag{5}$$

The $r \geq 1$ in (4) is a scaling factor which allows for more reuse of points and is a trick borrowed from Conn, Scheinberg, and Vicente (2009). Condition i) guarantees spread of the points, condition ii) guarantees the points are within the (possibly extended) trust-region, and iii) guarantees the points used for modelling is feasible with respect to the bounds of the unit.

### 2.1.2 Geometry requirement - Regression

In the case when the cardinality of $\mathcal{U}_i$ is greater than two, $n_i > 2$, the model making process may be done through least squares regression instead of interpolation. Let $\mathcal{U}_{i,\text{all}}$ be the set of (all) previously evaluated points. Further, let $\mathcal{U}_i$ be the subset of $\mathcal{U}_{i,\text{all}}$ which contains all the points that satisfies conditions ii)-iii). Next, the $\mathcal{U}_i$ is split into two sets: $\mathcal{U}_i^0 = \{u_{i,j} \in U_i : |u_{i,\text{cp}} - u_{i,j}| < \frac{1}{2}\Delta_i\}$ and $\mathcal{U}_i^1 = \{u_{i,j} \in U_i : |u_{i,\text{cp}} - u_{i,j}| \geq \frac{1}{2}\Delta_i\}$. If $\mathcal{U}_i^1$ is a non-empty set, $\mathcal{U}_i$ is deemed poised (or suitable) for regression.

This classification of poisedness for regression is aligned with the one in Conn, Scheinberg, and Vicente (2009): A set of bounded (and moderate) amount of points will be poised for regression if a subset is poised for interpolation. It may have a worse "poisedness", but only by a scaling factor.

### 2.1.3 Filtering

If there is a large imbalance in the cardinalities, the smallest may end up having close to no influence. A solution could be to set a cap on the cardinality of the sets $\mathcal{U}_i^0$ and $\mathcal{U}_i^1$. That is, instead of adding all points satisfying the aforementioned requirement when making $\mathcal{U}_i^0$ and $\mathcal{U}_i^1$, only select up to the $n_{\max}$ newest.

To avoid that restricting the cardinality results in only identical points, which could be the case if the unit is not altered for several iterations, a filtering method should be applied. If the applied filtering method is based on merging the new point with older points, then care must be taken to avoid unwanted effects of the merging. E.g., if a new point is found to satisfy Conditions i) - iii), we must ensure that the merged point also satisfies the criteria.

### 2.1.4 Geometry requirement for units turned off

The optimization problem that should be solved in (1) includes the binary variables to turn on and off units. To respect consideration iv), evaluating $f$ is costly, the geometry requirement is not imposed for units which are off at the current operation point.

## 2.2 Geometry improvement

So far, it has been discussed requirements on the sets of points used for interpolation (and regression). If these requirements are not satisfied, the algorithm must prioritize creating such sets of points before continuing looking for the optimum.

Finding the required set of new $u_i$'s to evaluate is not a trivial task due to the interconnection between the units given by the availability of the shared resource (1b). *E.g.*, if one unit needs more resource, one or more of the other units must be allocated less. However, these units have their own lower bounds on the resource requirement (1c). An idea could be to always decrease the use of resource, as then no resource needs to be "borrowed" from another unit. However, the unused resource must be allocated somewhere and the issue of where to route it without hitting the upper bounds arises.

Some terminology and notation will be introduced next. A unit that does not need a new point to pass the geometry requirement is referred to as a *neutral unit* and the index $k$ will be used for these units. Similarly for a unit that needs a point, we will use the term *lacking unit* and index $j$. Finally, index $i$ is used when it concerns all the units regardless of the geometry requirement.

The new point to improve geometry will be found by solving an optimization problem. Finding a point $u_j$ that satisfies the conditions i) - ii) for a single unit may be

formulated as three constraints using the two binary variables $x_{j,l}$ and $x_{j,r}$:

$$u_j \geq (u_{j,\text{cp}} - \Delta_j)x_{j,l} + (u_{j,\text{cp}} + 0.5\Delta_j + \Delta_=)x_{j,r} \tag{6a}$$
$$u_j \leq (u_{j,\text{cp}} - 0.5\Delta_j - \Delta_=)x_{j,l} + (u_{j,\text{cp}} + \Delta_j)x_{j,r} \tag{6b}$$
$$x_{j,l} + x_{j,r} = z_j \tag{6c}$$

The subscript $l$ and $r$ refer to "left" and "right", respectively. The $z_j$ will be set to one for these units, but is included to keep the notation similar to what will be presented later. Notice that the $\Delta_=$ is included to avoid that the new point will be merged back into the area $u_{j,\text{cp}} \pm 0.5\Delta_j$ if a filtering method based on merging is applied. The feasible areas for $u_j$ with $u_{j,\text{cp}} = 0$ is shown as the shaded gray areas in Fig. 1.

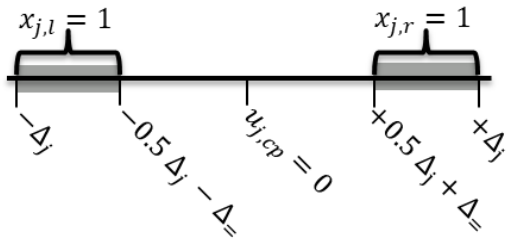


Figure 1: The feasible areas for the wells that needs new points are shown in gray.

In addition to the new constraints above, the constraint on the lower and upper limit (1c) must be imposed. To help the solver, the on/off variable $z_i$ may be manually set to 1 for the units that need a new geometry improving point.

It should perhaps be noted that finding a single point, a $\mathbf{u}$, that provides all the necessary points may be infeasible. Furthermore, it may be infeasible to find such a point even with only one lacking and one neutral unit. *E.g.*, assume there are two units: $100 \leq u_1 \leq 200$, and $2 \leq u_2 \leq 4$ with a current operating point of $u_1 = 100$ and $u_2 = 2$, and that unit 2 needs another point. Considering that unit 1 is also at its lower limit, we need to shut unit 1, but then there is too much resource to be distributed. Hence, this problem is infeasible by problem definition. This type of infeasibility is hereafter referred to as *problem infeasibility*. Nonetheless, this type of "problem infeasibility" was never encountered in the case study and it is less likely when the amount of units increases. Furthermore, it could be argued that (i) such input problems should be avoided by the user, or (ii) the algorithm could exit with the current operation point as the solution when it occurs.

The resulting optimization, or feasibility, problem for improving the geometry is a

Mixed-Integer Linear Program (MILP):

$$\min_{\mathbf{u},\mathbf{z},\mathbf{x}} \quad 0 \tag{7a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_u} u_i = u_{\max} \tag{7b}$$

$$u_j \geq (u_{j,\text{cp}} - \Delta_j)x_{j,l} + (u_{j,\text{cp}} + 0.5\Delta_j + \Delta_=)x_{j,r} \tag{7c}$$

$$u_j \leq (u_{j,\text{cp}} - 0.5\Delta_j - \Delta_=)x_{j,l} + (u_{j,\text{cp}} + \Delta_j)x_{j,r} \tag{7d}$$

$$x_{j,l} + x_{j,r} = z_j \tag{7e}$$

$$z_i \underline{u}_i \leq u_i \leq z_i \overline{u}_i \tag{7f}$$

$$u_i \in \mathbb{R} \tag{7g}$$

$$z_j = 1 \tag{7h}$$

$$z_i \in \{0, 1\} \tag{7i}$$

$$x_{j,l}, x_{j,r} \in \{0, 1\} \tag{7j}$$

As mentioned above, the task of finding a single point satisfying (7c) and (7d) for all the lacking units may be infeasible. Provided that the "problem infeasibility" is not the issue, several evaluations of the true function would be required. Let $\mathcal{J}$ be the set of all the $j$ indices, *i.e.*, it contains the indices of all the lacking units. One could, for example, solve (7) two times where the set $\mathcal{J}$ is split into two (equally large) sets $\mathcal{J} = \mathcal{J}_0 \cup \mathcal{J}_1$. If one (or both) fails, repeat the branching process until feasibility is reached in all branches. For each branching, the required amount of lacking units that receives a new point is decreased and, thus, the optimization problem becomes less constrained.

There may be several $\mathbf{u}$, $\mathbf{z}$ and $\mathbf{x}$ that satisfies the constraints in (7). In Section 3, these unused degrees of freedom will be exploited to try to minimize the increase in cost while performing a geometry improvement step.

## 2.3 Solving the subproblem

When all units have enough points to create a decent model, it is time to look for a new operation point that should further minimize $f$. This problem is referred to as the *subproblem* in trust-region literature. Let $F_i$ be the (linear) surrogate model of

$f_i$. The subproblem is given as

$$\mathbf{u}^+, \mathbf{z}^+ = \arg\min_{\mathbf{u},\mathbf{z}} \quad F(\mathbf{u}) = \sum_{i=1}^{n_u} F_i(u_i) \tag{8a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_u} u_i = u_{\max} \tag{8b}$$

$$u_i \geq z_i(u_{i,\text{cp}} - \Delta_i) \tag{8c}$$

$$u_i \leq z_i(u_{i,\text{cp}} + \Delta_i) \tag{8d}$$

$$z_i \underline{u}_i \leq u_i \leq z_i \overline{u}_i \tag{8e}$$

$$u_i \in \mathbb{R} \tag{8f}$$

$$z_i \in \{0, 1\} \tag{8g}$$

As opposed to the standard derivative-free trust-region methods, we may employ an off-the-shelf Mixed-Integer Linear Program (MILP) solver to solve the subproblem. A standard method for finding $\mathbf{u}^+$ is to only take a single step that provides sufficient decrease of the objective function, see Conn, Scheinberg, and Vicente (2009) for details. If the solution of the subproblem yields a better value of $f$, then it is used as the next operation point.

## 2.4 Trust-region radius update

Different from the standard approach in derivative-free trust-region methods, several surrogate models are in use. Each unit will have its own surrogate model and trust-region radius. Let $u_i^+$ be the point where the model prediction quality $\rho_i$ should be evaluated:

$$\rho_i = \frac{f_i(u_{i,\text{cp}}) - f_i(u_i^+)}{F_i(u_{i,\text{cp}}) - F_i(u_i^+)} \tag{9}$$

The trust-region radius for unit $i$ is updated according to:

$$\Delta_i \leftarrow \begin{cases} \Delta_i & \text{if } \rho_i \geq \eta_1 \\ \max(\gamma \Delta_i, \Delta_{i,\min}) & \text{else} \end{cases} \tag{10}$$

with $0 < \gamma < 1$. In words, if the model prediction quality is deemed good, the radius is kept fixed, otherwise it is reduced.

The functionality for turning on/off units and using a surrogate model for each unit introduces some additional scenarios that must be dealt with. If either,

- the unit is off at the next point (*i.e.*, $z_i^+ = 0$), or
- the unit has never been on, or

- the resource allocation remains the same for the unit, *i.e.*, $u_{i,\text{cp}} = u_i^+$

then do not calculate $\rho_i$ and skip updating the trust-region radius.

Finally, it should perhaps be noted that despite the sets of points used to create the $F_i$'s are all deemed poised, the prediction quality may still be bad. The poisedness only guarantee a certain amount of spread in the points. The aim is that the spread should help getting points that capture valuable information that can be used for modeling. However, there is no (strong) guarantee on the resulting prediction quality.

## 2.5 Acceptance of new point

A point $\mathbf{u}^+$ found while solving (8) will be accepted as the new operation point as long as:

$$\frac{f(\mathbf{u}_{\text{op}}) - f(\mathbf{u}^+)}{F(\mathbf{u}_{\text{op}}) - F(\mathbf{u}^+)} > 0 \tag{11}$$

Which means that as along as the new point has a lower true function value, it will be accepted as the new operation point.

Notice that the subscript "op" is used to refer to the operation point, and it should not be confused with the center point of the model. Each unit, will have its own model center point. If the unit is on, the center point will be the last solution of (8). If the unit is off, its center point will be the last solution of (8) when it was on, *i.e.*, its $z_i$ was 1. The term operation point is used for the point that is currently the best $\mathbf{u}^+$ point encountered. This means that $\mathbf{u}_{\text{op}}$ will always satisfy the constraints of (1), whereas $\mathbf{u}_{\text{cp}}$ may not. The individual $u_{i,\text{cp}}$ will satisfy the upper and lower input constraints for that unit.

## 2.6 Criticality step

Another building block for the trust-region method is the *criticality step*. It ensures a relationship between the size of the trust-region radius and a measurement of stationarity (Conn, Scheinberg, and Vicente, 2009). Its task is to make the models more accurate when this measure is close to zero. This will drive the trust-region radius towards zero.

The criticality step presented in Conn, Scheinberg, and Vicente (2009) is intended for the minimization of a function without any constraints. In the unconstrained case, the measure of stationarity is the gradient of the objective function. For the constrained case, we should use the gradient of the Lagrangian instead:

$$\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u},\mathbf{z},\lambda,\boldsymbol{\mu}) = \begin{bmatrix} \nabla_{u_1}\mathcal{L} \\ \nabla_{u_2}\mathcal{L} \\ \vdots \\ \nabla_{u_{n_u}}\mathcal{L} \end{bmatrix} \tag{12}$$

$$\nabla_{u_i}\mathcal{L} = \nabla_{u_i}F(\mathbf{u}) + \lambda - \underline{\mu}_i + \overline{\mu}_i \tag{13}$$

where

$$\mathcal{L}(\mathbf{u},\mathbf{z},\lambda,\boldsymbol{\mu}) = F(\mathbf{u}) + \lambda\left(\sum_{i=1}^{n_u} u_i - u_{\max}\right) + \sum_{i=1}^{n_u}\left[\underline{\mu}_i(z_i\underline{u}_i - u_i) + \overline{\mu}_i(u_i - z_i\overline{u}_i)\right]$$

Notice that the constraints limiting the search area to the trust-region (8c) and (8d) are not included. If they were, the gradient of the Lagrangian would simply be zero each time at the optimum of (8).

Using the Lagrangian instead of the objective function introduces several issues which will be discussed next.

The first issue will be illustrated by a small example. Consider that there are two units, and that the current operation point is far away from the lower and upper bounds of each unit. Then we have

$$\nabla_{u_1}\mathcal{L} = \nabla_{u_1}F_1(\mathbf{u}) + \lambda$$
$$\nabla_{u_2}\mathcal{L} = \nabla_{u_2}F_2(\mathbf{u}) + \lambda$$

where the bounds are left out for simplicity. The $\lambda$ will receive its value based upon the two equations above. Further, assume that the model prediction quality has been found to be good for one unit ($\rho_1 = 1$) and bad for the other ($\rho_2 = 10^{-4}$). As the underlying functions $f_i$'s are assumed to be convex, it is fair to say that a low prediction quality indicates a bad gradient prediction too. If one of the two gradients $\nabla_{u_1}F_1(\mathbf{u})$ and $\nabla_{u_2}F_2(\mathbf{u})$ are incorrect, then the quality, or usefulness, of all the elements of the gradient of the Lagrangian is at stake as they are all intertwined through the $\lambda$ multiplier. To deal with this issue, the gradient of the Lagrangian is only evaluated whenever $\rho_i \geq \eta_1$ for all the considered units (as explained in Section 2.4).

Another issue when going from the gradient of objective function to the gradient of the Lagrangian is where to evaluate the gradient. In Conn, Scheinberg, and Vicente (2009), the gradient of the model is evaluated at the center point of the model, or the operation point in our setting. The Lagrange multipliers in (13) complicate this choice. The multipliers that satisfies the First Order Necessary Conditions (FONC) of optimality (Nocedal and Wright, 2006) are only available at a local optimum. However, the current operation point will rarely (or never) be such a point. Once the

operation points for the unit models are moved, so will (most likely) the solution of (8), even though none of the $F_i$'s are updated. The reason for the change is due to the application of linear models combined with that the operation point is moved, and thus also the trust-regions. As a consequence, finding the gradient of the Lagrangian at the current operation point is (close to) impossible.

If the norm of the gradient of the Lagrangian was evaluated at the solution of (8) before the operation point is moved, then this point satisfies the FONC, and the multipliers are available. Despite this not being the gradient at the operation point, it will be used as the measure of stationarity. If the norm of this gradient is small, it indicates that the gradient's elements are close to zero without considering the trust-region bounds. And in this case, we are indeed close to a stationary point (of the model) and a reduction of the trust-region radii are in-order to "zoom in" on the true function.

Another complicating factor of implementing the criticality step for the constrained case is a result of the previous issues. The criticality step in Conn, Scheinberg, and Vicente (2009) is an iterative procedure which terminates when there is a satisfactory relationship between the norm of the gradient and the trust-region radius: $\Delta \leq \mu_c||g||$, where $g$ is the model gradient and $\mu_c$ is a tuning parameter. This iterative procedure will be exited as long as the norm of the gradient of the true function at the center point is not zero. Roughly explained, each iteration of the criticality step consists of (i) reducing the radius ($\tilde{\Delta} \leftarrow \omega_c\tilde{\Delta}, \omega_c \in (0,1)$), (ii) making sure the (new) set of points are poised in the new region, (iii) creating the new surrogate model with gradient $\tilde{g}$, and finally (iv) check if its gradient is satisfactory. When the procedure exits, the final radius is chosen as

$$\Delta = \min(\max(\tilde{\Delta}, \beta_c||\tilde{g}||), \Delta_0) \tag{14}$$

where $\Delta_0$ is the radius when entering the criticality step. The constants must satisfy $\mu_c > \beta_c > 0$. The final radius $\Delta$ of the criticality step is selected as the radius in the range $[\tilde{\Delta}, \Delta_0]$ closest to $\beta_c||\tilde{g}||$, which helps avoiding that the radius is reduced too much.

Such an iterative procedure is not feasible for our setup due to the first issues. The infeasibility will be illustrated through an example. Assume that the radius was reduced because the gradient was found too small compared to the radius: $\tilde{\Delta} \leftarrow \omega_c\tilde{\Delta}, \omega_c \in (0,1)$. Next step would be to create (potentially new) sets of points that are deemed poised. Further, the surrogate models are made, the subproblem is solved and the true function is evaluated at the solution. Now, if the prediction quality was not satisfied for all the considered units, then the gradient of the Lagrangian could not be evaluated. Without this gradient, the criticality step cannot continue, and thus the iterative setup of the criticality step in Conn, Scheinberg, and Vicente (2009) is not feasible.

To deal with this, we suggest a single pass approach which tries to copy the most important feature of the original criticality step in Conn, Scheinberg, and Vicente

(2009). More specifically, the radii are reduced if at least one of the elements of the gradient is too small compared to the corresponding radius. The proposed criticality step consists of only one pass, and is hereafter referred to it as the criticality *substep*. The substep is given in Algorithm 1.

This criticality substep will result in the radii (potentially) decreasing faster towards zero, and maybe even too fast. As opposed to the trust-region selection in (14), we simply get $\Delta = \tilde{\Delta} = \omega_c \Delta_0$ which would correspond to setting $\beta_c = 0$. As alluded to in (10), we have introduced a $\Delta_{\min}$, which also will be used in the criticality step, see Algorithm 1. This hard limit will avoid that the algorithm gets stuck due to the trust-region radii going too fast to zero.

**Algorithm 1** Criticality substep

**Parameters:** $\mu_c > 0$, $0 < \omega_c < 1$, $\boldsymbol{\Delta}_{\min}$

**procedure** CRITICALITY($\boldsymbol{\Delta}$, $\mathcal{U}$, $\mathbf{z}_{\text{op}}$, $\mathbf{u}^+$, $\mathbf{z}^+$, $\lambda^+$, $\underline{\boldsymbol{\mu}}^+$, $\overline{\boldsymbol{\mu}}^+$)
  **Require:** All $U_i$'s are deemed poised for interpolation (or regression).
  Calculate $\nabla_{\mathbf{u}} \mathcal{L}$ in (12) at the solution.
  reduce = False
  **for** $i = 1, 2, \ldots, n_u$ **do**
    **if** $z_i^+ == 1$ and $\mu_c \left| \nabla_{\mathbf{u}_i} \mathcal{L} \right| < \Delta_i$ **then**
      reduce = True
    **end if**
  **end for**
  **if** reduce **then**
    **for** $i = 1, 2, \ldots, n_u$ **do**
      **if** $z_i^+ == 1$ and $\mu_c \left| \nabla_{\mathbf{u}_i} \mathcal{L} \right| < \Delta_i$ **then**
        $\Delta_i \leftarrow \max(\omega_c \Delta_i, \Delta_{i,\min})$
      **end if**
    **end for**
  **end if**
  **return** $\boldsymbol{\Delta}$
**end procedure**

The setup of having several units, trust-region radii and models allows for some design choices. Specifically, if one element of the gradient is zero, the corresponding unit's radius or all radii could be decreased. The latter has shown to result in better performance. This is further illustrated and discussed in Section 5.2. This choice is not without its challenges. The more aggressive reduction of the radii may indeed hinder fast convergence. However, this may also be the case for either approach, and will be both problem specific and rely on the chosen parameters. For example, starting with small trust-region radii, and/or having strong contraction parameters (*i.e.*, $\omega_c \approx 0$ and $\gamma \approx 0$), will all result in the radii hitting the $\Delta_{\min}$ quickly.

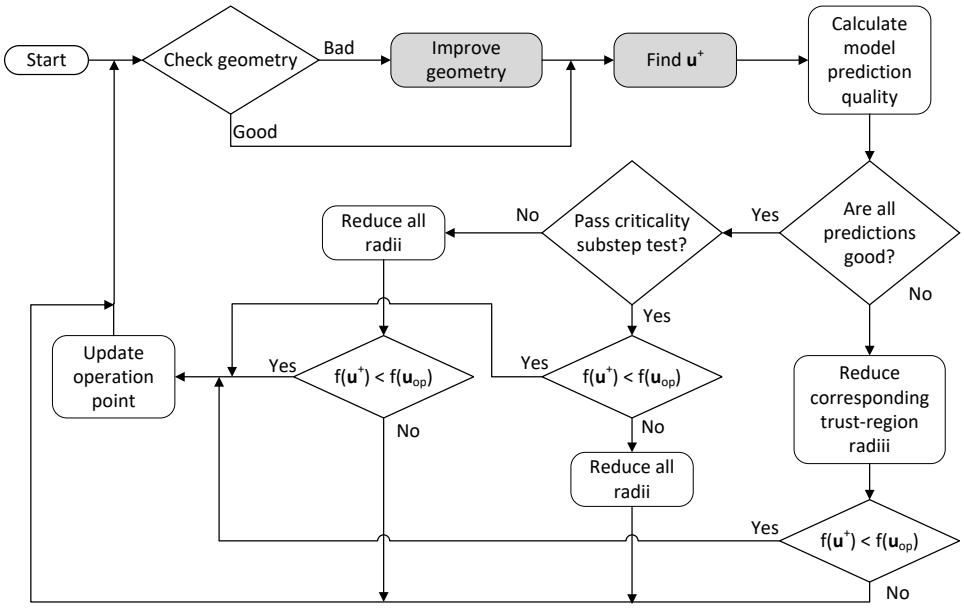


Figure 2: A visual flowchart for Algorithm 2. The gray boxes are where the true function is evaluated.

## 2.7 The complete algorithm

All the required building blocks of the algorithm have been detailed in the previous sections. The only extension that is missing is how to exploit the unused degrees of freedom while finding geometry improvement points. The presentation is postponed to after this section. The complete derivative-free trust-region model-based approach is given in Algorithm 2. A flowchart of the algorithm is shown in Fig.2.

The algorithm is inspired by the framework presented in Conn, Scheinberg, and Vicente (2009), and specifically Algorithm 10.1. They differ in the following aspects.

- First, the way poisedness is checked and improved is different. In this paper, emphasis is put on not disturbing the to-be-optimized process more than necessary when poisedness needs to be improved, see Section 3. Such considerations are not made in Conn, Scheinberg, and Vicente (2009).
- Second, the constrained optimization problem (1) introduces the need of looking at the gradient of the Lagrangian instead of the objective function in the criticality step.
- Third, the integer variables $z_i$'s introduce additional design choices.
- Fourth, each unit is modelled individually with its own trust-region radius,

whereas they use only one model.

- Fifth, due to the poisedness being checked and improved in the beginning of the while loop in Algorithm 2, the remaining part of the algorithm concerning the poisedness property is simplified compared to Algorithm 10.1 in Conn, Scheinberg, and Vicente (2009).
- Sixth, all radii will be reduced if (i) the solution of the subproblem was not better than the current $\mathbf{u}_{\text{op}}$, and (ii) none of the radii has been reduced during the current iteration of the algorithm. This ensures that the radii will decrease towards $\Delta_{\min}$ whenever the algorithm may get stuck. More explicitly, if all sets are deemed poised, all model prediction qualities satisfy the criterion, and the norm of the gradient of the Lagrangian is sufficiently large, but the solution of the subproblem is worse (or equally good) as the one at $\mathbf{u}_{\text{op}}$, then all the radii are reduced.

Deciding a termination criterion that consistently provides both a good final solution and avoiding unnecessarily many perturbation is not straightforward. The easiest criterion is to stop after a certain amount of true function evaluations. This could be a valid choice if the user decides that a certain amount of time should be used for improving the performance. Then by knowing how long it takes to perform one true function evaluation, one may calculate how many evaluations are available. Another option could be to track the relative improvement of the solutions of (8). Then, once the trust-region radii (for the considered units) are all at their corresponding minima, one could terminate once the experienced improvement is below some threshold.

# 3 Geometry improvement: Unused degrees of freedom

The optimization formulation in (7), assuming it is feasible, provides a geometry improvement point for all the units that require such a point. Depending upon how many units need a new point, there might be several degrees of freedom left in the optimization. Put differently, there are may be many different $\mathbf{u}$ vectors that will satisfy the geometry requirement. This freedom will be exploited next.

We emphasize that nothing of what is added from here on out will actually restrict the feasible area. In other words, if (7) is feasible, so is the to-be-presented problem. The extensions suggested next could be altered to fit the specific application.

