POLITECNICO DI MILANO

School of Industrial and Information Engineering

Department of Chemistry, Materials and Chemical Engineering
"Giulio Natta"

Master of Science in Chemical Engineering

Distributed Optimization of a Gas-Lifted Wells Network using Transient Measurements

SUBPRO

Subsea Production and Processing

Supervisors :

Flavio MANENTI

Sigurd SKOGESTAD

Dinesh KRISHNAMOORTHY

Candidate :

Carlo VALLI

Matr. n. 899932

Academic Year 2018 – 2019

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols and Acronyms

Symbols - Roman letters

A_a	Cross-section area of anulus
A_{bh}	Cross-section area of well at bottom hole
A_w	Cross-section area of well
C_{iv}	Flow coefficient for downhole injecton valve
C_{pc}	Flow coefficient for production valve
D_a	Diameter of annulus
D_{bh}	Diameter of well at bottom hole
D_w	Diameter of well
GOR	Gas-Oil Ratio
$\overline{GOR}$	Average Gas-Oil Ratio
g	Acceleration of gravity constant
H_a	Height of annulus
H_{bh}	Height of well at bottom hole
H_w	Height of well
J	Cost function
L_a	Length of annulus
L_{bh}	Length of well at bottom hole
L_w	Length of well
M_w	Molecular weight of gas
m_{ga}	Mass of gas in annulus
m_{gt}	Mass of gas in tubing
m_{ot}	Mass of oil in tubing

n_w	Number of wells
n_{w1}	Number of wells first cluster
n_{w2}	Number of wells second cluster
PI	Reservoir productivity index
p_a	Annulus pressure
p_{bh}	Bottom hole pressure
p_m	Manifold pressure
p_r	Reservoir pressure
p_{wh}	Wellhead pressure
p_{wi}	Well injection point pressure
R	Gas constant
T_a	Temperature in annulus
T_w	Temperature in well tubing
w_{gl}	Gas lift rate
w_{iv}	Well injection flow rate
w_{pc}	Wellhead total production rate
w_{pg}	Wellhead gas production rate
w_{po}	Wellhead oil production rate
w_{ro}	Oil rate from reservoir
w_{rg}	Gas rate from reservoir

Symbols - Greek letters

α	Master problem updating parameter
ε	Tollerance
Θ	Estimated parameter
λ	Lagrangian multiplier
σ_{GOR}	GOR standard deviation

Acronyms

c-NMPC	Centralized Nonlinear Model Predictive Control

CV	Controlled Variable
DAE	Differential Algebraic Equation
DD	Dual Decomposition
d-NMPC	Distributed Nonlinear Model Predictive Control
DOF	Degrees of Freedom
DRTO	Dinamic Real Time Optimization
EKF	Extended Kalmann Filter
HRTO	Hybrid Real Time Optimization
IPOPT	Interior Point OPTimizer
MPC	Model Predictive Control
MILP	Mixed Integer Linear Programming
MV	Manipulated Variable
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
ODE	Ordinary Differential Equation
PD	Primal Decomposition
PI	Proportional Integral Controller
RTO	Real Time Optimization
SFI	Norwegian Center for Research Based Innovation
SRTO	Static Real Time Optimization
SUBPRO	Subsea Production and Processing
SuPER	Sustainable Process Engineering Research

Abstract

The scope of this thesis is to develop and analyze a distributed optimizer based on Hybrid Real Time Optimization (HRTO). The traditional Static RTO (SRTO) and dynamic RTO (DRTO) are widespread in modern industry, however both the approaches have limiting factors concerning respectively the necessity of waiting for stationary conditions and the high computational effort required. HRTO consists in a static optimization coupled with a dynamic estimator, in order to use transient measurements to perform the model adaptation step. This framework is able to achieve performances similar to DRTO in terms of convergence rate to the optimal point, at computational times similar to SRTO. HRTO's advantages have already been outlined, however it has never been applied to a distributed optimizer. In large and complex plants, system decomposition allows to reduce the effort in modelling, control and to decrease the computational time required to solve the optimization problem. Moreover, in particular system in which the sharing of informations between subsystems must be limited due to privacy issues, a distributed optimizer can be the only feasible option. In this work, system decomposition have been performed on a six gas-lifted oil wells network. The main feature of this network is the presence of two wells clusters, connected to the same processing facility but managed by two different operators. In such a scenario, sensitive data must not be shared between the clusters. However, the presence of global constraints requires a distributed RTO to optimize and coordinate the production. Two decomposition techniques have been applied and tested: primal decomposition, which is based on an iterative reallocation of the available resources, and dual decomposition, which formulates the subproblems by means of Lagrangian Relaxation. At first, a Nonlinear Model Predictive Controller (NMPC) has been implemented and its advantages respect to traditional controllers have been outlined. Then, a distributed optimizer has been compared to a centralized optimizer used as reference. Despite both the optimizers converge to the same optimal solution, the distributed framework needs to introduce local constraints that can lead to suboptimal choices whenever significant disturbances occur. Lastly, the two decomposition techniques have been compared. Primal decomposition is able to find the global optimum with a lower average computational time, even though dual decomposition requires a lower number of iterations. Primal decomposition can also assure a higher level of privacy, requiring the sharing of only non-sensitive informations, as the gradients of subproblems' solutions.

Keywords Distributed optimization; RTO; Gas Lift; NMPC; Oil production.

Estratto

Lo scopo di questa tesi è lo sviluppo e analisi di un ottimizzatore distribuito basato su Ottimizzazione Real-Time ibrida (HRTO). Le tradizionali RTO statica (SRTO) e RTO dinamica (DRTO) sono diffuse nell'industria moderna, tuttavia entrambi gli approcci presentano dei fattori limitanti relativi rispettivamente alla necessità di attendere condizioni stazionarie e all'elevato tempo di calcolo necessario. HRTO consiste in un'ottimizzazione statica accoppiata ad un estimatore dinamico, in modo da poter effettuare lo step di adattazione del modello anche durante il transitorio. Questo approccio è in grado di ottenere performances simili a DRTO per quanto riguarda i profitti, con tempi di calcolo paragonabili a SRTO. I vantaggi di HRTO sono già stati evidenziati, tuttavia non è stata mai applicata ad un ottimizzatore distribuito. In impianti complessi e di grandi dimensioni, decomporre il sistema può ridurre le difficoltà legate alla modellazione e controllo, oltre a diminuire il tempo di calcolo richiesto per risolvere il problema di ottimizzazione. Inoltre, in sistemi dove la condivisione di informazioni deve essere limitata per questioni di riservatezza, un ottimizzatore distribuito può essere l'unica opzione disponibile. In questo lavoro, la decomposizione è stata applicata ad una rete di sei pozzi petroliferi gas lift. La principale caratteristica di questo network è la presenza di due raggruppamenti di pozzi, collegati allo stesso stabilimento a valle, ma controllati da due diversi operatori. In questo scenario le informazioni sensibili relative alla produttività non devono essere condivise. Tuttavia, la presenza di vincoli globali richiede di introdurre un ottimizzatore distribuito per massimizzare e coordinare la produzione. Due tecniche di decomposizione sono state testate: decomposizione primaria, basata sulla ripartizione delle risorse disponibili, e decomposizione duale, che formula i sottoproblemi attraverso rilassamento Lagrangiano. Inizialmente un NMPC è stato sviluppato e i suoi vantaggi rispetto a controllori tradizionali sono stati evidenziati. In seguito l'ottimizzatore distribuito è stato confrontato con uno centralizzato, usato come riferimento. Benchè entrambi gli ottimizzatori convergano alla stessa soluzione, la necessità, da parte dell'ottimizzatore distribuito, di introdurre vincoli locali nel sistema può condurre a scelte sub-ottimali. In seguito, le due tecniche di decomposizione sono state confrontate. La decomposizione primaria è in grado di trovare un ottimo globale con un tempo di calcolo inferiore, benchè la decomposizione duale richieda un minor numero di iterazioni. Inoltre, la decomposizione primaria può garantire un maggior livello di privacy, richiedendo la condivisione solamente di informazioni non sensibili, quali i gradienti delle soluzioni dei rispettivi sottoproblemi.

Parole Chiave Ottimizzazione distribuita; RTO; Gas Lift; NMPC; Estrazione di petrolio.

Introduction

The offshore extraction of oil and gas is a complex process, in which many decisions must be taken in order to maximize the production, satisfying process and environmental constraints, and taking in account disturbances or uncertainities. A continuous improvement and innovation of the existing technologies permits the producers to be competitive in the fast-moving market. In the recent years, subsea technology has emerged as an interesting field development tool, able to make oil's extraction safe and efficient without the necessity of building platforms. The reduction of investment costs opens more possibilities in the profitable exploitation of smaller and difficulty located fields. The full automation of the production requires a deep knowledge of subsea equipment: being operation failures and rehabilitation significantly more expensive in this kind of operations, methods to prevent and deal with unexpected maintenance must be considered even in the design phase. Likewise, a good control strategy of the system can help avoid costly unplanned down-time (Moreno and Markeset, 2002). Together with process and equipment innovations, research and improvements in the optimal control field are required, due to the large volume of production that characterizes oil production systems, the possibility of guaranteeing a continuous optimal control of the process is a great advantage, able to remarkably increase profits.

The optimal operation of an oil and gas production system is a challenging task, because of the necessity of meeting goals and objectives ranging from planning and scheduling to fast corrective actions. This complex decision making process is usually decomposed in a hierarchical way, depending on the time-horizon of the decisions and goals to be achieved, as described in (B. A. Foss and Jensen, 2011). This framework is widely accepted as industrial standard, and it is also well studied in academic literature, see for example (Skogestad, 1999), (Skogestad, 2004), (Larsson and Skogestad, 2000), (Rangaiah and Kariwala, 2012). A common decision-making hierarchy adopted in the oil and gas production is shown in picture 0.1. Long-term decision-making involves asset management and investment strategies, while medium-term decisions refer to plant wide scheduling, as reservoir management, drilling and production planning. Below that is the control and automation layer, that continuously handles disturbances and regulations of process's operating conditions.

Within this lower level, the first stage consists in a continuous optimization of the proces, called Real Time Optimization (RTO). RTO provides an optimal set point to the lower control layers, that can be broadly divided in supervisory and regulatory control. The supervisory controller computes the optimal trajectory to approach RTO's set point, dealing

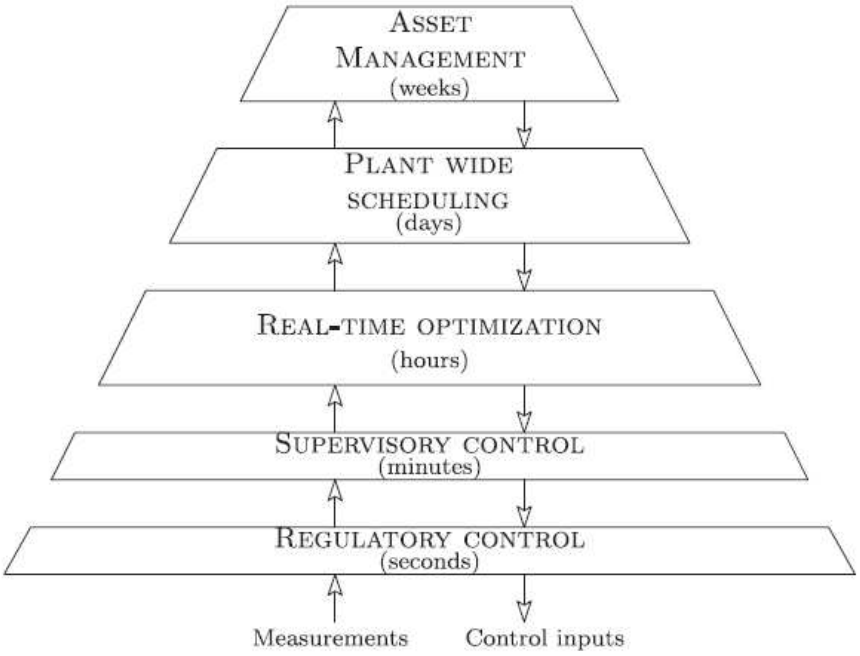

Figure 0.1: Decision making hierarchy in process control. Figure taken from (Krishnamoorthy, B. A. Foss, *et al.*, 2018).

with interactions with other variables and constraints. In the lowest level, fast controllers actually regulate the production system.

This work deals with the lower layers of the decision-making hierarchy, specifically with the development of a Real Time Optimizer. Traditional static RTO (SRTO) is a well estabilished tool and is widely adopted in the modern industry. However, the necessity of waiting for steady state measurements to update the model's parameters, reduces significantly the profits (Darby *et al.*, 2011) whenever disturbances, uncertainities and slow dynamics makes difficult to rapidly achieve stable, stationary conditions. Moreover, SRTO is not able to account for dynamic violations of the constraints, due to the adoption of static models in the optimization step.

In order to prevent these problems, Dynamic RTO (DRTO) and the closely related Economic MPC (EMPC) have been studied and developed. The use of a dynamic model during optimization and model adaptation allows to continuosly evaluate the optimal process conditions, even during transients. The disadvantages of this approach are a higher modelling and compution effort, due to the difficulties in formulating and solving a dynamic nonlinear optimization problem for large-scale systems.

Recently, a new RTO technique called Hybrid RTO (HRTO) has been presented in the paper *"Steady-state real-time optimization using transient measurements"* (Krishnamoorthy, B. A. Foss, *et al.*, 2018), in which a static optimizer has been coupled with a dynamic estimator. Thus it is possible to continuosly and efficiently perform a static optimization,

without waiting for stationary conditions. HRTO is able to provide similar performances to dynamic optimization, with a computational time similar to static RTO.

The aim of this thesis is to expand and continue the aforementioned study (Krishnamoorthy, B. A. Foss, *et al.*, 2018), developing a distributed optimizer using HRTO's framework. Optimization problems within oil and gas are large and complex, with hundred decision variables and strongly coupled constraints, as explained in (Hauge and Horn, 2005). System decomposition is a powerful tool, that allows to reduce the efforts in model identification, debugging and numerical solving. In this work, a six gas-lifted oil wells network has been considered and decomposed in two subsystems, composed by three wells each. The two clusters are controlled by two different operators, but the connection to a common downstream processing facility introduces global constraints that couple the two subsystems. The presence of privacy issues regarding well's properties, production's parameters and process conditions of the two subsystems is the main reason for performing a distributed optimization.

With the development and spreading of subsea technology, the networks of wells, platforms and processing facilities will increase in dimension and complexity, deeply modifying the interactions between different companies within the same geographical area. Therefore, distributed optimization is an interesting aspect of optimal control in oil production, that could lead to practical applications in the future.

This study is the author's final thesis and represents the conclusion of a M.Sc. in Chemical Engineering at Politecnico di Milano. This work has been carried out during an exchange period within the Process System Engineering research group of the Norwegian University of Science and Technology (NTNU), under the supervision of Sigurd Skogestad and Dinesh Krishnamoorthy. The work is part of SUBPRO (Subsea Production and Processing), a project sponsored by the Norwegian Center for Research Based Innovation (SFI).

This thesis is organized as follows: in chapters 1 and 2 the theory of Real Time Optimization and Distributed Optimization is explained, in chapter 3 the case study is presented, while in chapter 4 the system's optimization's procedure is described. In chapter 5 the results are presented, before concluding the work in chapter 6.

Chapter 1

Real time optimization

1.1 Introduction

Real time optimization (RTO) refers to all the techniques that allow the continuous evaluation of the optimal process conditions, in order to maximize the profits of the plant taking in account its constraints. RTO characterizes all the decisions that have to be made in a timescale of hours, to fulfill objectives decided during plant wide scheduling. Process optimization has been long considered an appealing tool applicable only to academic problems. In the last decades great improvements have been made in the process systems research field, permitting to make Real Time Optimization viable for current processes and, in some cases, mandatory to achieve competitive productivities. One of the main difficulties of process optimization is to find an effective mathematical formulation of the problem to be solved. Even when process models are available, the presence of plant-model mismatches and disturbances make the direct use of model-based optimal inputs hazardous. RTO is a valid technique to overcome these problems, relying not only on a mathematical model, but also on measurements and online process informations. Indeed, RTO is usualy carried out in two steps: a model adapation step and a model-based optimization step. In this way, the informations from the plant are implemented in the model during the first step, in order to compute the optimal solution in the second step.

Many RTO techniques exist and they can be roughly divided in two main groups: static RTO and dynamic RTO. Static RTO relies on a static model to perform the optimization. It is quite spread due to the presence of a simple model, and because the main interest of the optimization is the steady state response of a system, being the transients periods usually shorter than the steady state's ones. In order to solve a static optimization problem, steady state measurements are necessary.

If the optimization of transients is required, dynamic RTO is a possible answer. This kind of optimization technique relies on more complex dynamic models, in which the modelling and computational effort is compensated by the possibility of optimizing the system even in non-stationary conditions. For the model adaptation step a dynamic estimator is required.

In the last years a newly born RTO technique have been developed: hybrid RTO. It consists in a static optimizer that is able to use transient measurements to perform the

optimization. The advantage of this technique is the possibilities of obtaining performances comparable to DRTO with the computational time of a static optimization.

1.2 Static RTO

1.2.1 Description

Many chemical processes operate at steady state. Steady state operation is attractive from an implementation point of view, because the decision variables need to be simply kept constant over long time periods. A mathematical model of the steady state operation is usually available, permitting the usage of Static Real Time Optimization. The lower modelling and computational effort required is the main reason for which SRTO is the most common technique adopted to continuously optimize a chemical process.

Traditional SRTO is based on a two-steps approach:

- Static parameter estimation - In this step the model's parameters are adjusted to match the current data. Before this step a Steady State Detection system is necessary, that is able to individuate if the plant is operating close enough to steady state. After this, the model is adapted to match the current data, by means of techniques like data reconciliation.

- Static Optimization - In this step the optimization is carried out. The model is coupled to an objective function that must be maximized or minimized. The optimization is mathematically formulated as an NLP problem and then solved.

A representation of a SRTO optimization routine is shown in figure 1.1

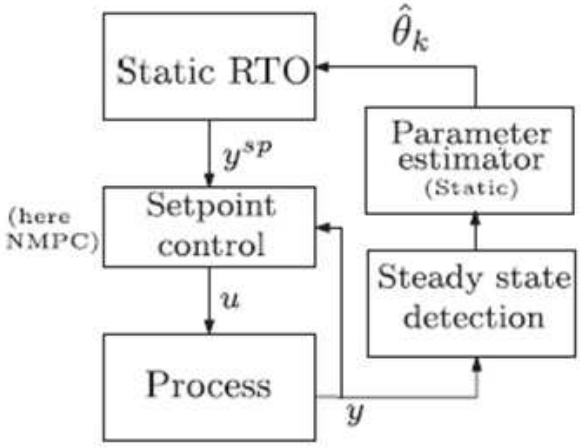

Figure 1.1: Static Real Time Optimization, from (Krishnamoorthy, B. A. Foss, *et al.*, 2018).

1.2.2 Model adaptation step

Before starting the real computation of the optimal process conditions, measurements from the plant are required. In this way, it is possible to continuously adapt the model to match the real parameters that characterize the system. SRTO needs steady state measurements, thus, at first, a steady state detector is required.

Many possibilities are available in order to individuate the end of a transient and the presence of a steady state, usually statistical or heuristic methods are adopted. A description of the most common steady state detection routines can be found in (Camara *et al.*, 2016)

Once the system has reached steady state, the model adaptation can be carried out. The model adaptation is based on data reconciliation, in which a real model optimization is carried out. It can be expressed as follow:

$$\boldsymbol{\Theta}_i = arg \min_{\Theta_i} |y_{meas} - y_{calc}(u_k, \boldsymbol{\Theta}_i)|^2; \tag{1.2.1}$$

Where $\boldsymbol{\Theta}_i$ are the unknown parameters that must be estimated, y_{meas} the real plant measurements, y_{calc} the plant variables as computed from the model, u_k the input parameters, or manipulated variables of the plant. It can be noted that data reconciliation is not a regression, but it is a real constrained optimization, with a number of degrees of freedom equal to the number of unknowns parameters and the model itself as constraint.

Once the parameter estimation step is completed, the model is updated and the optimization problem can be formulated and solved.