## 3.1 Minimize change

When performing a geometry improvement step, it could be tempting to keep the objective function as in (1a), but where the $f_i$'s were replaced by their linearized

**Algorithm 2** Derivative-free trust-region optimization method for resource allocation problems

**Input:** Operation point $(\mathbf{u}_{\text{op}}, \mathbf{z}_{\text{op}})$, and center-point $\mathbf{u}_{\text{cp}}$. Trust-region radii parameters $\boldsymbol{\Delta}$, $\boldsymbol{\Delta}_{\min}$, and $r$. Trust-region radii update parameters $0 < \gamma < 1$ and $0 < \eta_1 < 1$. Criticality step parameters $\mu_c > 0$ and $0 < \omega_c < 1$

**Require:** $\mathbf{u}_{\text{op}}$ is feasible

**Require:** Either all units have a model, or all units are on $z_i = 1$

**while** not converged **do**
  $\boldsymbol{\Delta}_p \leftarrow \boldsymbol{\Delta}$
  Check if all $\mathcal{U}_i$'s are deemed poised according to definition in Section 2.1.1 (or 2.1.2)
  If any units need more points, solve *e.g.*, (7) or (32), and evaluate the true function
  Find $(\mathbf{u}^+, \mathbf{z}^+, \lambda^+, \underline{\boldsymbol{\mu}}^+, \overline{\boldsymbol{\mu}}^+)$ by solving the subproblem (8)
  Evaluate the true function
  Calculate all considered units' $\rho_i$'s in (9)
  **if** All calculated $\rho_i$'s $\geq \eta_1$ **then**
    $\boldsymbol{\Delta}$ = Criticality($\boldsymbol{\Delta}$, $\mathcal{U}$, $\mathbf{z}_{\text{op}}$, $\mathbf{u}^+$, $\mathbf{z}^+$, $\lambda^+$, $\underline{\boldsymbol{\mu}}^+$, $\overline{\boldsymbol{\mu}}^+$)
  **else**
    **for** $i = 1, 2, \ldots, n_u$ **do**
      **if** $\rho_i < \eta_1$ **then**
        $\Delta_i \leftarrow \max(\gamma \Delta_i, \Delta_{i,\min})$
      **end if**
    **end for**
  **end if**
  **if** $f(\mathbf{u}^+) < f(\mathbf{u}_{\text{op}})$ **then**
    $\mathbf{u}_{\text{op}} \leftarrow \mathbf{u}^+$
    $\mathbf{z}_{\text{op}} \leftarrow \mathbf{z}^+$
  **else**
    **if** $\boldsymbol{\Delta} == \boldsymbol{\Delta}_p$ **then**
      **for** $i = 1, 2, \ldots, n_u$ **do**
        $\Delta_i \leftarrow \max(\gamma \Delta_i, \Delta_{i,\min})$
      **end for**
    **end if**
  **end if**
  Add $\mathbf{u}^+$ to the set $\mathcal{U}_{\text{all}}$.
**end while**

counterpart. This way, the true function would be (attempted) minimized while obtaining the required points. However, it is important to remember that the reason new points are required is because the unit models are not to be trusted.

Before presenting the suggested objective function, we highlight the difference between operation point and center point. The operation point and the center point for a unit will be the same if the unit is on. If a unit is off, its operation point is 0, but its center point will be kept as the last non-zero operation point.

We suggest to minimize the normalized squared distance from the operation point:

$$P_u \sum_{i \in \mathcal{Z}_{\text{on}}} \left( \frac{u_i - u_{i,\text{op}}}{\overline{u_i} - \underline{u_i}} \right)^2 \tag{15}$$

where $\mathcal{Z}_{\text{on}}$ is the set of indices of the units that are on at the current operation point, and $P_u$ is a scaling parameter used for prioritizing different objectives later on. A squared distance would be indifferent to $u_i$ increasing or decreasing. However, there could be available information that would give preference to either increase or decrease.

## 3.2 Priority to increase/decrease

The following extension allows for prioritized exploration directions for both the lacking units and the neutral units. Let $k$ be an index of a neutral unit, *i.e.*, one that does not need a new point. The binary variable $x_{k,l}$ will be 1 if $u_k \leq u_{k,\text{cp}}$ and $x_{k,r} = 1$ if $u_k \geq u_{k,\text{cp}}$. This is similar to the use of the $x_{j,l}$'s and $x_{j,r}$'s in (7). The constraints below must be added to (7)

$$u_k \geq u_{k,\text{cp}} x_{k,r} + \underline{u}_k x_{k,l} \tag{16a}$$
$$u_k \leq u_{k,\text{cp}} x_{k,l} + \overline{u}_k x_{k,r} \tag{16b}$$
$$z_k = x_{k,l} + x_{k,r} \tag{16c}$$

In addition, the objective function should be extended with:

$$P_{l,r} \sum_{i=1}^{n_u} x_{i,l} \cdot v_{i,l} + x_{i,r} \cdot v_{i,r} \tag{17}$$

where the $v_{i,l}$'s and $v_{i,r}$'s are the scalars giving priority to changing $u_i$ in either direction:

$$v_{i,l} < 0 \text{ and } v_{i,r} = 0 \text{ if prioritize to decrease } u_i \tag{18}$$
$$v_{i,l} = 0 \text{ and } v_{i,r} = 0 \text{ if no preference} \tag{19}$$
$$v_{i,l} = 0 \text{ and } v_{i,r} < 0 \text{ if prioritize to increase } u_i \tag{20}$$

The $v_{i,l}$'s and $v_{i,r}$'s for the neutral units could for example be chosen as:

$$v_{k,l} = -1, v_{k,r} = 0 \quad \text{if } |a_k| \leq 10^{-4} \text{ or } a_k > 0 \tag{21a}$$
$$v_{k,l} = 0, v_{k,r} = -1 \quad \text{else} \tag{21b}$$

where $a_k$ refers to the slope parameter in the linear model equation (2). For the lacking units, it is less obvious how to choose $v_{j,l}$'s and $v_{j,r}$'s. If no knowledge is available, set them to 0.

With these two extensions, it will be prioritized to stay as close as possible to the operation point and the exploration directions can be prioritized. Next up, an extension to prioritize which neutral unit to pertub will be presented.

### 3.3 Prioritize which neutral unit to alter

When some units need additional points to make a good linear model, other units must be perturbed too. Ideally, the resource should be distributed only between the units that need changes, but such redistribution of the resource may not always be sufficient. There may be some units that one consider as prioritized as they have an low associated cost, and these should not be altered unless it is necessary. This mindset is perhaps more intuitive in the maximization setting where one would not like to alter units that gives a high revenue.

A new set of binary variables will be used to impose this prioritization. Let $w_i$ be a binary variable to indicate if a unit $i$ is allowed ($w_i = 1$) to be altered or not ($w_i = 0$).

$$u_i \leq (z_i - w_i)u_{i,\text{op}} + w_i\overline{u}_i \tag{22}$$

$$u_i \geq (z_i - w_i)u_{i,\text{op}} + w_i\underline{u}_i \tag{23}$$

$$w_i \leq z_i \tag{24}$$

with $i \in \mathcal{Z}_{\text{on}}$, and where subindex "op" refers to the current operation point. The equations above will only be imposed for units that are currently on ($z_{i,\text{op}} = 1$). Turning off a unit ($z_i = 0$) does not count as altering, and such changes will be handles later.

The objective function then needs to be extended with

$$P_w \sum_{i \in \mathcal{Z}_{\text{on}}} w_i W_i \tag{25}$$

and $\mathbf{W} = [W_1, W_2, \ldots, W_{n_u}]^\top$ is a vector with non-negative elements and contains the prioritization weight for perturbing the units. A lower value means it is more acceptable to perturb it. A value of 0 should be given to the lacking units.

Deciding the remaining $W_i$'s is up to the user of the method. It could be based on application/engineering knowledge, or some kind of measurement of "importance". One idea could be to take an inverted average over all previously gathered points ($U_{i,\text{all}}$), including those outside the trust-region for the neutral units:

$$W_{i,\text{avg}} = \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{1}{\mathcal{F}_{i,\text{all}}[j]} \tag{26}$$

where $\mathcal{F}_{i,\text{all}}$ is an ordered set containing the measured value $f_i(u_{i,j})$ for each element $u_{i,j} \in \mathcal{U}_{i,\text{all}}$. Further, the $\mathbf{W}_{i,\text{avg}}$ should be shifted and scaled to obtain non-negative elements:

$$W_{i,\text{shifted}} \leftarrow W_{i,\text{avg}} + |\min(0, \mathbf{W}_{\text{avg}})| + 1 \tag{27}$$

$$W_i \leftarrow \frac{W_{i,\text{shifted}}}{\max(\mathbf{W}_{\text{shifted}}) - \min(\mathbf{W}_{\text{shifted}})} \tag{28}$$

where the +1 is added to ensure that the weight for the neutral units is higher than 0, $W_k > 0$, and thus more costly to alter than the lacking units ($W_j = 0$).

One last set of constraints is added to (7) to obtain the desired prioritized altering. Let $\mathcal{P}$ be an ordered set of $n_u$ indices of the units ordered according to an ascending importance. *I.e.*, it is most acceptable to perturb the unit whose index is found at first element of $\mathcal{P}$. Further, let $\mathcal{M}$ be a one-to-one mapping from unit $i$ to its corresponding index in $\mathcal{P}$. The inverse mapping (from priority index to unit index) is given as $\mathcal{M}^{-1}$

$$w_{\mathcal{M}^{-1}(n_u)} \leq w_{\mathcal{M}^{-1}(n_u-1)} \leq \ldots \leq w_{\mathcal{M}^{-1}(1)} \tag{29}$$

## 3.4 Prioritize to not change on/off status

Depending upon the application area, it may be costly and/or undesirable to turn on/off units simply to get geometry improvement points. To penalize closing a unit, the following can be added to the objective function:

$$P_z \sum_{i \in \mathcal{Z}_{\text{on}}} -z_i \tag{30}$$

For the units which are currently off, a similar penalization scheme can be used. However, a preferred turning on order can be imposed by adding the single extension:

$$\sum_{i \in \mathcal{Z}_{\text{off}}} (P_z + P_w(1 - W_i))z_i \tag{31}$$

where $(1 - W_i)$ is used because a value of $W_i$ closer to one indicates it is a unit which is believed to have a low associated cost.

### 3.5 Geometry improvement - summary

An optimization problem with all the aforementioned extensions is now presented.

$$
\begin{aligned}
\min_{\mathbf{u},\mathbf{z},\mathbf{x},\mathbf{w}} \quad & P_u \sum_{i\in\mathcal{Z}_{\text{on}}} \left(\frac{u_i - u_{i,\text{op}}}{\overline{u}_i - \underline{u}_i}\right)^2 + P_{l,r} \sum_{i=1}^{n_u} x_{i,l} v_{i,l} + x_{i,r} v_{i,r} \\
& + \sum_{i\in\mathcal{Z}_{\text{on}}} (P_w w_i W_i - P_z z_i) + \sum_{i\in\mathcal{Z}_{\text{off}}} (P_z + P_w(1 - W_i)) z_i && \text{(32a)} \\
\text{s.t.} \quad & \sum_{i=1}^{n_u} u_i = u_{\max} && \text{(32b)} \\
& \underline{u}_i z_i \le u_i \le \overline{u}_i z_i && \text{(32c)} \\
& u_j \ge \underline{u}_j z_j + (u_{j,\text{cp}} - \Delta_j - \underline{u}_j) x_{j,l} + (u_{j,\text{cp}} + 0.5\Delta_j + \Delta_= - \underline{u}_j) x_{j,r} && \text{(32d)} \\
& u_j \le \overline{u}_j z_j + (u_{j,\text{cp}} + \Delta_j - \overline{u}_j) x_{j,r} + (u_{j,\text{cp}} - 0.5\Delta_j - \Delta_= - \overline{u}_j) x_{j,l} && \text{(32e)} \\
& x_{j,l} + x_{j,r} = z_j && \text{(32f)} \\
& u_k \ge u_{k,\text{cp}} x_{k,r} + \underline{u}_k x_{k,l} && \text{(32g)} \\
& u_k \le u_{k,\text{cp}} x_{k,l} + \overline{u}_k x_{k,r} && \text{(32h)} \\
& x_{k,l} + x_{k,r} = z_k && \text{(32i)} \\
& w_{\mathcal{M}^{-1}(n_u)} \le w_{\mathcal{M}^{-1}(n_u-1)} \le \ldots \le w_{\mathcal{M}^{-1}(1)} && \text{(32j)} \\
& u_i \ge w_i \underline{u}_i + (z_i - w_i) u_{i,\text{op}}, \quad \forall i \in \mathcal{Z}_{\text{on}} && \text{(32k)} \\
& u_i \le w_i \overline{u}_i + (z_i - w_i) u_{i,\text{op}}, \quad \forall i \in \mathcal{Z}_{\text{on}} && \text{(32l)} \\
& w_i \le z_i, \quad \forall i \in \mathcal{Z}_{\text{on}} && \text{(32m)} \\
& u_i \in \mathbb{R} && \text{(32n)} \\
& z_j = 1 && \text{(32o)} \\
& z_i \in \{0,1\} \qquad \text{On/Off} && \text{(32p)} \\
& x_{i,l}, x_{i,r} \in \{0,1\} \qquad \text{Preferred perturbation direction} && \text{(32q)} \\
& w_i \in \{0,1\} \qquad \text{Preferred perturbation order} && \text{(32r)} \\
& && \text{(32s)}
\end{aligned}
$$

where indices $i$, $j$ and $k$ indicate it concerns all units, lacking units, and neutral units, respectively, unless otherwise explicitly states. In contrast to (7), which was a MILP, the new geometry improvement problem is a Mixed-Integer Quadratic Program.

The scaling parameters, or prioritization parameters, $P$'s, in (32a) can be used to give priority to the different extensions. The prioritization order could depend upon the application area. We suggest to use the following decreasing order of importance:

1. Do not alter the on/off status (unless necessary).
2. Alter the neutral units in (inverse) order of importance.
3. The perturbation direction.

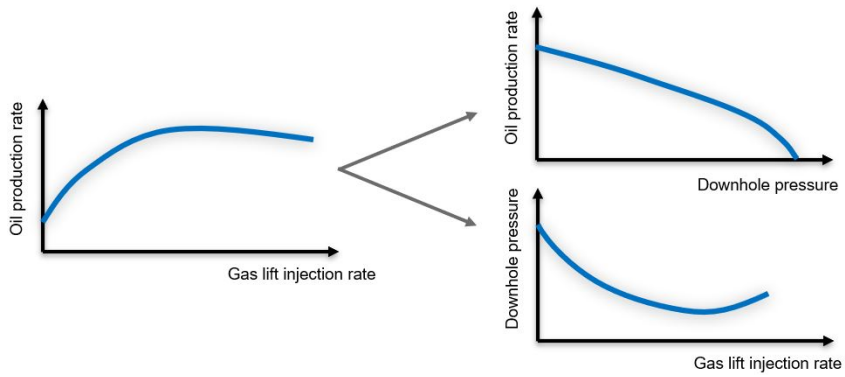


Figure 3: Illustration of the suggested splitting of the GLPC.

4. Stay as close as possible to the current operation point

To obtain this behavior, the $P$'s are selected as follows. Both $P_u$ and $P_{l,r}$ are set to 1. Considering that the first term of (32a) is indifferent to the direction, it is not in conflict with the second term. We assume that the $v$'s are selected according to (21), which ensures that the two first terms for a single unit is in the range $[-1, 0]$. $P_w$ must be selected large enough to ensure that perturbing left/right will not be more advantageous than not perturbing at all, $P_w = n_u + 1$. Finally, $P_z$ must be sufficiently large such that opening/closing *one* unit cannot lead to an improved objective function, $P_z = n_u(n_u + 1)$.

# 4 Application study - Gas lift allocation

This section starts by first explaining the motivational example of this work. Thereafter, a decomposition of a specific curve which allows for utilization of the available sensor data will be introduced. To the best of the authors' knowledge, this decomposition has not previously been used to exploit data in such a manner. The suggested algorithm will be applied to a small simulated version of the motivational example. Problems of larger sizes will be tackled in the next section.

This work was motivated by a part of the Daily Production Optimization (DPO) challenge in the oil and gas industry. The DPO is concerned with optimally utilizing the available resources while still obeying system and operational constraints. The study consists of a set of wells producing to a separator. The wells produce fluids comprised of oil, gas and water.

Initially, the pressure inside the reservoir below the seabed is, typically, sufficiently high to drive the flow of hydrocarbons to the surface on its own. As fluids are

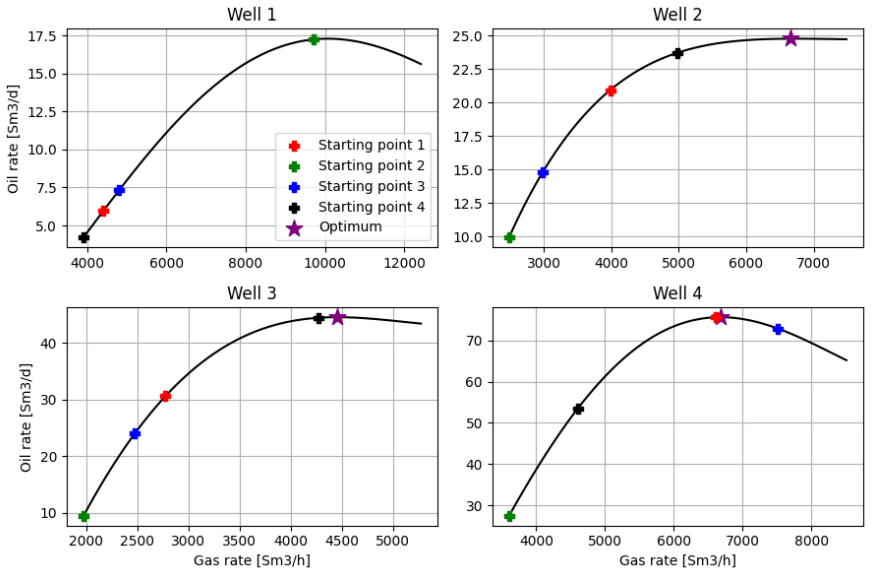


Figure 4: The GLPCs for the four wells case study in Section 4.3. Notice that Well 1 is shut at the optimum and therefore does not have a star marker. Each set of equally colored plus symbols shows the start point for one case.

drained, the reservoir pressure decreases and at some point it is too low to maintain an economically beneficial production.

One method to increase production when the reservoir pressure has decreased, is to inject gas into the wellbore. The gas, or lift gas, is mixed with the fluids coming from the reservoir, and this mixture will achieve a lower density, and, thus, the production of fluids will increase. However, there is a limit on how much gas may be injected before the production actually starts to decrease. This is due to that the friction of the mixture will increase faster than the mass of the mixture decreases.

Each well may have a limitation on the lower bound on the gas lift injection rate. A well may exhibit unstable behavior, *e.g.* slugging, if too little gas is injected. In addition, an upper limit may exist. A well may produce sand (from the reservoir) if the pressure difference between the reservoir and the wellbore is too large. In this work, these constraints are given as bounds on the gas lift injection rates.

Distributing the available lift gas between the wells can be seen as a resource allocation problem on the form (1). The resource, or lift gas, must be allocated between the wells, or units. In addition, the natural choice of opening or shutting wells is considered.

A standard method of approaching this problem is to create a Gas Lift Performance

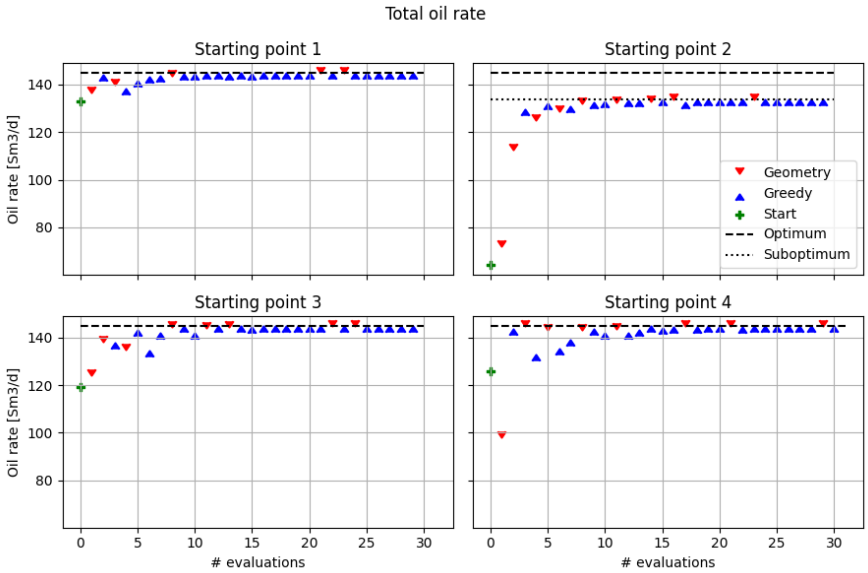


Figure 5: The evolution of the total oil rate for the four wells case study in Section 4.3. The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry.

Curve (GLPC) for each well. This concave curve gives the relationship between the gas lift rate and the oil production in steady-state conditions for the well. If these curves were available and reliable, they should be used and the resource allocation problem is straight forward. However, the focus of this example is when they are not available.

The gatherable data for each well can be split into two categories; production data and test-separator data. The production data are collected from sensors attached to that specific well and can be read any time. For our setup, that includes the downhole pressure gauge and the gas lift rate sensor. There is typically no rate sensor attached to the output of each well which is why the second set of data is acquired. The test-separator data is only available when the considered well is routed to a test-separator. For our setup, the relevant data is the oil rate.

An offshore oil field typically has many wells, but only one test-separator. In addition, letting a well produce to the test-separator invariably implies a cost of lost production. Therefore, test-separator data may be old, and only a few steady-state measuring points are taken. Also, in many fields well production may be off-design with respect to test-separator instrumentation, which implies significant uncertainties in test-separator measurements. Taken together, these issues imply that getting more

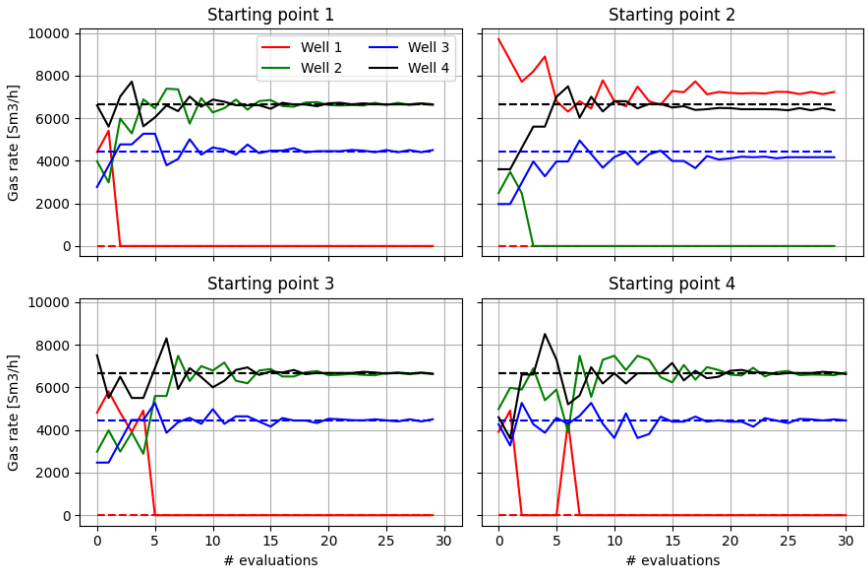


Figure 6: The evolution of the gas lift rates for the four wells case study in Section 4.3.

than first-order (linear) information from test-separator data is a challenge.

## 4.1 Gas Lift Performance Curve Decomposition

We suggest to decompose the GLPC into two curves. One curve relates the gas lift rate to the downhole pressure, and another which relates the downhole pressure to the oil rate, see the illustration in Fig. 3.

This decomposition allows for an exploitation of the two different categories of data. The first curve will only be updated based on data collected from the test-separator analysis. As mentioned above, due to the challenges with the test-separator measurements, this first curve will be linear. In addition, the true relationship between the downhole pressure and the oil rate is typically close to linear for a large range of pressures.

The real advantage of this decomposition surfaces when the second curve is considered. This curve, which relates the gas lift rate to the downhole pressure, can be updated based on data collected during production as both these measurements are available. The relationship between gas lift rate and downhole pressure may better be modelled by a quadratic equation. However, the extrapolation capabilities of a quadratic model, or any polynomials of degree 2 or higher, may be unreliable. Thus, a linear model is

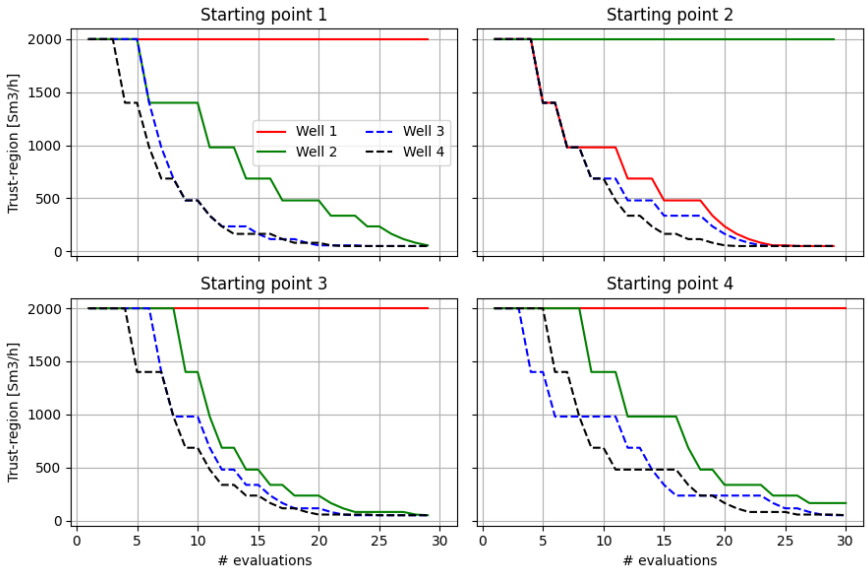


Figure 7: The evolution of the trust-region radii for the four wells case study in Section 4.3.

chosen for both curves. To deal with the inaccuracy of the assumed linear relationship between gas lift rate and downhole pressure, the presented trust-region framework will be applied.

## 4.2 Setup information

This section contains details on the setup and parameters. The setup given here is used in the subsequent examples unless otherwise stated.

The true linear relationship between the downhole pressure and the oil rate for each well is assumed measured perfectly. Introducing significant errors in these measurements would make for a more realistic case. However, the proposed algorithm will only find good solutions to the extent that the given set of relationships reflects reality. Moreover, the comparison of the results of the algorithm with the optimal solution becomes clearer when these relationships are modelled correctly. It is not expected that the algorithm will converge to a local solution if the downhole pressures versus the oil rates are modelled incorrectly. These relationships can be thought of some kind of ranking between the different wells.

In order to focus on the core of the algorithm, and not on *e.g.* filtering, no noise is introduced in the measurements. The modelled linear relationship between the gas

lift rate and the downhole pressure for each well is found using the linear least squares regression method `lsq_linear` in Python's Scipy. Only two points for each well are used to create the models: center point plus one more.

The parameters used in the algorithm is given in Tab. 1.