1.2.3 Optimization step

With the newly updated model, a mathematical formulation of the optimization problem can be generated. An objective function and the process constraints must be defined, then a Non Linear Programming (NLP) problem as the following can be formulated:

$$u^{opt} = arg \min_{u}\{J(y, u)\}; \tag{1.2.2}$$

$$s.t. \quad y = \boldsymbol{f}(u, \boldsymbol{\Theta}_i); \tag{1.2.3}$$

$$\boldsymbol{g}(y, u) \leq 0. \tag{1.2.4}$$

Where J is the objective function, y the process variables, u the input variables, u^{opt} the optimal input variables, or the solution of the optimization problem. $\boldsymbol{\Theta}_i$ are the parameters estimated in the previous step, $\boldsymbol{f}$ the model's equation and $\boldsymbol{g}$ the process's constraints. Please note that the objective function is often the plant's profit, defined as the difference between incomes and expenses. Many NLP solvers, as the one adopted in this work, are able to solve only minimization problems. Of course the aim of economic optimization is to maximize the profits, it is anyway possible to use the same NLP solvers by simply defining the objective function as the opposite of the profits, and then minimizing it.

1.2.4 Problems related to SRTO

As stated before, static RTO is the most used tool for continuous optimization in industrial processes, due to the lower modelling effort required. Moreover, the utilization of a static model reduce greatly the computational time of the optimization. However, there are some problems, mostly related to the model adaptation step, that may make other RTO techniques preferable.

The main problem related to SRTO is the necessity to wait for steady state conditions. SRTO is not able to optimize the system during non stationary conditions due to the presence of a static model, thus it will not compute any optimal solution during tranisents. In real plants, the presence of disturbances and uncertainities make really difficult to achieve stable steady state conditions. It is also quite common to deal with process with slow responses, which means that even a small disturbance can perturb the system for a long time. The necessity of waiting for steady state is a huge problem, that can make SRTO not advisable for most process, due to the presence of long transients in which the system will work sub-optimally. For example, in the case study considered in this work, the system is able to work in steady state for 62% of all the simulation time. If we decide to implement a SRTO in this system, it will work in a sub optimal way for almost 40% of the time, that will imply a consistent loss of profits.

Moreover, the presence of statistical or heuristic methods during the steady state detection require the introduction of tolerances by the user to decide if the system is "close enough" to steady state. If this tolerances are specified without enough attention, the system can be erranously considered at steady state, leading to the propagation of error to the optimization step.

In order to deal with transient optimization, it is possible to adopt more complex models, able to describe also system's beahaviour during transients. This is the concept at the basis of Dynamic Real Time Optimization.

1.3 Dynamic RTO

1.3.1 Description

In the recent years, there have been many developments in DRTO and the closely related Economic Model Predictive Control (EMPC), that are able to provide an optimal input trajectory even during transients. In DRTO, dynamic models are implemented in the optimization routine: the greater modelling effort is compensated by the possibility of continuously optimizing the system at each sample time, without the necessity of waiting for stationary conditions. Again, the optimization is based on a two step approach:

- Dynamic State Estimation - The optimizer's model is updatd by means of a dynamic estimator, that usually consists in a filter able to remove the noise from the plant's measurements.

- Dynamic Optimization - The optimization itself is carried out in this second step, after the update of model's parameters. A dynamic model is used.

A rappresentation of a DRTO optimization routine is shown in figure 1.2

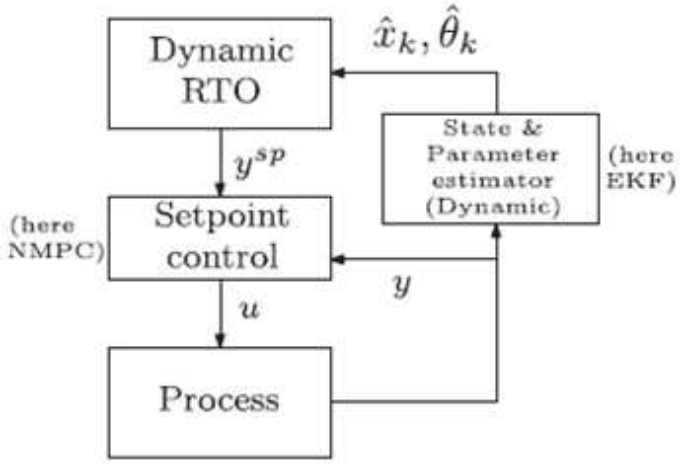

Figure 1.2: Dynamic Real Time Optimization, from (Krishnamoorthy, B. A. Foss, *et al.*, 2018).

1.3.2 Dynamic estimator

The first step is a model adaptation step, in which some model's parameters are updated to match the plant's measurements. A dynamic estimator is adopted, which is able to update those parameters at every sample time, independently from stationary or not conditions. These estimators are usually filters, which aim is to remove the noise from a series of measurements in order to give an estimation of the desired parameters. A dynamic model is implemented in the filter, and by means of a covariance matrix it is possible to modify how much the filter should trust the model or the measurements. In this work, the estimator adopted is an Extended Kalman Filter (EKF). EKF is the non linear version of the Kalman Filter and it is considered the standard in nonlinear state estimation, as stated in (Julier and Uhlmann, 2004) and (Wan, 2016). The equations that make up the EKF are presented in chapter 4.

1.3.3 Optimization step

The optimization problem is similar to the static one, with the only difference that the model is now a dynamic and it considers the time as a variable:

$$u^{opt} = arg \min_{u}\{J(y,u)\}; \tag{1.3.1}$$

$$s.t. \quad y = \boldsymbol{f}(u, \boldsymbol{\Theta}_i); \tag{1.3.2}$$

$$\boldsymbol{g}(y,u) \leq 0. \tag{1.3.3}$$

Where J is the objective function, y the process variables, u the input variables, u^{opt} the optimal input variables. $\boldsymbol{\Theta}_i$ are the parameters estimated in the previous step, $\boldsymbol{f}$ the model's equation and $\boldsymbol{g}$ the process's constraints.

The main difference between the SRTO's and DRTO's optimization step is that while in SRTO the optimization is carried out once per sample time, in DRTO, at each sample time, the system is simulated and optimized for a longer time. This period is called Prediction Horizon (PH) and it must be sufficiently long to ensure stability (Maciejowsky, 2002).

To continuously solve the dynamic optimization problem, it is possible to approximate the function as a sum of Lagrangian Polinomial. This is the idea on which the method adopted in this work(Direct collocation) is based. This method is explained in chapter 4, regarding the implementation of the Model Predictive Controller (MPC).

1.3.4 Problems related to DRTO

Although the use of dynamic models for model adaptation and optimization is interesting, due to the possibility of eliminating the necessity of steady state detection, solving a dynamic nonlinear optimization problem for large system may be challeging. If compared to the solving of a static optimization, the computational effort is larger, as stated in (Krishnamoorthy, B. A. Foss, *et al.*, 2018) it can increase even 100 times. The reason of this increase is in the greater number of variables that must be optimized. For example in the case study described in the same article, while SRTO must solve a system composed by 22 optimization variables, DRTO has 3056 variables to be optimized at each sample time. The variables's number is so larger because of the necessity for DRTO of optimizing the whole process for not only one time step, but for the whole prediction horizon.

The higher computational time and modelling effort make DRTO difficult to implement in industrial applications. Even if it seems very appealing, this technology is not enough developed to spread widely in the modern industry. A very interesting solution can be found in hybrid RTO, which could be able to take the place of static RTO during this transient period towards Dynamic RTO.

1.4 Hybrid RTO

1.4.1 Description

Hybrid Real Time Optimization (HRTO) has been developed in the recent period. SRTO uses the same static model in both the model adaptation and in the optimization step and, similarly, DRTO uses the same dynamic model in both the steps. As described in the paper "Steady-state real-time optimization using transient measurements" (Krishnamoorthy, B. A. Foss, *et al.*, 2018), if the main problem of SRTO lays in the model adaptation step, while in DRTO the optimization step is troublesome from a computational point of view, one possibility could be to try to consider a hybrid structure, in which the model adaptation is performed using a dynamic estimator, while static models are used in the optimization step, as shown in figure 1.3.

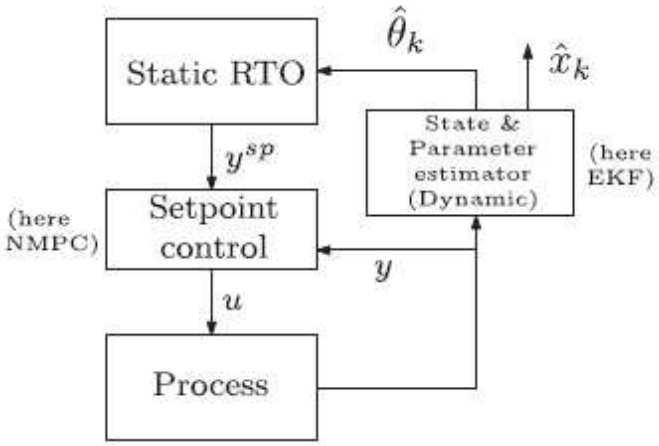

Figure 1.3: Dynamic Real Time Optimization, from (Krishnamoorthy, B. A. Foss, *et al.*, 2018).

This is the idea of Hybrid Real Time Optimization, which consists in a static optimization using transient measurements. In this way, it is possible to continuously carry out a static optimization, without the need of waiting for steady state conditions. HRTO can be a very powerful tool due its capacity of obtaining performances similar to DRTO with computational times comparable to SRTO (see (Krishnamoorthy, B. A. Foss, *et al.*, 2018)) .

This technique is composed by the two aforementioned steps:

- Dynamic State Estimation - The optimizer's model is updated by means of a dynamic estimator, that usually consists in a filter able to remove the noise from the plant's measurements.
- Static Optimization - In this step the optimization is carried out. The model is coupled to an objective function that must be maximized or minimized. The optimization is mathematically formulated as an NLP problem and then solved.

1.4.2 Dynamic estimator

The first step is a model adaptation step, in which the model's parameters are updated to match the plant's measurements. This step is exactly the same used in Dynamic RTO. The reason for opting for a dynamic estimator instead of a static one is the possibility of updating continuously the model, without the necessity of waiting for steady state. As described before, non linear estimation is a well known and estabilished study field and many estimators have been developed. In this work an Extended Kalman Filter has been considered, its framework can be found in chapter 4.

1.4.3 Optimization step

The optimization is performed using a static model. The optimization problem is the same as the system 1.2.2, that can be found in SRTO's description. A static model is chosen because

the primary objective is to optimize the steady state performance of the process, while the transients can be dealt through the use of a set point control layer. In this framework, the control layer is split in two: an upper control layer and a lower one. In the upper layer, an advanced controller, as a Model Predictive Controller (MPC) receives a set point from the RTO's layer. The MPC computes an optimal set point trajectory that is sent to the lower control layer, where standard controllers, as a Proportional Integral controller (PI controller) deals with the effective control on the regulation devices. The use of the set point tracking layer permits to deal with dynamic violations of constraints.

1.4.4 Comparison with SRTO and DRTO

SRTO requires a low computational time, but it needs to wait for stationary conditions to start the optimization. DRTO is able to optimize continuously the system, even during transients, but the great number of optimization variables that compose the NLP increases the computational time, wich can be prohibitive for online application. As discussed in (Krishnamoorthy, B. A. Foss, *et al.*, 2018), HRTO requires a low computational effort due to the static optimization step, similar to SRTO. On the other hand, the model adaptation step is dynamic, which means that the model can be corrected at each time step, without waiting for steady state.

HRTO shows very interesting performances whenever coupled with an advanced control layer, as a Model Predictive Controller (MPC). In this way, even if HRTO optimizes the stationary conditions (being the model static), the MPC deals with the transients, controlling the system in order to not dynamically violate the constraints. In this framework, the HRTO generates a steady state set point, that is sent to the upper control layer. The MPC acts as a set point tracker, generating a new set point for the lower control layer (as a Proportional Integral controller), in order to approach HRTO's set point without violating the constraints, even during transients. This framework is able to behave in a similar way to DRTO, where the optimization layer computes directly the best set point trajectory. Taking as example the case study discussed in (Krishnamoorthy, B. A. Foss, *et al.*, 2018), HRTO's profit loss is equal to 2% if compared with DRTO, while SRTO's loss is around 34%. For what concerns computational time, HRTO has a lightly higher computational time respect to SRTO, however it is almost 100 times lower than DRTO's computational time, as shown in table 1.1.

It is evident that the loss of HRTO respect to DRTO is low, while the gain in terms

Table 1.1: SRTO, DRTO and HRTO performances' comparison. The table is taken from (Krishnamoorthy, B. A. Foss, *et al.*, 2018), the case study deals with a gas lift oil production system.

	avg. time [s]	max time [s]	Integrated Profit [x 10^6\$]
SRTO	0.0184	0.0223	1.8256
HRTO	0.0199	0.0282	2.7019
DRTO	0.9025	3.3631	2.7509

of computational effort is huge. It has been proved that HRTO is a valid technique, that can replace SRTO with considerable benefits, even in more complex systems where DRTO's computational effort would be too large for online applications. Therefore in this thesis, we will consider the Hybrid RTO approach in detail.

Chapter 2

Distributed optimization

In this chapter the reasons and the methodologies to perform a distributed optimization are presented and discussed.

2.1 Introduction and reasons

In the classical formulation of an optimization problem, the system is modelled and optimized as a whole. This approach is conceptually obvious, but whenever the system considered is large, it can be advisable to opt for a distributed approach. In distributed optimization, the system is decomposed in more subsystems, that are modelled and optimized individually. A central coordinator is required, in order to allocate effectively the avialable resources and achieve global optimal conditions, as shown in figure 2.1.

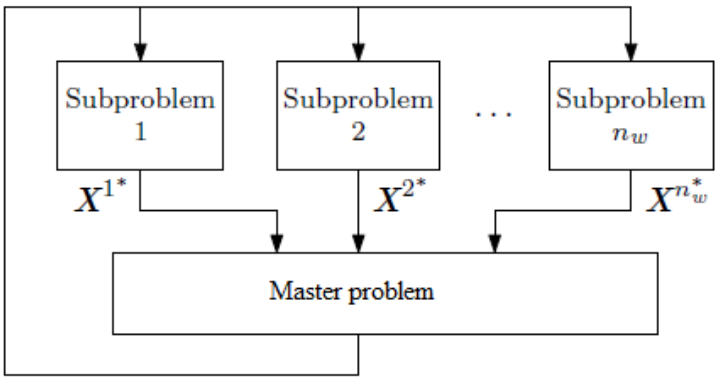

Figure 2.1: Distributed optimization block diagram.

This kind of aproach may seems not very effective, due to the necessity of solving many optimization problem iteratively, but there are many reasons that can lead the choice from a centralized optimizer to a distributed one:

- First of all, complications can arise since the beginning, in the modelling step. Large systems can be difficult to be modelled, due to the great number of unknown parameters

that must be estimated. An empirical model is not always available or feasible. As stated in (Krishnamoorthy, Aguiar, *et al.*, 2018), modelling effort is often complimented with system identification or plant experimentation. It is without any doubt easier to evaluate, identify and update fewer models for smaller subsystems, instead of one large model.

- In complex chemical plants, each unit can act as a consumer or producer of resources (heat, steam, electricity, water,...). In this case, it is preferable to optimize each unit individually and manage the usage of these resources with a global coordinator, in order to fullfill the overall domand-supply balance constraints.
- Sub-problems can be solved in parallel, decreasing the computation time required for the optimization. Moreover, a network between the subsystems and a central coordinator can be created, in order to let all the subsystems solve their own subproblems. This is very important when we have privacy concerns, as described in the next point.
- One of the main advantages of using a distributed optimizer is the possibility to carry out a global optimization without requiring the subsystems to exchange informations between them. In the modern off-shore oil and gas industry several wells can be coupled to the same production manifold or to the same processing facility. If the wells are owned by different companies, it is possible that each operator refuse to share the details of their operational constraints, costs and profits. In such a situation, individual optimizers should be responsible for the optimization of the individual units, while an overall coordinator deals with modifying the cost function or the constraints of the individual optimization problems, possessing only limited information. This is the main reason for which a distributed optimization may be not only suggested, but mandatory in order to find a solution to these privacy issues.

Due to its importance and the possibility to reduce the optimization effort, decomposition has been deeply studied. It is a well known and utilized technique, and multiple methods of decomposition are described in literature.

2.2 Decomposition methods

The first step to perform a distirbuted optimization is the system's decomposition. While in the introduction a time-scale based approach has been presented to vertically decompose the management of a plant, this kind of decomposition can be considered "horizontal", being based on the division of the system according to its geography or its process flow. The decomposition of a system should be performed intiuitively, creating subsystems that can be close from a geographical point of view or a functional point of view. In the case study considered in chapter 3, a large oil wells network, has been divided in two subsystems, according to the owners of those clusters of wells. Another example can be the decomposition of a very large chemical plants, in which many components are producted and sold. An

intuitively way to decompose and optimize the production and the profits could be to consider every production line as a stand-alone subsystem, that can be optimize individually with the help of a master coordinator, whose task is to optimally allocate reactants, steam, pressurized air, electricity and so on.

In this work two decomposition techniques have been considered and compared. They are Primal Decomposition and Dual Decomposition. Although the solution of a distributed optimization is invariant from the choice of the decomposition technique, this two techniques adopt different approaches to decouple the problem in subproblems, that requires different informations and can have a higher o lower convergence speed. That is why it is interesting to consider both the approaches and compare them.

2.2.1 Primal decomposition

Primal decomposition is an auction-based approach that iteratively reallocates the shared resources among the subsystems. This approach represents well the task of the global coordinator: to allocate the available resources to the subsystems and let them optimize their processes by themselves. Once the optimization of the subsystems is performed, the subsystems must share some informations with the central coordinator, according to which, resources' allocation is updated. This loop continues until achieving convergence, meaning the finding of a global optimum. The idea of "available resources" is intuitive whenever we consider a input to the system that is limited, for example reactants, heat power, water, steam, ... In this approach the idea of "available resources" is extended to every variable that has a limit, thus, to every process constraint. A processing capacity of a downstream equipment can be considered as a resource, and the productivities of all the upstreams equipment can be managed as an allocation of this particular resource.

In order to make an example of how primal decomposition works, let's consider a chemical plant that can be decomposed in two subsystems A and B. A simplified optimization problem can be the following:

$$\max_{\boldsymbol{u}} \; J^{tot} = (x^{tot} - u^{tot}); \tag{2.2.1}$$

$$s.t. \quad \boldsymbol{x} = \boldsymbol{F}_c(\boldsymbol{u}, \boldsymbol{\Theta}); \tag{2.2.2}$$

$$u^{tot} \leq u^{max}; \tag{2.2.3}$$

$$\tag{2.2.4}$$

where J is an objective function that represents the revenues of the plant, $\boldsymbol{x}$ is a variable that represent the income (the plant's main product), while $\boldsymbol{u}$ is the input of the plant, that is the optimization variable and it has an associated cost. $\boldsymbol{F}_c$ is the system's mathematical or empirical model, while $\boldsymbol{\Theta}$ are some parameters that characterize the system. $\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\Theta}$ are vectors, and they are so defined:

$$\boldsymbol{x} = \left[x^A, x^B\right]; \tag{2.2.5}$$
$$\boldsymbol{u} = \left[u^A, u^B\right] \tag{2.2.6}$$
$$\boldsymbol{\Theta} = \left[\Theta^A, \Theta^B\right] \tag{2.2.7}$$
$$\tag{2.2.8}$$

Where the subscripts A and B stay for the variable's values relative to one on the two subsistem A or B. x^{tot} and u^{tot} are the sum of the elements of each vector, namely the values of those variables for the global system. Only one constraint is present: the total value of the input u^{tot} has an upper bound equal to u^{max}. The system is decomposed in two subsystems, thus, two subproblems are generated.

Subproblems

In the formulation of the two subproblems, the two subsystems are considered separately. A model of each subsystem is required, that can be completely different between them if we consider subsystems in series, or very similar if the subsystems are just the same equipments working in parallel. Similarly, two objective functions are required. For the decomposition to be succesfull it is important that the global objective function is the sum of the subsystems' objective function.