Table 1: The default parameters used for the algorithm

| **Parameter** | **Value** |
|---|---|
| $\eta_1$ | 0.1 |
| $\gamma$ | 0.7 |
| $\omega_c$ | 0.7 |
| $\mu_c$ | $10^9$ |
| $\boldsymbol{\Delta}_0$ | 2000 |
| $\boldsymbol{\Delta}_=$ | 0.0 |
| $\boldsymbol{\Delta}_{\min}$ | 50.0 |
| $r$ | 1.0 |

The optimization problems (8) and (32) are formulated using CasADi (Andersson et al., 2019), and solved with the Mixed-Integer Nonlinear Program (MINLP) solver BONMIN (Bonami et al., 2008).[1] The computational study was carried out on Dell XPS 15 9570 with Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz.

The algorithm is initialized with, or started from, the current operation point, which is the natural choice. The available lift gas throughout subsequent iterations of the algorithm is given as the sum of the applied gas lift rates at the initial operation point. Experience shows that solving the geometry improvement optimization problem (32) with many units requiring new points can be a time consuming task. Applying a better solver and/or take advantage of high performance computing could ease this task. However, to circumvent this waiting time, a limit on the amount of lacking units for each time (32) is solved is imposed and set to 10. If, for any reason, the solver can not find a solution to (32), then that set of lacking units is split in two and the formulation is solved twice with the easier problem. This splitting may happen as many times as is required.

Throughout the entire application study the curves and points referred to as "optimal" or "optimum" are found by using BONMIN on the problem with all the correct Gas Lift Performance Curves (GLPCs) available. *I.e.*, the underlying problem is formulated as in (1) with the $f_i$'s available, and solved by an off-the-shelf solver. This is in contrast to the proposed algorithm, which was designed to only have noisy samples of the GLPCs (the $f_i$'s) available.

The termination criterion of Algorithm 2 is set as a threshold on true function eval-

[1] Unfortunately, the Lagrange multipliers required for evaluating the gradient (12) is not available directly from BONMIN. To overcome this, the subproblem (8) is first solved with BONMIN, then it is solved again with the Nonlinear Program (NLP) solver IPOPT (Wächter and Biegler, 2006) keeping the binary variables fixed.

uations. At the end of each iteration of the algorithm, it checks if the true function has been evaluated equally or more times than this threshold.

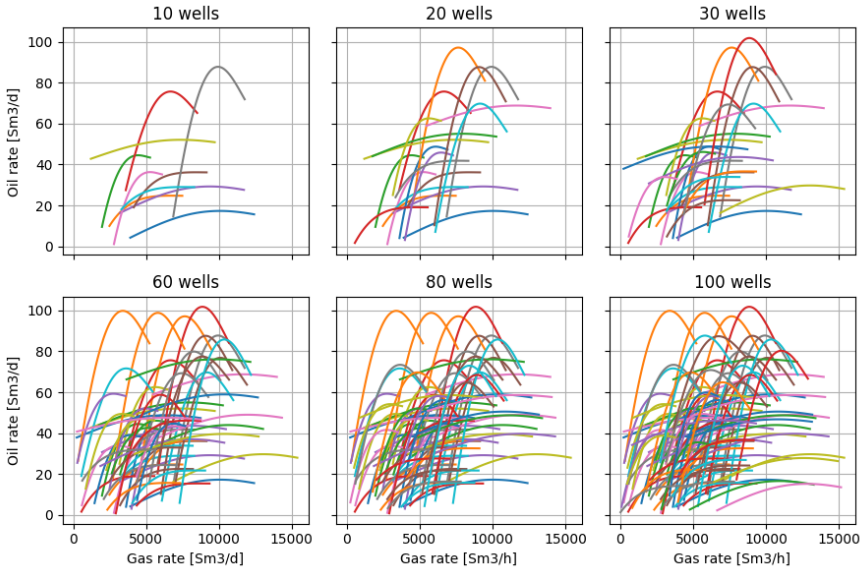


Figure 8: The GLPCs for all the wells used in Section 5.1.

### 4.2.1 Consequences of using two linear models

As a result of decomposing the gas lift performance curve into two curves, each unit will have two linear models associated to it. One which relates the gas lift rate to the oil rate (through the downhole pressure),

$$Q_i(u_i) = \alpha_i P_i(u_i) + \beta_i \tag{33}$$

where $P_i(u_i)$ is the second linear model relating the gas lift rate to the downhole pressure

$$P_i(u_i) = a_i u_i + b_i \tag{34}$$

The parameters of (33) remains fixed, whereas those in (34) are updated by the algorithm. In the theory section, only one linear model where used. The tuple $(u_i, Q_i)$ is used in most steps, except when the model prediction quality is calculated in (9). The model prediction quality requires a comparison between the predicted and measured values, thus, the tuple $(u_i, P_i)$ is used for this.

The algorithm is presented as a minimization method. The goal is to maximize the (estimated) oil rate, thus, (33) must be multiplied by negative one in the objective function (8) such that the oil rate is maximized and not minimized.

### 4.3 Results for a four well case

A small example consisting of four wells, or units, is presented. This allows for easier illustration of how each well behaves. In this four well example, we investigate how the algorithm evolves as a function of the number of true function evaluations. Four different starting operation points are considered. For all starting points, the total available lift gas are the same.

The four GLPCs and the four sets of starting points are shown in Fig. 4. For example, the starting points for Starting point 1 may be found as all the red marks in the four subplots. The optimum is the same for all different initial conditions. The lower and upper bounds for the wells are available in Fig. 4 as the minimum and maximum x-values for the curves. There is no marker for the optimum for Well 1 as this well is shut at the optimum.

It is emphasized that the concave GLPCs are unknown to the optimizer and they may only be evaluated.

The evolution of the total oil rate, gas lift rates, and trust-region radii are shown in Fig. 5, Fig. 6, and Fig. 7, respectively. In almost all cases, the trust-region radii have reached the lower limit $\Delta_{\min}$ for the wells that are open, see Fig. 7.

In Fig. 5, it can be seen that the algorithm has more or less reached the (local) optimum in less than 15 evaluations of the true functions. For Starting point 1, 3 and 4, the optimum found by the algorithm is the same as the one found by BONMIN. For Starting point 2, we can see that the algorithm does not reach the optimum. Looking at Fig. 6, it can be seen that the algorithm converges to a point where one of the wells are closed. The proposed algorithm is not a global algorithm, and converging to a local optimum is expected behaviour. Nonetheless, for the given set of open wells, the found solution is indeed a local optimum. The dotted line in Fig. 5 is found by BONMIN when the binary variables are fixed. In this case, the solutions found by BONMIN and the algorithm coincide.

For all the different starting points, no wells were ever closed for the reason of improving the geometry of other wells.

## 5 Investigating properties of the algorithm

In this section, different aspects of the algorithm will be analyzed. The examples in this section will be different instantiations of the motivational example presented in

the previous section.

This section is structured as follows. First, the algorithm is applied to several setups with an increasing number of wells to see how the algorithm performs. Second, an algorithmic design choice made in the theory section is justified through examples. Third, the importance of the minimum trust-region radius $\Delta_{\min}$ parameter is illustrated by examples. Fourth, parameters of the algorithm related to the radii are discussed. Fifth, the importance of the starting point, for a certain amount of lift gas, is investigated.

## 5.1 Complexity: Increasing the number of wells

In this section, we will see how the algorithm performs when the number of wells, or units, increases. Six different scenarios will be considered: 10, 20, 30, 60, 80 and 100 wells. The GLPCs for the wells are shown in Fig. 8. The GLPCs themselves are not too interesting, but are included to illustrate that the wells are all represented by different concave functions. Further, the lower and upper bounds on the gas lift rates are also visible in the figure.

For each set of wells, three different starting points are considered. As opposed to before, the total available lift gas varies for each starting point, see Tab. 2.

Table 2: Amount of available lift gas for the scenarios with an increasing number of wells in Section 5.1. All numbers are in Sm3/h.

| | N = 10 | N = 20 | N = 30 | N = 60 |
|---|---|---|---|---|
| Starting point 1 | 33524 | 72172 | 105982 | 200805 |
| Starting point 2 | 43524 | 92172 | 135982 | 260805 |
| Starting point 3 | 61974 | 126422 | 188682 | 363555 |

| | N = 80 | N = 100 |
|---|---|---|
| Starting point 1 | 1268271 | 335670 |
| Starting point 2 | 2348271 | 435670 |
| Starting point 3 | 3485271 | 606920 |

The evolution of the total oil rate for the scenarios with 10, 20 and 30 wells are shown in Fig. 9. *E.g.*, the first row of subplots are the three different starting points for the case with 10 wells. A similar plot for the scenarios with 60, 80 and 100 wells are available in Fig. 10. However, due to the complexity of solving the underlying MINLP for such an high amount of wells, the optimal oil rate lines calculated by BONMIN are not included. Nonetheless, it illustrates how the proposed algorithm increases in number of required true function evaluations for an increasing amount of units before convergence is reached. The scenarios with 10, 20 and 30 wells already have illustrated that the algorithm converges to good local optima.

In both of these two plots, we can see the required number of function evaluations to

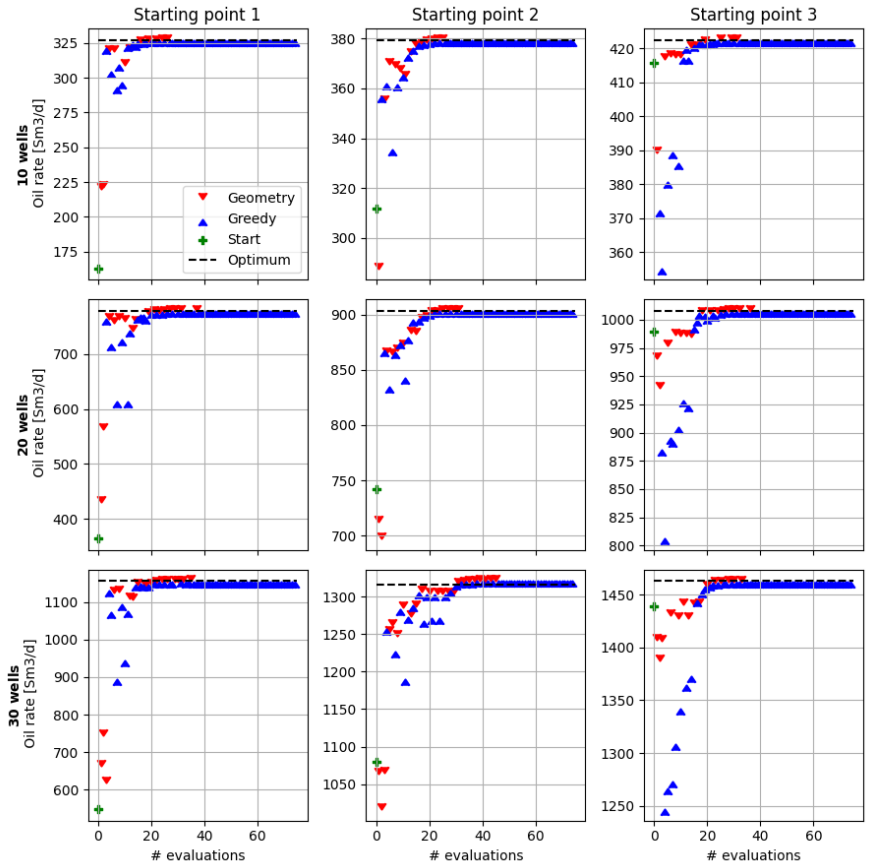


Figure 9: The evolution of the total oil rate for the setups with 10, 20 and 30 wells. The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry. The available gas is different for each starting point.

reach convergence is increased with a higher number of wells. However, the increase does not appear to be exponential.

The time taken to run the algorithm for all three starting points are shown in Fig. 11. The computational time clearly grows exponentially with an increasing amount of wells.

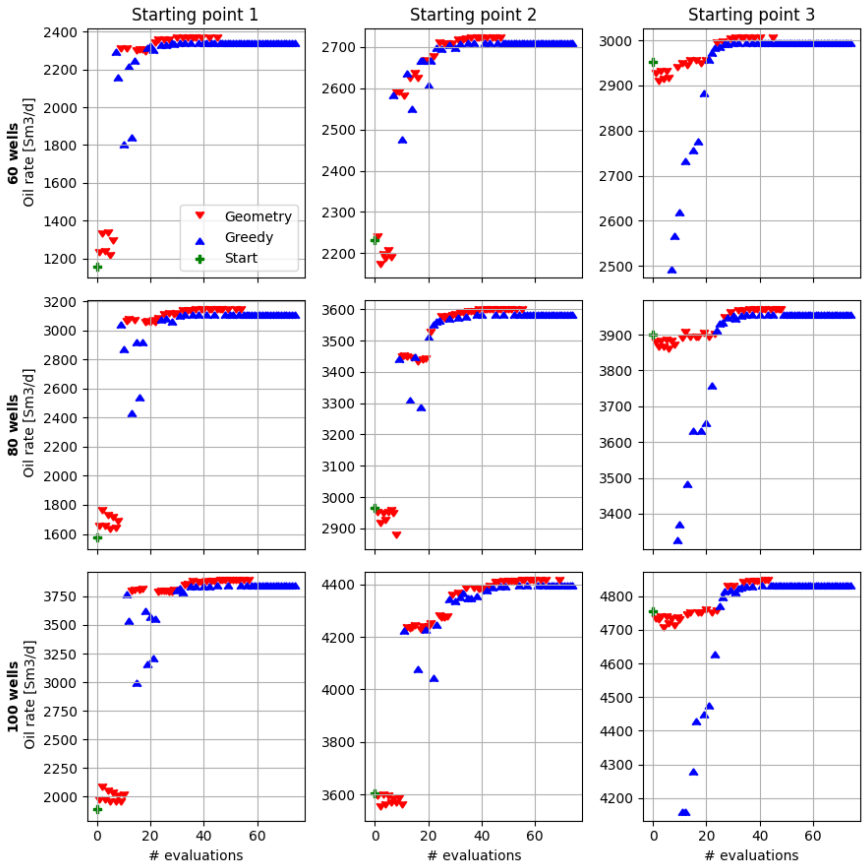


Figure 10: The evolution of the total oil rate for the setups with 60, 80 and 100 wells. The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry. The available gas is different for each starting point.

## 5.2 Criticality substep design choice justification

It was mentioned in Section 2.6 that a design choice was taken based upon experience. More specifically, it was decided to reduce all the radii in case that at least one of the radii was too large compared to the corresponding gradient element. The other design option was to only reduce the corresponding radius. In the following comparison, a third option is included: Skip the criticality substep altogether. This corresponds to setting the $\mu_c$ parameter to $\infty$. The three options are summarized below:

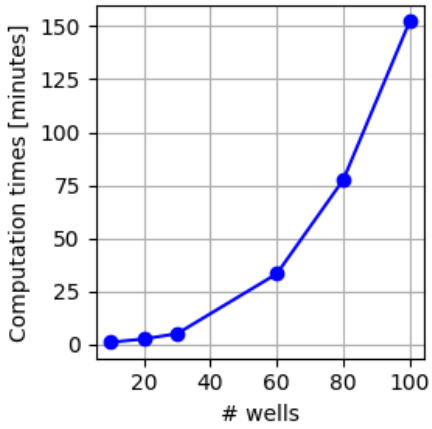


Figure 11: The computational time to solve 75 evaluations of the algorithm for three different starting points with an increasing number of wells.

- **Option 1:** Reduce all the radii if at least one element of the gradient of the Lagrangian is small compared to the corresponding radius.

- **Option 2:** Only reduce the radius for a unit if the corresponding element of the gradient of the Lagrangian is small compared to its radius.

- **Option 3:** Skip the criticality substep altogether.

A setup with 10 wells are used in this comparison. The evolution of the total oil rate is shown in Fig. 12 where each row of subplots presents one of the three options, and the columns are different starting points. It is evident that Option 2 is an inferior option for all the three starting points compared to Option 1. The performance of Option 3 is closer to the one obtained in Option 1. However, Option 1 is superior.

The reason why Option 3 is exhibiting a slower convergence than Option 1 may be explained as follows. In Option 3, all the radii will only be reduced when the if-test at line 21 of Algorithm. 2 is true. *I.e.*, all the radii are reduced when all the model prediction qualities are deemed satisfied and the solution of the subproblem did not provide a better value of the true function. In contrast, Option 1 will in addition reduce all the radii if one of the relationships between the gradient elements and the radii are unsatisfactory regardless of the value of the true function.

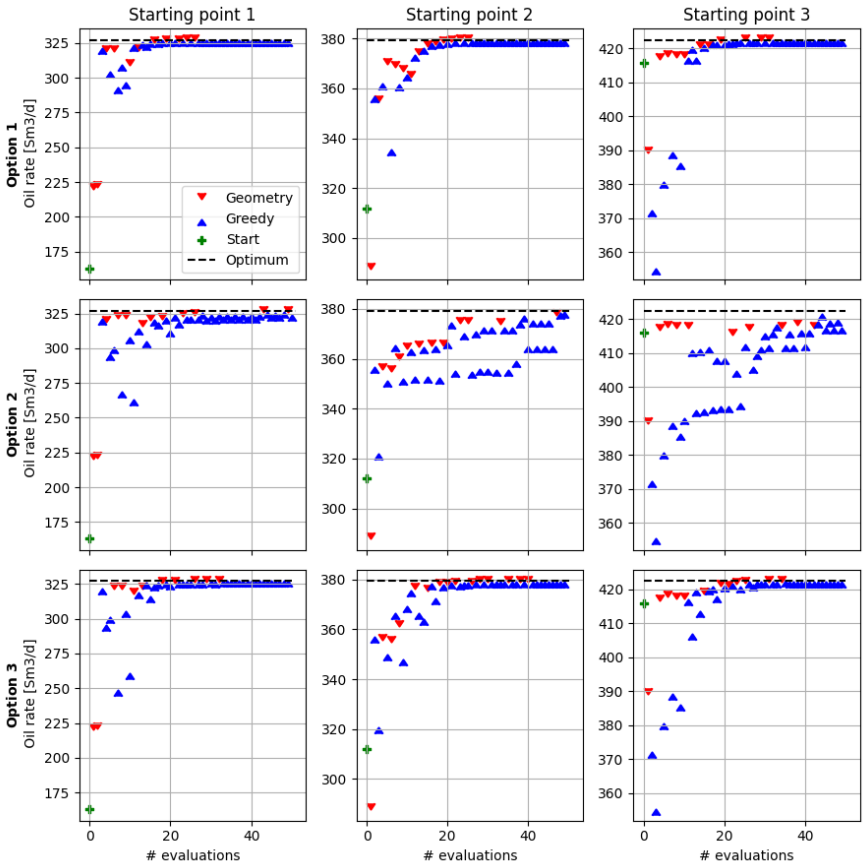


Figure 12: The evolution of the total oil rate for the setup with 10 wells in Section 5.2. The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry. The available gas is different for each starting point.

## 5.3 Importance of the minimum radii parameter

A key parameter of Algorithm 2 regarding convergence of the algorithm is $\boldsymbol{\Delta}_{\min}$. In this section we will illustrate two different aspects of this parameter. From one aspect, a smaller value of $\boldsymbol{\Delta}_{\min}$ is beneficial, whereas the other aspects advocates a larger value. To illustrate its impact on the performance of the algorithm, the setup used in Section 4.3 with the second set of starting points (Starting point 2) will be used. Recall that the algorithm converged with a sub-optimal set of units being on/off for that case. This is acceptable as it is not a global algorithm. In the examples below,

the algorithm also converges to sub-optimal sets with regards to the binary variables. For each example, the optimal gas lift rates with respect to the final set of binary variables are used as the dotted lines in the figures.

The parameters used for the following examples are found in Tab. 1 except for the initial radii $\boldsymbol{\Delta}_0 \in \{500, 2000\}$ and the minimum radii $\boldsymbol{\Delta}_{\min} \in \{0, 50, 200\}$.

In the first example, the initial start radii are set to 2000. The resulting evolution of the gas lift rates and the total oil rate for the three different minimum radii values are plotted in Fig. 13 and Fig. 14, respectively. Both figures show that a smaller value of $\boldsymbol{\Delta}_{\min}$ yields a more desirable response. A higher value of the parameter results in a zigzagging response of the gas lift rates, see Fig. 13. As a result of this alternating response around the optimal actions, the resulting total oil rates is negatively impacted, see Fig. 14. A tempting conclusion from this small example is to set $\boldsymbol{\Delta}_{\min} = 0$. In the next example we will illustrate why this is discouraged.

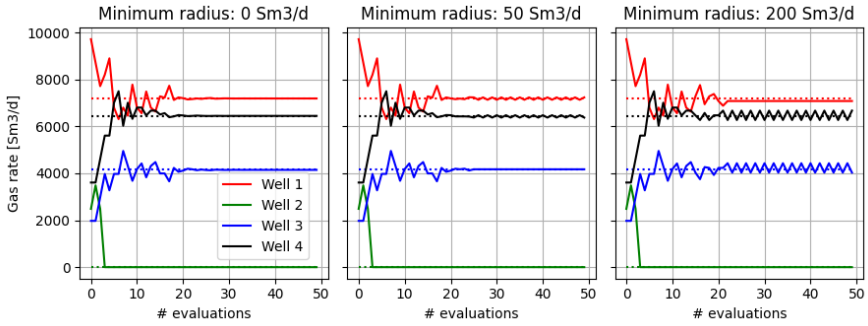


Figure 13: The evolution of the gas lift rates for different minimum radii values and with starting radii set to $\boldsymbol{\Delta}_0 = 2000$.

In the second example, the initial start radii are set to 500, and the same types of plots are available in Fig. 15 and Fig. 16. It becomes evident why setting $\boldsymbol{\Delta}_0$ to zero is discouraged. The algorithm converges to a point, but it is not a (local) optimum. The radii are going too fast to zero which then hinder the algorithm to move further in the solution space. When the minimum radii is increased to 50 and 200, the responses are as expected. The algorithm converges towards the (local) optimum. The larger $\boldsymbol{\Delta}_{\min}$ gives a faster response, but also ends with more zigzagging around the optimum.

These two examples have illustrated the importance of the $\boldsymbol{\Delta}_{\min}$ regarding safeguarding against unwanted convergence, or stopping, of the algorithm due to the $\boldsymbol{\Delta}$ approaching zero.

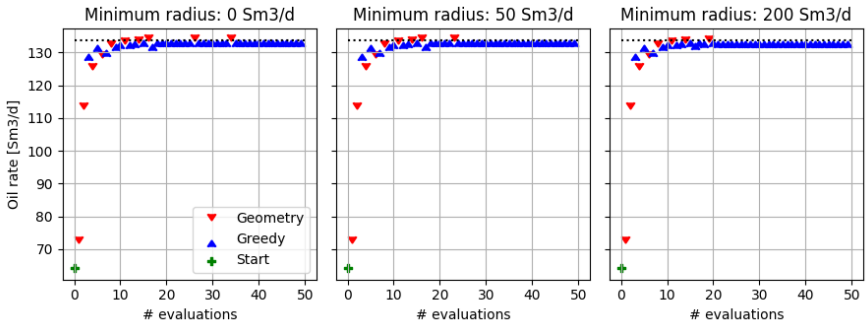


Figure 14: The evolution of the total oil rate for different minimum radii values and with starting radii set to $\boldsymbol{\Delta}_0 = 2000$ . The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry.

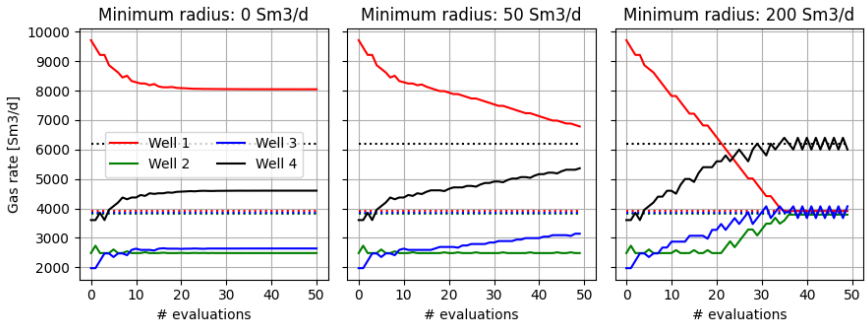


Figure 15: The evolution of the gas lift rates for different minimum radii values and with starting radii set to $\boldsymbol{\Delta}_0 = 500$.

## 5.4 Parameters impacting the evolution of the radii

The examples in Section 5.3 made it evident that the $\boldsymbol{\Delta}_{\min}$ parameter is of high importance regarding the behaviour and convergence of the algorithm. In this section, we will discuss some of the other parameters of the algorithm and how they impact the radii and its convergence. The example with starting radii of 500, and $\Delta_{\min} = 0$ in the previous section is used as a base case. The algorithm did not converge to a local optimum in the base case, and we will here show how the choices of other parameters will impact the evolution of the radii.

The two most obvious parameters that will directly impact the radii are the contraction factors: $\gamma$ and $\omega_c$. Setting these parameters closer to one will ensure that the radii are reduced slower. On the other hand, a lower value will result in a more

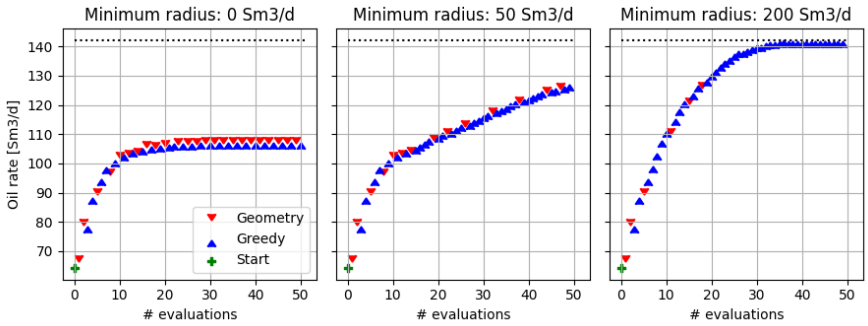


Figure 16: The evolution of the total oil rate for different minimum radii values and with starting radii set to $\boldsymbol{\Delta}_0 = 500$ . The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry.

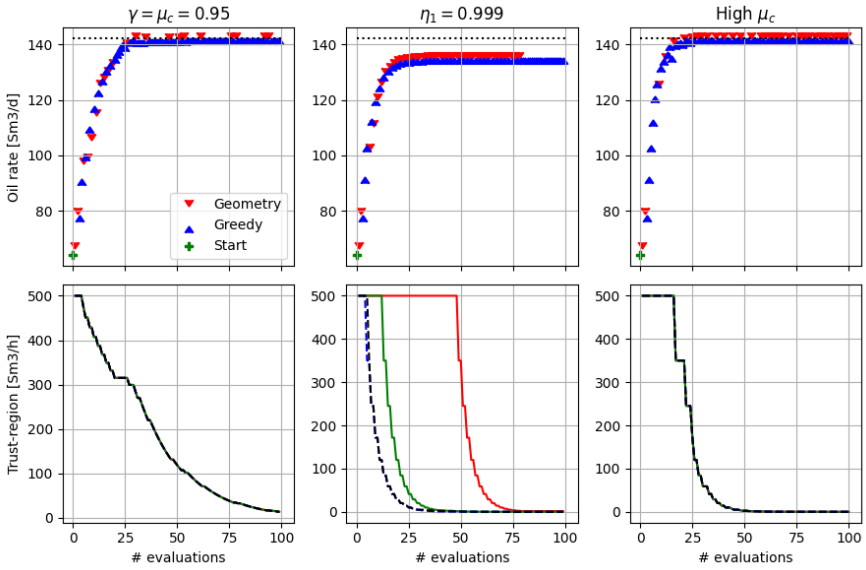


Figure 17: The behavioral differences of the algorithm due to selected changes of the default parameters. The applied parameters are the default ones given in Table 1, except for those noted in the titles of the subplots and the starting radii which were set to 500 for all three cases.

aggressive reduction of the radii. In the first column of subplots in Fig. 17, the evolution of the oil rate and the radii are shown for the case with high contraction factors,

$\gamma = \omega_c = 0.95$. As opposed to the base case, the algorithm converges to the local optimum as the radii does not reach zero before the local optimum is approached.

Another parameter is the threshold on what is considered a sufficient model prediction: $\eta_1$. The result of tuning this parameter is less obvious. If the value is almost one, then most predictions will be deemed unsatisfactory. This means that the corresponding radii will be reduced. However, because one or more predictions are bad, then neither the criticality substep nor the "reduce all" functionality at line 21 of the algorithm will happen. The evolution with $\eta_1 = 0.999$ is shown in the second column in Fig. 17. Compared to the base case, the algorithm reaches closer to the local optimum, but also here it is stopped by the radii going to zero.

Yet another parameter that could help slowing down the reduction of the radii is the $\mu_c$. By setting this parameter high enough, the criticality substep will never happen. However, as we saw in Section 5.2, the additional reduction resulting from this step did result in better performance. Nonetheless, with a high enough value of $\mu_c$, one or more radii will only be reduced if (i) a model predicts badly, or (ii) all models predicts well and there was no improvement in the true function. The evolution with $\mu_c$ set to its default value squared is shown in the third column in Fig. 17. In this case, the local optimum is reached. The radii are kept constant for several iterations whilst the algorithm keeps improving the value of the true function.

## 5.5 Randomized starting points

In this section, we will quantitatively investigate if the starting point is of importance regarding experienced convergence.

A setup with 10 wells, and a 100 randomized starting points are used. All of the starting points have the same amount of available lift gas. For all starting points, the available lift gas is given as the sum of all the minimum gas lift rates multiplied by 1.2. Each well gets its minimum gas lift rate, and the remaining 20% is uniformly randomly divided between the 10 wells. The assigned initial gas lift rates were observed to always be below the upper limits for all wells.

The results are provided in Fig. 18 and Fig. 19. For each starting point, we checked the obtained best greedy point at several checkpoints. The checkpoints were set to 10, 15 and 25 evaluations of the true function. By "best greedy point" it is meant the best point observed so far amongst the points that were found by solving the subproblem (8), *i.e.*, ignoring the geometry improvement points. Two metrics to discuss convergence are used. First, in Fig. 18, the total oil production of the best point is compared to the one found by solving the same problem with BONMIN. In this metric, a point is counted as successful if it has 99% or more of the total oil rate found by BONMIN. Second, in Fig. 19, the z variable of the best points are considered. In this metric, a point is counted as successful if it has the same z-vector as the one found by BONMIN.

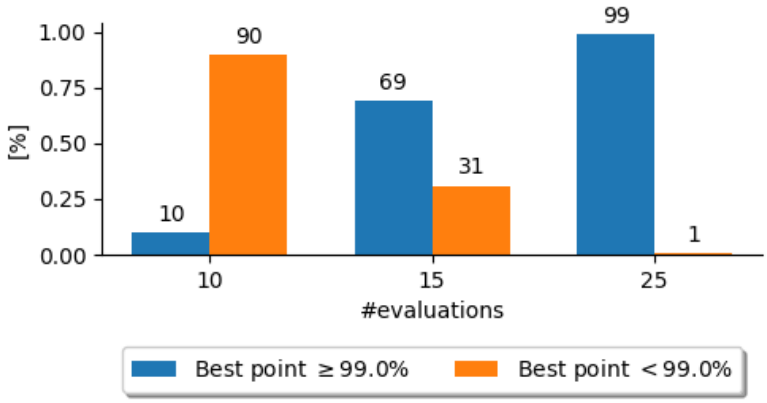


Figure 18: The blue bars indicate the number of instances where the best greedy point found were no worse than 1% of the point found by bonmin. The x value specifies the checkpoints in evaluations. The orange bars are the opposite of the blue.

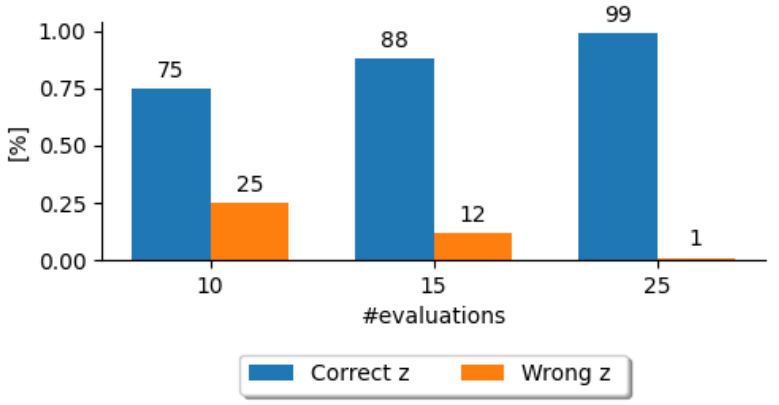


Figure 19: The blue bars indicate the number of instances where the z value corresponding to the best greedy point found were the same as the one found by bonmin. The x value specifies the checkpoints in evaluations. The orange bars are the opposite of the blue.

As we can see in the two figures, in 99 out of the 100 different starting point, the algorithm converges to a point very close to the point found by BONMIN. In only 1

case does the algorithm converge to a solution with an incorrect z-variable, see Fig. 19.

The reason why the first metric uses a comparison percentage less than 100, is that the algorithm is not expected to converge exactly to the (local) optimum. This is due to the $\Delta_{\min}$ parameter, and the zigzagging behaviour that follows from it, see the rightmost plot in Fig.13.

# 6 Discussion

The proposed algorithm has been designed to tackle a resource allocation problem where the most challenging obstacle was that the relationship between the allocated resource and the corresponding cost of a unit was unknown. The lack of models motivated the use of a data-driven derivative-free method. In the previous section, it was illustrated by examples that the algorithm converges to local optima. In 99 out of 100 cases when considering different starting points, the final vector of binary variables were the same as the presumably globally optimal solution found by BONMIN. However, the algorithm is not designed to be a global optimizer, and converging to a local solution is expected behavior. Moreover, when the binary variable vector, $\mathbf{z}$, was kept fixed, the algorithm converged to the local optimum in all cases.

An attractive feature of the algorithm is that it seems the number of required evaluations of the true function is moderate, and does not grow exponentially with the number of units. A hidden cost when looking at the convergence time, is the time taken to solve all the optimization formulations within the algorithm. The geometry improvement optimization formulation (32) is a Mixed Integer Nonlinear Program (MINLP) which is a complicated problem structure to solve. Nonetheless, in this work the true function evaluations are considered to be the most time consuming task. In the case study, after a set of gas lift rates are imposed, it is required to wait for a steady-state of the system before measurements may be taken. The transition time from a steady-state to the next is expected to be in the range of several hours. The increase in computational time is an expected result as a MINLP solver typically exhibits such a phenomenon. Nonetheless, we only need to solve either the geometry improvement optimization problem or the subproblem once before the inputs are applied to the system.

A disadvantage of this tailored algorithm is that a proof along the lines of Conn, Scheinberg, and Vicente (2009) to show global convergence to first-order critical points does no longer apply as the proposed method deviates from the framework in Conn, Scheinberg, and Vicente (2009). The major differences being the single pass criticality substep, as explained in Section 2.6, the required minimum trust-region radii parameter $\boldsymbol{\Delta}_{\min}$ to avoid the radii going to zero, and the use of several trust-region radii and models. These changes are not aligned with the framework of Conn, Scheinberg, and Vicente (2009), and thus, the proof of convergence cannot be applied. Nonetheless, the proposed method contains the same type of building blocks as the one in Conn,

Scheinberg, and Vicente (2009). Furthermore, global convergence was experienced in all examples.

A final advantage of the method is that it only evaluates the true function at points that are feasible with respect to the underlying problem (1). This is the case as the constraints are imposed at all times when any new point is selected.

# 7 Conclusion

We proposed a derivative-free trust-region model-based algorithm to tackle a resource allocation problem with some specific characteristics that were inspired by a real world problem. The algorithm converged to local optima in all the compared cases.

In addition, we proposed a decomposition of the gas lift performance curve to exploit available sensor data. Furthermore, we illustrated how this decomposition can be combined with the suggested algorithm.

# Acknowledgment

Partial financial support from Wintershall Dea Norge AS is acknowledged. They had no involvement regarding content nor writing of this article.

We acknowledge Mammad Mirzayev for stimulating discussions on the industrial task that inspired this work.

Parts of the original ideas for the gaslift application study in this paper are due to Professor Alexey Pavlov, for which he is gratefully acknowledged.

# References

Andersson, J. A. E. et al. (2019). "CasADi – A software framework for nonlinear optimization and optimal control". In: *Math. Program. Comput.* 11.1, pp. 1–36.

Bajaj, I. and M. F. Hasan (2019). "UNIPOPT: Univariate projection-based optimization without derivatives". In: *Computers & Chemical Engineering* 127, pp. 71–87.

Bajaj, I., S. S. Iyer, and M. Faruque Hasan (2018). "A trust region-based two phase algorithm for constrained black-box and grey-box optimization with infeasible initial point". In: *Computers & Chemical Engineering* 116. Multi-scale Systems Engineering – in memory & honor of Professor C.A. Floudas, pp. 306–321.

Bonami, P. et al. (2008). "An algorithmic framework for convex mixed integer nonlinear programs". In: *Discret. Optim.* 5.2. In Memory of George B. Dantzig, pp. 186–204.

Conn, A. R., K. Scheinberg, and L. N. Vicente (2009). *Introduction to derivative-free optimization*. SIAM.

Giuliani, C. M. and E. Camponogara (2015a). "Derivative-free methods applied to daily production optimization of gas-lifted oil fields". In: *Comput. Chem. Eng.* 75, pp. 60–64.

— (2015b). "Derivative-free optimization with use of problem structure: Applications to oil production". In: *2015 IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, pp. 764–768.

Giuliani, C. M., E. Camponogara, and A. Plucenio (2013). "A computational analysis of nondifferentiable optimization: Applications to production maximization in gas-lifted oil fields". In: *2013 IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, pp. 286–291.

Katoh, N., A. Shioura, and T. Ibaraki (2013). "Resource Allocation Problems". In: *Handbook of Combinatorial Optimization*. New York, NY: Springer New York, pp. 2897–2988.

Krishnamoorthy, D., B. Foss, and S. Skogestad (2016). "Real-Time Optimization under Uncertainty Applied to a Gas Lifted Well Network". In: *Processes* 4.4.

Krishnamoorthy, D., A. Pavlov, and Q. Li (2016). "Robust Extremum Seeking Control with application to Gas Lifted Oil Wells". In: *IFAC-PapersOnLine* 49.13. 12th IFAC Workshop Adapt. Learn. Control Signal Process. (ALCOSP) 2016, pp. 205–210.

Nocedal, J. and S. J. Wright (2006). *Numerical Optimization*. New York, NY: Springer New York.

Peixoto, A. J. et al. (2015). "Modelling and Extremum Seeking Control of Gas Lifted Oil Wells". In: *IFAC-PapersOnLine* 48.6. 2nd IFAC Workshop Autom. Control Offshore Oil Gas Prod. (OOGP) 2015, pp. 21–26.

Powell, M. J. (1994). "A direct search optimization method that models the objective and constraint functions by linear interpolation". In: *Advances in optimization and numerical analysis*. Springer, pp. 51–67.

— (2009). "The BOBYQA algorithm for bound constrained optimization without derivatives". In: *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge* 26.

Rashid, K. (2010). "Optimal Allocation Procedure for Gas-Lift Optimization". In: *Ind. Eng. Chem. Res.* 49.5, pp. 2286–2294.

Rashid, K., W. J. Bailey, and B. Couet (2012). "A survey of methods for gas-lift optimization". In: *Model. Simul. Eng.* 2012, p. 16.

Wächter, A. and L. T. Biegler (2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Math. Program.* 106.1, pp. 25–57.

# B Real time optimization of systems with fast and slow dynamics using a lookahead strategy

**Andersen, J. R.**, Lima Silva, T., Imsland, L., Pavlov, A., "Real time optimization of systems with fast and slow dynamics using a lookahead strategy". In: *2020 59th IEEE Conference on Decision and Control (CDC)* (2020), pp. 2342–2349. DOI: 10.1109/CDC42340.2020.9304460

## CRediT authorship contribution statement

**Joakim Rostrup Andersen:** Conceptualization, Methodology, Software, Writing - Original Draft, Visualization.
**Thiago Lima Silva:** Conceptualization, Software, Writing - Original Draft.
**Lars Imsland:** Conceptualization, Methodology, Writing - Review & Editing, Supervision.
**Alexey Pavlov:** Writing - Review & Editing, Supervision.

# Real time optimization of systems with fast and slow dynamics using a lookahead strategy

Joakim Rostrup Andersen[1], Thiago Lima Silva[2], Lars Imsland[1], and Alexey Pavlov[2]

[1]Department of Engineering Cybernetics, NTNU, Trondheim, Norway
[2]Department of Geoscience and Petroleum, NTNU, Trondheim, Norway

**Abstract:** Systems with fast and slow dynamics give rise to objectives in different time scales which may not be aligned. The existing dynamic optimal control methods might become computationally infeasible due to the fine discretization required to capture the fast dynamics. On the other hand, a real time optimization (RTO) method based on steady-state models, which is computationally efficient, can greedily drive the plant towards optimal operation. The drawback of the RTO approach is that it may yield actions that only focus on near future goals and the objectives involving the slower dynamics are neglected. In this paper, we propose to extend RTO with a lookahead strategy by introducing a predictor to capture the effect of changing the current controls on the long-term objective. In this way, we introduce the long-term objectives in RTO while maintaining its computational efficiency and not losing focus of short-term objectives. The proposed approach is demonstrated in a simulation study from offshore petroleum production, that compares the proposed method with both an "industry-standard" RTO method, and a full fledged dynamic optimization method that takes both slow and fast dynamics into account. The proposed methodology performs almost as well as the dynamic optimization method while maintaining a low computational effort.

## 1 Introduction

Complex industrial processes have goals in different time scales ranging from the long-term planning and scheduling to automatic actions for stable and safe operation. Such goals can be of different nature, and it can be challenging to achieve an overall good performance treating the problem as a whole. Thus, a proper manner to deal with such problems is to decompose the decision making process into several layers (Findeisen et al., 1980). This hierarchical methodology is accepted as a standard industrial practice (Darby et al., 2011) and also well-known in academic literature as plant-wide control (Skogestad, 2000; Skogestad, 2004; Larsson and Skogestad, 2000; Rangaiah and Kariwala, 2012).

The decisions in a hierarchical control system are divided in layers depending on their

time frame (Skogestad, 2000) as shown in Fig. 1. The top layer, commonly known as Asset Management, focuses on long-term decisions which typically involve risk handling, investments and infrastructure planning. Next are the decisions taken in a time frame of a couple of days or weeks such as operations scheduling and plant-wide operations. Next follows the Real-time Optimization (RTO) layer, which is responsible for decisions that have to be taken in a time scale of hours or days. Shorter-term operational decisions regarding the control and automation of the plant aim to keep operations stable and mitigate disturbances in a range of seconds to hours, being performed automatically without human interference.

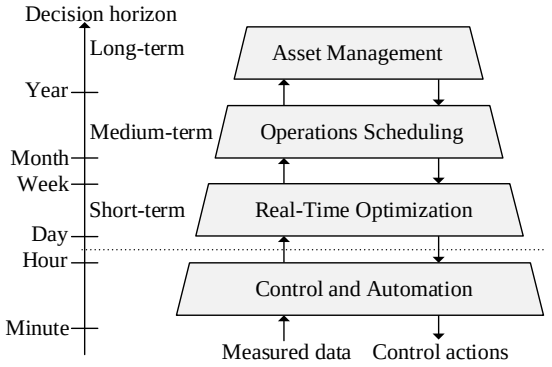


Fig. 1: Multi-level control hierarchy, adapted from Foss (2012).

Traditionally the interplay between the layers functions with the upper layers providing setpoints to the layers below such that near-optimal economic operation of the process is achieved. This hierarchical scheme is suitable in many industrial applications, especially when the objectives in the layers are aligned. However, in some applications, these goals might be conflicting in different time scales such that the overall objective of the control system becomes less obvious.

An example of such problems appears in resource-constrained production optimization. In the short-term, the top-ranked objective is the maximization of economic revenues obtained with the current production, while in a long-term perspective this might lead to suboptimal utilization of resources. In other words, the decision variables might be shared between different layers, but the objectives are not necessarily aligned, which means that prioritizing short-term economic performance comes at the cost of long-term economic performance.

In some cases, problems of this type are approached by constructing objectives that weigh control performance against usage of inputs, yielding thereby a multi-objective optimization problem for which some tuning effort is required (Van Essen, Van den Hof, and J.-D. Jansen, 2011; Zavala and Flores-Tlacuahuac, 2012).

Depending on the research field, this approach may be known as Dynamic Real Time Optimization (DRTO) or Economic Model Predictive Control (EMPC). These frame-

works are capable of handling complex dynamical systems with conflicting objectives and constraints. The main limitations of this approach are the requisite of computational power, and the potential unexpected behaviour due to an improper choice of weights, which is typically problem-dependent. The former limitation becomes evident as the size of the decision variable vector grows due to the increasing amount of control steps needed to capture the dynamics of the faster system.

The challenge addressed in this paper is inspired by production optimization problems involving multiple time scales within the petroleum industry. The proposed solution is generalized into a systematic approach which can possibly be extended to other application areas. It consists of a novel control method that reconciles problems with conflicting goals and heterogeneous time scales by incorporating long-term objectives in the RTO layer using a lookahead strategy. This involves some level of integration between the models from different layers, but in such a way that the integrated problem remains computationally tractable.

The paper is organized as follows: Section 2 presents a motivating example from the petroleum industry. In Section 3 we provide a general formulation for control problems of systems containing both fast and slow dynamics. Section 4 contains the RTO with a lookahead strategy and the solution method. In Section 5 we demonstrate the feasibility of the proposed methodology by applying it to the motivating example. Section 6 contains a discussion. We conclude with Section 7.

## 2 Motivating example

In offshore petroleum production, there is typically a set of wells producing from a hydrocarbon reservoir to the topside facilities, see Fig. 2. While the reservoir consists of considerable amounts of hydrocarbons trapped in subsurface porous media, the network is a set of pipelines connecting the wells to the platform.

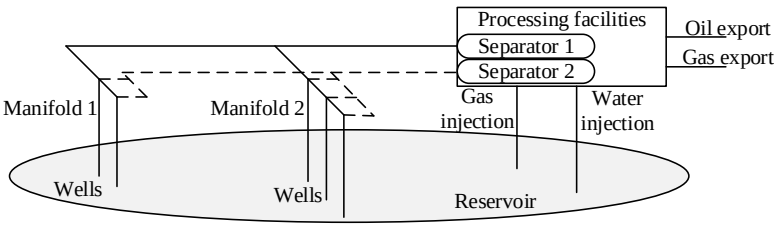


Fig. 2: Integrated production system with a reservoir, a production gathering network, and the processing facilities (Foss, 2012).

The reservoir fluids are produced by the wells and collected by a manifold before reaching the surface for separation, processing and further exportation. The oil is stored in tanks, while the water is treated before disposal or reinjection, and the gas

is either flared, reinjected or exported. There are also some wells injecting either water or gas into the reservoir to provide pressure support and enhance the overall recovery factor of the asset.

The control hierarchy presented in Fig. 1 is also suitable to describe the decisions in petroleum applications. In asset management the production infrastructure and topside facilities are designed. This provides a constraint in terms of number of wells and processing capacity for the reservoir management layer. This layer, on the other hand, determines well locations and drainage strategies that improve the long-term gains. Then, the RTO layer determines the well controls and the platform settings to maximize the production such that the operational constraints are honored. Finally, the control and automation layer guides the system towards the optimum, and autonomously stabilizes the rates and pressures of the wells. Notice that rate control is performed in both the reservoir management and the RTO layers, often with fairly different operational strategies. While in reservoir management the focus is to maintain the reservoir pressure and improving the overall recovery factor, the RTO layer acts greedily, prioritizing short-term gains often ignoring effects on the long-term objective.

## 2.1 Reservoir management: long-term optimization

The oil-water fluid flow in the reservoir can be described with a set of differential equations based on the mass-conservation principle, the Darcy law, and the capillary pressure principle (Chen, Huan, and Ma, 2006). After the discretization in time and space, the pressure of oil $p_{\text{oil},j,k}$ and the water saturation $S_{\text{water},j,k}$ typically suffice to describe the conditions in each grid block $j \in \mathcal{G}$ at time step $k \in \mathcal{K}$. The flows through the wells are a boundary condition for two-phase fluid flow in porous media, with $q_{\text{p},i,k}$ denoting the flow rate of phase p in well $i \in \mathcal{N}_{\text{w}}$, and $p_{\text{bh},i,k}$ being its Bottom-Hole Pressure (BHP) at time step $k$. A state space model can be utilized to formulate the aforementioned set of equations:

$$x_{j,k} = (p_{\text{oil},j,k}, S_{\text{water},j,k}), \forall j \in \mathcal{G}, \tag{1a}$$

$$v_{i,k} = (q_{\text{p},i,k}, p_{\text{bh},i,k}), \qquad \forall i \in \mathcal{N}_{\text{w}}, \tag{1b}$$

where the states $x_k$ are the pressure of oil and water saturation in the grid blocks, and the algebraic variables $v_k$ are the flows and pressures in the wells. The control vector $u_k$ contains one variable of $v_k$ per well which are used as a setpoint whereas the remaining variables in $v_k$ are calculated (cf. (2c)). Notice that, while the algebraic vector $v_k$ has the length of the same magnitude of the number of wells, the state vector $x_k$ have the same magnitude of the number of grid blocks in $\mathcal{G}$, which might range from thousands to a few million. The implicit equations to be solved at each

simulation step are:

$$0 = R(x_{k-1}, x_k, v_k), \tag{2a}$$
$$0 = V(x_k, v_k), \tag{2b}$$
$$0 = B(v_k, u_k), \tag{2c}$$

where $R(\cdot)$ represents the fluid flow in the reservoir, $V(\cdot)$ maps the reservoir states and the well algebraic variables, and $B(\cdot)$ are the relationship between the setpoints and the algebraic variables, see Chen, Huan, and Ma (2006) for more details.

The long-term rate control problem can be formulated as the maximization of the undiscounted Net Present Value (NPV):

$$\sum_{k=1}^{N} t_k \left[ \sum_{i \in \mathcal{N}_{\mathrm{w,prod}}} (q_{o,i,k} r_o - q_{w,i,k} r_w) - \sum_{i \in \mathcal{N}_{\mathrm{w,inj}}} (q_{w,i,k} r_i) \right] \tag{3}$$

subject to the implicit system of equations (2) and the well control bounds:

$$u_{\mathrm{lb},i,k} \leq u_{i,k} \leq u_{\mathrm{ub},i,k}, \quad \forall i \in \mathcal{N}_{\mathrm{w}}, k \in \mathcal{K}, \tag{4}$$

where $\mathcal{N}_{\mathrm{w,prod}}$ and $\mathcal{N}_{\mathrm{w,inj}}$ are the sets of production and injection wells, respectively, with $\mathcal{N}_{\mathrm{w}} := \mathcal{N}_{\mathrm{w,prod}} \cup \mathcal{N}_{\mathrm{w,inj}}$. The flow rates $q_{o,i,k}$ and $q_{w,i,k}$ are part of the algebraic variables $v_{i,k}$ of well $i \in \mathcal{N}_{\mathrm{w}}$. Further, the oil price, and the water production and injection costs are denoted by $r_o$, $r_w$ and $r_i$, respectively. Finally, the $t_k$ is the step length.

## 2.2 RTO: short-term optimization

In a shorter time-scale, rate control consists in determining well controls and platform settings that optimize the daily or weekly production. In this case, since the dynamics in the reservoir are considerably slower than in the gathering network, it is common to assume the reservoir at steady-state, and disregard the long-term effects of current actions (Codas, Campos, et al., 2012; Krishnamoorthy, Foss, and Skogestad, 2016; Camponogara et al., 2017; Hülse et al., 2020). Thus, the system of difference equations (2a) is replaced with a static curve known as the Inflow Performance Relationship (IPR), which describes the relationship between the well BHP and its flow rates, and needs to be adjusted routinely to match field measurements (Jahanbani and Shadizadeh, 2009). The short-term rate control problem in a given time step $k$ can

be formulated as follows:

$$\min_{u_k} \quad -\sum_{i\in\mathcal{N}_{\text{w,prod}}} (\hat{q}_{o,i,k+1} r_o - \hat{q}_{w,i,k+1} r_w) \tag{5a}$$

$$\text{s.t.} \quad \hat{q}_{l,i,k+1} = a_{i,k} u_{i,k} + b_{i,k}, \qquad \forall i \in \mathcal{N}_{\text{w,prod}}, \tag{5b}$$

$$\hat{q}_{w,i,k+1} = \hat{q}_{l,i,k+1} \gamma_{i,k}, \qquad \forall i \in \mathcal{N}_{\text{w,prod}}, \tag{5c}$$

$$\hat{q}_{o,i,k+1} = \hat{q}_{l,i,k+1}(1-\gamma_{i,k}), \quad \forall i \in \mathcal{N}_{\text{w,prod}}, \tag{5d}$$

$$\Delta\hat{P}_{k+1} = h(u_k, \hat{q}_{l,k+1}) \geq 0, \tag{5e}$$

$$u_{\text{lb},i,k} \leq u_{i,k} \leq u_{\text{ub},i,k}, \qquad \forall i \in \mathcal{N}_{\text{w,prod}}, \tag{5f}$$

where the control $u_{i,k}$ is the BHP and $\hat{q}_{l,i,k+1}$, $\hat{q}_{w,i,k+1}$, and $\hat{q}_{o,i,k+1}$ are the flow rates of well $i$ obtained with a linear IPR with coefficients $a_{i,k}$ and $b_{i,k}$. The IPR can be assumed linear because there is no gas being produced at reservoir conditions. The parameter $\gamma_{i,k}$ is the proportion of water in the liquid flow rate, also known as water cut. Further, the lower bounds $u_{\text{lb},i,k}$ are typically defined as the pressure at the inlet of the topside facilities, and the upper bounds $u_{\text{ub},i,k}$ are the minimum bottom-hole pressure given as reservoir boundary conditions. In practice, this is a safe margin to avoid the injection of oil into the reservoir. Each well has a valve at the top, called choke, which can adjust the rates to ensure feasibility with respect to the network infrastructure. The pressure drops over the chokes $\Delta\hat{P}_{i,k+1}$, $\forall i \in \mathcal{N}_{\text{w,prod}}$ are calculated by a network simulation procedure, here denoted by $h(u_k, \hat{q}_{l,k+1})$, and must be positive such that the chokes do not become virtual pumps.