Lastly, the constraints must be managed. In primal decomposition the constraints are considered as resources, and are allocated between the two subsystems. While the first-try allocation for the first subsystem can be arbitrary, for the second subsystems it is necessary to choose a complementary value to not exceed the global constraint. Thus, the two subproblems are the following:

$$\max_{\boldsymbol{u^A}} \; J^A = (x^A - u^A) \tag{2.2.9}$$
$$s.t. \quad x^A = \boldsymbol{F}_c^A(u^A, \Theta^A); \tag{2.2.10}$$
$$u^A \leq t^A; \tag{2.2.11}$$
$$\tag{2.2.12}$$

$$\max_{\boldsymbol{u^B}} \; J^B = (x^B - u^B) \tag{2.2.13}$$
$$s.t. \quad x^B = \boldsymbol{F}_c^B(u^B, \Theta^B); \tag{2.2.14}$$
$$u^B \leq t^B; \tag{2.2.15}$$
$$\text{with} \quad t^B = \; u^{max} - t^A; \tag{2.2.16}$$
$$\tag{2.2.17}$$

As stated before, we can find in the two subproblems the two objective functions J^A, J^B and the two models $\boldsymbol{F}_c^A, \boldsymbol{F}_c^B$. u^A and u^B are the inputs of each subsystem, x^A and x^B the

variable of interest of each subsystem. Θ^A, Θ^B are parameters characteristic of the two subsystems. For what concerns the constrains, two new variables have been introduced: t^A, t^B that represent the allocated resources to the two subsystems, or the maximum value of input variable that the two subsystem can use. These quantities are chosen so that $t^A + t^B = u^{max}$, in order to not violate the global constraint. It can be noted that, the sum of the two subproblems is exactly equal to the original optimization problem.

Master problem

The optimization of the two subproblems is performed individually. In this way it is possible to obtain a local optimum, depending on the allocation of resources. In order to achieve a global optimum a central coordinator, which must be able to allocate those resources in an effective way. The problem that the central coordinator must solve is called master problem.

In primal decomposition, a common method to formulate the master problem is a subgradient method. Subgradient methods require the subsystems to compute a subgradient of their solution, which is then used to update the allocated resources. This update can be carried out as follow:

$$t^A_{k+1} = t^A_k - \alpha(\Lambda^B - \Lambda^A); \tag{2.2.18}$$

$$t^B = u^{max} - t^A; \tag{2.2.19}$$

Where the subscripts represent the current iteration k and the next iterations $k+1$, while α is a tuning parameter of the central coordinator that control the update of the allocation of resources. Λ^A and Λ^B are the subgradients of the solutions of the two subsystems. They must be computed by the two subsystems and they are the only information that they are required to share. A block diagram of the optimization loop is presented in figure 2.2.

With this simple master problem, the central coordinator can update the allocation of resources, the two subproblems are updated and solved and the master problem is called again. This loop continues until achieving convergence or after reaching a certain number of iterations. Three parameters must be chosen for the tuning of the central coordinator: α, the tolerance and the maximum number of iterations.

- α define the velocity of the updating. High values of α can lead to a lower number of iterations, but problems related to robustness can arise. A compromise must be found, but it is advisable to prefer slower but more robust coordinators, in order to be sure to find a global optimum.
- Tolerance is the value which convergence can be considered to be achieved. The master loop continues until the difference between the newly calculated allocation of resources and the previous one is lower than tolerance:

$$tol \geq t^A_{k+1} \; - \; t^A_k = \alpha(\Lambda^B - \Lambda^A) \tag{2.2.20}$$

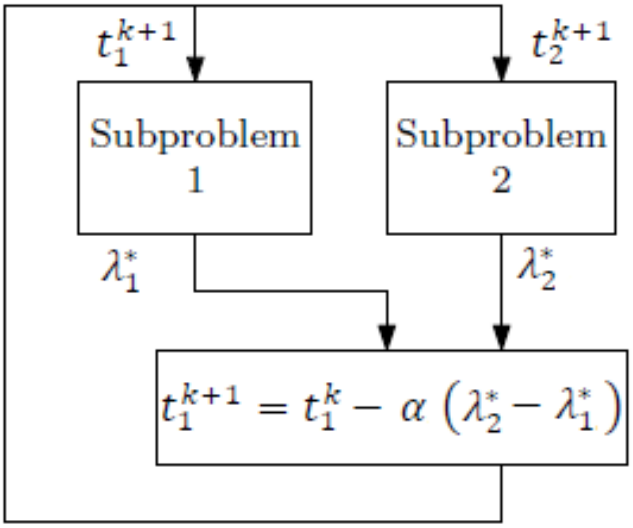

Figure 2.2: Primal decomposition block diagram.

It is interesting to observe that convergence is achieved according to the value of $\alpha(\Lambda^B - \Lambda^A)$. Higher values of α makes convergence more difficult to be reached. Moreover, the term $(\Lambda^B - \Lambda^A)$ shows that the only way to obtain a convergence and, thus, a global optimum is to have very similar gradients. Observing this term three cases are possible:

- $\Lambda^A \neq \Lambda^B \neq 0$: The two gradients are different. Their difference is not close to zero and convergence is not achieved. This is the case in which the subproblems have been solved, but both the solutions lay on the constraints and a global optimum is not found. The master loop must update the allocation of resources.
- $\Lambda^A = \Lambda^B = 0$: The two gradients are equal to zero. Both the problems have been solved and global optimum has been achieved. In particular, the solution does not lay on the constraint, the problem is unconstrained.
- $\Lambda^A = \Lambda^B \neq 0$: The two gradients are equal and convergence is achieved. The gradients are equal but not null, this means that the two constraints are active. The master coordinator weight the two subsystems in the same way, thus, global optimum is achieved when they gradients are equal.

Tolerance must be a low number, to assure that the solution found will not change significately and there is no need to perform another iteration. However, a higher number can be acceptable whenever the increase of precision of the solution does not appreciably change the value of the objective function. A lower value of tolerance leads to a greater precision and to a larger number of iterations.

- The maximum number of iterations is the maximum number of calls of the master problem, after which, even if convergence is not achieved, the solution is accepted. This

value depends on the process and on the values of α and tolerance. The maiximum number of iterations must be large enough to permit to reach convergence anytime it is feasible. Anyway it must have an upper bound, because of the risk of entering in an infinite loop whenever the problem is not feasible. Its value should be chosen regarding to the time available to perform the optimization.

2.2.2 Dual decomposition

While Primal Decomposition is an auction-based approach, based on the allocation of resources, Dual Decomposition is a price-based approach, in which the aim is to seek for the resources' prices that allows a market clearing, a situation in which the maximum profit for each subsystem is achieved, as described in (Wenzel *et al.*, 2018).

Dual decomposition is also called Lagrangian Decomposition, because it adopts Lagrangian relaxation to perform the system's decomposition. Lagrangian relaxation converts a constrained optimization problem in an unconstrained one, integrating the constraints in the objective function by means of the so called Lagrangian Multipliers. Considering the same system briefly presented in the primal decomposition discussion, applying Lagrangian Relaxation the global problem can be expressed as follow:

$$\max_{\boldsymbol{u}} \; L = (x^{tot} - u^{tot}) + \lambda(u^{max} - u^{tot}); \tag{2.2.21}$$

$$s.t. \quad \boldsymbol{x} = \boldsymbol{F}_c(\boldsymbol{u}, \boldsymbol{\Theta}); \tag{2.2.22}$$

$$\tag{2.2.23}$$

where L is the Lagrangian of the objective function and λ the respective lagrangian multiplier. $\boldsymbol{x}$ is a variable that represent the income, while $\boldsymbol{u}$ is the input and optimization variable of the plant. $\boldsymbol{F}_c$ is the system's model, while $\boldsymbol{\Theta}$ is a vector containing some parameters that characterize the system.

We can easily observe that this problem is unconstrained, being the constraint introduced inside the objective function. However the solution of this problem is the same as the original constrained problem. The violation of the constraints is dealt as a penalty in the objective function.

Again, the system is decomposed in two subsystems and a master problem is introduced to find the global optimum.

Subproblems

The decomposition is performed allocating between the two systems a fixed quantity of resources and varying the lagrangian multiplier of the related local constraints. We have:

$$\max_{\boldsymbol{u^A}} \; L^A = (x^A - u^A) + \lambda(\frac{u^{max}}{2} - u^A); \tag{2.2.24}$$
$$s.t. \quad x^A = \boldsymbol{F}_c^A(u^A, \boldsymbol{\Theta}); \tag{2.2.25}$$
$$\tag{2.2.26}$$

$$\max_{\boldsymbol{u^B}} \; L^B = (x^B - u^B) + \lambda(\frac{u^{max}}{2} - u^B); \tag{2.2.27}$$
$$s.t. \quad x^B = \boldsymbol{F}_c^B(u^B, \boldsymbol{\Theta}); \tag{2.2.28}$$
$$\tag{2.2.29}$$

Where λ is the Lagrangian multipliers characteristic of the given constraint. We can find again the objective functions J^A, J^B and the two models $\boldsymbol{F}_c^A, \boldsymbol{F}_c^B$. u^A, u^B , x^A and x^B are respectively the inputs and the variables of interest of each subsystem. Θ^A, Θ^B are parameters characteristic of the two subsystems.

It is possible to observe that the total available input variable u^{max} is allocated equally between the two subsystems. Any allocation could have been chosen, as long as the sum between the two would give u^{max}. We suppose that the two subsystems are similar so it is accettable to allocate the resources equally. It is important to notice that, in dual decomposition, this is not the real allocation of resources, but just a mathematical expedient to formulate the constraints.

Master problem

As stated before, dual decomposition is a price-based approach. The lagrangian multiplier can be seen as the price of the used resource and the task of the global coordinator is to update it in order to maximize the global profit.

The master problem is formulated as follow:

$$\lambda_{k+1} = \lambda_k + \alpha(u_k^A + u_k^B - u^{max}) \tag{2.2.30}$$
$$s.t. \quad \lambda_{k+1} \geq 0 \tag{2.2.31}$$
$$\tag{2.2.32}$$

Where u_k^A, u_k^B are the optimal input variable computed at the actual k step. α is a parameter that control λ's updating velocity. The global coordinator updates the lagrangian multiplier proportionally to the difference between the optimal usage of resources and the their total availability. This means that the information it requires from the two subsystems is the solution itself of the respective subproblem.

Another important point is the presence of a second condition on the lagrangian multiplier: it can not be negative. The lagrangian multiplier's final value depends on the properties of problem considered. It is easy to note that convergence is achieved whenever $\alpha(u_k^A + u_k^B - u^{max}) \to 0$. We have two possibilites:

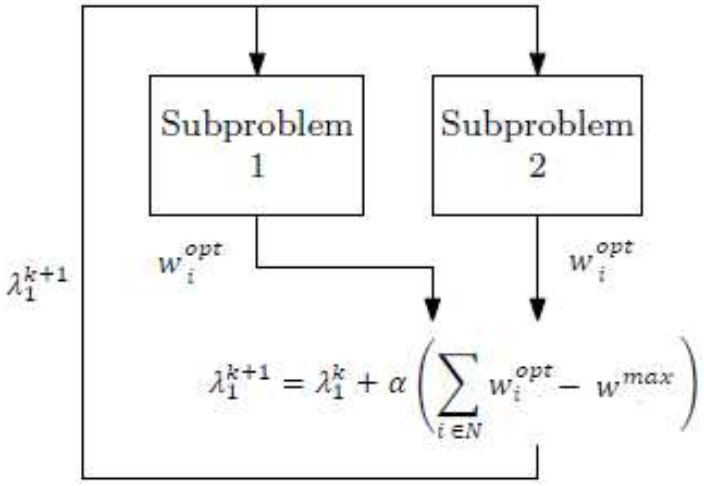

Figure 2.3: Dual decomposition block diagram.

- $(u_k^A + u_k^B - u^{max}) = 0$. The optimal value of input is equal to the maximum allowed value. The constraint is active and $\lambda \neq 0$.
- $(u_k^A + u_k^B - u^{max}) \neq 0$. The optimal value of input is lower than the maximum allowed value. The constraint is not active and λ must be null to reach convergence.

Thus, it is easy to understand that λ's value can give us important informations about the constraint. λ is equal to zero whenever the problem is unconstrained or the constraint is not active, while it assumes a positive value if the constraint is active.

For what concerns the tuning of the global coordinator, again, three parameters must be chosen: α, the tolerance and the maximum number of iterations. Same considerations can be made as in the case of primal decomposition.

Chapter 3

Case study: Gas Lift Oil Wells Network

3.1 Introduction and overview of the system

In order to study and analize the performances of a HRTO-based distributed optimizer, a gas lifted oil wells network has been considered. Offshore oil production is usually performed using large and complex networks of wells and processing facilities. All the elements of the network have many and complex connections and interactions between each other. Performing a decomposition on such a sistem can make analysis, control and optimization simpler, more efficient and more transparent.

3.1.1 Gas lifted well

Whenever the reservoir pressure is not sufficient to lift economically liquids to the surface, artificial lift methods are employed. One of the most common artificial lift technique is gas lift, which is often used for offshore oil production. In a gas lifted well, part of the produced gas is compressed and reinjected at the bottom of the well. The gas mixes with the oil coming from the reservoir, reducing its density and, consequently, reducing the hydrostatic pressure of the fluid column inside the tube. In this way, it is possible to increase the liquid flow from the reservoir. An example of a gas lift oil well is shown in picture 3.1.

The well is composed by an inside tubing surrounded by the annulus, an outer pipeline through which gas lift can flow downwards. At the injection point, far below the surface, gas lift flows through a valve and mixes with the oil from the reservoir. Then the mixture of oil and gas flows upwards through the tubing until reaching the production choke. The production choke is connected to a common production manifold, that collects all the flows coming from the network's wells. Lastly, the mixture is sent from the manifold to the processing facility, where the two phases are separated and finally processed. In order to deal with a possible excess of gas in the equipment, that can lead to an uncontrolled increase of pressure, a flare is always present, through which the gas in excess can be burnt.

In the case considered, the gas lift injection rate for each well is the manipulated variable (MV) on which the optimization is performed. Gas lift is a complex MV: increasing the gas injected in the well favours oil's production, but a too high flowrate can decrease oil's

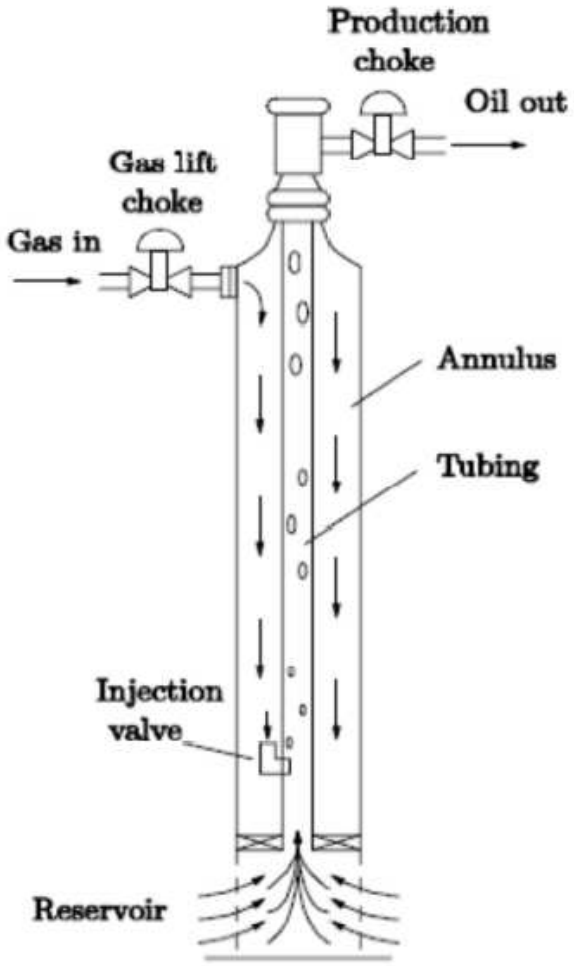

Figure 3.1: A gas lift oil well. The picture is taken from (Eikrem *et al.*, 2004).

flowrate. The reason of this behaviour is due to the higher bottom hole pressure and the increase in frictions in the well associated to a higher gas lift flowrate, as described in (Wang, 2003) and (Abdalsadig *et al.*, 2016). Thus, the choice of the optimal gas lift injection rate is complex and must be dealt with numerical optimization.

3.1.2 Network and decomposition

The network considered in this work is composed by six gas lifted wells. The wells have similar geometry, their main differences regard the oil's properties. Amall differences are important too and can change significantly the process conditions that maximize the well's productivity. The parameters that characterize the network's wells are presented in table 3.1

The main feature of this network is the presence of two "clusters" of wells, with two different owners and operators. The whole system is thus split in two smaller subsystems, composed by three wells each. The two clusters are operated separately and maintain a high

Table 3.1: Oil wells network's parameters.

Parameter	Unit of measure	Well 1	Well 2	Well 3	Well 4	Well 5	Well 6
L_w	[m]	1500	1500	1500	1500	1500	1500
H_w	[m]	1000	1000	1000	1000	1000	1000
D_w	[m]	0.121	0.121	0.121	0.121	0.121	0.121
L_{bh}	[m]	500	500	500	500	500	500
H_{bh}	[m]	500	500	500	500	500	500
L_a	[m]	1500	1500	1500	1500	1500	1500
H_a	[m]	1000	1000	1000	1000	1000	1000
D_a	[m]	0.189	0.189	0.189	0.189	0.189	0.189
ρ_o	$[kg/m^3]$	800	800	790	800	820	805
C_{iv}	$[m^2]$	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
C_{pc}	$[m^2]$	$2 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$2 \cdot 10^{-3}$
GOR	[kg/kg]	0.100	0.120	0.090	0.108	0.115	0.102
σ_{gor}	[kg/kg]	0.02	0.02	0.02	0.02	0.02	0.02
P_m	[bar]	20	20	20	20	20	20
P_{res}	[bar]	150	155	155	160	155	155
PI	$[kg/(bar\ s)]$	3.5	3.5	3.5	3.5	3.5	3.5
T_a	[°C]	28	28	28	28	28	28
T_w	[°C]	28	28	28	28	28	28

level of autonomy. However, they are strictly interacting, because of the connection to the same processing facility. The presence of common equipment downstream introduces global constraints that the system, considered as a whole, must not violate. Thus, a supervisory routine is necessary in order to coordinate the productions and satisfy the constraints imposed.

Moreover, privacy issues can arise in this particular system. The two different owners are also competitors, thus, it is important not to share sensible informations with the counterpart. Data regarding production, well parameters and reservoir properties must be kept secret, in order to avoid sharing informations from which higher strategies and plans can be deducted. In such a scenario, the wells' models and the objective functions of the respective clusters are not shared, even with the global coordinator. The optimization of the subsystems must be committed to the respective operators, while the global coordinator must be designed in order to work granting a certain privacy level to the two companies. These privacy issues can make the development of the centralized coordinator more difficult and complex. This is the main reason for which a distributed optimizer must be chosen: a centralized optimizer would require the knowledge of all the informations about the wells, reservoirs, process conditions and production objectives, in order to optimize the global production. In a distributed optimizer, the two operators can solve the optimization problems relative to their systems by their own and then share only part of their data to the coordinator, to let it supervise and manage the resources' allocation.

A simplified representation of the oil network is shown in figure 3.2. In the picture, the flare and all the downstream equipment have been omitted.

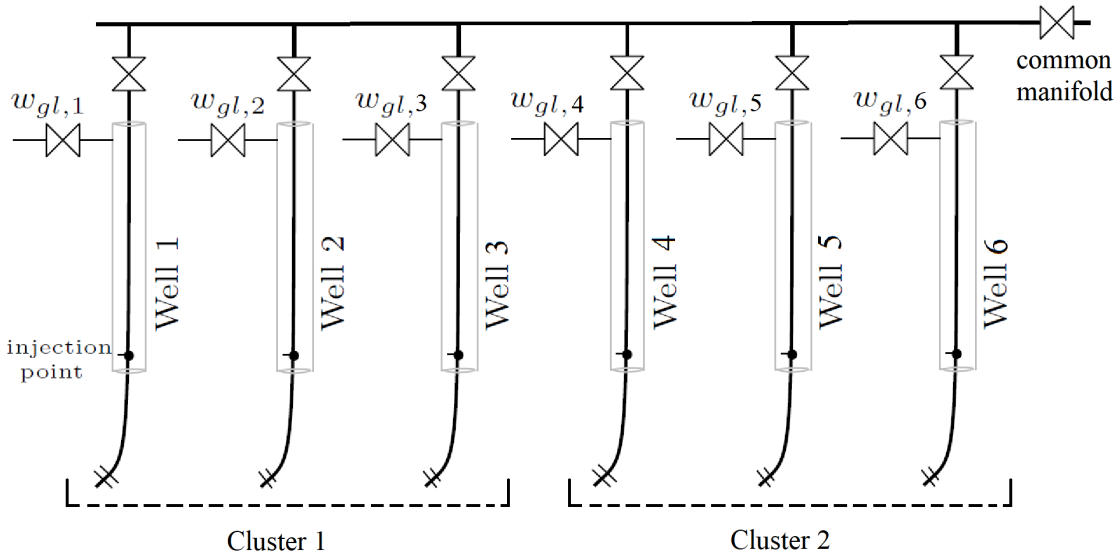

Figure 3.2: Oil wells network considered in this work. The figure has been modified from (Krishnamoorthy, B. A. Foss, *et al.*, 2018).