## 2.3 Long-term vs. short-term optimization

The rate control of reservoirs in the long-term is often performed in a resolution of a couple of months to a year, meaning that the same settings should be kept at a constant value for a long time. A review of methods for long-term production optimization may be found in J. Jansen (2011). In practice, such controls are typically changed every week, daily or even several times a day on a platform. Further, because many constraints are ignored while performing long-term rate control planning, these controls may not be feasible with respect to the gathering facilities. Among the attempts to overcome this issue is a framework to handle output constraints using a multiple shooting paradigm (Codas, Foss, and Camponogara, 2015), and a further extension to handle network constraints (Silva et al., 2019).

While it is common to treat RTO and reservoir management separately, the limitations of standalone approaches are evident: in the long-term, a coarse solution is obtained at a high computational cost, while the short-term solutions are greedy and disregard long-term effects of current actions. We propose an optimization procedure that considers both short- and long-term gains while keeping the computational effort low, thereby enabling the use of the proposed method in real-world decision-making workflows.

# 3 Optimal control problems with fast and slow dynamics

In this section, a mathematical formulation of the problem is presented followed by two fairly standard approaches to tackle it. A conceptual formulation of an optimal control problem with fast and slow dynamics is given below:

$$\min_{u(\cdot)} \quad \int_{t_0}^{t_f} \alpha L_c^s(x(t), z(t), u(t)) + (1-\alpha) L_c^l(x(t), z(t), u(t))\, dt \tag{6a}$$

$$\text{s.t.} \quad \dot{x}(t) = f_c(x(t), z(t), u(t)), \qquad \forall t \in [t_0, t_f], \tag{6b}$$

$$\dot{z}(t) = g_c(x(t), z(t), u(t)), \qquad \forall t \in [t_0, t_f], \tag{6c}$$

$$0 \leq h_c(x(t), z(t), u(t)), \qquad \forall t \in [t_0, t_f], \tag{6d}$$

where $L_c^s$ and $L_c^l$ are the possibly conflicting economical objectives for the short-term and long-term perspective, respectively, $f_c(\cdot)$ and $x$ represent the dynamics and states of the system with slow dynamics, respectively, while $g_c(\cdot)$ and $z$ are the equivalent for the system with fast dynamics. Equation (6d) are the path constraints. The decision variables are the control inputs $u(t)$ $\forall t \in [t_0, t_f]$, and the objective function is given as the integral in (6a) with $\alpha \in [0, 1]$.

In this paper we focus on a modified version of (6) in which the fast dynamics are seen at steady-state conditions, i.e., $\dot{z}(t) = 0$. This assumption is made because we assume the slow dynamics to be much slower than the fast dynamics such that the transients of the fast dynamics system do not affect the integrated system. Thus, the fast dynamics are replaced with algebraic equations (cf. (7c) below). The resulting formulation can be written in discrete time domain as follows:

$$\min_{u} \sum_{\forall k \in \mathcal{K}} t_k \left[\alpha L^s(x_{k+1}, u_k) + (1-\alpha) L^l(x_{k+1}, u_k)\right] \tag{7a}$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k), \qquad \forall k \in \mathcal{K}, \tag{7b}$$

$$0 = g(x_k, u_k), \qquad \forall k \in \mathcal{K}, \tag{7c}$$

$$0 \leq h(x_k, u_k), \qquad \forall k \in \mathcal{K}, \tag{7d}$$

where $\mathcal{K} := [0, 1, \ldots, N-1]$ is the set of time steps. The functions in (7) are the discrete versions of the same functions in (6). The long-term rate control problem, given in the motivating example, in combination with the network gathering constraints in (5e) would be an example of (7) if we weigh the two objective functions (3) and (5a). Note that variables with subscript $k$ denote (in general) vector-valued quantities corresponding to the time instant $k$. The same variable without the subscript will be a matrix, e.g., $u = [u_0, u_1, \ldots, u_{N-1}]$ and $x = [x_1, x_2, \ldots, x_N]$. The control $i$ at the time step $k$ is denoted $u_{i,k}$ whereas $u_k$ indicates a vector of all these controls.

### 3.1 Solving the optimal control problem

A common way to solve the optimal control problem (6) is by a direct transcription method such as Multiple Shooting (MS) or Single Shooting (SS) (Biegler, 2010). However, these may run into computational limitations as the complexity is typically cubic in state- and input dimension, and time horizon.

### 3.2 Solving a sequence of quasi steady-state optimization problems

To circumvent the computational issues of solving the problem as a large dynamic optimization problem, one alternative is to decompose it into smaller and separate subproblems, one for each time step $k$, such that the integrated state from step $k$ becomes the initial state for step $k+1$. The formulation of an individual subproblem at time step $k$ resembles an RTO problem, and is given by:

$$\min_{u_k} \quad L^s(x_{k+1}, u_k) \tag{8a}$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k), \tag{8b}$$

$$0 = g(x_{k+1}, u_k), \tag{8c}$$

$$0 \leq h(x_{k+1}, u_k). \tag{8d}$$

Each subproblem (8) is a quasi steady-state optimization problem since the fast dynamics are assumed to be steady-state whereas the slow dynamics is integrated one step ahead. Notice that, if possible, it is only necessary to integrate the states that are needed to calculate the constraints and the objective function while solving the RTO. In the subproblem (5) in the motivating example, we use a linear approximation of the integration.

This approach greedily solves a sequence of quasi steady-state problems such that the optimized variables are purely based upon what is the best action in the current step. In other words, it is expected that the final value of (7a) will be worse than the one from a direct optimal control approach. On the other hand, the greedy strategy is computationally tractable since each subproblem is solved individually. As a result of each step being treated separately, this strategy avoids running into dimensionality issues due to an increase in the number of control steps. Thus, the computational bottleneck for this method can be taken as the time it takes to solve a single instance of (8).

## 4 Lookahead real time optimization

The greedy strategy is efficient but might yield considerable losses in the long-term objective in comparison to the direct optimal control solution. The latter, on the

other hand, might become computationally infeasible for a fine resolution in the discretization scheme. We propose a methodology that extends the greedy approach with a lookahead strategy in order to improve the cumulative objective (7a). The elements of the methodology and an algorithm with the solution method are provided in what follows.

## 4.1 The predictor

A key ingredient of the methodology is the predictor. It is used to predict the impact of changing the current controls, $u_k$, on the final value of the long-term objective, $\sum L^l(\cdot)$, in (7a):

$$P(u_k; k, N, x_k, u^{\text{init}}), \tag{9}$$

where $N$ is the total amount of time steps. The last parameter, $u^{\text{init}}$, is a matrix containing an initial guess for the optimal controls over the whole time period $k \in \mathcal{K}$. This is needed since it is required to simulate the slow dynamics from step $k$ to $N$ to find the impact of changing $u_k$. For simplicity, the size of $u^{\text{init}}$ is fixed. The predictor is a scalar function.

As an example of a predictor, which will be used in this work, consider a linear predictor which can be used to estimate the impact of changing $u_k$ on the long-term objective, $L^l(\cdot)$, in (7a). The linear predictor can be obtained by perturbing $u_k$, as described in Algorithm 1.

**Algorithm 1** Linear Predictor

**Require:** $x_k, u^{\text{init}}, e_k$.
**Ensure:** $x_k, u^{\text{init}}$ are feasible w.r.t. (7b)-(7d).
1: **for** $i \leftarrow 1, 2, \ldots, n_u$ **do**
2: $\quad u^{\text{copy}} \leftarrow u^{\text{init}}$
3: $\quad u^{\text{copy}}_{i,k} \leftarrow u^{\text{copy}}_{i,k} + e_{i,k}$
4: $\quad \hat{\Delta}_{i,k} \leftarrow \dfrac{I(x_k, u^{\text{copy}}, k, N) - I(x_k, u^{\text{init}}, k, N)}{e_{i,k}}$
5: **end for**
6: $P(u_k; k, N, x_k, u^{\text{init}}) \leftarrow (u_k - u^{\text{init}}_k)^T \hat{\Delta}_k$

The perturbation vector $e_k$ has the same size of the control vector $u_k$. To calculate the impact of changing one element in $u_k$, the slow dynamics are simulated from $x_k$ to $x_N$. This requires the use of an initial guess containing the controls for the whole horizon, $u^{\text{init}}$, for the matrix $u$. We assume here that this guess is feasible with respect to all the constraints in (7). The notation $u^{\text{init}}_{i,k}$ denotes the element $i$ of the controls at step $k$ of the initial guess $u^{\text{init}}$. Further, the function $I(x_k, u, k, N) = \sum_{i=k}^{N-1} L^l(x_{i+1}, u_i)$ where the future states of the slow dynamics are obtained by integration.

## 4.2 Conflicting objectives

The predictor consists of an estimate of how the value of the long-term objective changes with respect to the current control actions, while the short-term objective prioritizes immediate profit. To deal with these possibly conflicting objectives, we propose a two-stage optimization approach. The first stage consists of optimizing the short-term objectives by solving the quasi steady-state RTO (8). We denote the integrated state and optimized control actions by $x'_{k+1}$ and $u'_k$. The second stage is performed with the lookahead, i.e., using the predictor $P(u_k; k, N, x_k, u^{\text{init}})$, but allowing only a certain deviation from the objective function value obtained with the RTO. The second optimization stage can be formulated as:

$$\min_{u_k} \quad P(u_k; k, N, x_k, u^{\text{init}}) \tag{10a}$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k), \tag{10b}$$

$$0 = g(x_{k+1}, u_k), \tag{10c}$$

$$0 \leq h(x_k, x_{k+1}, u_k), \tag{10d}$$

$$L^s(x_{k+1}, u_k) \leq \beta L^s(x'_{k+1}, u'_k), \tag{10e}$$

where the additional constraint (10e) imposes a limit on the deviation from the first-stage RTO objective $L^s(x'_{k+1}u'_k)$. Note that $L^s(x'_{k+1}u'_k)$ is assumed to be negative. The parameter $\beta \in [0, 1]$ determines how much the short-term objective function $L^s(\cdot)$ may worsen, e.g., $\beta = 0.9$ implies an acceptable worsening of 10%. The objective function in (10a) is the predictor replacing the first stage RTO objective.

In a given time step $k$, the solution obtained at the first stage RTO (8) is optimal with respect to the objective (8a). Thus, the solution obtained with the Lookahead RTO (LRTO) is expected to worsen the instantaneous objective in exchange for a potential improvement in the long-term objective $\sum L^l(\cdot)$ in (7a). An alternative to solve the two-stage optimization problem is to combine the two, possibly conflicting, objective functions into one by introducing weights. However, by doing so, the guarantee given by the parameter $\beta$ would be lost. This guarantee is of interest because it limits the risk taken by reducing the near future objectives. This risk reflects that it is likely that the models used in optimization of $L^s(x'_{k+1}u'_k)$ are more certain than those used for calculation of the predictor (9).

## 4.3 The algorithm

The building blocks of the proposed methodology are put together in a two-stage optimization method described in Algorithm 2. For a given initial state $x_0$ and nominal path $u^{\text{init}}$ that are feasible with respect to the slow dynamics system (8b), the algorithm first solves the RTO to obtain the greedy solution $u'_k$. Then a predictor is calculated and used in the second stage optimization to obtain the current optimized control $u^*_k$. The calculated control is applied to the plant (or simulator) to obtain the

next state. The process is repeated until the end of the control interval. Notice that Lookahead RTO is defined as the procedure performed at lines 2 to 4 of Algorithm 2.

**Algorithm 2** The workflow of Lookahead RTO

**Require:** $x_0, u^0$.
**Ensure:** $x_0, u^0$ are feasible w.r.t. (8b).
1: **for** $k \leftarrow 0, 1, \ldots, N-1$ **do**
2: $x'_{k+1}, u'_k \leftarrow$ Solve RTO in (8).
3: Obtain the predictor, (9).
4: $u^*_k \leftarrow$ Solve modified RTO in (10) with $x'_{k+1}, u'_k$.
5: $x_{k+1} \leftarrow$ Apply $u^*_k$ to the plant.
6: **end for**

# 5 Simulation analysis

The Lookahead RTO method is assessed in the rate control optimization problem described in Section 2. We consider an oil production system consisting of an oil-water reservoir, a production gathering network, and the topside facilities.

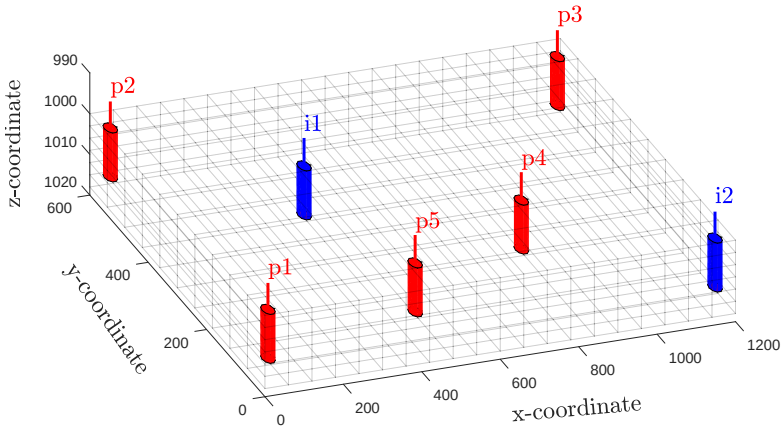


Fig. 3: Illustration of the reservoir model.

## 5.1 The simulator setup

The reservoir model is an adapted case instance from Roysem and Alfsen (2012), which is based on the SPE1 benchmark, and implemented in the Matlab Reservoir

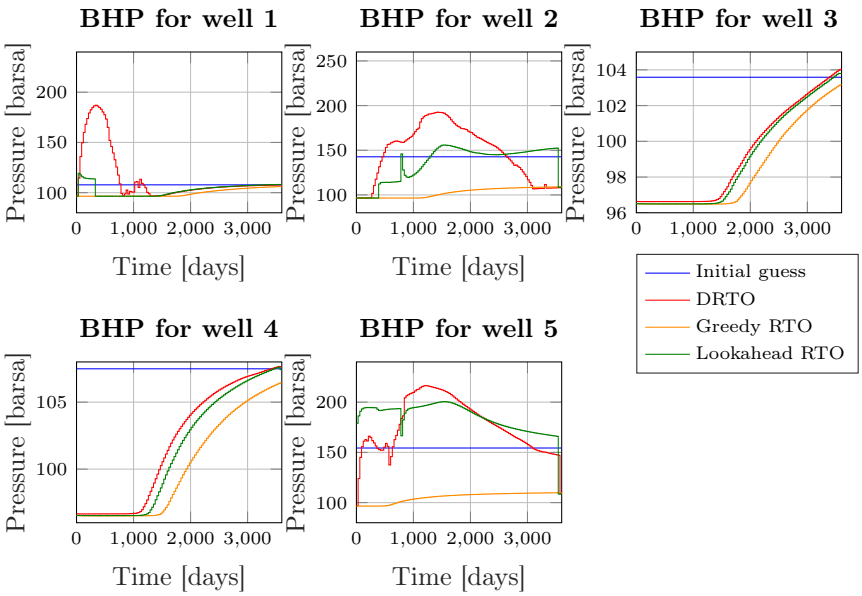


Fig. 4: Different control strategies.

Simulation Toolbox 2019a (Lie, 2019). As illustrated in Fig. 3, it consists of a Cartesian three-dimensional box, split in 1200 regular grid blocks being $20 \times 10 \times 6$ in x, y, and z coordinates, respectively. Each block is 60 meters long, 60 meters wide and 3 meters high. The fluid utilized is the incompressible black-oil model, and the initial oil pressure $p_{\text{oil},i,0}$ is set to 200 bar, while the water saturation $S_{\text{water},i,0}$ is set to 0.15 in all grid blocks $i \in \mathcal{G}$. The field has 5 production wells and 2 water injectors, while its life time is taken as 10 years of 360 days, with time steps of 30 days, resulting in 120 time steps. We chose such a coarse resolution to maintain the problem computationally feasible for the aforementioned dynamic optimization approach.

The gathering network has 5 wells producing to a common manifold at the inlet of the topside facilities. Each producer is equipped with a choke at the top of the well, i.e. the well-head, which allows the regulation of the well flow. The set of pipelines transporting the fluids are divided into the production tubings and the flowlines. The tubings are pipes in the well bore through which the reservoir fluids are produced. The tubings of all wells have the same features, a length of 250 meters, diameter of 76 millimeters, inclination of 90 degrees, i.e. they are vertical. The flowlines are the pipes that take the fluids from wells to the platform. Each well has a separate flowline with the same features, a total length of 1000 meters, diameter of 240 millimeters, and an inclination angle of 55 degrees. The downstream boundary condition of the network is a constant pressure of 5 bar at the inlet of the processing facilities.

The fluid flow in the gathering network is modeled and simulated with a network

solver developed in Silva et al. (2019). The pressure drops are calculated using the framework for nodal analysis in J. D. Jansen (2017), which consists of several correlations to compute fluid properties and pressure-drop derivatives. The calculations are performed by an integrator that utilizes an interface to the ODE solver CVODES, which is part of Suite of Nonlinear and Differential/Algebraic Equation Solver (Sundials), version 2.6.2 (Hindmarsh et al., 2005).

The long-term objective given in (3) has the parameters $r_o, r_w$ and $r_i$ set to 60, 10 and 10 USD/stb, respectively. We chose $\alpha = 1$ and $L^s(\cdot) = L^l(\cdot)$ in (7a) such that the stage costs of LRTO, RTO and the dynamic optimization method remain the same. Note that the objectives are still conflicting, due to the different horizons. This selection implies that the injectors are not a part of the decision variables. The injection rates of wells "i1" and "i2" are set to 302.0778 $m^3/d$ and 88.7446 $m^3/d$, respectively. The IPRs for the wells which are needed for the production optimization, as explained in Section 2, are identified by perturbation of the control inputs by -10 bar. This way we can find a linear approximation of what the flows will be at step $k+1$ given a control action $u_k$. The IPRs are reconstructed at each iteration of Algorithm 2. The separator pressure is used as the lower bound for the BHPs of the wells in all time steps. The upper bound for the BHP for each well is set to 99% of the minimum value of the pressures in the surrounding grid blocks. Just like the IPRs, these values will be updated at each iteration. Ideally, the upper bounds should be made for the state at $k+1$, but to avoid having to add a nonlinear constraint, the upper bound is created by first integrating along the nominal path for one step and then use this state to determine the upper bound. The reservoir pressure is a part of the slow dynamics, and thus this approximation is considered justified.

The second stage optimization problem for this experiment, solved at step 4 of Algorithm 2, will be similar to the one solved at the first stage, which is given the motivating example (5), but with the two following changes. The first is the additional constraint

$$-\sum_{i \in \mathcal{N}_w} (\hat{q}_{o,i,k+1} r_o - \hat{q}_{w,i,k+1} r_w) \leq \beta L(\hat{q}'_{o,k+1}, \hat{q}'_{w,k+1})$$

with $\beta = 0.75$. The second change is that the objective function is replaced by the linear predictor given in Algorithm 1:

$$P(u_k; k, N, x_k, u^{\text{init}}) = (u_k - u_k^{\text{init}})^T \hat{\Delta}_k.$$

A small modification was done to Algorithm 2. In addition to applying $u_k^*$ to the plant, it was also copied into $u_{k+1}^{\text{init}}$. This change was done due to the assumption that the controls should not aggressively change from one step to another. This would provide a better linearization point for the linear predictor and the IPRs.

## 5.2 Results

Four different strategies are compared in the simulation analysis. In addition to the methods Greedy RTO and Lookahead RTO addressed in the paper, two different strategies obtained by a multiple shooting framework Silva et al., 2019 are presented. The first, named DRTO, is configured to have a 30-day step size. In the second trajectory each well will keep the same control for 10 years. This trajectory is used as an initial guess for the other strategies and is named Initial guess.

The quality of the prediction obtained with the linear predictor described in Algorithm 1, with a perturbation size of 1.5 bar, is measured as the ratio of the actual change to the predicted change, which means that a value of 1 indicates a perfect prediction. Some characteristics of the quality of the linear predictor are given in Tab. 1. The predictor performed well on average, which justifies its choice as the predictor for the experiments.

Table 1: Measurements of the quality of the linear predictor.

| **Min** | **Max** | **Mean** | **RMSE** |
|---|---|---|---|
| -2.34 | 3.70 | 1.01 | 0.47 |

The different trajectories for the controls for the four strategies are shown in Fig. 4. The controls in the Greedy RTO case are all changing smoothly. This is because the only constraint is that the pressure loss over the choke has to be greater than zero and the wells still produce a good amount of oil at the end, and thus $q_{o,i,k+1}r_o \geq q_{w,i,k+1}r_w$ throughout the 10 year period. This implies that it is most valuable to simply produce as much as possible as the production network allows at all times. If $\Delta P = 0$, then maximum flow rate is achieved. At the very end of the simulation we can see that there is a large jump in the control for both well 2 and well 5 for the Lookahead RTO strategy, this is due to the fact that the sign of the linear predictor changes and the greedy strategy aligns with the predictor. Phrased differently, the short-term and long-term objectives are aligned.

The values of the long-term goal for the different strategies may be seen in Fig. 5. We can see, not unexpectedly, that the Greedy RTO strategy gives a higher income in the beginning. Around day 1000, the Lookahead RTO takes the lead and remains most profitable until around day 2600. From this point forward, the DRTO strategy is the best. The two most interesting comparisons are the two RTO approaches, and the Lookahead RTO vs. DRTO. The solution obtained with the Lookahead RTO method is 7.1 % (or 16.3 million USD) better NPV than the greedy RTO. When compared to DRTO, the Lookahead RTO method achieved a solution that is only 0.4% (or 1.1 million USD) lower in terms of NPV. Although the original objective of Lookahead RTO was to improve the greedy strategy, which is similar to daily routines in real world platforms, it also performed almost as well as DRTO.

To illustrate the differences in terms of production achieved by the different methods,

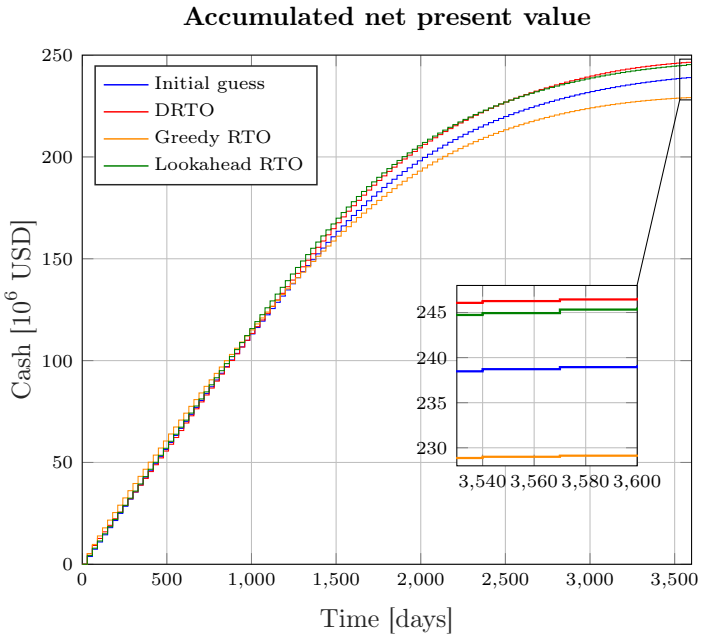


Fig. 5: Accumulated NPV for the different control strategies.

the accumulated flows of oil and water are plotted in Fig. 6. The Greedy RTO approach produces the least oil and most water. The Lookahead RTO and the DRTO approaches produces almost identical amounts of water throughout the entire horizon, implying that the shape of the NPV curves and accumulated oil flow curves will be similar due to (3). This means that, with the lookahead strategy, it is possible to avoid the excessive water production that the greedy method causes, while maintaining a low computational effort.

The solving time for the most computationally heavy iteration of the lookahead method was less than 1 minute, while the DRTO took around 3.5 hours. Keep in mind that we are using a toy reservoir with a long step size. The computational efficiency of the lookahead method allows its use in a RTO fashion in daily operations, while, in a similar manner, using the multiple shooting approach in a Model Predictive Control fashion would probably not be reasonable. Further, the entire trajectory for the Lookahead RTO in the figures took less than 1 hour to compute.

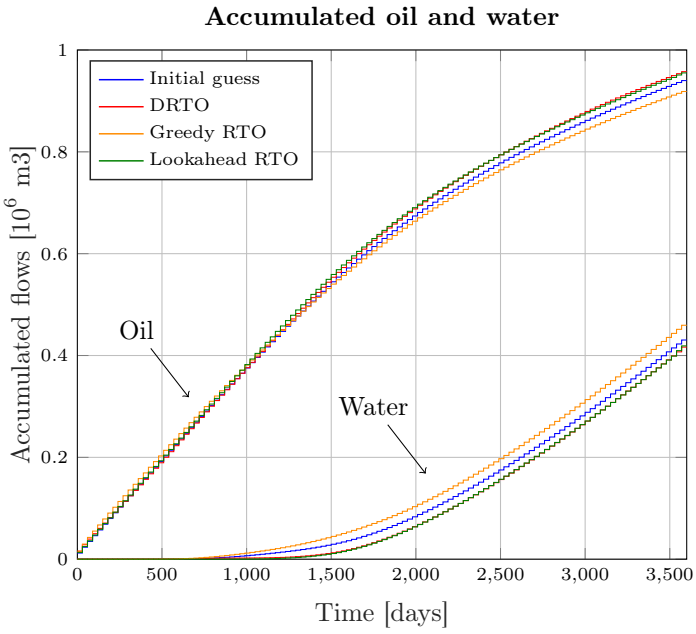


Fig. 6: Accumulated flows for the different control strategies.

# 6 Discussion

The suggested methodology is computationally inexpensive because it only uses the simulator of the slow dynamics to create the predictors and to evolve in time. Since the simulator is not embedded into the smaller optimization problems, the method does not inherit its computational complexity and the problem dimensionality is kept low. This strength enables the use of Lookahead RTO in similar fashion to the standard RTO.