In order to perform a numerical optimization, a mathematical model describing the system is required.

3.2 Modelling

In this section a brief description of the gas lifted well model that is used in the optimzation process is presented. The model adopted is based on the work (Krishnamoorthy, B. Foss, *et al.*, 2016).

The differential algebraic equation system (DAE) is built up of 3 differential equations and 12 algebraic equations for each well. The reservoir model has not been considered, being it not crucial for the network model. Similarly, the downstream equipment as the riser, the separator and the flare are not considered in the model.

The wells that compose the network are all gas lifted and present similar geometry, thus the model adopted is exactly the same for all the system's elements. The focus of the work is the development of the distributed optimizer, thus, some assumptions have been made in order to reduce the modelling effort:

Reservoir

The reservoir properties are assumed to vary from well to well, and only the reservoir inflow model is included in this model. The wells can be producing from different sections of the reservoir, or even different reservoirs. The reservoir model is not crucial for the network model.

Gas lift injection

The gas lift injection is adjusted by a flow controller on the gas lift injection valve. This variable is considered perfectly controlled to give the desired flowrate, delays and dead time are considered to be negligible.

Manifold dynamics

The manifold to which all the wells are connected is considered to be small enough to assume its dynamics negligible. The pressure in the manifold is supposed to be constant.

Chokes

The wellhead choke valves are assmed to be kept at a fully open position, in order to avoid energy losses that are then considered negligible. The downhole gas lift injection valve is modeled without a choke opening, because it is usually a mandrel with fixed opening.

3.2.1 Mass balance of different phases

It is necessary to write three different mass balances for each well, respectively for the gas in the anulus, the gas in the tubing and the oil in the tubing:

$$\dot{m}_{ga} = w_{gl} - w_{iv}, \tag{3.2.1}$$

$$\dot{m}_{gt} = w_{iv} - w_{pg} + w_{rg}, \tag{3.2.2}$$

$$\dot{m}_{ot} = w_{ro} - w_{po}, \tag{3.2.3}$$

Here m_{ga} is the mass of gas in the anulus, m_{gt} the mass of gas in the well tubing, m_{ot} the mass of oil in the tubing; w_{gl} the gas-lift injection rate, w_{iv} the gas flow from the anulus into the tubing, w_{pg} and w_{po} the produced gas and oil flow rates, w_{rg} and w_{ro} the gas and oil flow rates from the reservoir.

3.2.2 Density models

The densities of gas in the anulus ρ_a and of the fluid mixture in the tubing ρ_m are given by:

$$\rho_a = \frac{M_w p_a}{T_a R}, \tag{3.2.4}$$

$$\rho_m = \frac{m_{gt} + m_{ot} - \rho_o L_r A_r}{L_w A_w}, \tag{3.2.5}$$

Here M_w is the molecular weight of the gas, p_a is the anulus pressure, T_a is the temperature in the anulus, R is the gas constant, ρ_o is the oil density, L_r and L_w are the length of the

well above and below the injection point and A_r and A_w the cross-sectional area of the well above and below the injection point.

3.2.3 Pressure models

The anulus pressure p_a, wellhead pressure p_{wh}, well injection point pressure p_{wi} and bottom hole pressure p_{bh} are given by:

$$p_a = (\frac{T_a R}{V_a M_w} + \frac{g H_a}{L_a A_a}) m_{ga}, \tag{3.2.6}$$

$$p_{wh} = \frac{T_w R}{M_w} (\frac{m_{gt}}{L_w A_w + L_a A_a - \frac{m_{ot}}{\rho_o}}) m_{ga}, \tag{3.2.7}$$

$$p_{wi} = p_{wh} + \frac{g}{A_w L_w} (m_{ot} + m_{gt} - \rho_o L_r A_r) H_w, \tag{3.2.8}$$

$$p_{bh} = p_{wi} + \rho_w g H_r, \tag{3.2.9}$$

Here L_a and A_a are the length and the cross-sectional area of the anulus, H_a is the vertical height of the annulus, T_w is the temperature in the well tubing,H_w and H_r the vertical height of the well tubing below and above the injection point and g is the acceleration of gravity constant. The cross-sectional area of the anulus and the tubing have been computed from their diameters D_a and D_w.

3.2.4 Flow models

The flow through the downhole gas lift injection valve, w_{iv}, total flow through production choke w_{pc}, produced gas w_{pg} and oil w_{po} flow rate and the reservoir gas w_{rg} and oil flowrate w_{ro} are given by:

$$w_{iv} = C_{iv} \sqrt{\rho_a max(0, p_{ai} - p_{wi})}, \tag{3.2.10}$$

$$w_{pc} = C_{pc} \sqrt{\rho_w max(0, p_{wh} - p_m)}, \tag{3.2.11}$$

$$w_{pg} = \frac{m_{gt}}{m_{gt} + m_{ot}} w_{pc}, \tag{3.2.12}$$

$$w_{po} = \frac{m_{ot}}{m_{gt} + m_{ot}} w_{pc}, \tag{3.2.13}$$

$$w_{ro} = PI(p_r - p_{bh}), \tag{3.2.14}$$

$$w_{rg} = GOR \cdot w_{ro}, \tag{3.2.15}$$

Here C_{iv} and C_{pc} are the valve flow coefficients for the downhole injection valve and the production choke, PI is the reservoir productivity index, p_r is the reservoir pressure, p_m the manifold pressure and GOR is the gas-oil ratio.

GOR is assumed as the only time-varying parameter, in particular its value is included in the range:

$$GOR \in \{GOR \pm 2\sigma\}, \tag{3.2.16}$$

Lastly, the cross sectional area for the annulus A_a and tubing A_w foreach well are calculated by using the respective diameters D_a and D_w:

$$A_a = \frac{\pi {D_a}^2}{4} - \frac{\pi {D_w}^2}{4} \tag{3.2.17}$$

$$A_w = \frac{\pi {D_w}^2}{4} \tag{3.2.18}$$

3.2.5 DAE formulation

The system is modeled as a semi-explicit DAE system that has the following form:

$$\dot{\boldsymbol{x}} = f(x, z, u) \tag{3.2.19}$$

$$\boldsymbol{g}(x, z, u) = 0 \tag{3.2.20}$$

Here $f(x, z, u)$ denotes the set of differential equations 3.2.1 - 3.2.3 and $\boldsymbol{g}(x, z, u)$ is the set of algebraic equations 3.2.4 - 3.2.15. This gives a set of differential states x, algebraic states z and decision variables u that are shown below:

$$x = [m_{ga} \quad m_{gt} \quad m_{ot}]^T \tag{3.2.21}$$

$$z = [p_{bh} \quad p_{wi} \quad p_{wh} \quad p_a \quad p_m \quad w_{iv} \quad w_{pc} \quad w_{pg} \quad w_{po} \quad w_{ro} \quad w_{rg}] \tag{3.2.22}$$

$$u = w_{gl} \tag{3.2.23}$$

Where each element of x, z and u is a vector, containing all the values assumed by that variable in the six wells.

3.3 Optimization

Up to now, the system have been described and all the equations needed to mathematically model the network have been presented. In order to perform a mathematical optimization, it is necessary to define the processes contraints and the objective function that must be maximized.

3.3.1 Constraints

As mentioned before, the two subsystems have high independence for what concerns strategies, objectives and the control of the respective wells. However they are coupled by means of global constraints. These global constraints are related to the common equipment, as the gas lift pumps, the piping and the downstream equipment. A gas lift process can have many kinds of constraint, concerning flows, pressures and temperatures. In this work, only the two most common process constraints are considered: a limit in the injection rate of compressed gas in the well and a limited capacity of the produced gas processing.

Gas Lift injection rate

In a gas lift well, part of the produced gas in reinjected in the well, to increase the oil production. Before injecting the gas in the annulus, it is neccessary to compress it to higher pressures. Thus, it is reasonable to assume that the potential maximum rate of gas lift has an upper limit, because of limitations in the compressors' power. Moreover, it is assumed that this maximum allowed gas lift flowrate can vary in time. This is an hard constraint, that can not be violated even during transients.

Gas produced capacity

Another important constraint in the oil and gas extraction process is the gas produced capacity. This constraint must be taken in account when downstream equipments act as a process bottleneck. The downstream equipment has the role of purifying and processing the produced oil and gas. While the oil is the main product of the reservoir, in this case the gas is considered as a by-product, from which no profit can be made. Moreover, while the oil production can be reduced according to strategies and plans, the production of gas depends entirely on the produced oil rate and on the properties of the reservoir. These properties can be unknown and also vary in time, thus it is impossible to predict how much gas will be produced in the future. It can happen that the production of gas is higher than expected during the equipment sizing, thus, not all the produced gas can be processed. This limit of the produced gas capacity can be considered a soft constraint: during stationary conditions the total amount of produced gas can not overcome the limit, but during transients it is considered acceptable to violate, locally, the constraint. In this case, the excess gas is flared, however, the gas flaring is subject to a fine, that can be considered as a carbon tax. This tax is not negligible during the computation of the profits.

3.3.2 Objective function

The objective of any kind of production system is the maximization of the profits, according to process contraints and company's strategies. In the oil production system considered, oil is the main and only source of profits. No profits are associated with the gas production. For what concerns the process costs, the only one considered in this model is related to the

compression and injection of gas lift. The objective function adopted in the optimization problem can be written as in equation 3.3.1:

$$J = (\$_{oil} \sum_{i \in \mathcal{N}} w_{po_i} - \$_{gl} \sum_{i \in \mathcal{N}} w_{gl_i}); \tag{3.3.1}$$

where J is the total income from the plant, $\$_o$ and $\$_{gl}$ represent the price and the cost associated respectively to oil and gas lift compression, while w_{po_i} and w_{gl_i} are the amount of produced oil and injected gas for each well. The values of $\$_o$ and $\$_{gl}$ are supposed to be equal and costant in the two subsystems, but their values can differ, because they are decided in the higher layers of the decision-making gerarchy described in the introduction. Being the two operators independent, it is possible for them to choose different strategies and to associate a higher or lower value to the incomes or outcomes of the plant. Thus, the aim of the global coordinator is not to maximize the plant's profits in an absolute way, but to permit to the two operators to equally maximize their own objective functions with the available resources.

Lastly, it is important to observe that the optimizers developed are static, which means that constraints' violations are not considered in the objective function. Thus, the carbon tax associated with the gas flaring is not mentioned in the plant's objective function. In order to deal with it, it is necessary to implement a more advanced controller (as MPC), able to make choices during transients according to the profits and penalties associated to an excessive gas production.

Chapter 4

Case study optimization

4.1 Introduction

In chapter 3, the system considered as case study and its mathematical model are presented. In this chapter, after a brief description of the softwares used, all the choices and the steps related to the development and implementation of the optimization loop are illustrated. The optimization loop implemented in this work is shown in picture 4.1.

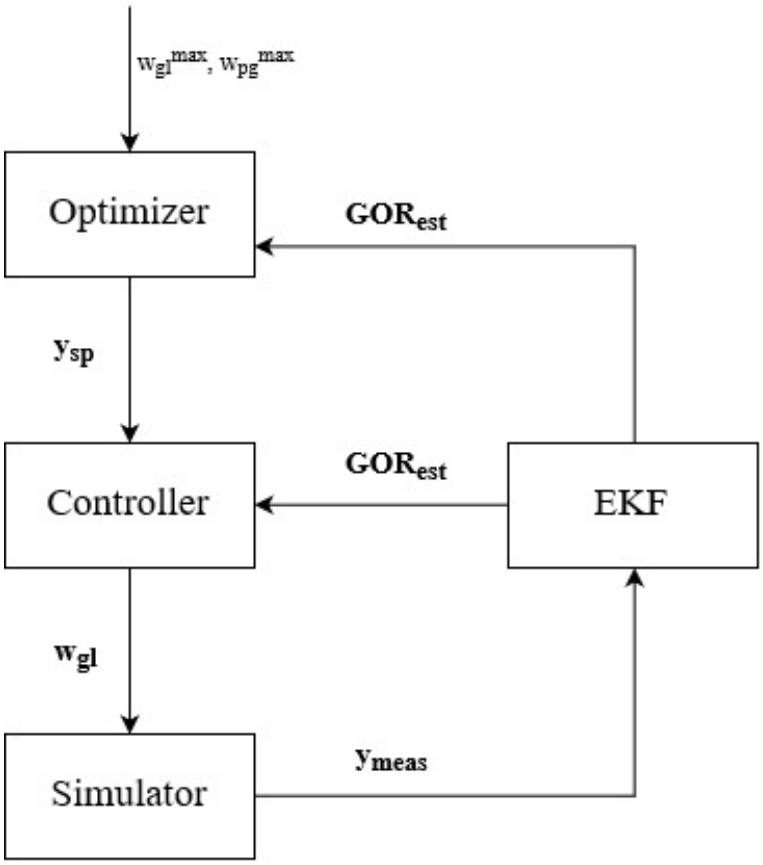

Figure 4.1: Optimization and process loop in the case study.

In this case study, the data describing the plant's behaviour do not come from real measurements. The plant have been simulated by means of a simulator built ad hoc. The data measurements produced by the simulator ($\boldsymbol{y_{meas}}$) are used as input by the dynamic estimator, that consists in an Extended Kalman Filter (EKF). Its role is to estimate the system's states and the unknown parameters. The unknown parameter that EKF must predict is the Gas to Oil Ratio (GOR) for each well. Once they have been evaluated, the optimization can be carried out. Three different optimizers have been developed: one centralized optimizer and two distributed optimizers, based respectively on primal and dual decomposition. The optimizer computes the optimal gas lift injection rate for each well, according to the constraints at the current time step and the estimated value of GOR for each well. These optimal values are used as set point for the control layer. Two possibilities have been tested: a classical Proportional-Integral controller, and a more complex Non Linear Model Predictive Controller (NMPC). This advanced controller computes an optimal set point trajectory that can be used by a lower control layer, as PI controllers. For the sake of simplicity, during the simulations with the NMPC, the lower control layer has not been considered: the optimal set point trajectory is directly implemented in the plant, assuming the lower control layer's dynamic negligible. This optimization loop is carried out once each sample time, that has been chosen to be equal to five minutes. This means that, while the simulator and EKF works continuously, the optimizers and the NMPC solve the respective problem once every five minutes. The EKF must work constinuosly in parallel with the plant simulator in order to keep the system's model always updated.

4.2 Softwares

The case study considered in this work has been simulated and optimized in MATLAB R2017b, a programming environment developed by Mathworks, which is widely used in the academic world. All the documentation regarding this programming language can be found in [https://it.mathworks.com/help/], see also (MathWorks, 2017), (MATLAB, 2017). The mathematical model and the NLP problems have been formulated with CasADi framework (CasADi v.3.4.5), an open source software tool for numerical optimization and optimal control [https://web.casadi.org/docs/, see also (Andersson *et al.*, 2018) (Andersson *et al.*, In Press, 2018)]. CasADi permits a symbolic formulation of the NLP problems, that are then solved with IPOPT (Interior Point OPTimizer), an open source software package for large scale nonlinear optimization. IPOPT implements an interior point line search filter method to find a local solution of standard NLPs. The mathematical details of the algorithm can be found in [https://www.coin-or.org/Ipopt/documentation/, see also (Watcher and Biegler, 2018), (Watcher and Biegler, 2006)]. Lastly, the DAE systems have been integrated and solved using IDAS integrator (IDAS v3.1.0, Implicit Differential-Algebraic solver), a general purpose solver from SUNDIALS (SUite of Nonlinear and DIfferenctial/ALgebraic equation Solvers) for initial value problems (IVPs). More informations about the integrator can be found in [https://computing.llnl.gov/projects/sundials/idas], see also (Hindmarsh *et al.*, 2005a), (Hindmarsh *et al.*, 2005b). The optimization and simulations are carried out on a 3.0 GHz

workstation with 12 GB memory.

4.3 Simulator

The simulator takes the place of the real plant in these tests. It is reasonable to assume that not all the variables are measured in a real plant. Thus, the only measured variables considered are the wellhead pressure (P_{wh}), the bottom hole pressure (P_{bh}), the produced oil and gas flowrates (w_{po}, w_{pg}) and the oil and gas flowrates from the reservoir (w_{ro}, w_{rg}). These measurements are used by the EKF to produce an estimation of states and parameters. The final sistem obtained from this model is the following:

$$\dot{\boldsymbol{x}}_i = f_i(x_i, z_i, u_i, p_i), \tag{4.3.1}$$

$$g_i(x_i, z_i, u_i, p_i) = 0 \qquad \forall i \in \mathbb{N} = \{1, \dots, 6\} \tag{4.3.2}$$

where $f_i(x_i, z_i, u_i, p_i)$ is the set of differential equations (3.2.1) - (3.2.3), $g_i(x_i, z_i, u_i, p_i)$ is the set of algebraic equations 3.2.4 - 3.2.15. x_i,z_i are the differential and algebraic states, u_i are the control inputs (w_{gl_i}) and p_i the time varying parameter (GOR) for the i^{th} well. Statistical noise can be introduced to make the measurements more realistic. However, this noise is unnecessary for the optimizer's analysis, thus, it has been chosen to remove it in order to have simpler and clearer results and graphs.

The simulator runs continuously, and its initial conditions are obtained from past simulations, through which a feasible set of data have been found.

For what concerns the simulator's implementation in the code, the model's equation have been inserted in a CasADi's function to create the Differential Algebraic Sistem (DAE) (4.3.1). The function is called every second of simulation to continuously simulate the system's behaviour. Part of the code defining the simulator can be found in appendix A

4.4 Estimator

In the HRTO or DRTO framework, a dynamic estimator is necessary to provide an estimation of the unmeasured or uncertain variables (in this case the GOR for each well) to the optimizer. This estimator is dynamic, which means that it can work also with transient measurements and produce reasonable estimations. The possibility to carry out the estimation and to solve a static optimization problem even during transients is the real improvement from SRTO to HRTO, as explained in chapter 1. In this work an Extended Kalmann Filter (EKF) has been used as dynamic state estimator. Many estimation methods have been presented in literature, see (Krishnamoorthy, B. A. Foss, *et al.*, 2018) for a brief discussion about other dynamic estimators that could be employed.

4.4.1 Extended Kalman Filter

Extended Kalman Filter is the non-linear counterpart of the classical, linear Kalman Filter. Its mathematical foundations were published between 1959 and 1961 in the papers (R. Kalman, 1960) , (R. E. Kalman and Bucy, 1961). EKF utilizes numerical techniques, as Taylor's expansion, to linearize a model about a working point. If the model is not well-known, Monte Carlo methods are employed.

Unlike its linear counterpart, EKF is not always an optimal estimator. Moreover if the initial estimate of states is wrong, convergence problems can arise. However, EKF needs only simple arithmetic and matrix computation, which means that it is fast and does not require too much computational effort. This is the reason why EKF has been chosen as states dynamic estimator in this work.

Equations

In the EKF, the state transition and observations models must be differentiable functions, while it is not necessary for them to be linear. The state vectors are given by:

$$\boldsymbol{x_k} = f(\boldsymbol{x_{k-1}}, \boldsymbol{u_k}) + \boldsymbol{w_k}; \tag{4.4.1}$$

$$\boldsymbol{z_k} = h(\boldsymbol{x_k}) + \boldsymbol{v_k} \tag{4.4.2}$$

The states $\boldsymbol{x_k}, \boldsymbol{z_k}$ have process and observation noises $\boldsymbol{w_k}, \boldsymbol{x_k}$ with zero mean and gaussian distribution, with covariances $\boldsymbol{Q_k}, \boldsymbol{R_k}$ respectively. The discrete-time extended Kalman filter is presented in the following equations, as stated in (Dan, 2006).

Predict Predict state estimate:

$$\hat{\boldsymbol{x}}_{k|k-1} = f(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k); \tag{4.4.3}$$

Where $\hat{\boldsymbol{x}}_{k|k-1}$ is the a posteriori state estimate at time k given observations up to and including at time k-1, while $\boldsymbol{u_k}$ is the input vector.

Predict covariance estimate:

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k1} \boldsymbol{H}_k^T + \boldsymbol{Q}_k; \tag{4.4.4}$$

Where $\boldsymbol{P}_{k|k-1}$ is the a posteriori error covariance matrix, that measure the estimated accuracy of the state estimate. $\boldsymbol{H}_k$ is the observation model, $\boldsymbol{Q}_k$ the process noise covariance.

Update Innovation or measurement residual

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1}); \tag{4.4.5}$$

Innovation (or residual) covariance:

$$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T + \boldsymbol{R}_k; \tag{4.4.6}$$

Near-Optimal Kalman gain:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1}; \tag{4.4.7}$$

Update state estimate:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k; \tag{4.4.8}$$

Update covariance estimate:

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k|k-1}; \tag{4.4.9}$$

Where the state transition and observation matrices are defined as follows:

$$\boldsymbol{F}_k = \frac{\partial f}{\partial \boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k}; \tag{4.4.10}$$

$$\boldsymbol{H}_k = \frac{\partial h}{\partial \boldsymbol{x}}\Big|_{\hat{\boldsymbol{x}}_{k|k-1}}; \tag{4.4.11}$$

Implementation and validation

For what concerns EKF's implementation, a specific MATLAB function has been developed, in which the model's equation are used to generate the state transition matrix $\boldsymbol{F}_k$ and the state observation matrix $\boldsymbol{H}_k$. Every second the matrix calculations are carried out, and a new value of GOR is estimated constinuously.

In order to perform the estimation, EKF requires some plant's measurement. The data coming from the simulator are the values of $P_{wh}, P_{bh}, w_{po}, w_{pg}, w_{ro}, w_{rg}$. In figures 4.2 - 4.5 the results of a simplified validation are shown. The simulation regards a single gas lifted well, simulated for a total time of six hours. The unknown parameter is the GOR, which is also the only source of disturbances in the system.

Figures 4.3 - 4.4 - 4.5 show the measurements used by EKF to carry out the estimation. In figure 4.2 the continuous blue line is the real value of GOR, that varies in time. The dashed red line is the estimated GOR value. As we can see, the estimated value is exactly the same as the real value. Moreover, being EKF a dynamic estimator and running it with the same frequency as the simulator itself, no delays are present. Thus, it is possible to state that EKF produces valid estimation of the GOR. In appendix B all the plots regarding measurements and GOR estimation during the extended simulation are reported, while the EKF function implemented in the code can be found in appendix A.

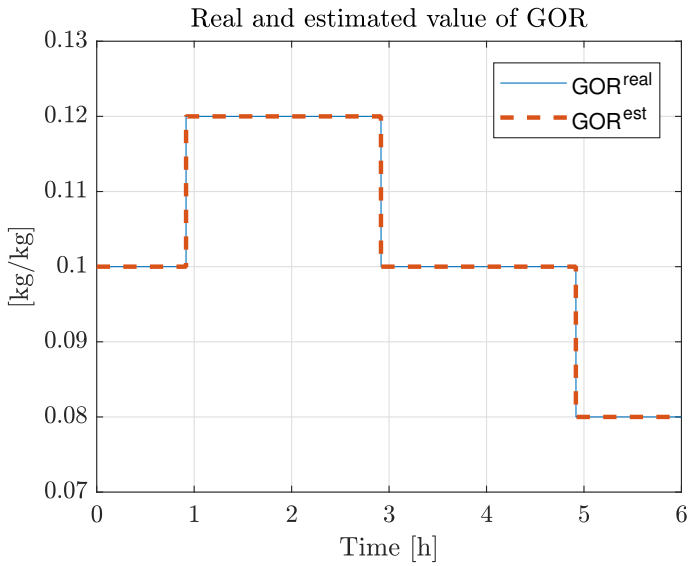

Figure 4.2: Comparison between real and estimated GOR.

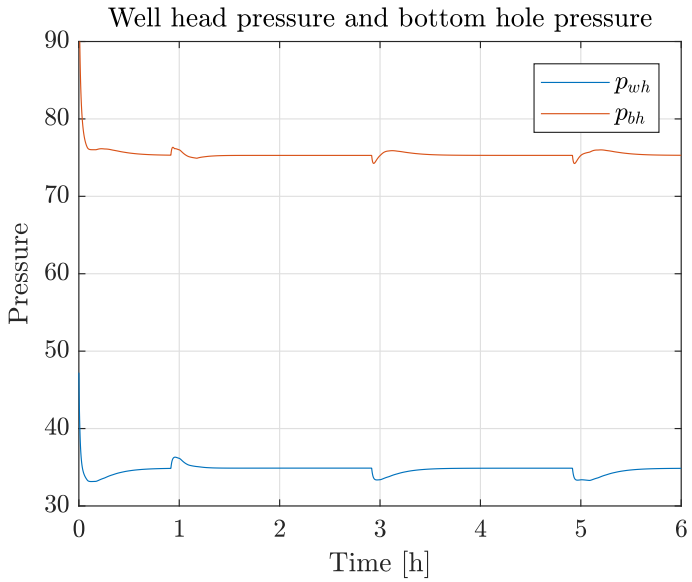

Figure 4.3: Pressure measurements during EKF validation.

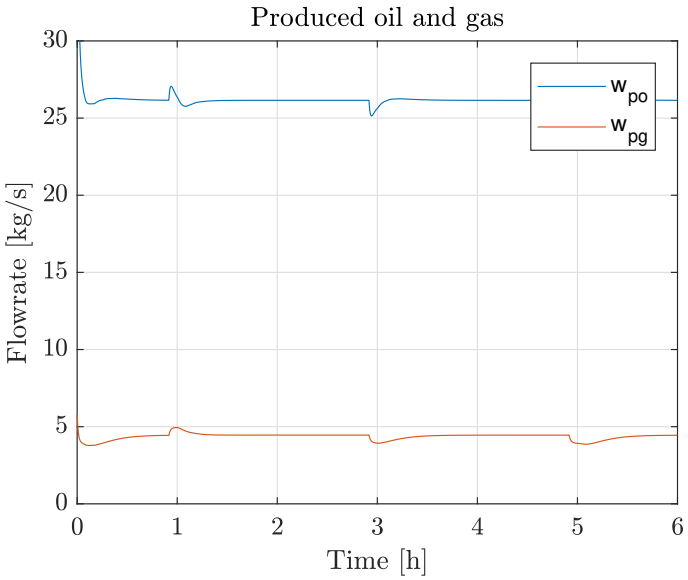

Figure 4.4: Produced flows measurements during EKF validation.

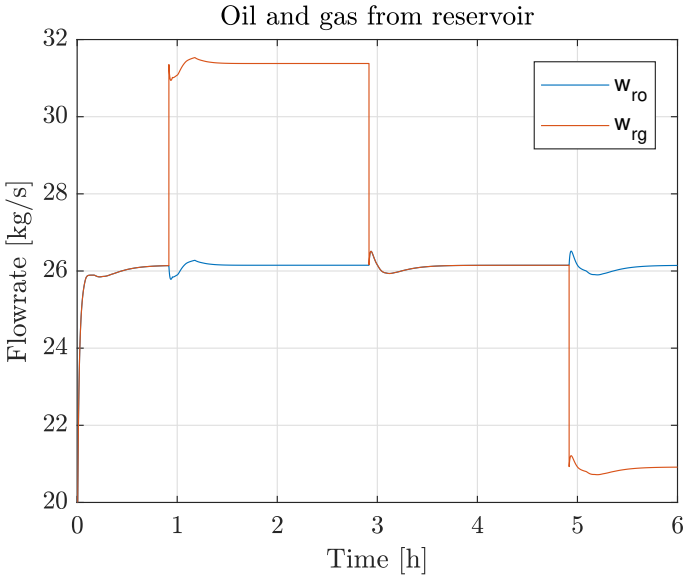

Figure 4.5: Reservoir flows measurements during EKF validation.

4.5 Optimizer

The general framework of the optimizer used in this work is the Hybrid RTO (HRTO), described in chapther 1. HRTO fundamental steps are a dynamic state estimator (EKF) and a static optimizer (see figure 4.6), that computes the optimal value of the controlled variables. In this work, three different optimizers have been developed: a classical centralized RTO and two distributed optimizers, in which the problem has been decomposed by means of primal and dual decomposition.

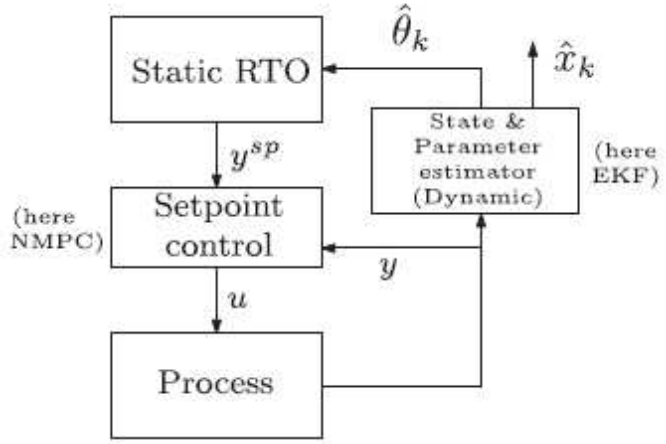

Figure 4.6: Hybrid Real Time Optimization, from (Krishnamoorthy, B. A. Foss, *et al.*, 2018).

All the optimizer works with a sample time equal to 300 seconds. A low sample time value would grant a high promptness and accuracy of the optimizer regarding the occurance of disturbances. However, it is necessary to consider also the time required to perform the optimization, thus it is necessary to discretize the simulation time in sample times long enough to solve the optimization problem. 300 seconds has been chosen as a good compromise between accuracy and computational efficiency, allowing to consider negligible delays in the computation of new optimal condition and to finish in time the calculations required.

The input of all the optimizers are the current constraints, that are supposed to vary in time, and the estimated GOR values from the EKF. Their output are the optimal setpoints for the controlled variables in each well.

4.5.1 Centralized optimizer

The centralized optimizer is the classical static optimizer used in SRTO or HRTO. The system is considered as a whole, and only one large NLP is formulated. The static optimization problem that must be solved is the following:

$$\min_{w_{gl_i}} \; -J = -(\$_{oil} \sum_{i \in \mathcal{N}} w_{po_i} - \$_{gl} \sum_{i \in \mathcal{N}} w_{gl_i}); \tag{4.5.1}$$

$$s.t. \quad \boldsymbol{x} = \boldsymbol{F}_c(x, z, u, \theta); \tag{4.5.2}$$

$$\boldsymbol{G}(x, z, u, \theta) = 0; \tag{4.5.3}$$

$$\sum_{i \in \mathcal{N}} w_{pg_i} \leq w_{pg}^{max}; \tag{4.5.4}$$

$$\sum_{i \in \mathcal{N}} w_{gl_i} \leq w_{gl}^{max}; \tag{4.5.5}$$

where: J is the objective function, that represent the total income from the plant, $\$_o$ and $\$_{gl}$ are the incomes and costs associated to the produced oil and the gas lift injection, respectively. Equations (4.5.2) - (4.5.3) represent the system's model, respectively the set of differential equations (3.2.1) - (3.2.3) and of algebraic equation (3.2.4) - (3.2.15). Equations (4.5.4) - (4.5.5) are the constraints.

The NLP problem (4.5.1) - (4.5.5) is developed in CasADi v3.0.1 using the MATLAB programming environment and solved using IPOPT. It is interesting to observe that this is a minimization problem and the objective function (4.5.1) is defined as the opposite of plant's revenues. The reason of this choice is that IPOPT is able to solve only minimization problems. In order to perform a maximization, it is necessary to simply change the sign of the objective function and minimize it.

In the MATLAB code, a new NLP problem is formulated at each sample time, updating the previous problem with the new constraints and the newly estimated GORs. To carry out the optimization, a first guess value is necessary. At each sample time, the optimal value computed in the previous sample is used as first guess. This is known as "warm start", and it allows to reduce the computational effort: if the GOR and the constraints have not changed between the previous and the current samples, the optimization problem is unchanged, and the solution is the same. Thus, if we suppose major disturbances to be not common, the optimizer will need only one iteration for most of the time, focusing the computational effort only in the samples in which disturbances arise.

This centralized optimizer was already developed in previous works (see **??**). Although it was applied for smaller networks composed by no more than two wells, its validation was already performed. Thus, it is considered valid and it is used as a reference for the validation of the distributed optimizers.

4.5.2 Distributed Optimizers

In the distributed optimizer, the two subsystems are optimized individually. A global coordinator is required to solve the master problem, that iteratively updates the allocation of resources (primal decomposition) or their price (dual decomposition), so that the whole system satisfies the global constraints. The two optimizers are implemented in a similar way: at each sample time two NLP subproblems are formulated, one for each cluster. The

subproblems are solved individually and the value of the objective function's subgradient (primal decomposition) or of the optimal values for the CVs (dual decomposition) are used as input for the master problem. The solution of the master problem permits to update the subproblems for the next iteration. This inner coordination routine continues until convergence is achieved.

As in the centralized optimizer, a warm start is introduced: the first guess values are the optimal values computed at the previous iteration. In this way, it is possible to lower significantly the average number of iterations required. However, the necessity of solving the master problem in an iterative way increases the number of calls of IPOPT at each sample time. The system considered is not large enough to have an appreciable reduction of computational time by means of system's decompostion. Thus, we suppose that the distributed optimizers will require, on average, more time to find a solution respect to the centralized one.

Primal decomposition

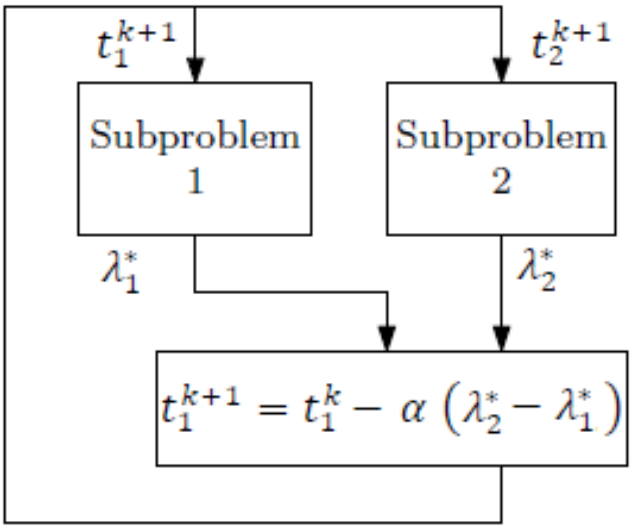

Figure 4.7: Primal decomposition block diagram, as described in algorithm 4.5.1.

Primal decomposition is an auction-based approach that iteratively reallocates the shared resources among the subsystems. In the system considered, the "resources" are the constraints themselves: the limited gas lift and gas produced flowrate. At each iteration the master problems allocate part of the available resources to the subsystems, through which the local constraints can be updated and the subproblems solved. The allocation of resources is updated according to the gradient of the solution with respect to w_{gl} or w_{pg}. The following two NLP subproblems are solved separately:

$$\min_{w_{gl_i}} \; -J_1 = -(\$_{oil} \sum_{i \in \mathcal{N}_1} w_{po_i} - \$_{gaslift} \sum_{i \in \mathcal{N}_1} w_{gl_i}); \tag{4.5.6}$$

$$s.t. \quad \boldsymbol{x} = \boldsymbol{F}_c(x, z, u, \theta); \tag{4.5.7}$$

$$\boldsymbol{G}(x, z, u, \theta) = 0; \tag{4.5.8}$$

$$\sum_{i \in \mathcal{N}_1} w_{pg_i} \leq t_1^{pg}; \tag{4.5.9}$$

$$\sum_{i \in \mathcal{N}_1} w_{gl_i} \leq t_1^{gl}; \tag{4.5.10}$$

$$\min_{w_{gl_i}} \; -J_2 = -(\$_{oil} \sum_{i \in \mathcal{N}_2} w_{po_i} - \$_{gaslift} \sum_{i \in \mathcal{N}_2} w_{gl_i}); \tag{4.5.11}$$

$$s.t. \quad \boldsymbol{x} = \boldsymbol{F}_c(x, z, u, \theta); \tag{4.5.12}$$

$$\boldsymbol{G}(x, z, u, \theta) = 0; \tag{4.5.13}$$

$$\sum_{i \in \mathcal{N}_2} w_{pg_i} \leq t_2^{pg} = w_{pg}^{max} - t_1^{pg}; \tag{4.5.14}$$

$$\sum_{i \in \mathcal{N}_2} w_{gl_i} \leq t_2^{gl} = w_{gl}^{max} - t_1^{gl}; \tag{4.5.15}$$

Where N_1, N_2 are the wells in each cluster, t_1^{pg} and t_1^{gl} are the allocated resources for the first cluster, t_2^{pg} and t_2^{gl} the allocated resources for the second cluster. In primal decomposition, the system decomposition is performed on the coupling constraints, that are split between the two subsystems. In order to make the two subproblems equivalent to the original problem, $\boldsymbol{t_2}$ is defined as the difference between the total maximum value of the global constraint and the local constraint $\boldsymbol{t_1}$.

The two optimization problems (4.5.6) - (4.5.11) can be solved independently by the two cluster's operators. A central coordinator is required to solve the master problem and to update $\boldsymbol{t}^{wgl}$ and $\boldsymbol{t}^{pg}$ at each k iteration:

$$t_{1,k+1}^{wgl} = t_{1,k}^{w_g l} - \alpha_{wgl}(\lambda_2^{gl} - \lambda_1^{gl}) \tag{4.5.16}$$

$$s.t. \quad t_{1,k+1}^{wgl} \leq w_{gl}^{max} \tag{4.5.17}$$

$$t_{1,k+1}^{wpg} = t_{1,k}^{w_p g} - \alpha_{wpg}(\lambda_2^{pg} - \lambda_1^{pg}) \tag{4.5.18}$$

$$s.t. \quad t_{1,k+1}^{wpg} \leq w_{pg}^{max} \tag{4.5.19}$$

Where α_{wgl} and α_{wpg} are parameters that governs the update rate, $t_{1,k+1}^{wgl}$ and $t_{1,k+1}^{wpg}$ are the new values for the constraints of the first cluster, $\lambda_{1,2}^{gl}$ and $\lambda_{1,2}^{pg}$ are subgradients of the two solutions respect to the constraints. The master problem is solved until the values of t_1^{gl}

and t_1^{pg} converge to a definitive value. The primal decomposition optimizer's algorithm is the following:

Algorithm 4.5.1: Master problem for primal decomposition.

Input : $GOR^{est}, w_{gl}^{max}, w_{pg}^{max}$

Output : Optimal solutions w_{gl}^{opt}, w_{pg}^{opt}; Allocated resources t_1^{pg}, t_1^{gl},t_2^{pg}, t_2^{gl}

Data : Convergence rates α_{wgl}, α_{wpg}, Tolerance ε, Maximum number of iterations N_{it}^{max}

▷ *Initialization: set initial time and difference*
$k \leftarrow 1$, $\Delta t_{wg} \leftarrow 1$, $\Delta t_{gl} \leftarrow 1$

▷ *Optimization cycle*
while $k < N_{it}^{max}$ **do**

▷ *Set optimization constraints*
$g_1 = [t_{1,k}^{pg}, t_{1,k}^{gl}]; g_2 = [t_{2,k}^{pg}, t_{2,k}^{gl}]$

▷ *Solve optimization subproblems*
find $\max_{w_{gl_i}} J_1$ s.t $\left[\sum_{i\in\mathcal{N}_1} w_{gl,1}^{opt}; \sum_{i\in\mathcal{N}_1} w_{pg,1}^{opt}\right] \leq g_1$
find $\max_{w_{gl_i}} J_2$ s.t $\left[\sum_{i\in\mathcal{N}_2} w_{gl,2}^{opt}; \sum_{i\in\mathcal{N}_2} w_{pg,2}^{opt}\right] \leq g_2$

▷ *Get optimal solutions w_{gl}^{opt}, and subgradients $\Lambda_{1,2}^{gl}, \Lambda_{1,2}^{pg}$*

▷ *Master Problem: compute new allocation of resources and differences*
$t_{1,k+1}^{gl} = t_{1,k}^{gl} - \alpha_{wgl} * (\Lambda_2^{gl} - \Lambda_1^{gl})$
$t_{1,k+1}^{pg} = t_{1,k}^{pg} - \alpha_{wpg} * (\Lambda_2^{pg} - \Lambda_1^{pg})$
$t_{1,k}^{gl} = \min(t_{1,k+1}^{w_{gl}}; t_{1,k}^{w_{gl}})$
$t_{1,k}^{pg} = \min(t_{1,k+1}^{w_{pg}}; t_{1,k}^{w_{pg}})$
$t_2^{gl} = w_{gl}^{max} - t_1^{wgl}$
$t_2^{pg} = w_{pg}^{max} - t_1^{pg}$
$\Delta t^{gl} = t_{1,k+1}^{gl} - t_{1,k}^{gl}$
$\Delta t^{pg} = t_{1,k+1}^{pg} - t_{1,k}^{pg}$

if $\Delta_{gl} \leq \varepsilon \wedge \Delta_{pg} \leq \varepsilon$ **then**
▷ *Keep optimal values and constraints for subproblems*
return
end
end

The algorithm 4.5.1 describes the procedure adopted during the implementation: as long as the number of iterations is lower than N_{it}^{max}, the two subproblems are solved, the optimal solutions collected and the subgradients of the solutions Λ^{pg} and Λ^{gl} stored. After this, the master problem is formulated and new values of t_1^{pg} and t_1^{gl} are computed. If convergence is achieved, the values obtained are valid and an optimal solution is given as an output,

otherwise the loop continues. A block diagram of algorithm 4.5.1 is presented in figure 4.7.