These days it is becoming more common to have easily available aggregated plant data. The LRTO may incorporate such information in the same manner as RTO. For example, if we gather data and new knowledge about the system, we may update the models in between iterations of Algorithm 2. Further, the $u^{\text{init}}$ trajectory may also be updated if we obtain a better model for the slow dynamics.

The $\beta$ parameter indicates how much of a worsening of the first stage objective function is accepted. It could also be tuned based upon how much we trust the model for the slow dynamics, or the projected oil price. If the slow dynamics are highly uncertain, then this parameter could be increased. This parameter can also be used in another way. Assume we have a good model for the slow dynamics. It would make sense to allow for a larger deviation in the beginning, because there is less future

to consider when the end is getting closer. In this sense, the $\beta$ could be gradually increased as time passes. This would result in a more greedy approach towards the end.

One issue one might run into when using Algorithm 2 is infeasibility in the slow dynamics. The initial guess, $u^{\text{init}}$, is feasible. However, after solving (10) we change $u_k^*$ ($\neq u_k^{\text{init}}$). The fact that $u^{\text{init}} = [u_0^{\text{init}}, u_1^{\text{init}}, \ldots, u_{N-1}^{\text{init}}]$ is feasible does not imply feasibility of $[u_0^*, u_1^{\text{init}}, \ldots, u_{N-1}^{\text{init}}]$. This could perhaps be solved by, e.g., by running the multiple shooting method again with a coarser resolution such that the infeasibility is avoided. Note that this optimization should happen after one has applied $u_k^*$ to the plant, but before a new iteration of Algorithm 2 is started.

A consequence of solving the optimal control problem in (7) as a sequence of smaller optimization problems, instead of one larger problem, is that the predictor, which is based upon a guess of the future control actions, will not be perfect unless this guess is followed.

# 7 Conclusions

In the simulation study, it was shown that including the long-term effects into the short-term decision making process was valuable. The existing methods in literature, to optimize systems with both fast and slow dynamics, may suffer from the computational requirements due to the dimensionality. Disregarding the long-term effects gives rise to a method that is fast but non-optimal in the long run. The suggested methodology combines the advantages of the traditional real time optimization with a lookahead strategy to improve the long-term goal. The simulation study shows promising results for this novel methodology.

# References


Biegler, L. T. (2010). *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. Vol. 10. Siam.

Camponogara, E. et al. (2017). "Integrated methodology for production optimization from multiple offshore reservoirs in the Santos Basin". In: *IEEE Trans. Autom. Sci. Eng.* 14.2, pp. 669–680.

Chen, Z., G. Huan, and Y. Ma (2006). *Computational methods for multiphase flows in porous media*. Vol. 2. Siam.

Codas, A., S. R. V. Campos, et al. (2012). "Integrated production optimization of oil fields with pressure and routing constraints: the Urucu field". In: *Computers & Chemical Engineering* 46, pp. 178–189.

Codas, A., B. Foss, and E. Camponogara (2015). "Output-constraint handling and parallelization for oil-reservoir control optimization by means of multiple shooting". In: *SPE Journal* 20.04, pp. 856–871.

Darby, M. L. et al. (2011). "RTO: An overview and assessment of current practice". In: *Journal of Process control* 21.6, pp. 874–884.

Findeisen, W. et al. (1980). *Control and coordination in hierarchical systems*. Wiley.

Foss, B. (2012). "Process control in conventional oil and gas fields—Challenges and opportunities". In: *Control Engineering Practice* 20.10, pp. 1058–1064.

Hindmarsh, A. C. et al. (2005). "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers". In: *ACM Transactions on Mathematical Software (TOMS)* 31.3, pp. 363–396.

Hülse, E. O. et al. (2020). "Introducing approximate well dynamics into production optimization for operations scheduling". In: *Computers & Chemical Engineering* 136, p. 106773.

Jahanbani, A. and S. Shadizadeh (2009). "Determination of Inflow Performance Relationship (IPR) by well testing". In: *Canadian International Petroleum Conference*. Petroleum Society of Canada.

Jansen, J. (2011). "Adjoint-based optimization of multi-phase flow through porous media – A review". In: *Computers & Fluids* 46.1, pp. 40–51.

Jansen, J. D. (2017). *Nodal Analysis of Oil and Gas Production Systems*. Society of Petroleum Engineers.

Krishnamoorthy, D., B. Foss, and S. Skogestad (2016). "Real-Time Optimization under Uncertainty Applied to a Gas Lifted Well Network". In: *Processes* 4.4, p. 52.

Larsson, T. and S. Skogestad (2000). "Plantwide control - A review and a new design procedure". In: *Modeling, Identification and Control* 21.4, pp. 209–240.

Lie, K.-A. (2019). *An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST)*. Cambridge Univ. Press.

Rangaiah, G. P. and V. Kariwala (2012). *Plantwide control: Recent developments and applications*. Wiley.

Roysem, E. H. and M. Alfsen (Dec. 2012). "Optimization of investment strategy related to subsea water separator using reservoir simulation". MA thesis. Norwegian Univ. of Science and Technology.

Silva, T. L. et al. (2019). "Network-constrained production optimization by means of multiple shooting". In: *SPE Reservoir Evaluation & Engineering* 22.02, pp. 709–733.

Skogestad, S. (2000). "Plantwide control: The search for the self-optimizing control structure". In: *Journal of process control* 10.5, pp. 487–507.

— (2004). "Control structure design for complete chemical plants". In: *Computers & Chemical Engineering* 28.1-2, pp. 219–234.

Van Essen, G., P. Van den Hof, and J.-D. Jansen (2011). "Hierarchical long-term and short-term production optimization". In: *SPE Journal* 16.01, pp. 191–199.

Zavala, V. M. and A. Flores-Tlacuahuac (2012). "Stability of multiobjective predictive control: A utopia-tracking approach". In: *Automatica* 48.10, pp. 2627–2632.

# C Application of Data-Driven Economic NMPC on a Gas Lifted Well Network

**Andersen, J. R.**, Imsland, L., "Application of Data-Driven Economic NMPC on a Gas Lifted Well Network". In: *IFAC-PapersOnLine* 54.3 (2021), pp. 275–280. DOI: https://doi.org/10.1016/j.ifacol.2021.08.254

## CRediT authorship contribution statement

**Joakim Rostrup Andersen:** Conceptualization, Software, Writing - Original Draft, Visualization.
**Lars Imsland:** Writing - Review & Editing, Supervision.

# Application of Data-Driven Economic NMPC on a Gas Lifted Well Network

Joakim Rostrup Andersen[1] and Lars Imsland[1]

[1]Department of Engineering Cybernetics, NTNU, Trondheim, Norway

**Abstract:** The Daily Production Optimization (DPO) problem is the task of maximizing production of hydrocarbons subject to operational constraints. Handling of uncertainty in model structure and parameters is of high importance to the usefulness of the solution. Ignoring these challenges will, most likely, render the solution either infeasible or the solution will not be an optimum of the plant. We suggest to apply a data-driven methodology to use state- and output-measurements from the plant to iteratively update the Optimal Control Problem (OCP) which are used to control the plant. The goal of the method is to tune the OCP such that the solution will go towards an optimum of the plant as the parameters are being updated. A Reinforcement Learning updating technique is used to update the optimization formulation.

## 1 Introduction

The oil and gas industry produce value by drilling wells to extract hydrocarbons which are trapped in subsurface reservoirs. The Daily Production Optimization (DPO) activity within this industry aims at optimizing production from these wells to maximize revenue, while obeying a set of production constraints. There are many different sources giving rise to constraints, the most typical being limitations in how much gas and water, typically extracted along with oil, that can be processed. If the downhole pressure in the wells are not high enough to lift the fluids to the topside at an acceptable flowrate, then artificial lift can be used. One commonly used artificial lift method is the injection of gas into the wellbore to lower the density of the fluid mixture. Since a production facility typically may have many wells, and a limited amount of available gas, this also gives rise to a constrained optimization problem, namely the allocation of available lifting gas. The controllable elements in production optimization are typically chokes (valves) to control flows, and the settings of compressors and pumps.

Mathematical programming is well suited to tackle the DPO problem (Foss, Knudsen, and Grimstad, 2018). Kosmidis, Perkins, and Pistikopoulos (2004) formulated the DPO problem as a mixed integer nonlinear programming problem, where the integers were used to piece wise linearize nonlinear functions. Kosmidis, Perkins, and

Pistikopoulos (2005) used integers to also decide the routing of the streams. Other works looking at the DPO as a mixed integer (non)linear programming problem are Misener, Gounaris, and Floudas (2009), Codas and Camponogara (2012), Silva and Camponogara (2014), and Grimstad et al. (2016).

Modelling of the multiphase reservoir inflow, wells, risers and topside facilities are a complex task, with limited information available of subsurface conditions. Thus, the models used in DPO are typically uncertain, which may lead to poor prediction capabilities. This uncertainty is a serious challenge for the DPO problem (Foss, Knudsen, and Grimstad, 2018). Nonetheless, most works in production optimization simply ignores this issue altogether (Krishnamoorthy, Foss, and Skogestad, 2016). The quality and usefulness of the solution are heavily affected by such negligence. The solution could be infeasible for the real system, or it could be suboptimal.

Uncertainty in the DPO problem was first explicitly handled by Bieker, Slupphaug, Johansen, et al. (2007) according to Foss, Knudsen, and Grimstad (2018).

Dynamic scenario-based optimization was used to deal with parametric uncertainty by Krishnamoorthy, Foss, and Skogestad (2016) for production optimization of gas lifted wells. They reduced the conservativeness of the solution compared to the classical worst-case optimization, while still being robustly feasible.

To avoid the need of steady-state measurements, Krishnamoorthy, Foss, and Skogestad (2018) suggested to use transient measurements to perform model updating, whilst only using a steady-state model in the optimization. They demonstrated the hybrid method on a gas lifted well network simulation case study.

The modifier adaptation optimization method was applied to a similar setup by Matias, Le Roux, and Jäschke (2018), also considering parametric uncertainty. At convergence, the plant and model optimum coincided. Modifier adaptation has the advantage that both model structure and model parameter mismatch can be handled. However, the applied method needs a steady-state detection phase.

The contribution of this paper is the introduction of a methodology into the field of DPO. The method is able to compensate for both error in the model structure and uncertainty in parameters. Steady-state detection is not applicable as dynamic models are used. Similar to the modifier adaptation method, the goal is to use the data to optimally control the plant, and not necessarily to correctly model the system dynamics. The data-driven methodology was proposed by Gros and Zanon (2020). The method combines Economic Nonlinear Model Predictive Control (ENMPC) with Reinforcement Learning (RL). The ENMPC is used to control the system, whereas the RL is used to tune the ENMPC such that it more optimally controls the system.

# 2 Theory

In this Section, the relevant background theory will be provided. It gives an introduction to the relevant parts of Reinforcement Learning (RL). The newly suggested function approximator(Gros and Zanon, 2020) is put into perspective of the other parts, and an updating scheme for the function is provided. For a deeper understanding of RL, we refer to Sutton and Barto (2018).

## 2.1 Reinforcement Learning

Reinforcement Learning is a kind of machine learning, where an *agent* is supposed to learn how to operate a system to maximize a numerical reward signal. The agent is not explicitly told what the reward will be for taking an action, but it must instead try the action and learn from it. In Reinforcement Learning, there are three main components. The first component is the *reward* which is used as a metric to evaluate the goodness of the immediate return of the applied action in the current state. The second component is the *value function*, which gives the value of being in a state. It is the discounted sum of all future rewards given that we follow the *policy*. Finally, we have the *policy* which provides the next action to apply given a state. There is also a component known as the *action-value function*, which is the same as the value function except that it provides the value of a state given that we apply a specified action at the first step.

Given a state $s$, and an action $a$, we have the policy $\pi_\theta(s)$, value function $V_\theta(s)$ and action-value function $Q_\theta(s, a)$, parameterized by the parameter vector $\theta$. The relationship between them is given as follows:

$$V_\theta(s) = \min_a Q_\theta(s, a), \tag{1a}$$

$$\pi_\theta(s) = \arg\min_a Q_\theta(s, a). \tag{1b}$$

Let $s'$ denote the next state, and $r(s, a)$ denote the reward, then (1a) can be rewritten as the Bellman Equation:

$$V_\theta(s) = \min_a r(s, a) + \gamma V_\theta(s'), \tag{2}$$

where $\gamma$ is a discount factor.

Finding the optimal action-value function for all possible combinations of states and actions is in general not possible. However, a process is typically working in only a small subset of such combinations. The aim is to create an approximation of the (action-) value function in the operating region.

## 2.2 Action-value approximation

In this paper, we will use a parametric Economic Nonlinear Model Predictive Control (ENMPC) as the function approximator for the action-value function. This approximator was proposed by Gros and Zanon (2020), and extended with system identification capabilities by Martinsen, Lekkas, and Gros (2020). A parametric ENMPC is presented below.

$$
\begin{aligned}
Q_\theta(s,a) = \min_{x,u,\sigma} \; & \sum_{k=0}^{N-1} \left[\gamma^k (L_\theta(x_k,u_k) + \omega^\intercal \sigma_k)\right] \\
& + \gamma^N L_\theta^f(x_N) + \lambda_0(x_0) && \text{(3a)} \\
s.t. \quad & x_{k+1} = f_\theta(x_k,u_k) \text{ for } k \in [0,\dots,N-1] && \text{(3b)} \\
& h_\theta(x_k,u_k) \leq \sigma_k \text{ for } k \in [0,\dots,N-1] && \text{(3c)} \\
& x_0 = s && \text{(3d)}
\end{aligned}
$$

where $L_\theta(\cdot)$ is an estimated reward function, $\omega$ is used to penalize constraint violation given by the slack variable $\sigma$, $f_\theta(\cdot)$ is the dynamic model, $N$ is the control horizon, and $x_0$ is the initial state. Finally, $L_\theta^f(\cdot)$ is the terminal cost that should capture all future rewards, and $\lambda_0(x_0)$ is an initial cost which does not impact the solution of the optimization problem, but may help the RL. For the rest of the article, we will use $s, a$ to denote real states and actions, whereas $x, u$ will denote predicted states and inputs.

This choice of function approximator is actually rather intuitive. An MPC gives you the next action to apply given current state, just like the policy. It also provides the objective function value given current state, just like the value-function. If we impose an additional constraint: $u_0 = a$, then we get the same behaviour as an action-value function.

## 2.3 Goal of Reinforcement Learning

The overall goal in Reinforcement Learning (RL) is to find a policy $\pi_\theta(\cdot)$ that minimizes the following expected value:

$$
\mathbb{E} \sum_{i=0}^{\infty} \gamma^i \bar{L}(s_i, \pi_\theta(s_i)), \tag{4}
$$

where $\bar{L}$ is given as:

$$
\bar{L}(s_i,a_i) = L(s_i,a_i) + \hat{\omega}^\intercal \max(0, h(s_i,a_i)), \tag{5}
$$

where the first term is the experienced reward, and the latter is the penalization of any unwanted behaviour due to that experience. For example, in our case study

$L(s_i, a_i)$ is the negative value of the produced fluids, whereas the latter term represents any gas that must be flared due to excessive gas production. The weight $\hat{\omega}$ is used to tell the RL how critical it is to not violate the constraints $h(\cdot)$. It should be noted that this does not allow for hard constraints to be included since this would indicate a value of $+\infty$ in $\bar{L}(\cdot)$, which most RL methods cannot handle (Martinsen, Lekkas, and Gros, 2020). Moreover, having hard constraints in any MPC scheme may lead to an infeasible optimization problem. One could also argue that if one has state/output constraints that never can be violated, then either one must control the system extremely conservatively, or one would need a perfect model of the system and its constraints. And if one has the perfect model, then RL is superfluous either way.

## 2.4 Q-learning

To minimize (4) we will apply the concept of Q-learning (Watkins, 1989). The tuning of the action-value function, $Q_\theta(s, a)$, is driven by the minimization of what is known as the temporal-difference error $\delta_t$:

$$\delta_t = y_t - Q_\theta(s_t, a_t), \tag{6}$$

where $y_t$ is the *fixed target value* given by $y_t = \bar{L}(s_t, a_t) + \gamma V_\theta(s_{t+1})$. In the parameter updating step, it is assumed that $y_t$ is independent of $\theta$. This assumption makes the updating a bootstrapping strategy which means caution should be used when selecting the step size.

A batch of samples will be used each time the parameters are updated to help avoid overfitting. A Gauss-Newton method will be used to minimize the sum of squares:

$$\min_\theta \sum_{t=1}^{n_b} \delta_t^2, \tag{7}$$

where $n_b$ is the number of elements in the batch. The update law with learning rate, or step size, $\alpha$, is given by:

$$\theta \leftarrow \theta + \alpha (J_Q^\intercal J_Q)^{-1} J_Q^T \delta, \tag{8}$$

where

$$J_Q = \begin{bmatrix} \nabla_\theta Q_\theta(s_{t,1}, a_{t,1}) \\ \nabla_\theta Q_\theta(s_{t,2}, a_{t,2}) \\ \vdots \\ \nabla_\theta Q_\theta(s_{t,n_b}, a_{t,n_b}) \end{bmatrix}, \quad \delta = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_{n_b} \end{bmatrix}. \tag{9}$$

Inverting the approximate Hessian $H = J_Q^\intercal J_Q$ will lead to problems if the matrix is singular, or badly conditioned. To circumvent this issue, let $\tilde{H}^\dagger$ denote the "pseudo-inverse" of the symmetric $H$:

$$\tilde{H}^\dagger = \mathcal{F}(\mathcal{F}^\intercal H \mathcal{F})^{-1} \mathcal{F}^\intercal, \tag{10}$$

where $\mathcal{F}$ is chosen as the "near" fullspace of $H$, with the modification that the singular value of a direction has to be strictly greater than $\sigma_{\min}$ to be included in the fullspace. The new update-law then becomes:

$$\theta \leftarrow \theta + \alpha \tilde{H}^{\dagger} J_Q^T \delta. \tag{11}$$

This updating law does not guarantee that a global optimum of the parameters for the nonlinear $Q_\theta$ function is found. However, most applications of nonlinear function approximators in RL will suffer from this limitation (Martinsen, Lekkas, and Gros, 2020). Nonetheless, as long as the $Q_\theta$ function is fitted to the operating range of interest, a local optimum is sufficient. If the operating area changes, then we could apply online updating to try to fit the $Q_\theta$ function to the current operating range.

# 3 Simulation study

In this study, we will apply the above theory on a hydrocarbon production system consisting of two gas lifted wells, see Fig. 1, which produce to a common manifold with a fixed pressure of 50 bar. The goal is to distribute the available lift gas between the two wells such that the oil production is maximized. The processing facility also has a limitation on how much gas it can handle.

## 3.1 The model of a gas lifted well

The utilized gas lifted well model is explained in great details in Binder (2012). All constants may be found in that reference, we used well 2 and 4.

This model is based on mass balance of the different phases in the annulus and the tubing. This kind of model has been developed and studied in several papers, e.g., see Jahanshahi (2013), and Imsland (2002). The version used in this paper takes into account oil, water and gas phases, but flow friction is neglected.

The states of the model are the mass of gas in the annulus $m_{ga}$, mass of gas in the tubing $m_{gt}$, and mass of liquid in the tubing $m_{lt}$. The ordinary differential equations, which are based upon mass balance, for well $j$ is given below:

$$\dot{m}_{ga,j} = w_{gl,j}(u_{gl,j}) - w_{gi,j}(\cdot), \tag{12a}$$
$$\dot{m}_{gt,j} = w_{gr,j}(\cdot) + w_{gi,j}(\cdot) - w_{gp,j}(u_{pc,j}), \tag{12b}$$
$$\dot{m}_{lt,j} = w_{lr,j}(\cdot) - w_{lp,j}(u_{pc,j}), \tag{12c}$$

where $w_{gl,j}$ is the injected gas into the annulus, $w_{gi,j}$ is the gas going from the annulus into the tubing, $w_{gr,j}$ and $w_{lr,j}$ are the gas and liquid coming from the reservoir into the tubing, finally, $w_{gp,j}$ and $w_{lp,j}$ is the gas and liquid produced by the well. For each well, there are two manipulated variables: $u_{gl,j}$ and $u_{pc,j}$. The first is $u_{gl,j} = w_{gl,j}$

which is the amount of gas in kg/s injected into the annulus, and the second is $u_{pc,j} \in [0, 1]$ which is the production choke opening (cf. Fig. 1). In (12) we have indicated which flows directly depend upon the manipulated variables.

The model contains two one-way chokes to make sure that fluids from the tubing may not enter the annulus and to make sure that the well does not drain fluids from the manifold. E.g., the flow through the gas injection valve is modelled as:

$$w_{gi,j} = C_{iv,j}\sqrt{\rho_{gi,j}\max(0, p_{ai,j} - p_{ti,j})}, \tag{13}$$

where $C_{iv,j}$ is the valve specific constant for the injection valve, $\rho_{gi,j}$ is the density of the gas in the annulus at the injection valve, and $p_{ai,j}$ and $p_{ti,j}$ are the pressures at each side of the valve. The Optimal Control Problem (OCP) is implemented using CasAdi (Andersson et al., 2019), which allows to use max-functions in the formulation. To avoid the issue with an unbounded gradient of the square root in (13) when the argument is close to zero, we used a smooth approximation when $p_{ai,j} - p_{ti,j}$ becomes small.

If one would like to apply a constraint on, or penalize, how fast the control inputs may change, then one could augment the state space with two more states for each well:

$$\dot{u}_{gl,j} = v_{gl,j}, \tag{14a}$$
$$\dot{u}_{pc,j} = v_{pc,j}, \tag{14b}$$

where the $v$'s represent the rate of change of the controls and will be the new manipulated variables in the Model Predictive Control (MPC). The state vector $x$ contains five elements:

$$[m_{ga,j}, m_{gt,j}, m_{lt,j}, u_{gl,j}, u_{pc,j}],$$

and the control vector has two elements, $[v_{gl,j}, v_{pc,j}]$, for each well at each time step.

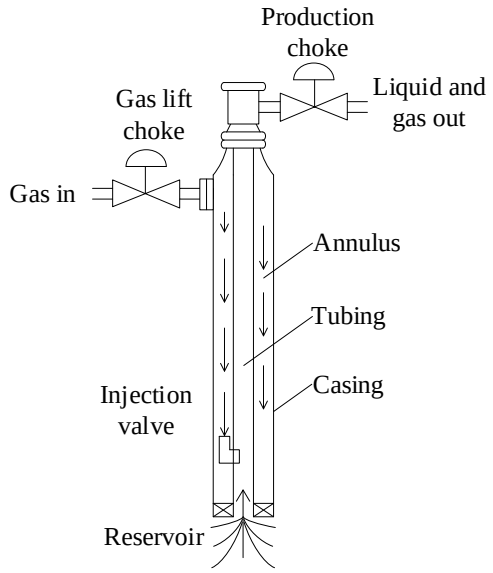


Figure 1: Schematic of the gas lifted well model.

### 3.2 The RL function approximator

Based on the theory outlined in the previous section, we use the function approximator $Q_\theta(a, s)$ implicitly defined by the solution of:

$$\min_{x,v,\sigma} \ \lambda_0 + \sum_{k=0}^{N-1} \left[\gamma^k (w_{\theta,op}(x_k, v_k) + \omega\sigma_{k+1})\right] + \gamma^N \frac{\gamma}{1-\gamma} w_{\theta,op}(x_{N-1}, v_{N-1}) \tag{15a}$$

$$s.t. \ \ x_{k+1} = f_\theta(x_k, v_k), \text{ for } k \in [0, \ldots, N-1] \tag{15b}$$

$$w_{\theta,gp}(x_k) + \lambda_1 \leq \Gamma + \sigma_k, \text{ for } k \in [1, \ldots, N] \tag{15c}$$

$$\underline{x} \leq x_k \leq \bar{x}, \text{ for } k \in [1, \ldots, N] \tag{15d}$$

$$\underline{v} \leq v_k \leq \bar{v}, \text{ for } k \in [1, \ldots, N] \tag{15e}$$

$$x_N \text{ is a steady-state} \tag{15f}$$

$$x_0 = s \tag{15g}$$

where $w_{\theta,op}(x_k, v_k)$ is the integrated negative oil production from both wells from step $k$ to $k+1$, $w_{\theta,gp}(x_k)$ is the instantaneous gas production at step $k$, and $f_\theta(x_k, v_k)$ is defined by (12). The gas handling constraint is enforced by (15c), where $\Gamma = 3.8$kg/s is the maximum gas that should be produced at a time instant, and $\lambda_0$ and $\lambda_1$ are two tunable bias parameters. The control horizon was set to $N = 100$ steps, over a

simulation time $T = 3600$s (1 hour), giving a time step of 36s. The penalty factor was set $\omega = 1000$, and the discount factor was set to $\gamma = 0.98$.

We impose (15f) to make sure the system reaches a steady-state. We do this in the example by requiring $x_N = x_{N-j}$, for $j \in [1, \ldots, 5]$. The steady-state allows us to use the property of a geometric series to efficiently express the discounted terminal cost:

$$\sum_{k=N}^{\infty} \gamma^k = \gamma^N \frac{\gamma}{1-\gamma}, \quad \text{for } |\gamma| < 1, \tag{16}$$

multiplied by the oil produced at the last interval. Note that constraint (15f) could lead to infeasibility issues, if that was encountered the simulation time $T$ and control horizon $N$ could be extended.

The lower bounds, $\underline{x}$, for the masses and production chokes were set to zero, and for the gas injection it was set to 1.2 and 0.1 kg/s for well 2 and 4, respectively. The upper bounds, $\bar{x}$ for all states were set to $\infty$ except for the production chokes that had an upper bound of 1. The upper and lower bounds for all the $v$'s were set to $\pm 0.25/(T/N)$ to limit aggressive behaviour.

For this case study, we chose four tunable parameters in the function approximator (15). Two of them are directly related to the optimization formulation: $\lambda_0$ and $\lambda_1$. For each well, the valve specific constant for the injection valve, see Fig. 1, is selected as a parameter that the RL can tune. This means that the parameter vector $\theta$ has the elements $C_{iv,2}$, $C_{iv,4}$, $\lambda_0$, and $\lambda_1$. This selection of parameters is not unique and other choices could give better, or worse, performance.

### 3.3 Implementation details

The optimal control problem (OCP) was implemented using CasADi (Andersson et al., 2019) version 3.5.0 with IPOPT version 3.12.3 (Wächter and Biegler, 2006) as the nonlinear program (NLP) solver. IPOPT was ran with default parameters except that maximum iterations was increased to 5000. Direct Collocation, with Legendre collocation points and a polynomial order of degree of 2, was used as a direct transcription method, see e.g., Biegler (2010) for more information on the method.