Dual decomposition

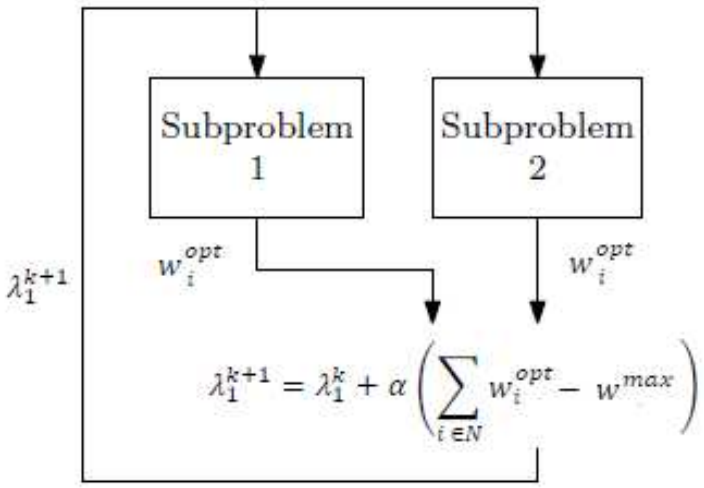

Figure 4.8: Dual decomposition block diagram, as described in algorithm 4.5.2.

In dual decomposition the constrained optimization problem is converted in an unconstrained one, by means of the Lagrangian of the objective function. Differently from primal decomposition, in dual decomposition there is no allocation of resources: at each iteration the price of the resources is updated proportionally to the difference between the resources utilized and the constraints. These resources' prices are represented by the Lagranian multipliers associated to the constraints. As mentioned in chapter 2, it is important to remember that the lagrangian multiplier will be null only when the constraint is active, otherwise it will assume a positive value. The following NLP problems are solved for both the clusters:

$$\begin{aligned} \min_{w_{gl_i}} \; -L_1 = -\,(\$_{oil} \sum_{i \in \mathcal{N}_1} w_{po_i} - \$_{gaslift} \sum_{i \in \mathcal{N}_1} w_{gl_i} \\ + \lambda_{gl}(\sum_{i \in \mathcal{N}_1} w_{gl_i} - \frac{w_{gl}^{max}}{2}) + \lambda_{pg}(\sum_{i \in \mathcal{N}_1} w_{pg_i} - \frac{w_{pg}^{max}}{2})); \end{aligned} \tag{4.5.20}$$

$$s.t. \quad \boldsymbol{x} = \boldsymbol{F}_c(x, z, u, \theta); \tag{4.5.21}$$

$$\boldsymbol{G}(x, z, u, \theta) = 0; \tag{4.5.22}$$

$$\sum_{i \in \mathcal{N}_1} w_{pg_i} \leq t_1^{pg}; \tag{4.5.23}$$

$$\sum_{i \in \mathcal{N}_1} w_{gl_i} \leq t_1^{gl}; \tag{4.5.24}$$

$$\min_{w_{gl_i}} \; -L_2 = -\,(\$_{oil} \sum_{i\in\mathcal{N}_2} w_{po_i} - \$_{gaslift} \sum_{i\in\mathcal{N}_2} w_{gl_i} + \lambda_{gl}(\sum_{i\in\mathcal{N}_2} w_{gl_i} - \frac{w_{gl}^{max}}{2}) + \lambda_{pg}(\sum_{i\in\mathcal{N}_2} w_{pg_i} - \frac{w_{pg}^{max}}{2}));$$ (4.5.25)

$$s.t. \quad \boldsymbol{x} = \boldsymbol{F}_c(x, z, u, \theta);$$ (4.5.26)

$$\boldsymbol{G}(x, z, u, \theta) = 0;$$ (4.5.27)

$$\sum_{i\in\mathcal{N}_2} w_{pg_i} \leq t_1^{pg};$$ (4.5.28)

$$\sum_{i\in\mathcal{N}_2} w_{gl_i} \leq t_1^{gl};$$ (4.5.29)

Where L is the Lagrangian of the cost function, λ_{gl} and λ_{pg} are the lagrangian multipliers associated to the two constraints. Similarly to the previous case, the decomposition is performed on the coupling constraints. It is important that the sum of the local constraints in the equations (4.5.20) - (4.5.25) is equal to the value of the global constraint. The two optimization problems (4.5.20) - (4.5.25) can be solved independently by the two cluster's operators. A central coordinator is required to solve the master problem, that updates the lagrangian multipliers at each iteration as follows:

$$\lambda_{gl}^{k+1} = \lambda_{gl}^{k} + \alpha_{gl}(\sum_{i\in\mathcal{N}} w_{gl_i}^{opt} - w_{gl}^{max})$$ (4.5.30)

$$s.t \quad \lambda_{gl}^{k+1} \geq 0$$ (4.5.31)

$$\lambda_{pg}^{k+1} = \lambda_{pg}^{k} + \alpha_{pg}(\sum_{i\in\mathcal{N}} w_{pg_i}^{opt} - w_{pg}^{max})$$ (4.5.32)

$$s.t \quad \lambda_{pg}^{k+1} \geq 0$$ (4.5.33)

Where λ_{gl}^{k+1} and λ_{pg}^{k+1} are the Lagrangian multipliers of the constraints at the next iteration, $w_{gl_i}^{opt}$ and $w_{pg_i}^{opt}$ are the optimal values computed from the optimizations of the two subsystems. α_{gl} and α_{pg} are, again, two parameters that govern the update rate. It is important to impose the conditions (4.5.31) and (4.5.33), to avoid wrong results in the optimization. The loop continues until convergence is achieved. The dual decomposition optimizer's algorithm is the following:

Algorithm 4.5.2: Master problem for dual decomposition.

Input : $GOR^{est}, w_{gl}^{max}, w_{pg}^{max}$

Output : Optimal solutions w_{gl}^{opt}, w_{pg}^{opt}

Data : Convergence rates α_{wgl}, α_{wpg}, Tolerance ε, Maximum number of iterations N_{it}^{max}

▷ *Initialization: set initial time and difference*
$k \leftarrow 1$, $\Delta t_{wg} \leftarrow 1$, $\Delta t_{gl} \leftarrow 1$

▷ *Optimization cycle*
while $k < N_{it}^{max}$ **do**

▷ *Define Lagrangians of cost functions*
$L_1 = \$_{oil} \sum_{i \in \mathcal{N}_1} w_{po_i} - \$_{gaslift} \sum_{i \in \mathcal{N}_1} w_{gl_i} + \lambda_{gl}(\sum_{i \in \mathcal{N}_1} w_{gl_i} - \frac{w_{gl}^{max}}{2}) + \lambda_{pg}(\sum_{i \in \mathcal{N}_1} w_{pg_i} - \frac{w_{pg}^{max}}{2})$
$L_2 = \$_{oil} \sum_{i \in \mathcal{N}_2} w_{po_i} - \$_{gaslift} \sum_{i \in \mathcal{N}_2} w_{gl_i} + \lambda_{gl}(\sum_{i \in \mathcal{N}_2} w_{gl_i} - \frac{w_{gl}^{max}}{2}) + \lambda_{pg}(\sum_{i \in \mathcal{N}_2} w_{pg_i} - \frac{w_{pg}^{max}}{2})$

▷ *Solve optimization subproblems*
for $i = 1, 2$ **do**
find $\max_{\boldsymbol{w_{gl}}} L_i$;
end

▷ *Get optimal solutions* w_{gl}^{opt}

▷ *Master Problem: compute new cost of resources and differences*
$\lambda_{pg}^{k+1} = \max(0; \lambda_{pg}^{k} + \alpha_{pg}(\sum_{i \in \mathcal{N}} w_{pg_i}^{opt} - w_{pg}^{max}))$
$\lambda_{gl}^{k+1} = \max(0; \lambda_{gl}^{k} + \alpha_{gl}(\sum_{i \in \mathcal{N}} w_{gl_i}^{opt} - w_{gl}^{max}))$
$\Delta\lambda_{gl} = \lambda_{1,k+1}^{w_{gl}} - \lambda_{1,k}^{w_{gl}}$
$\Delta\lambda_{pg} = \lambda_{1,k+1}^{w_{pg}} - \lambda_{1,k}^{w_{pg}}$

if $\Delta_{gl} \leq \varepsilon \wedge \Delta_{pg} \leq \varepsilon$ **then**
▷ *Keep optimal values*
return
end
end

At each iteration k, the Lagrangian of the cost function is formulated, and the two unconstrained problems solved. Then, the values of the lagrangian multipliers are updated according to the solutions obtained from the two subproblems and the current constraints. In this way, if the constraint is not active, the Lagrangian multipliers are reduced, while if the constraint is active, its lagrangian multiplier is constant and equal to zero. A block diagram of algorithm 4.5.2 is presented in figure 4.8.

4.6 Control layer

The optimizer computes the optimal values for each variable in the system. These values are given as set points to the control layer, whose aim is to keep the variables aligned to the respective optimal values. Two different control layers have been tested: a classical Proportional-Integral controller (PI controller) and a Non-Linear Model Predictive Controller (NMPC).

4.6.1 Proportional-Integral controller

Introduction

The Proportional-Integral controller is a common feedback control loop, widely used in industrial control system and in a variety of other applications. Its main advantages are the very easy implementation, that can be performed even in a mechanical way, and the almost null computational effort required. As the name suggest, the PI controller is able to control the system by means of an action composed by two terms: a proportional term and an integral term. The proportional term implements in the system a control action proportional to the error ε, defined as the difference between the actual value of the controlled variable (CV) and its desired set point (SP). This kind of action is characteristic of the pure Proportional controllers (P controllers). Its main limit is the impossibility in aligning the CV exactly to SP: the presence of a steady state offset is an intrinsic characteristic of this kind of controller. In order to fix this problem, that can make the P controller unsuitable for applications in which a great accuracy is required, the PI controller have been developed. The PI controller implements, parallely to the P action, a control term that depends on the integral of ε in time. In this way, small errors extended in time acquire a greater importance and can be removed, reducing the offset to zero. However, the implementation of a control loop based on integral action increases the system dynamic's order. Thus, even linear system can become unstable because of the presence of a PI controller. A higher attention must be put in the tuning of a PI controller, in order to control the closed loop dynamics and to avoid undesired responses, as overshoots or fast oscillations.

Implementation

For what concerns the implementation of this controller, two different CVs have been tested: the wellhead pressure (P_{wh}) and the produced gas flowrate (w_{pg}). The most obvious way to control this system would be to control the manipulate gas lift injection rate by means of the compressors' power or the opening of the Gas Lift choke. However, these elements are not considered in our model, thus, we chose to use these two variables as CVs and to compare the respective controller's behaviour. For both the CVs, the optimal setpoint is computed by the optimizer, then a simple law is implemented to manipulate the inlet gas flowrate (manipulated variable, MV):

$$\varepsilon = y - y_{sp} \ ; \tag{4.6.1}$$

$$y = y_0 + K_c \cdot (\varepsilon + 1/\tau_I \int_0^t \varepsilon \ dt \ ); \tag{4.6.2}$$

where ε is defined as error, y is the current value of the CV, y_{sp} is CV's setpoint, K_c is the proportional gain and τ_I is the integral time.

As we can see, there are two parameters that characterize the controller action: K_c and τ_I. K_c is defined as proportional gain, and it quantifies the proportional action of our controller, while τ_I is a separate weight for the integral action.

The tuning of these parameters is of the primary importance, because it permits to decide the properties of the closed loop response. High values of K_c/τ_I increases the controller promptness and can make the response underdamped, while low values of K_c/τ_I make the response smoother and overdamped. Tuning parameters have been found with SIMC rules, following the steps described in [http://folk.ntnu.no/skoge/prosessregulering/lectures/SiS6SIMC_tuning.pdf, see also (Skogestad, 2017)].

SIMC rules are an improvement of the Internal Model Control (IMC) tuning procedure and they have been presented in 2003 in the paper "*'Simple analytic rules for model reduction and PID controller tuning*" (Skogestad, 2003). The first step to perform the tuning according to these rules, is to find a first order plus delay model of the system considered. Thus, an open loop step response simulation have been carried out. This simulation is shown in figure 4.9.

The open loop step response is performed disconnecting the controller from the system and introducing an unitary step change disturbation in the MV. The CV's response is monitored and from the graph obtained it is possible to obtain infromations about the process reaction curve (PRC). The system is approximated by a first order model, whose parameters are the gain of the sistem k' and its carachteristic time τ_1. The gain k' is defined in equation (4.6.3), it represents the ratio between the increase in CV's response (Δy) and the MV's step change (Δu), while τ_1 is the time needed to reach the 63% of the steady state response value. k' and τ_1 for the system considered are presented in table 4.1.

After deciding the desired characteristic time (τ_C) for the closed loop response (in our case equal to 10 *min*), it is possible to easily compute the tuning parameters for the PI controller:

$$k' = \frac{\Delta y}{\Delta u}; \tag{4.6.3}$$

$$k_p = \frac{1}{k'} \frac{\tau_1}{\tau_c}; \tag{4.6.4}$$

$$\tau_I = \min\{\tau_1, 4 \ \tau_c\}; \tag{4.6.5}$$

The tuning parameters adopted in the work are presented in table 4.2

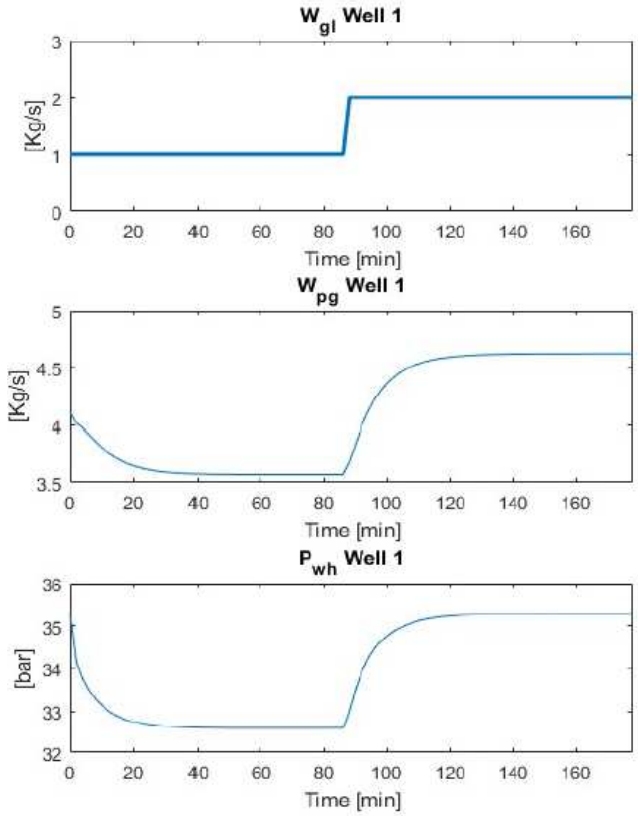

Figure 4.9: Open loop step response.

Table 4.1: Parameters computed from open loop step response simulation.

Controlled variable	Gain k'	Characteristic time τ_1 [min]
w_{pg}	1.04	11
P_{wh}	2.71	9

Table 4.2: Tuning parameters with SIMC rules.

Controlled variable	K_p	τ_I [min]
w_{pg}	0.8750	11
P_{wh}	0.3321	9

Once the two controllers have been tuned it is possible to implement them in the code. The controller's code lines are inserted just before the simulator. At each sample time, they receive a set point value for their CV from the optimizer and they continuously compute the respective manipulated variable w_{gl} according to equation 4.6.2.

4.6.2 Non Linear Model Predictive Controller

A Model Predictive Controller (MPC) uses a dynamic model to predict the system behaviour for a finite time-horizon and generate an optimal set point trajectory for the lower control layer. The predicted input variable is optimized according to a defined objective function implemented in the controller. For a detailed theory and examples of MPC design see "Model Predictive Control: Theory and design" by Rawlings and Mayne (Rawlings and Mayne, 2009) and "Nonlinear Model Predictive Control: Theory and Algorithm" by Grune (Grune and Pannek, 2011).

Traditionally, MPC consists of an optimization problem to achieve certain control objectives, such as set poin control, state and input constraints, rate of change constraints etc. The behaviour of the control variable is determined from an optimization starting at the current position and predicted for a given time-horizon into the future. The measured variables for the states of the system are given as new initial conditions for the next horizon optimization. The receding horizon strategy that define MPC is illustrated in figure 4.10, where the measured variables and the control variables are plotted both in the past and in the predicted future. The measured and the predicted output is compared to a reference trajectory.

The advantage of MPC method is that it can anticipate future changes and modify the controlled variable accordingly, hence diminishing the impact of changes on the operating conditions. For a plant with an implemented MPC, the benefits could be less down time, better performance of control and improved flexibility.

NMPC is a MPC with a nonlinear cost function or constraints equations. In this thesis, the control system is an NMPC with a cost function of eq. (4.6.6):

$$J = \sum((w_{gl_i} - w_{gl}^{SP})^2 + \$_{flare} \cdot s + \gamma(\Delta u)^2) \tag{4.6.6}$$

Where w_{gl_i} is the control input, w_{gl}^{SP} is the set point computed by the optimizer, s is a slack variable, that represents the dinamic violation of the system for the w_{pg} constraint. The slack variable represents the gas flowrate sent to the flare and $\$_{flare}$ is the penalty associated to the flared gas, that can be a process cost or a tax related to the gas emissions, as a carbon tax. Lastly, Δu is the difference between the control action at the current time step and the control action at the next time step, while γ is a penalty cost associated to this difference. This penalty is introduced to reduce the difference between consecutive control action, in order to have a smoother set point trajectory. Therefore, the MPC developed generates an optimal set point trajectory according to:

- the current error $\varepsilon = (w_{gl_i} - w_{gl}^{SP})$, as a proportional controller would do;

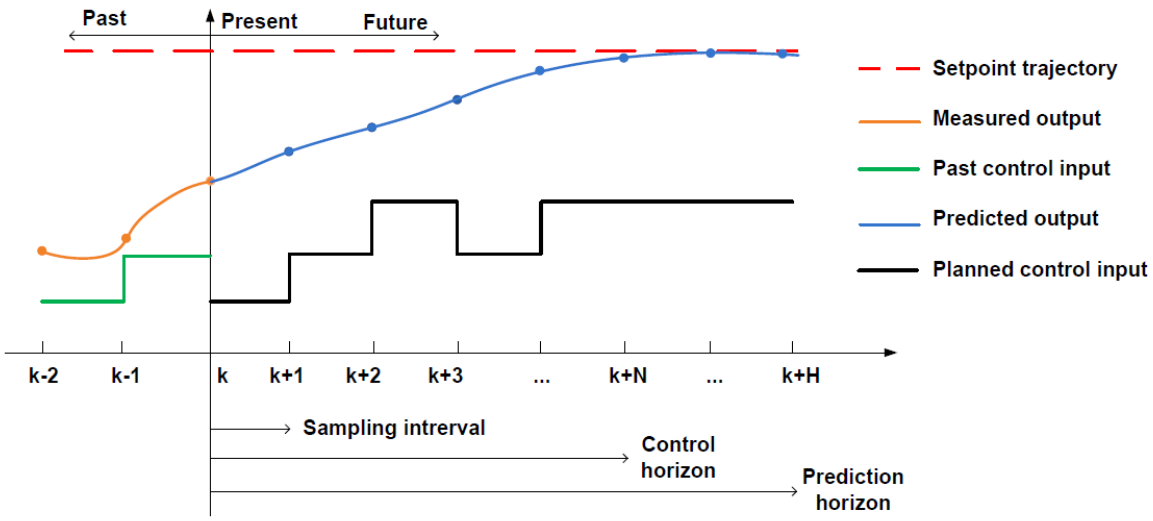

Figure 4.10: Model predictive controller, from (Moradzadeh *et al.*, 2014)

- the dynamic violation of constraint $\$_{flare} \cdot s$;
- a smoother gas lift manipulation $\gamma(\Delta u)^2$.