For testing, a plant replacement model was implemented using the same model, but with somewhat different parameters, as can be seen in Table 1. The plant was integrated with the explicit Runge-Kutta 4 method. Two integration steps were used for each control step. The penalty factor in (5) was set to $\hat{\omega} = 20000$, the $h$ was taken as the instantaneous excessive gas production at $t+1$, and the $L$ was set to the accumulated oil production from $t$ to $t+1$.

For the RL, we used a moving window of 50 elements. The batch was chosen to always include the newest point, in addition to 7 uniformly randomly selected from the window. The step size, or learning rate, was set to $\alpha = 0.01$. The treshold for counting a singular value as zero was set to $\sigma_{\min} = 0.1$.

Scaling was applied to the state vector, the control vector, the objective function and the constraints in the NLP to obtain the same order of magnitudes. When performing the update law (11), we only performed scaling on the tunable parameters such that the gradients of the Lagrangian of the NLP with respect to the parameters were approximately of the same magnitude.

Sometimes when performing the Gauss-Newton step, we experienced that the evaluation of the $Q_\theta$ function was unsuccessful. When this happened, we ignored that sample in the batch update and continued. There were 59, 7 and 1 batches where 1, 2 and 3 evaluations failed, respectively.

Table 1: Real and guessed values of the parameters.

| Parameter | Real value | Guessed value |
|---|---|---|
| $r_{\text{gor}}$ well 1 | 0.07 | 0.06 |
| $r_{\text{wc}}$ well 1 | 0.7 | 0.8 |
| $r_{\text{gor}}$ well 2 | 0.09 | 0.1 |
| $r_{\text{wc}}$ well 2 | 0.5 | 0.6 |
| $C$ both wells | 0.8 | 0.7 |
| $C_{pc}$ both wells | 0.0016 | 0.00176 |
| $C_{iv}$ both wells | 0.00016 | 0.000144 |

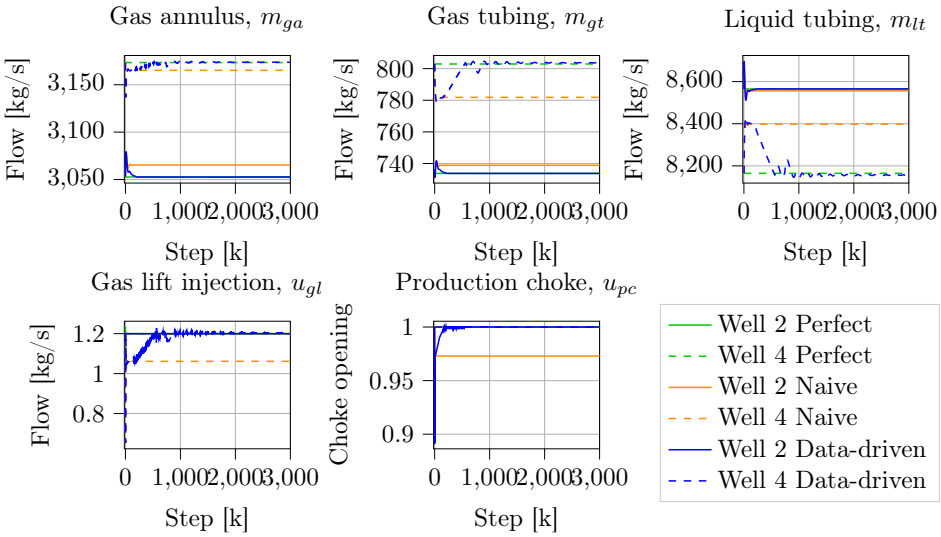


Figure 2: The states, including the controls, for the different strategies.

# 4 Results and discussion

In this Section, we present the results obtained in the simulation study. The method is compared to the "perfect" case where we have full knowledge of the model parameters, and also to a "naive" approach where an ENMPC was applied with wrong parameters, see Table 1, and nothing was done to improve the control model.

In Fig. 2, we see that as the RL keeps updating the parameter vector $\theta$, the MPC keeps suggesting another input. The data-driven ENMPC slowly goes towards the trajectories obtained by the perfect ENMPC.

In this specific application, two fully open production chokes and an active gas processing constraint are necessary for optimal operation. In Fig. 2, we can see that the chokes are fully open for both the ideal and data-driven approaches, but not for the naive one. In Fig. 3, we can see that the gas processing constraint is reached at the beginning by the ideal approach, and it is never reached by the naive approach. Further, we see that the constraint will eventually be reached also by the data-driven ENMPC. However, it does take some time for the RL to tune the parameters to some values that give a close to optimal control. It takes approximately 550 iterations before the constraint is active. The slow convergence is due to a small learning rate, or step size.

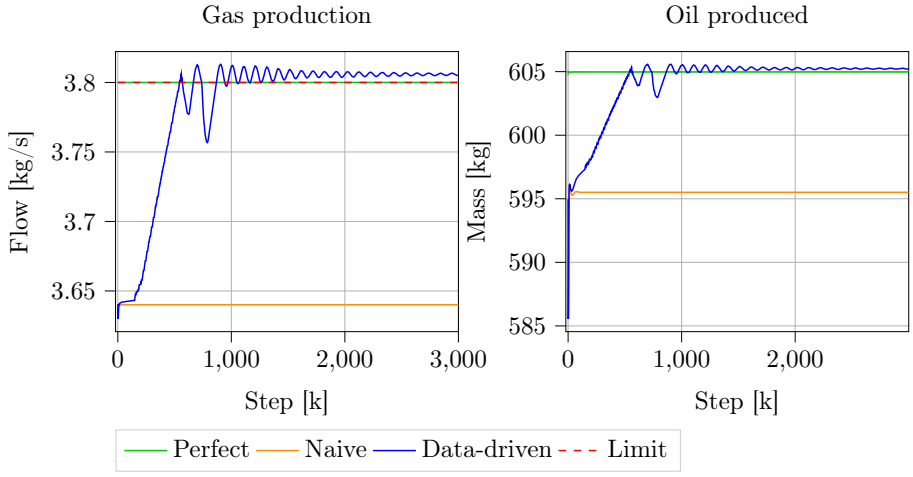


Figure 3: The gas production and produced oil for the different strategies.

If we increased the step size, then the constraint might be violated by a greater amount, leading to a large penalty-term. With a too small constraint violation penalty, $\hat{\omega}$, then the calculated optimal control input could be one that keeps violating the constraints. On the other hand, if it is too big, combined with a large step

size, it could lead to an updating step where the parameters are moved far away from the optimal values.

In Fig. 3, we can see that data-driven ENMPC does slightly violate the constraint. However, the amount is in the range of 6 grams of gas a second which most likely is unnoticeable due to non-perfect sensors. Further, it is reasonable to assume that operators would use a $\Gamma$ that is lower than the actual limitation of the plant. Increasing the $\hat{\omega}$ slightly and decreasing the step size would, most likely, decrease the gap between the produced gas and the limit $\Gamma$.

The integrated produced oil at each time interval is shown in Fig. 3. The figure shows that the data-driven ENMPC converges towards the same production as is achieved by the ENMPC with full knowledge of the system.

The evolution of the four tunable paramers can be seen in Fig. 4. The final values of the two valve specific constants are not the same as those in the "Real" column of Tab. 1. This is as expected, because the goal is to tune the $Q_\theta$ function such that we optimally control the system, and not to do system identification. If we set the two bias parameters to zero, and let the RL tune all the parameters in Tab. 1, then one may expect the parameters to converge to the real values. This could potentially also remove the small constraint violation in the gas processing capabilities.

# 5 Conclusion

We have studied a data-driven ENMPC method proposed by Gros and Zanon (2020), and applied it to a simulation study of two gas lifted oil wells. The results were promising in the sense that the data-driven ENMPC was able to tune the parameters to a set of values that gave a near-optimal operation of the process even though there were more incorrectly guessed parameters than tunable ones. It should be noted that the approach is a state-feedback approach, and relies on measurements of states that are not commonly available. In practice, this method must therefore be accompanied by a state estimation scheme.

# Acknowledgment

We would like to thank Andreas Bell Martinsen for sharing his valuable insight and knowledge on the methodology.

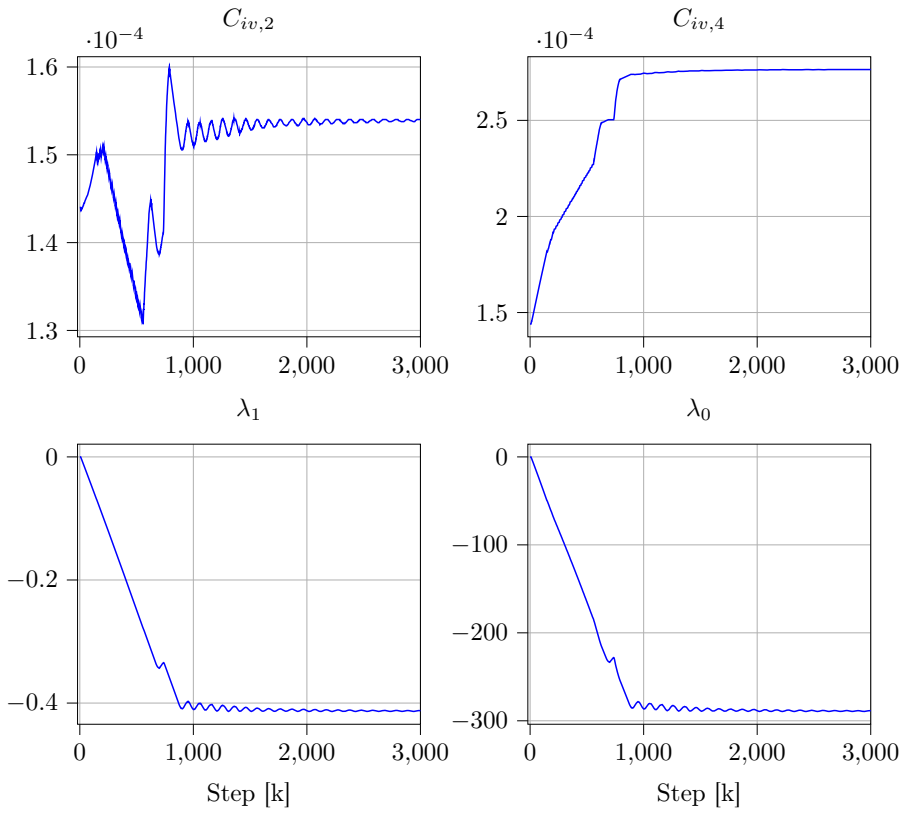


Figure 4: The evolution of the tunable parameters.

# References


Andersson, J. A. E. et al. (2019). "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1, pp. 1–36.

Biegler, L. T. (2010). *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM.

Bieker, H. P., O. Slupphaug, T. A. Johansen, et al. (Jan. 2007). "Well Management Under Uncertain Gas or Water Oil Ratios". In: *Digital Energy Conference and Exhibition*. Society of Petroleum Engineers.

Binder, B. J. T. (2012). "Production optimization in a cluster of gas-lift wells". Master's thesis. Institutt for teknisk kybernetikk, NTNU.

Codas, A. and E. Camponogara (2012). "Mixed-integer linear optimization for optimal lift-gas allocation with well-separator routing". In: *European Journal of Operational Research* 217.1, pp. 222–231.

Foss, B., B. R. Knudsen, and B. Grimstad (2018). "Petroleum production optimization – A static or dynamic problem?" In: *Computers & Chemical Engineering* 114. FOCAPO/CPC 2017, pp. 245–253.

Grimstad, B. et al. (2016). "Global optimization of multiphase flow networks using spline surrogate models". In: *Computers & Chemical Engineering* 84, pp. 237–254.

Gros, S. and M. Zanon (2020). "Data-Driven Economic NMPC Using Reinforcement Learning". In: *IEEE Transactions on Automatic Control* 65.2, pp. 636–648.

Imsland, L. (2002). "Topics in nonlinear control.: Output Feedback Stabilization and Control of Positive Systems". 112, 0809-103X. PhD thesis. Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Jahanshahi, E. (2013). "Control Solutions for Multiphase Flow: Linear and nonlinear approaches to anti-slug control". 271, 1503-8181. PhD thesis. Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Kosmidis, V. D., J. D. Perkins, and E. N. Pistikopoulos (2004). "Optimization of Well Oil Rate Allocations in Petroleum Fields". In: *Industrial & Engineering Chemistry Research* 43.14, pp. 3513–3527.

— (2005). "A mixed integer optimization formulation for the well scheduling problem on petroleum fields". In: *Computers & Chemical Engineering* 29.7, pp. 1523–1541.

Krishnamoorthy, D., B. Foss, and S. Skogestad (Dec. 2016). "Real-Time Optimization under Uncertainty Applied to a Gas Lifted Well Network". In: *Processes* 4.4, p. 52.

— (2018). "Steady-state real-time optimization using transient measurements". In: *Computers & Chemical Engineering* 115, pp. 34–45.

Martinsen, A. B., A. M. Lekkas, and S. Gros (2020). "Combining system identification with reinforcement learning-based MPC". In: *arXiv preprint arXiv:2004.03265*.

Matias, J. O., G. A. Le Roux, and J. Jäschke (2018). "Modifier adaptation for real-time optimization of a gas lifted well network". In: *IFAC-PapersOnLine* 51.8, pp. 31–36.

Misener, R., C. E. Gounaris, and C. A. Floudas (2009). "Global Optimization of Gas Lifting Operations: A Comparative Study of Piecewise Linear Formulations". In: *Industrial & Engineering Chemistry Research* 48.13, pp. 6098–6104.

Silva, T. L. and E. Camponogara (2014). "A computational analysis of multidimensional piecewise-linear models with applications to oil production optimization". In: *European Journal of Operational Research* 232.3, pp. 630–642.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Wächter, A. and L. T. Biegler (2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical programming* 106.1, pp. 25–57.

Watkins, C. J. C. H. (1989). "Learning from delayed Rewards". PhD thesis. King's College.

# D Parametrization of learning based MPC for process control

**Andersen, J. R.**, Gros, S., Imsland, L., "Parametrization of learning based MPC for process control". In: *Journal of Process Control* (2023). (Submitted)

## CRediT authorship contribution statement

**Joakim Rostrup Andersen:** Methodology, Software, Writing - Original Draft, Visualization.
**Sebastien Gros:** Conceptualization, Methodology, Writing - Original Draft, Writing - Review & Editing
**Lars Imsland:** Writing - Review & Editing, Supervision.

This paper is not yet published and is therefore not included.

# E Bilevel programming as a means of infinite weighting in regression problems

**Andersen, J. R.**, Imsland, L., "Bilevel programming as a means of infinite weighting in regression problems". In: *IFAC-PapersOnLine* 55.7 (2022), pp. 851–856. DOI: https://doi.org/10.1016/j.ifacol.2022.07.551

## CRediT authorship contribution statement

**Joakim Rostrup Andersen:** Conceptualization, Methodology, Software, Writing - Original Draft, Visualization.
**Lars Imsland:** Writing - Review & Editing, Supervision.

# Bilevel programming as a means of infinite weighting in regression problems

Joakim Rostrup Andersen[1] and Lars Imsland[1]

[1]Department of Engineering Cybernetics, NTNU, Trondheim, Norway

**Abstract:** Linear regression is concerned about fitting a model to a set of data. The weighted least squares method is a standard tool for performing linear regression. In this paper, we focus on the case when some of the samples are given priority over others. The residuals for these samples should be given an infinite weighting compared to other samples. However, due to numerical limitations, a weight which is finite but sufficiently large must be chosen instead. We suggest an alternative approach that in practice allows infinite weighting. This is achieved by reformulating the regression optimization problem as a bilevel program. The method is illustrated in a numerical example study. The example shows that, without needing to determine a weighting factor, the proposed method yields the same solution, up to numerical precision, as to the one obtained by using a large weight.

## 1 Introduction

Linear regression is concerned about fitting a model, which is linear in its parameters, to a data set of inputs $x_j$'s and outputs $y_j$'s. The least squares method is a widely used method for linear regression. It finds the parameter vector $p$ that minimizes the squared errors between predictions and measurements

$$\min_p \sum_{j=1}^{n} w_j (y_j - \hat{y}(x_j; p))^2 \tag{1}$$

where $\hat{y}(\cdot)$ is the model and $n$ is the number of samples. In the ordinary least squares method it is assumed that the variance for all samples is constant, yielding $w_j = 1$. An extension of this method is the weighted least squares, where the weights are determined based upon the corresponding estimated variance for the samples, yielding possibly varying $w_j$'s. Seen from a less mathematical perspective, the idea would be to assign higher weights for the samples that one trusts the most. This latter mindset is the one taken in this article.

This work is inspired by a curve estimation task encountered in the oil and gas industry. We look into the case when there are two sources of data regarding modeling of

oil flow rates from individual wells. First, the oil flow rate for each well are measured on an infrequent basis. Second, the total oil flow rate for all the wells are measured during production. In this paper, it is assumed that all measurements are taken at steady-states. Further, it is assumed that the output of the total oil rate sensor is of significant higher quality, due to easier measurement principles (single phase flow compared to multiphase flow for the individual wells) and better maintenance as its data is directly related to the amount of sold products and therefore fiscal reporting. Thus, samples from this sensor are considered superior.

In this paper, we propose to use bilevel programming as a tool to achieve prioritization in a weighted linear regression problem for this estimation task.

Bilevel programs are hierarchical optimization problems. There is the *upper-level* problem which has a constraint that involves the solution of another optimization problem. The second optimization problem is referred to as the *lower-level* problem. Both the upper-level and lower-level problems may have their own set of constraints. Bilevel programming is an active field of research, see Dempe (2020) for an overview on theory, algorithms and applications of bilevel optimization.

From the viewpoint of multi-objective optimization, the method we suggest has similarities with the lexicographic method, which also treats different objectives hierarchically.

This paper is organized as follows. In Section 2, the suggested method is presented including approaches to solve the resulting bilevel program. In Section 3, the proposed theory is applied to a small example, and its performance is compared to a weighted optimization problem. Finally, in Section 4, a conclusion is provided.

## 2 Bilevel programming as an infinite weighting scheme

In this section, the application of bilevel programming as a means of infinite weighting in regression problems will be presented. The parameter estimation is performed by minimizing the sum of squared residuals in a least squares manner. The formulations are extended to also include constraints. Lastly, strategies to solve the resulting bilevel program are provided.

A common parameter estimation situation would be the need to fit a model to a set of data. In this paper, we are looking into a situation where there are two models, two data sets, and three sets of parameters. Each model may have its own parameters, and in addition, the two models share some, or all, parameters. The two models are:

$$\hat{y}_p(\hat{x}) := \hat{y}(\hat{x}; p_s, \hat{p}) \tag{2a}$$
$$\tilde{y}_p(\tilde{x}) := \tilde{y}(\tilde{x}; p_s, \tilde{p}) \tag{2b}$$

where $p_s \in \mathbb{R}^{n_{p_s}}, \hat{p} \in \mathbb{R}^{n_{\hat{p}}}$ and $\tilde{p} \in \mathbb{R}^{n_{\tilde{p}}}$ are tunable parameter vectors, $p_s$ are shared

between the two models, and the $\hat{x} \in \mathbb{R}^{n_{\hat{x}}}$ and $\tilde{x} \in \mathbb{R}^{n_{\tilde{x}}}$ are input vectors. To keep the notation clean, the shorter and slightly misleading left hand side of (2) is used. Further, it is assumed that both models map to scalar outputs:

$$\hat{y}_p : \mathbb{R}^{n_{\hat{x}}} \times (\mathbb{R}^{n_{p_s}} \times \mathbb{R}^{n_{\hat{p}}}) \to \mathbb{R} \tag{3a}$$
$$\tilde{y}_p : \mathbb{R}^{n_{\tilde{x}}} \times (\mathbb{R}^{n_{p_s}} \times \mathbb{R}^{n_{\tilde{p}}}) \to \mathbb{R}. \tag{3b}$$

If there were no shared parameters, *i.e.*, $n_{p_s} = 0$, then these two models could be treated separately, *e.g.*, by solving two least squares problems. A straightforward extension of that idea to the case with $n_{p_s} \neq 0$ would be to solve a weighted least square problem:

$$\min_{p_s,\hat{p},\tilde{p}} \hat{w} \sum_{j=1}^{m_{\hat{y}}} (\hat{y}_j - \hat{y}_p(\hat{x}_j))^2 + \tilde{w} \sum_{j=1}^{m_{\tilde{y}}} (\tilde{y}_j - \tilde{y}_p(\tilde{x}_j))^2 \tag{4}$$

where the subindex $j$ indicates it is a data point. The values of the weights $\hat{w}$ and $\tilde{w}$ will have a significant impact on the final parameters. The focus of this paper will be where one of these two will have priority over the other. This corresponds to setting the corresponding weight "infinitely large". However, due to limitations in numerical computing, it may be challenging to find a value which gives a reliable desired behavior.

Another way of phrasing the desire of prioritizing one over the other, is to say that the second curve fitting may only happen in the subspace of those directions which does not impact the first curve fitting. Such a subspace may for example exist if there are few data points for the one with highest priority. In this case, the remaining degrees of freedom of the shared parameters may be taken by the other minimization objective. This idea can be posed as a bilevel program:

$$\min_{p_s,\hat{p}} \sum_{j=1}^{m_{\hat{y}}} (\hat{y}_j - \hat{y}_p(\hat{x}_j))^2 \tag{5a}$$
$$\text{s.t.} \quad \min_{p_s,\tilde{p}} \sum_{j=1}^{m_{\tilde{y}}} (\tilde{y}_j - \tilde{y}_p(\tilde{x}_j))^2 \tag{5b}$$

where the lower-level problem (5b) has the highest priority.

To make the parameter estimation procedure more flexible, constraints may be added

to the upper and lower problem:

$$\min_{p_s,\hat{p}} \sum_{j=1}^{m_{\hat{y}}} (\hat{y}_j - \hat{y}_p(\hat{x}_j))^2 \tag{6a}$$

$$\text{s.t.} \quad \hat{h}(\hat{x}; p_s, \hat{p}) \geq 0 \tag{6b}$$

$$\hat{g}(\hat{x}; p_s, \hat{p}) = 0 \tag{6c}$$

$$\min_{p_s,\tilde{p}} \sum_{j=1}^{m_{\tilde{y}}} (\tilde{y}_j - \tilde{y}_p(\tilde{x}_j))^2 \tag{6d}$$

$$\text{s.t.} \quad \tilde{h}(\tilde{x}; p_s, \tilde{p}) \geq 0 \tag{6e}$$

$$\tilde{g}(\tilde{x}; p_s, \tilde{p}) = 0 \tag{6f}$$

where (6b) and (6e) also allows for setting bounds on the variables. The interpretation that the upper-level problem is minimized in the subspace defined by the remaining degrees of freedom of the lower-level problem is not completely valid any longer as the constraints (6b)-(6c), which contain $p_s$, may restrict the lower-level's feasible area. However, the lower-level's objective function still has a higher priority than the upper-level one.

## 2.1 Solving the bilevel program

A common approach to tackle this type of problem is to replace the lower-level optimization problem by its KKT conditions, as suggested by Fortuny-Amat and McCarl (1981). In the rest of this paper, it is assumed that the lower-level problem is convex and that the Linear Independence Constraint Qualification (LICQ) (Nocedal and Wright, 2006) holds at the optimum. In this case, the KKT conditions are both necessary and sufficient conditions of optimality. The objective functions will be convex for any models that are linear in the parameters. It should be pointed out that the linearity is only required for the parameters and not the variables. For example, a polynomial model of any degree will satisfy this criteria. The KKT conditions for the lower-level problem:

$$\nabla_{p_s,\tilde{p}} \mathcal{L}(\tilde{x}, p_s, \tilde{p}, \lambda_{\tilde{h}}, \lambda_{\tilde{g}}) = 0 \tag{7a}$$

$$\tilde{h}(\tilde{x}; p_s, \tilde{p}) \geq 0 \tag{7b}$$

$$\tilde{g}(\tilde{x}; p_s, \tilde{p}) = 0 \tag{7c}$$

$$\lambda_{\tilde{h}} \geq 0 \tag{7d}$$

$$\lambda_{\tilde{h}} \odot \tilde{h}(\tilde{x}; p_s, \tilde{p}) = 0 \tag{7e}$$

where $\odot$ represents element-wise multiplication, $\lambda_{hl}$ and $\lambda_{gl}$ are the Lagrange multiplier vectors, and the Lagrangian is defined as:

$$\begin{aligned}\mathcal{L}(\tilde{x}, p_s, \tilde{p}, \lambda_{\tilde{h}}, \lambda_{\tilde{g}}) = &\sum_{j=1}^{m_{\tilde{y}}} (\tilde{y}_j - \tilde{y}_p(\tilde{x}_j))^2 \dots \\ &- \lambda_{\tilde{h}}^T \tilde{h}(\tilde{x}; p_s, \tilde{p}) \dots \\ &- \lambda_{\tilde{g}}^T \tilde{g}(\tilde{x}; p_s, \tilde{p}). \end{aligned} \tag{8}$$

The final optimization problem becomes:

$$\min_{p_s, \hat{p}, \tilde{p}, \lambda_{\tilde{h}}} \sum_{j=1}^{m_{\hat{y}}} (\hat{y}_j - \hat{y}_p(\hat{x}_j))^2 \tag{9a}$$

$$\text{s.t.} \quad \hat{h}(\hat{x}; p_s, \hat{p}) \geq 0 \tag{9b}$$

$$\hat{g}(\hat{x}; p_s, \hat{p}) = 0 \tag{9c}$$

$$\nabla_{p_s, \tilde{p}} \mathcal{L}(\tilde{x}, p_s, \tilde{p}, \lambda_{\tilde{h}}, \lambda_{\tilde{g}}) = 0 \tag{9d}$$

$$\tilde{h}(\tilde{x}; p_s, \tilde{p}) \geq 0 \tag{9e}$$

$$\tilde{g}(\tilde{x}; p_s, \tilde{p}) = 0 \tag{9f}$$

$$\lambda_{\tilde{h}} \geq 0 \tag{9g}$$

$$\lambda_{\tilde{h}} \odot \tilde{h}(\tilde{x}; p_s, \tilde{p}) = 0. \tag{9h}$$

The optimization problem in (9) is a Nonlinear Program (NLP), and a regular NLP solver may be used. However, the constraint (9h), which arises from the complementary conditions of the first order conditions of optimality of the lower-level problem, may be a tough challenge for the solver.

## 2.2 The complementary condition

The complementary condition, *e.g.*, $\lambda_i \cdot h_i = 0$, says that either $\lambda_i$ or $h_i$ is zero, or both. However, both may not be nonzero simultaneously. In the bilevel literature, there exists several ways to handle (9h). The two approaches given next were suggested by Fortuny-Amat and McCarl (1981).

### 2.2.1 Big-M reformulation

The complementary condition can be formulated using a binary variable. If the integer variable takes the value $z_i = 1$, then the constraint is active, and a value of 0 indicates it is inactive.

$$\lambda_i \cdot h_i = 0 \Longleftrightarrow \begin{cases} h_i \leq M_1(1 - z_i) \\ \lambda_i \geq 0 \\ \lambda \leq M_2 z_i \end{cases} \tag{10}$$

where $z_i \in \{0, 1\}$, and $M_1$ and $M_2$ are chosen "large enough" such that desirable behavior is achieved. If there is a known finite upper value of $h_i$, then this may be used as $M_1$. The other constant, $M_2$, is more challenging to determine.

### 2.2.2 Special Ordered Set of Type 1

Another approach, which avoids the determination of any big-M, involves categorizing variables within Special Ordered Sets of Type 1 (SOS1). At most one variable within a SOS1 may be nonzero. For each complementary condition, a SOS1 is created containing two variables: $\{h_i, \lambda_i\}$, and the solver must be informed of these sets.

## 3 Example study

The methodology was inspired by a part of the daily production optimization (DPO) challenge within the oil and gas industry. The DPO focuses on utilizing the production system in an optimal manner. The goal is to increase revenue while obeying constraints arising from the facility. See Fig. 1 for an illustrative setup. The fluid flow from a well is typically a mixture of oil, gas and water which must be separated into three single-phase streams by the facility. As a simple example of a DPO, the revenues come from selling the oil and gas, whereas the constraints could be on how much water and gas the facility may process.

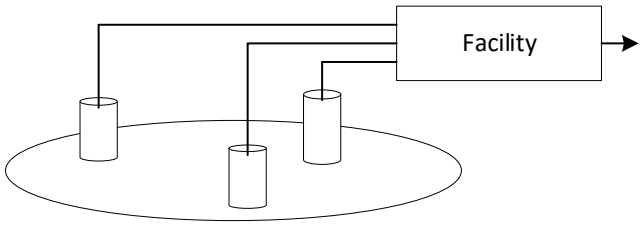


Figure 1: An example of a production network with one reservoir, three wells, and a production facility.

There are (at least) two different categories of oil flow rates. These will be discussed next.

First, a well's fluid production may be routed to what is known as a test separator. From the data collected at the test separator, the relationship between the downhole pressure, *i.e.* the pressure at the wellbore, and the oil flow rate can be estimated for that well. Because of a limitation on the availability of the test separator, the most recent test data for a well may be old.

Second, the total oil production, *i.e.* the oil production of all the wells, is measured during operation This measurement is often more accurate as it is both newer/more frequent and it is used to determine how much oil the platform exports which needs to be reported for fiscal reasons. The measurements of the downhole pressures for the wells are also available during operation.

Both for the well-based and total measurements, it is assumed that the wells must be in a steady-state before measurements can be taken. This may imply that there will be few recordings of the total oil production.

The problem that motivated this paper was the challenge of merging these two sources of measurements. In the rest of this section, we will study a constructed example where the proposed method is applied. The study focuses on utilizing the available data to estimate the relationship between the downhole pressure and gaslift rate for the wells.

## 3.1 Setup

In this setup, we have chosen to look at three wells. Each well has a linear model of its relationship between downhole pressure and oil production:

$$\hat{q}_i(P^w) = \alpha_i(P^w_{i,0} - P^w) + \beta_i \tag{11}$$

where $\alpha_i$ and $\beta_i$ are the two to-be-determined parameters, and $\hat{q}_i(P^w)$ is the predicted oil production for well $i$ at the downhole pressure $P^w$. Finally, $\beta_i$ is the predicted oil production at downhole pressure $P^w_{i,0}$. $P^w_{i,0}$ is set to 120,100 and 110 bar for the three wells. For each well test, it is assumed three data points. These points are given in Fig. 2 as the points marked by blue circles.

The upper-level objective function for the bilevel formulation in (6) is set as the sum of the squared prediction errors for the well tests of all three wells:

$$J_{ul} = \sum_{i=1}^{3}\sum_{j=i}^{3}\left(\hat{q}_i(P^w_{i,j}) - q_{i,j}\right)^2 \tag{12}$$

where $i$ and $j$ indicates the well and sample number, respectively.

The prediction of the total oil rate is a sum of these individual models:

$$\hat{q}_{\text{tot}}(\mathbf{P}_j) = \sum_{i=1}^{3}\hat{q}_i(P_{j,i}). \tag{13}$$

with $\mathbf{P}_j = [P_{j,1}, P_{j,2}, P_{j,3}]$. The lower-level objective function in (6) is set as the sum of the squared prediction errors of the total oil production:

$$J_{ll} = \sum_{j=1}^{n_{sl}} (\hat{q}_{\text{tot}}(\mathbf{P}_j) - q_{\text{tot},j})^2 \tag{14}$$

where $n_{sl}$ is the number of samples of the total oil rate. These samples are hereafter referred to as *system-level* samples. The system-level samples used in this example were arbitrarily chosen and are given in Tab. 1. These samples are assumed to be perfect. *I.e.*, both the sensors for downhole pressures and the sensor for the total oil rate have no type of measurement error.

Table 1: The table shows the different operation points for the system-level samples. Each column gives the data for one operation point, where the second to fourth rows give the downhole pressures in bar, and the last row tells the total oil production in Sm3/d.

| j | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Well 1 ($P_{j,1}$) | 117 | 96 | 100 | 100 | 90 |
| Well 2 ($P_{j,2}$) | 118 | 96 | 104 | 100 | 90 |
| Well 3 ($P_{j,3}$) | 119.5 | 99 | 108 | 100 | 95 |
| Oil ($q_{\text{tot},j}$) | 17.08 | 62.50 | 48.33 | 56.33 | 73.83 |

In this setup, there are only shared parameters, $n_{\tilde{p}} = n_{\hat{p}} = 0$ and $n_{p_s} = 6$. In the first example, there are no constraints.

The optimization formulations were formulated by using CasADi (Andersson et al., 2019) 3.5.5. The nonlinear programs were solved by Ipopt (Wächter and Biegler, 2006). The mixed integer nonlinear programs in Section 3.4 were solved by BONMIN (Bonami et al., 2008). No changes were made to the default settings of the solver.

## 3.2 Results

In the example, we started without any system-level samples in the curve fitting, and then added one sample at a time. First, the first column of Tab. 1 was added, then the second, and so on. The results are given in Fig. 2. For each well, we have plotted, in dashed-style, the resulting estimated linear relationships provided an increasing availability of system-level samples. The small numbers with arrows indicates the amount of system-level samples that was used in the estimation of that line. The graphs resulting from using 3,4 or 5 system-level samples all lie on top of each other. According to the figure, the slope of all the final lines are the same as those of the red lines. Comparing the blue lines, which are estimated purely based on the well-test data, with the dashed lines, it can be concluded that all the dashed lines provide a better approximation of the slopes of the red lines.

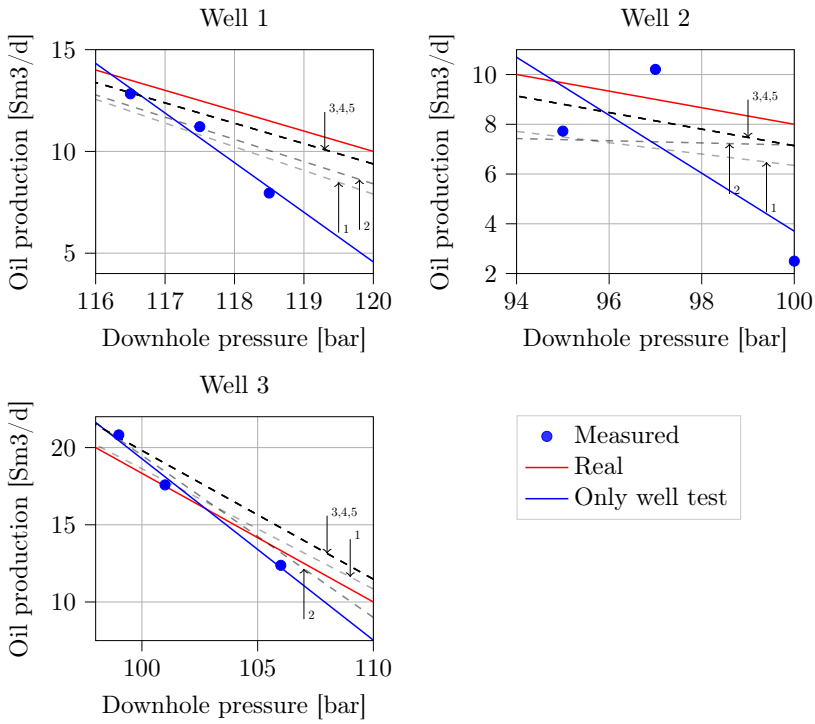


Figure 2: The blue filled circles are points measured during well tests. The blue lines are the linear relationship given only well-test data. The red lines are the real (unknown) relationships. The dashed lines are the estimated linear relationships given more and more system level points. The small numbers indicates the number of system level points used in the estimation.

It should be pointed out that the biases, the $\beta_i$'s, are not perfect. This is to be expected as the system-level samples does not provide information on the value of the individual $\beta_i$'s, only the sum of them. Having system-level samples where some of the wells are shut and other open, should improve the bias estimation. However, for the current example with 3 (or more) system-level samples the result is that the $\alpha_i$'s are determined by the lower-level problem, and also the sum of the biases. Then the distribution of this sum to the individual $\beta_i$'s happens through the minimization of the upper-level objective function which focuses on the well-test data.

In Fig. 3, the prediction of the total oil rate is plotted at a test point. The test point is $[119.5, 99.5, 109.5]$ bar which was chosen arbitrary except that it does not coincide with any of the system-level samples. The fact that the prediction does not change by adding more than 3 points can be explained by the lack of measurement error or

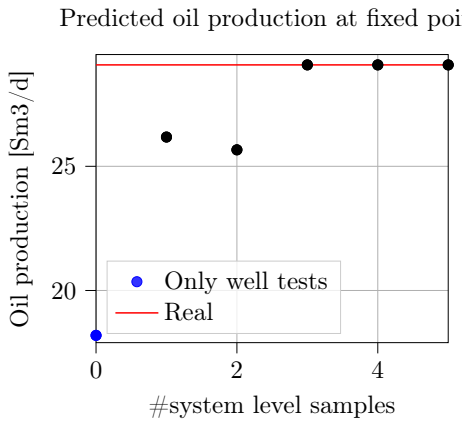


Figure 3: The red line is the real oil production at operation point $[119.5, 99.5, 109.5]$ bar. The blue dot is the prediction done with only well-test data, whereas the black also contains system level points. The number of samples is given on the x-axis.

noise in the total oil rate and the downhole pressures. From Tab. 1, it can be seen that any system-level point after the three first points must be a linear combination of the three previous ones. Thus, no more information can be extracted from the system for this setup unless wells are allowed to be shut.

### 3.3 Comparison with weighted optimization

As mentioned in Section 2, the suggested approach can be viewed as a way of performing infinite weighting in a weighted objective function. *E.g.*, $\hat{w} = 1$ and $\tilde{w} = \infty$ in (4). Here we will illustrate by example that solving this weighted minimization problem with an increasing $\tilde{w}$ will yield an almost identical solution as the one obtained by the bilevel programming. In this test, all the well-test samples and the system-level samples are used.

In Fig. 4 the value of (14) is plotted for different values of $\tilde{w}$. The weight start at $\tilde{w} = 10$ and increases by a factor of 10 up to the final value $\tilde{w} = 10^6$. The correct value of $J_{ll}$, the y-axis value, is zero. It should be zero because there is no measurement noise on the system-level samples, and both the model and the real relationship are linear. In other words, the prediction of the total oil rate should be perfect at the operation points given in Tab. 1. From the figure, it can be seen that the $J_{ll}$ keeps decreasing towards zero as the weight increases. Notice that the scales are logarithmic. The $J_{ll}$ from the weighting through bilevel approach gave a value of "numerically" 0, $\mathcal{O}(10^{-25})$.

As a side note, When increasing the weight to $10^{14}$, the solver fails. This is reason-

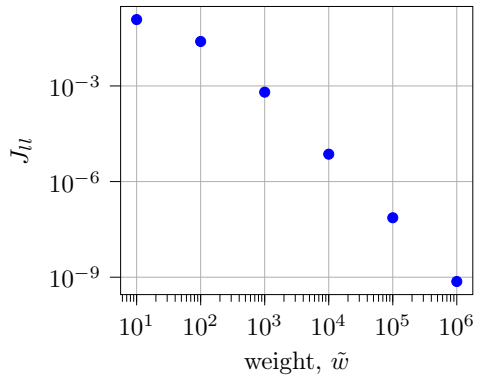


Figure 4: The objective function value of the lower-level problem converges towards the one resulting from the bilevel approach as the weighting factor increases.

able as the problem is ill-conditioned due to scaling. However, it highlights another weakness one may encounter when trying to find a "sufficiently" large weight.

To illustrate that the parameters found through the weighted optimization, with an increasing weight, converges towards those found by the bilevel program, the norm between the parameters are shown in Tab. 2. The norm decreases as the weight increases.

Table 2: The table shows the $\ell_2$-norm of the difference between the parameters obtained from solving the bilevel program and a weighted problem with the weight given in the first column. In the third column, the value of the lower-level objective function (14) is given.

| Weight | $\ell_2$-norm | $J_{ll}$ |
|---|---|---|
| $1.0 \cdot 10^{+01}$ | $1.85 \cdot 10^{+00}$ | $1.21 \cdot 10^{-01}$ |
| $1.0 \cdot 10^{+02}$ | $9.26 \cdot 10^{-01}$ | $2.48 \cdot 10^{-02}$ |
| $1.0 \cdot 10^{+03}$ | $1.5 \cdot 10^{-01}$ | $6.39 \cdot 10^{-04}$ |
| $1.0 \cdot 10^{+04}$ | $1.6 \cdot 10^{-02}$ | $7.24 \cdot 10^{-06}$ |
| $1.0 \cdot 10^{+05}$ | $1.61 \cdot 10^{-03}$ | $7.33 \cdot 10^{-08}$ |
| $1.0 \cdot 10^{+06}$ | $1.61 \cdot 10^{-04}$ | $7.34 \cdot 10^{-10}$ |

## 3.4 Example with constrained bias parameter

In this example the proposed method of weighting through bilevel programming is applied on a similar setup as in Section 3.1. Everything remains the same except that there is introduced a constraint on the bias parameter belonging to Well 2: $\beta_2 \geq 7.8$. The real bias value is 8. All the well-test samples and system-level samples are used

for the curve fitting.

The Big-M method mentioned in Section 2 was used to deal with the complementary condition. The $M$'s in (10) were set to $M_1 = M_2 = 100$. This value was large enough to give the correct behavior.

The resulting linear predictors are shown in Fig. 5. The bias parameter of Well 2 is at the given constraint: $\beta_2 = 7.8$. As before, the slopes of the estimated lines and the red lines are all the same according to the figure. The lines for both Well 1 and Well 3 is shifted downwards. By looking at the figures, one may be convinced that these two shifts do indeed reduce the prediction error with respect to the well-test data; the green lines are moved "closer to" the blue circles.

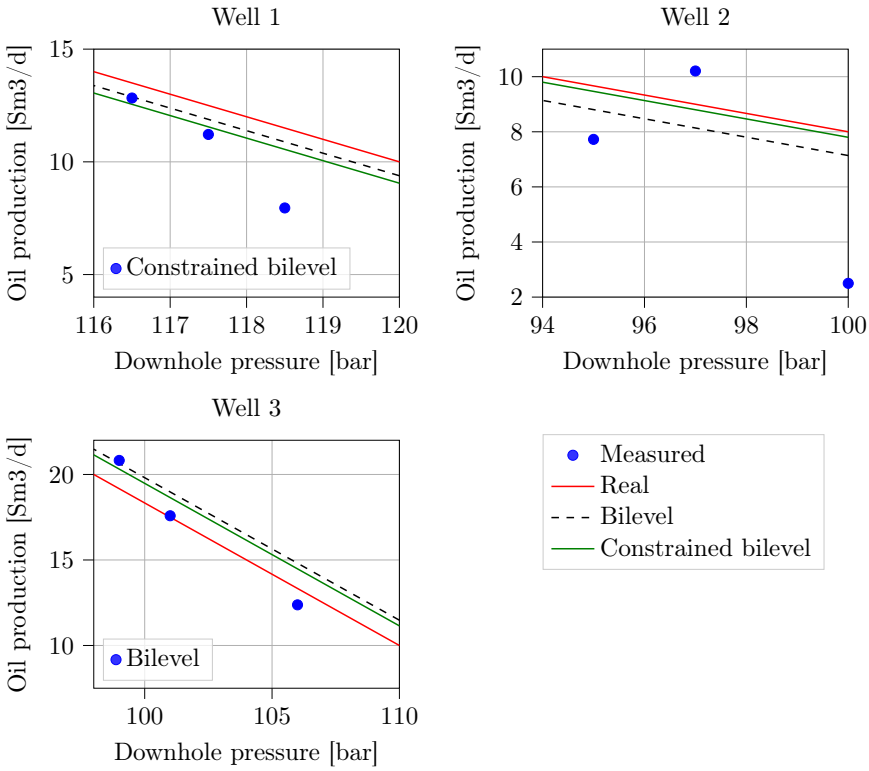


Figure 5: The blue filled circles are points measured during well tests. The red lines are the real (unknown) relationships. The dashed lines are those obtained with no constraints. The green lines were estimated with the constraint $\beta_2 \geq 7.8$

.

# 4 Conclusion

This paper presented an application of bilevel programming where the formulation was used to introduce infinite weighting. In the studied example, we saw that the proposed method allows for prioritization without having to determine a weighting factor. A disadvantage of the proposed method, compared to the weighting approach, is that integer decision variables will be introduced if there are constraints on the lower-level problem and a Mixed Integer Nonlinear Program (MINLP) solver is required instead of an NLP solver. MINLP solvers are typically slower than its integer-free counterpart. Nonetheless, the method removes issues with ill-conditioned formulations due to a large weight. Further, as no weight is used, the tuning process is eliminated.

# References


Andersson, J. A. E. et al. (2019). "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1, pp. 1–36.

Bonami, P. et al. (2008). "An algorithmic framework for convex mixed integer nonlinear programs". In: *Discrete Optimization* 5.2. In Memory of George B. Dantzig, pp. 186–204.

Dempe, S. (2020). "Bilevel optimization: theory, algorithms, applications and a bibliography". In: *Bilevel Optimization*. Springer, pp. 581–672.

Fortuny-Amat, J. and B. McCarl (1981). "A Representation and Economic Interpretation of a Two-Level Programming Problem". In: *The Journal of the Operational Research Society* 32.9, pp. 783–792.

Nocedal, J. and S. Wright (2006). *Numerical optimization*. Springer Science & Business Media.

Wächter, A. and L. T. Biegler (2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical programming* 106.1, pp. 25–57.

# References


[1] Foss, B., Knudsen, B. R., Grimstad, B., "Petroleum production optimization – A static or dynamic problem?" In: *Computers & Chemical Engineering* 114 (2018). FOCAPO/CPC 2017, pp. 245–253. DOI: https://doi.org/10.1016/j.compchemeng.2017.10.009.

[2] Skogestad, S. "Control structure design for complete chemical plants". In: *Computers & Chemical Engineering* 28.1 (2004). Escape 12, pp. 219–234. DOI: https://doi.org/10.1016/j.compchemeng.2003.08.002.

[3] Darby, M. L., Nikolaou, M., Jones, J., Nicholson, D., "RTO: An overview and assessment of current practice". In: *Journal of Process Control* 21 (2011), pp. 874–884.

[4] Krishnamoorthy, D. "Novel Approaches to Online Process Optimization Under Uncertainty: Addressing the limitations of current industrial practice". PhD thesis. NTNU, 2019.

[5] **Andersen, J. R.**, Imsland, L., "Data-driven derivative-free trust-region model-based method for resource allocation problems". In: *Computers & Chemical Engineering* (2023). (Submitted).

[6] **Andersen, J. R.**, Gros, S., Imsland, L., "Parametrization of learning based MPC for process control". In: *Journal of Process Control* (2023). (Submitted).

[7] **Andersen, J. R.**, Lima Silva, T., Imsland, L., Pavlov, A., "Real time optimization of systems with fast and slow dynamics using a lookahead strategy". In: *2020 59th IEEE Conference on Decision and Control (CDC)* (2020), pp. 2342–2349. DOI: 10.1109/CDC42340.2020.9304460.

[8] **Andersen, J. R.**, Imsland, L., "Application of Data-Driven Economic NMPC on a Gas Lifted Well Network". In: *IFAC-PapersOnLine* 54.3 (2021), pp. 275–280. DOI: https://doi.org/10.1016/j.ifacol.2021.08.254.

[9] **Andersen, J. R.**, Imsland, L., "Bilevel programming as a means of infinite weighting in regression problems". In: *IFAC-PapersOnLine* 55.7 (2022), pp. 851–856. DOI: https://doi.org/10.1016/j.ifacol.2022.07.551.

[10] Katoh, N., Shioura, A., Ibaraki, T., "Resource Allocation Problems". In: *Handbook of Combinatorial Optimization*. New York, NY: Springer New York, 2013, pp. 2897–2988. DOI: 10.1007/978-1-4419-7997-1_44.

[11] Conn, A. R., Scheinberg, K., Vicente, L. N., *Introduction to derivative-free optimization*. SIAM, 2009. DOI: 10.1137/1.9780898718768.

[12] Powell, M. J. "A direct search optimization method that models the objective and constraint functions by linear interpolation". In: *Advances in optimization and numerical analysis*. Springer, 1994, pp. 51–67.

[13] Powell, M. J. "The BOBYQA algorithm for bound constrained optimization without derivatives". In: *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge* 26 (2009).

[14] Giuliani, C. M., Camponogara, E., Plucenio, A., "A computational analysis of nondifferentiable optimization: Applications to production maximization in gas-lifted oil fields". In: *2013 IEEE Int. Conf. Autom. Sci. Eng. (CASE)*. 2013, pp. 286–291. DOI: 10.1109/CoASE.2013.6653975.

[15] Giuliani, C. M., Camponogara, E., "Derivative-free optimization with use of problem structure: Applications to oil production". In: *2015 IEEE Int. Conf. Autom. Sci. Eng. (CASE)*. 2015, pp. 764–768. DOI: 10.1109/CoASE.2015.7294173.

[16] Giuliani, C. M., Camponogara, E., "Derivative-free methods applied to daily production optimization of gas-lifted oil fields". In: *Comput. Chem. Eng.* 75 (2015), pp. 60–64. DOI: 10.1016/j.compchemeng.2015.01.014.

[17] Biegler, L. T. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. Vol. 10. Siam, 2010.

[18] Codas, A., Foss, B., Camponogara, E., "Output-constraint handling and parallelization for oil-reservoir control optimization by means of multiple shooting". In: *SPE Journal* 20.04 (2015), pp. 856–871.

[19] Silva, T. L., Codas, A., Stanko, M., Camponogara, E., Foss, B., "Network-constrained production optimization by means of multiple shooting". In: *SPE Reservoir Evaluation & Engineering* 22.02 (2019), pp. 709–733.

[20] Faulwasser, T., Grüne, L., Müller, M. A., "Economic Nonlinear Model Predictive Control". In: *Foundations and Trends® in Systems and Control* 5.1 (2018), pp. 1–98. DOI: 10.1561/2600000014. URL: http://dx.doi.org/10.1561/2600000014.

[21] Chachuat, B., Srinivasan, B., Bonvin, D., "Adaptation strategies for real-time optimization". In: *Computers & Chemical Engineering* 33.10 (2009), pp. 1557–1567. DOI: https://doi.org/10.1016/j.compchemeng.2009.04.014.

[22] Marchetti, A., Chachuat, B., Bonvin, D., "Modifier-Adaptation Methodology for Real-Time Optimization". In: *Industrial & Engineering Chemistry Research* 48.13 (2009), pp. 6022–6033. DOI: 10.1021/ie801352x.

[23] Marchetti, A. G., François, G., Faulwasser, T., Bonvin, D., "Modifier Adaptation for Real-Time Optimization—Methods and Applications". In: *Processes* 4.4 (2016). DOI: 10.3390/pr4040055. URL: https://www.mdpi.com/2227-9717/4/4/55.

[24] Faulwasser, T., Pannocchia, G., "Toward a Unifying Framework Blending Real-Time Optimization and Economic Model Predictive Control". In: *Industrial & Engineering Chemistry Research* 58.30 (2019), pp. 13583–13598. DOI: 10.1021/acs.iecr.9b00782.

[25] Vaccari, M., Bonvin, D., Pelagagge, F., Pannocchia, G., "Offset-Free Economic MPC Based on Modifier Adaptation: Investigation of Several Gradient-Estimation Techniques". In: *Processes* 9.5 (2021). DOI: 10.3390/pr9050901.

[26] Gros, S., Zanon, M., "Data-Driven Economic NMPC Using Reinforcement Learning". In: *IEEE Transactions on Automatic Control* 65.2 (2020), pp. 636–648. DOI: 10.1109/TAC.2019.2913768.

[27] Zanon, M., Gros, S., "Safe Reinforcement Learning Using Robust MPC". In: *IEEE Transactions on Automatic Control* 66.8 (2021), pp. 3638–3652. DOI: 10.1109/TAC.2020.3024161.

[28] Gros, S., Zanon, M., "Learning for MPC with stability & safety guarantees". In: *Automatica* 146 (2022), p. 110598. DOI: 10.1016/j.automatica.2022.110598.

[29] Nocedal, J., Wright, S., *Numerical optimization*. Springer Science & Business Media, 2006.

[30] Bennett, K. P., Kunapuli, G., Hu, J., Pang, J.-S., "Bilevel Optimization and Machine Learning". In: *Computational Intelligence: Research Frontiers: IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, China, June 1-6, 2008, Plenary/Invited Lectures*. Ed. by Jacek M. Zurada, Gary G. Yen, and Jun Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 25–47. DOI: 10.1007/978-3-540-68860-0_2. URL: https://doi.org/10.1007/978-3-540-68860-0_2.

[31] Mitsos, A., Bollas, G. M., Barton, P. I., "Bilevel optimization formulation for parameter estimation in liquid–liquid phase equilibrium problems". In: *Chemical Engineering Science* 64.3 (2009), pp. 548–559. DOI: https://doi.org/10.1016/j.ces.2008.09.034. URL: https://www.sciencedirect.com/science/article/pii/S0009250908005113.

ISBN 978-82-326-5577-9 (printed ver.)
ISBN 978-82-326-6850-2 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)





NTNU

Norwegian University of
Science and Technology