The prediction horizon (PH) has been chosen equal to two hours. This period must be sufficiently long to ensure stablity [Maciejowski, 2001] but it must be also as low as possible in order to reduce computational time: the adoption of a long PH will require a greater computational effort for the NMPC, that must simulate the system for all the PH's duration.

The NMPC computes the optimal set point trajectory for all the prediction horizon at each sample time (300 seconds). However, only the first control input computed is actually implemented in the real system, the input for the time step t_{k+2} will be computed again at the next sample time, in order to deal with unexpected disturbances. This controller has not any feedback from the plant during the sample time, but its model is updated with the newly estimated GOR at the beginning of each sample, alongside with the optimizer's model.

In the NMPC, a dynamic model of the system is implemented. A higher order method is required for solving the dynamic optimization problem. In this work, the dynamic model is discretized using a thid order direct collocation scheme. Collocation method is a higher order Runge-Kutta method, often used to solve Optimal Control Problems (OCP). Other direct methods used to discretize and solve OCP problems are the Single Shooting and Multiple Shooting, but Direct Collocation is considered as the default choice for DAE systems.

By means of direct collocation it is possible to approximate a function $f(t)$ over an interval $[t_k, t_{k+1}]$ using Lagrange polynomials of order K (equal to three in this work). The Lagrangian Polynomials are defined as follows:

$$P_{k,i}(t) = \prod_{j=1, j\neq i}^{k} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}, \tag{4.6.7}$$

of order K, with property:

$$P_{k,i}(t_{k,j}) = 1 \quad \text{if} \quad i = j; \tag{4.6.8}$$
$$P_{k,i}(t_{k,j}) = 0 \quad \text{if} \quad i \leq j; \tag{4.6.9}$$

The interpolation is performed as follows, using the real parameters $\Theta_{k,i}$

$$f(\Theta_{k,i}, t) = \sum_{i=0}^{K} \Theta_{k,i} \; P_{k,i}(t) \tag{4.6.10}$$

with the property:

$$f(\Theta_{k,i}, t) = \Theta_{k,i} \tag{4.6.11}$$

The collocation of the interpolation points depends on the order of the polynomial and the method adopted. The two most common methods for direct collocation are the Radau roots or Gauss-Legendre. For DAEs systems, Radau collocation points are suggested, while Gauss-Legendre are better suited for ODEs. Radau method is less exposed to oscillations and is stable for stiff differential system equations, due to the presence of a collocation point at the beginning and at the end of the sample time that also allows an easy implementation of constraints on the differential and algebraic states. The adoption of a direct collocation method, allows to formulate the NLP problem as a large, structured and sparse matrix, that makes the problem more efficient to solve. The position of the collocation points with Gauss-Legendre scheme and Radau scheme are shown in table 4.3.

Table 4.3: Collocation points with Legendre method and Radau method.

Polynomial degree	Legendre Roots	Radau Roots
1	0.500000	1.000000
2	0.211325	0.333333
	0.788675	1.000000
3	0.112702	0.155051
	0.500000	0.644949
	0.887298	1.000000
4	0.069432	0.088588
	0.330009	0.409467
	0.669991	0.787659
	0.930568	1.000000

An example of the polynomial integration over an interval $[k, k+1]$, using a third order direct collocation with Radau collocation scheme is shown below in figure 4.11.

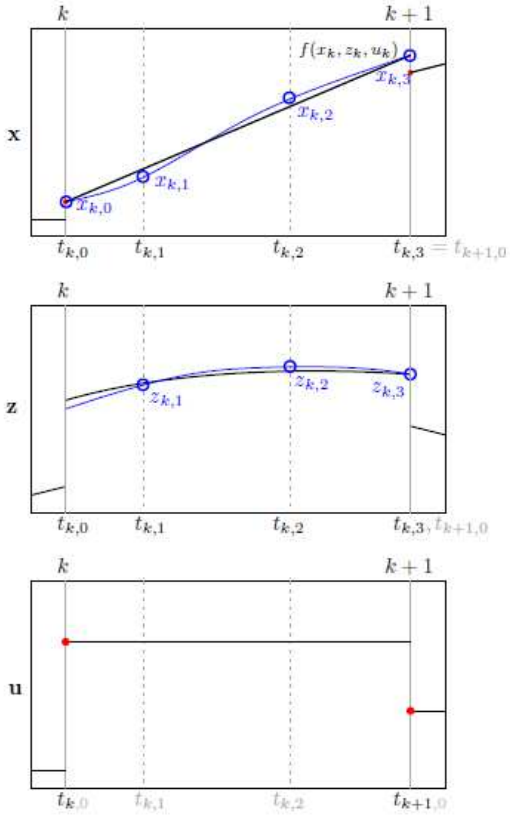

Figure 4.11: Schematic representation of third order direct collocation using Radau scheme, showing the polynomial approximation of the differential (x) and algebraic (z) states and the control input u for a sampling time $[k, k+1]$. Note the presence of a collocation point at $t_{k,0}$, which is used to ensure continuity in the differential state by enforcing the shooting gap constraints. The picture is taken from (Krishnamoorthy, B. Foss, *et al.*, 2016).

During the simulations with the centralized optimizer, a centralized NMPC (c-NMPC) is used. In order to solve the OCP problem, it is necessary to implement the process constraints. For the centralized NMPC, they are simply the values of w_{gl}^{max} and w_{pg}^{max}.

The idea at the basis of the case study is the presence of two different operators, that can control separately the two well clusters. Thus, it is reasonable to assume that also the control layer is decomposed. Hence, during the simulations with the distributed optimizer, a distributed NMPC (d-NMPC) is used (see also Farina *et al.*, 2016 for a comparison of d-NMPC schemes). Its framework and implementation are exactly the same as the centralized NMPC, the main difference is in the process constraints. While in c-NMPC the global process constraints are well determined, local constraints for d-NMPC are not defined. While in a simple PI controller the only input required is the CV's set point, for a MPC also constraints are required. Hence, the optimizer must compute feasible and

reasonable local constraints to make d-NMPC work. With primal decomposition the local constraints formulation is straightforward: the assumption of primal decomposition is exactly the allocation of resources, thus, the formulation of local constraints for the two subsystems. The two local constraints will be exactly the solutions of the master problem t_1^{gl}, t_1^{pg} for the first cluster and t_2^{gl}, t_2^{pg} for the second cluster.

In dual decomposition no allocation is performed, and no local constraints are formulated. A possible solution to deal with local constraints is to normalize the optimal solutions obtained rescpect to the global cosntraints themselves, as shown in equations (4.6.12) - (4.6.15)

$$t_1^{gl} = \frac{\sum_{i \in N1} w_{gl}^{opt}}{\sum_{i \in N} w_{gl}^{opt}}; \tag{4.6.12}$$

$$t_2^{gl} = \frac{\sum_{i \in N2} w_{gl}^{opt}}{\sum_{i \in N} w_{gl}^{opt}}; \tag{4.6.13}$$

$$t_1^{pg} = \frac{\sum_{i \in N1} w_{pg}^{opt}}{\sum_{i \in N} w_{pg}^{opt}}; \tag{4.6.14}$$

$$t_2^{pg} = \frac{\sum_{i \in N2} w_{pg}^{opt}}{\sum_{i \in N} w_{pg}^{opt}}; \tag{4.6.15}$$

In this way, the local constraints will be equal to the respective variable's optimal values whenever the constraint is active, while if the constraint is not active, the allocation of resources is proportional to the optimal solutions.

The formulation of local constraints is a delicate point, because they have not a real meaning, but just an expedient to coordinate the oil production. As described in chapter 5, the violation of local constraints does not mean necessarily the violation of global constraints, with the possibility of leading to suboptimal choices.

Lastly, two other constraints must be implemented in the MPC: the slack variable's constraint and the shooting gap constraint. The slack variable represents the quantity of gas flared during the w_{pg}^{max} constraint violation. The additional constraint is required in order to define how much gas can be flared, or how much the global or local constraint can be violated dynamically.

The shooting gap constraint force the continuity of the differential state, as shown in figure 4.11, imposing that:

$$x_{k,K} = x_{k+1,0} \tag{4.6.16}$$

Which means that the value of the differential state x, at sample time k in the last collocation point K (which is equal to three in our case) must be equal to the value of x at the next sample time $k+1$ at the first collocation point (0).

The MPC developed for this work is implemented in an external function that is called by the main code at each sample time. This function is reported in appendix A

Chapter 5

Results and discussion

The objective of this thesis is to investigate and analyze the performances of a distributed optimizer based on HRTO framework. The case study is described in chapter 3 while the development of the code and the optimization loop is presented in chapter 4. In this section, the results of the simulations are outlined and commented. The discussion follows a logical order, in which at first the disturbances characterizing the simulation are described, then the discussion focuses on the choice of control layer, on the comparison between centralized and ditributed optimizers and lastly on the comparison between the two decomposition methods: primal and dual decomposition.

5.1 Design of experiment

The simulation regards the case study presented in chapter 3. The simulation covers a total time of 24 hours, as to simulate a full day of production. Each second, the plant is simulated and from the measurements new values of the estimated GOR for each well are computed. The optimizers and the MPC works once each sample time, which is set equal to 300 seconds. The only disturbances that affect the systems regard the values of GOR. The Gas to Oil Ratio changes in time, as shown in figure 5.1.

The most common disturbance is a step change in GOR value, which is included in the intervals reported in table 3.1 and defined as follow:

$$GOR = \overline{GOR} \pm \sigma_{GOR}; \tag{5.1.1}$$

where $\overline{GOR}$ is the average value of GOR, while σ_{GOR} is its standard deviation. However, between 11 - 13 hours and 16 - 20 hours ramp disturbances occur, in which GORs slowly drift to other values. In this way it is possible to test both the cases: a sudden disturbance that significantly changes the optimal process conditions and a slower, less significant disturbance that continuously affects the plant.

The constraints too vary in time. They are considered as inputs for the optimizers, because they are not real unknown and unexpected disturbances, but they depends on

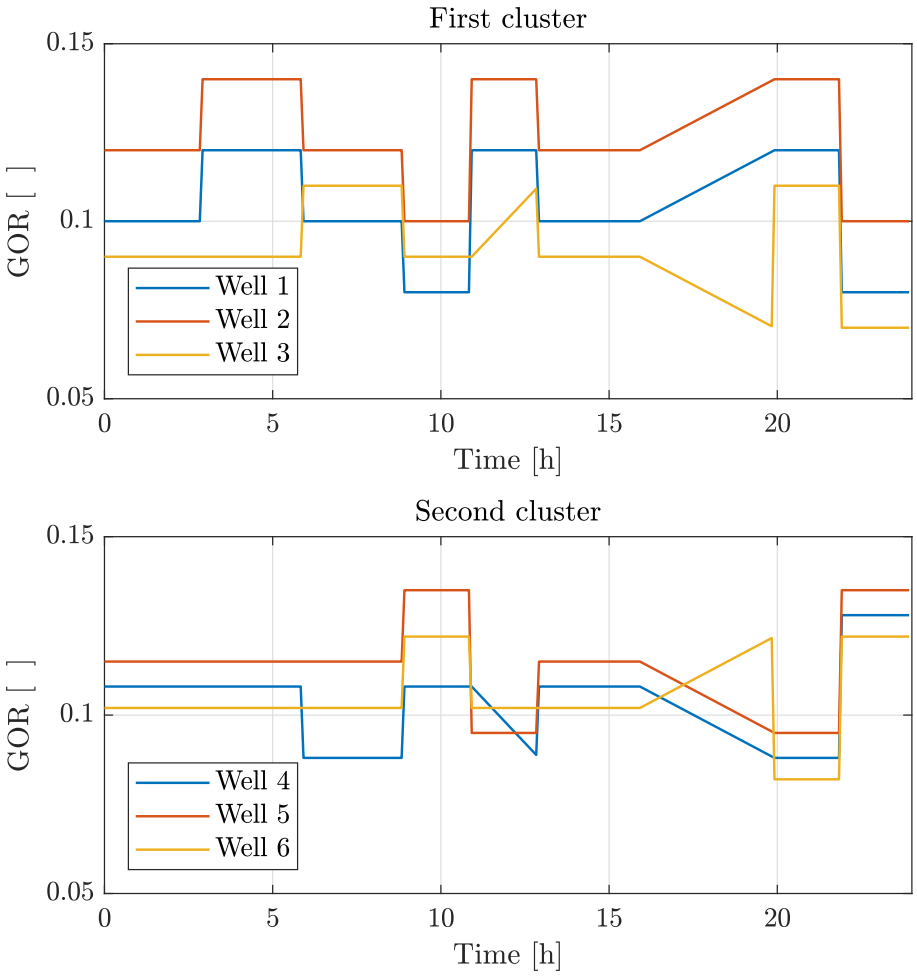

Figure 5.1: GOR values for each well during the simulation.

planned maintenance, strategies, or decisions regarding the upper levels of the hierarchy shown shown in figure 0.1.

The disturbances and variations in constraints have been chosen in order to test the behaviour of the optimization routine while facing different situations. In particular, it is important to verify if the optimizer is able to work both when one of the constraint is active or when the problem can be considered unconstrained. In order to check whether the constraints are active or not, it is possible to look at the values of the Lagrangian multipliers used for the distributed optimizer based on dual decomposition: they assume a non-null value whenever the respective constraint is active. Their values during all the simulation time are shown in figure 5.3.

From this figure, it is possible to observe that all the scenarios are considered in this simulation: in the intervals 0-3, 6-8, 16-21 hours the gas lift constraint is active, during the intervals 12-16, 21-24 hours the produced gas constraint is active, while during the rest of the simulation (3-6, 8-12 hours) the system is unconstrained. The different values of lambda represent a different severity of the constraint itself. The lagrangian multiplier at the solution

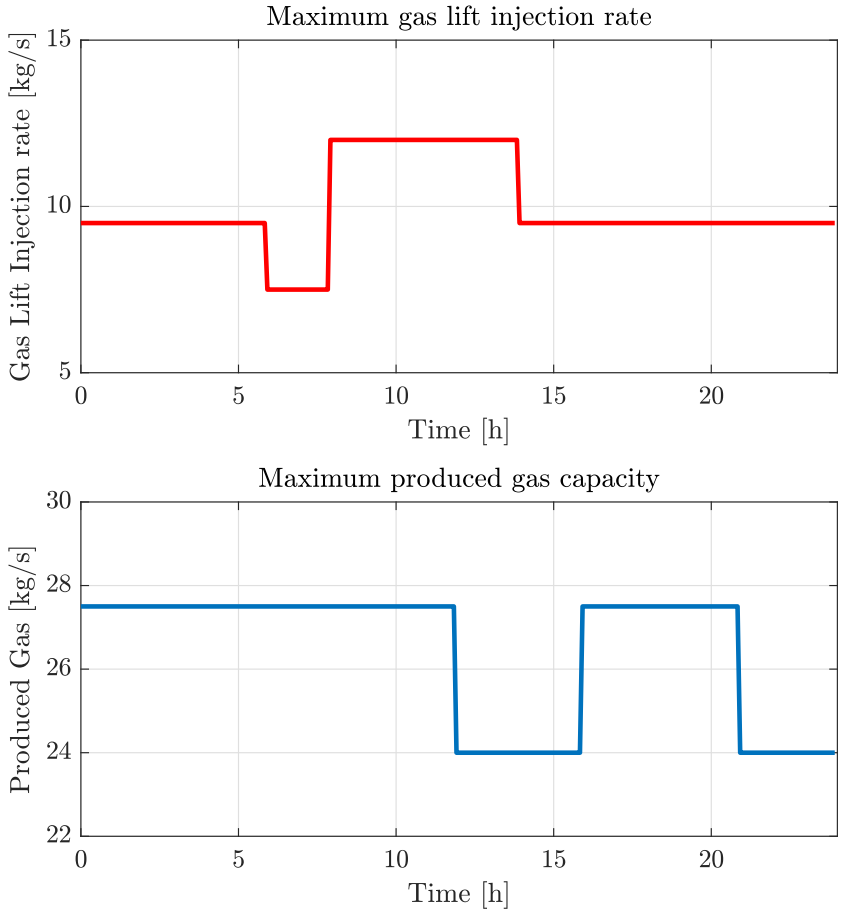

Figure 5.2: Variation of maximum gas lift injection rate and maximum gas produced capacity during the simulation.

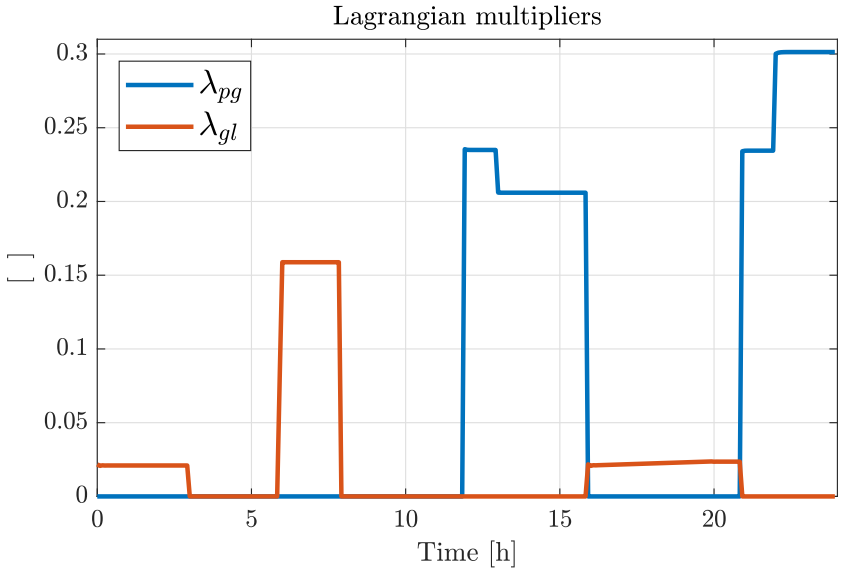

Figure 5.3: Lagrangian multipliers' values during simulation.

of the problem is equal to the rate of change in the maximal value of the objective function as the constraint is relaxed. This means that the higher the value of lambda, the further the system is from the uncinstrained optimal process conditions, thus the constraint itself can be considered more "severe".

It has been demostrated that the disturbances and the constraints introduced in the system permit to analyze the optimization routine in a wide range of situations. It is now possible to move to the next step, which is the choice of the control layer that must be implemented along the optimizers.

5.2 Choice of control layer

One of the first decisions that must be taken is how to effectively carry out the regulation and control of the system. In chapter 3 the development and tuning of two kind of controllers is illustrated: a classical Proportional Integral controller (PI controller) and a more advanced Model Predictive Controller (MPC). While the MPC directly control and manipulate the gas lift rate, for the PI controller two different controlled variables have been tested (produced gas and well head pressure), always manipulating the gas lift injection rate. The simulations are carried out according to the disturbances illustrated at the beginning of this chapter. Since the discussion is now focused on the control layer, the centralized optimizer's optimal set points are used. This is done in order to avoid any problems related to the use of a distributed optimizer and to be able to analyze the controllers' dynamics without any influences from to the optimizer's choice.

5.2.1 Controlled variable for the PI controller

In the PI controllers developed in this work, the controlled variable is not directly the gas lift injection rate, but the produced gas (w_{pg}) or the well head pressure (P_{wh}). This means that in these cases, the optimal set points computed by the optimizers will concerns one of this two variables and not w_{gl} as in MPC's case. The two CVs have been chosen in order to see if important differences can be noted in the use of a pressure or a production flowrate for what regards the controllers' performances. The most interesting plots for the analysis are the ones concerning the produced oil (figure 5.4, which is strictly connected to the profits of the production), and the graphs of the variables on which a constraint exists: the gas lift injection rate (figure 5.5) and the gas produced (figure 5.6).

It is possible to observe that both the controllers show a very similar behaviour. The stationary conditions are obviously the same, being the optimal process conditions depending on the same disturbances, while the transients are managed in a very similar way. Both the controllers do not show unstabilities related to oscillations or overshoots. It is possible to affirm that the two controllers are equivalent, their small differences depends exclusively on the tuning, in particular on the estimation of the process gain and characteristic time during the step change simulations, being the tuning rules adopted the same. The only reasons that could lead the choice towards one of the two CVs would be related to the measurement

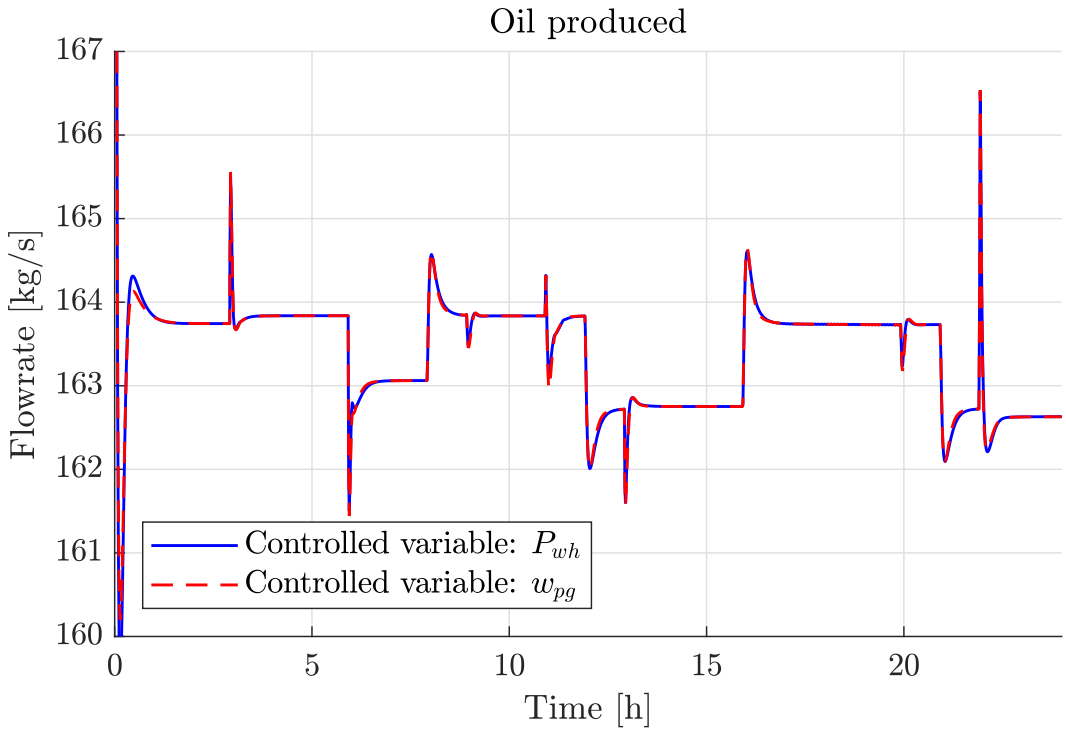

Figure 5.4: Comparison between PI controllers: produced oil.

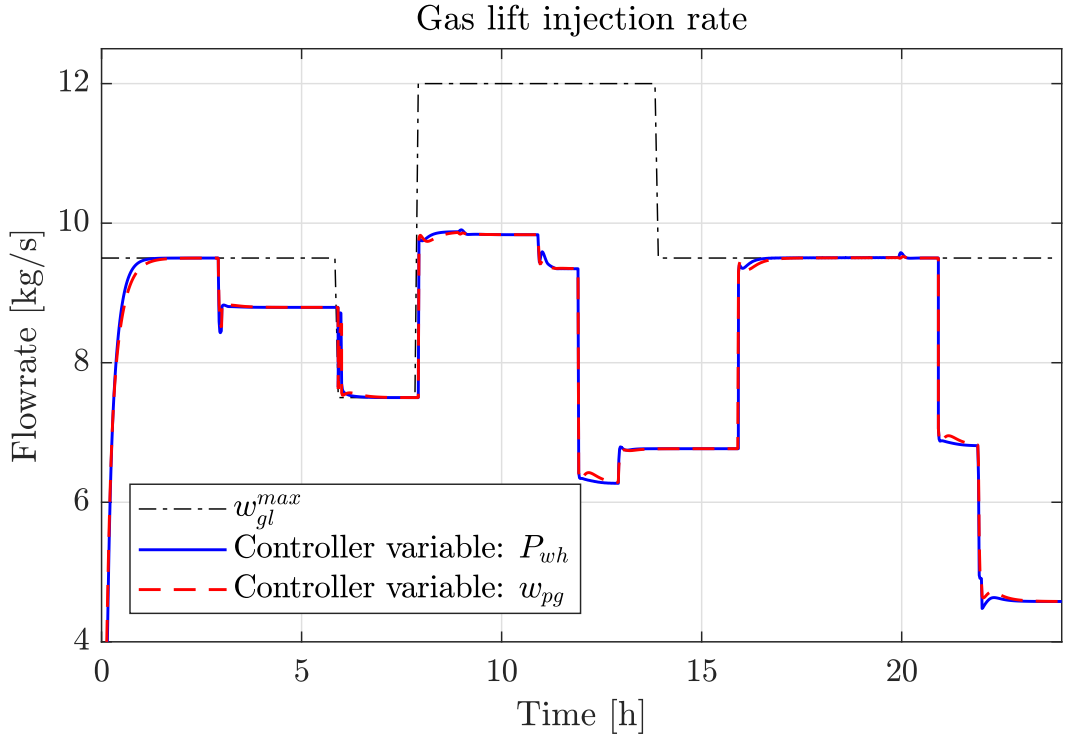

Figure 5.5: Comparison between PI controllers: gas lift injection rate.

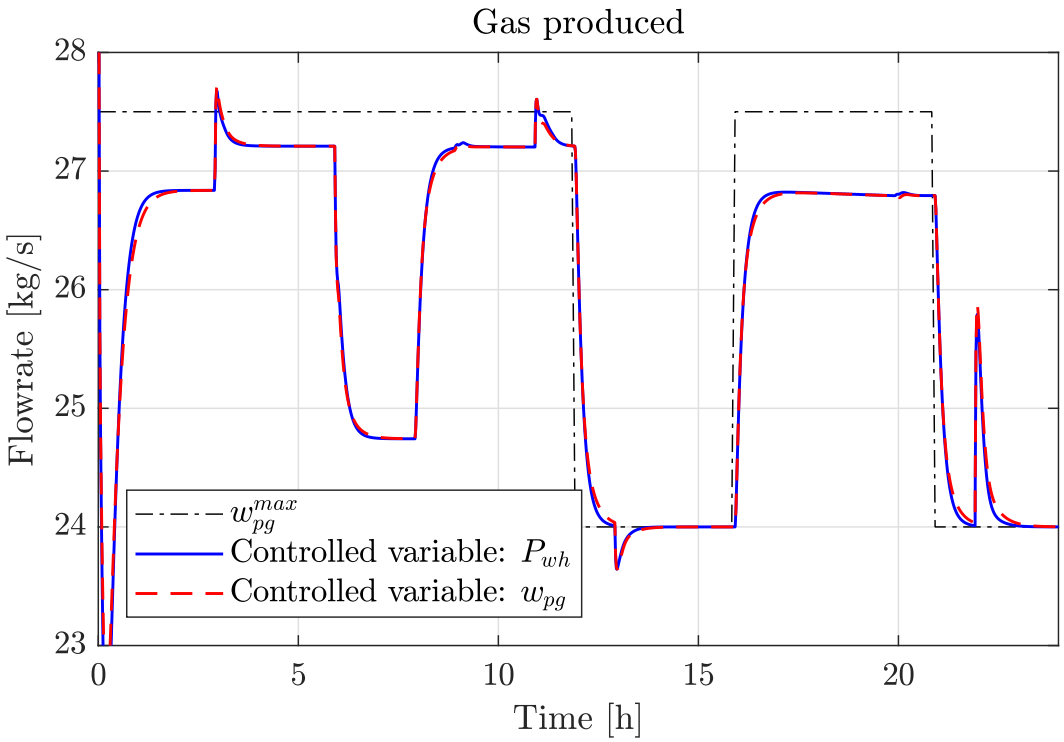

Figure 5.6: Comparison between PI controllers: produced gas.

devices or to the presence of dead times. While nothing can be said regarding the dynamics of valves and the presence of dead times, because a model of the regulatory system has not been implemented, from a measurement point of view using the well head pressure can be a better option. While for the measurement of pressure a simple and cheap pressure gauge can be adopted, having a precise and reliable measurement of the produced gas can be more challenging. The outflow of the production choke is a biphasic mixture containing both gas and liquid oil. A biphasic flow measurement device is complex and significantly more expensive than a pressure sensor, moreover it requires a more difficult and frequent maintenance. One possible solution would be to measure the gas flowrate after the liquid-gas separator, but because of the scenario considered in this case study, the separator could be very far from the common manifold, introducing in this way not negligible delays between measurements and control actions. Thus, although the behaviour of the two PI controller is almost the same, the well head pressure P_{wh} is chosen as CV for further comparisons.

5.2.2 PI controller and MPC

The MPC is a kind of advanced controller in which a system's model is implemented, in order to allow it to manage in a better way the manipulated variable. The PI controller's dynamics shown in figures 5.4 - 5.5 - 5.6 is pretty good, the controller is reactive but without introducing oscillations or overshoots. Thus, a comparison between the two controllers, PI and MPC, is now carried out, in order to see what kind of advantages an advanced controller

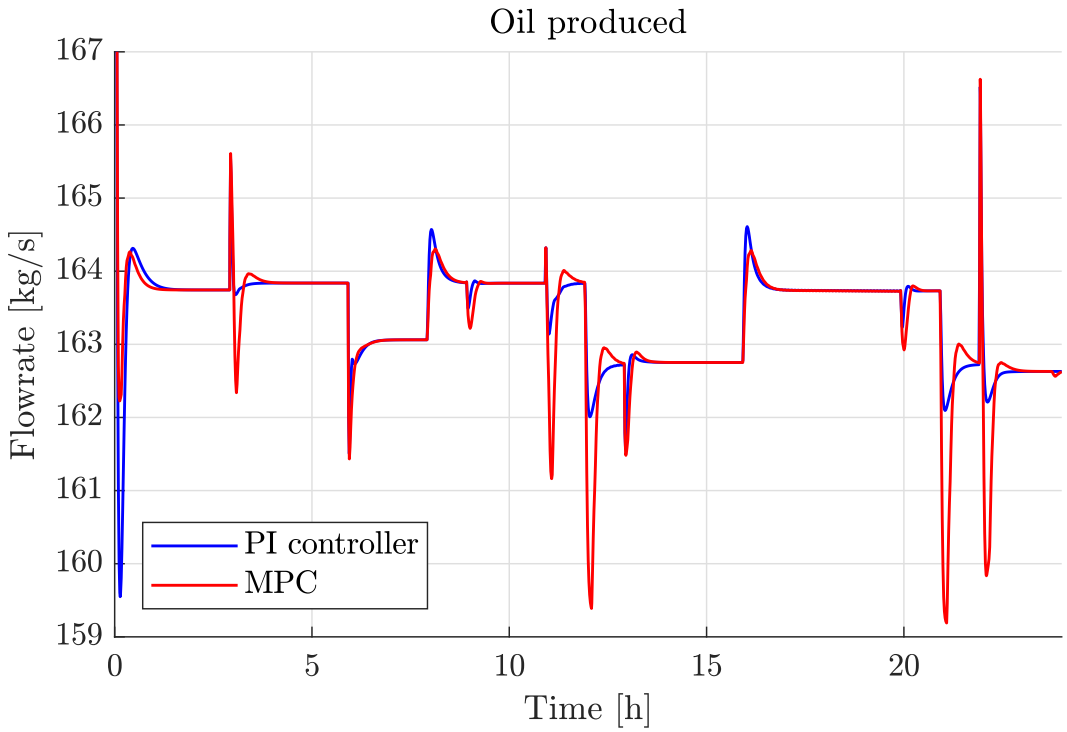

Figure 5.7: Comparison between PI and MPC controllers: produced oil.

can introduce. For both the controller, the set point is computed by a centralized optimizer. The plots resulting from the simulation are shown in figures 5.7 - 5.8 - 5.9.

Both the controllers shows acceptable dynamics. Their steady state response is exactly the same, as expected, but differences are present in the management of the transients (see 5.8 - 5.9). The most delicates points of the simulation are the ones in which the maximum produced gas capacity decreases, namely at 12 and 21 hours of simulation. In these moments, the sudden lowering of the constraints makes impossible to avoid the constraint violation. In these cases, the MPC is able to bring w_{pg} quickly below the maximum value, reducing significantly the gas lift injection rate for a short amount of time. The PI controller is slower, it is able to bring w_{pg} below the constraint only after 60 minutes on average. Although the PI controller shows a smoother behaviour and it is able to produce slightly more oil than MPC, the constraint violation for a so long time introduces a non negligible penalty in the plant's profits, as shown in figure 5.10 and in table 5.1.

In particular from the table it is possible to evaluate quantitative the advantages of MPC: while the total daily oil production is practically the same, the implementation of MPC

Table 5.1: Plant's income with PI controller and MPC controller.

Controller	Total oil produced [ton]	Total gas flared [ton]	Total income [10^7\$]
PI	14122	5.7134	1.3654
MPC	14117	2.8325	1.3710

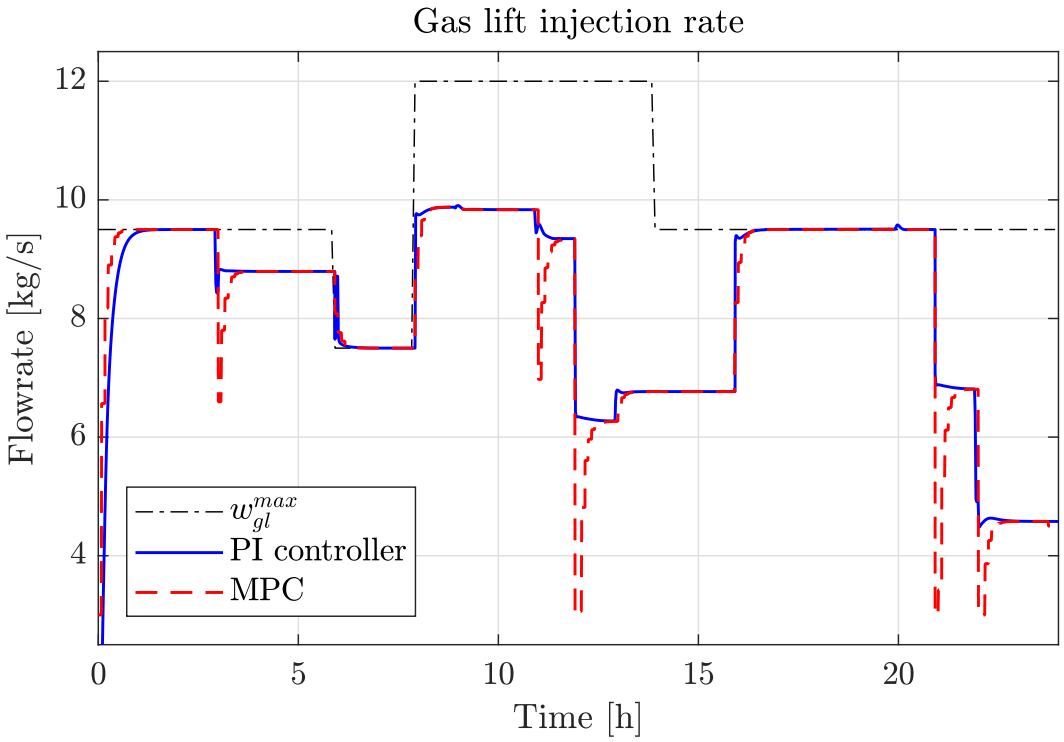

Figure 5.8: Comparison between PI and MPC controllers: gas lift injection rate.

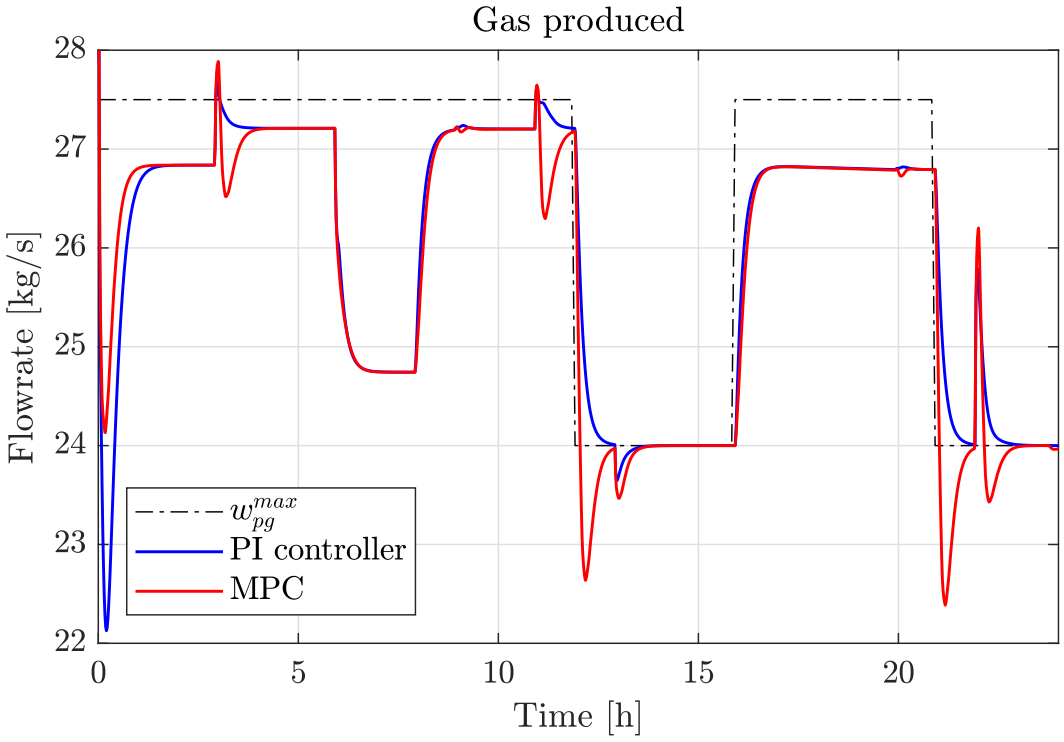

Figure 5.9: Comparison between PI and MPC controllers: produced gas.

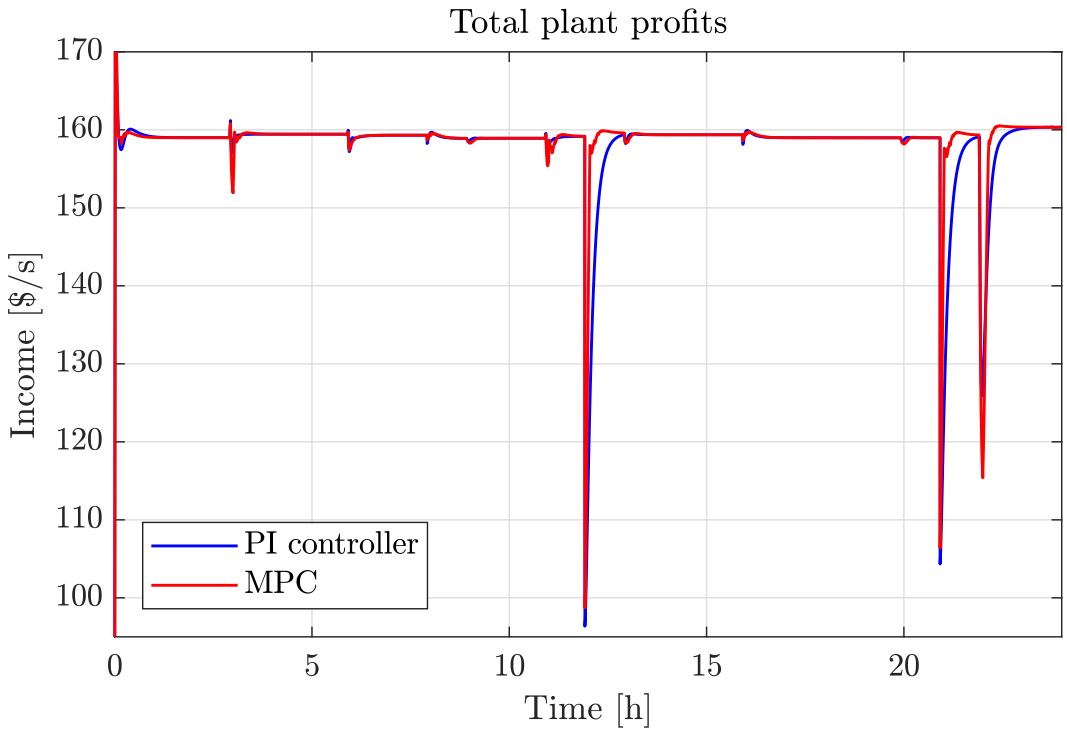

Figure 5.10: Comparison between PI and MPC controllers: plant's profits.

permits to halve the amount of gas flared, with significant improvements from a profits point of view, according to the process costs and carbon tax associated to the use of the flare. The possibility to reduce the periods in which constraints are violated and the more flexibility and promptness of the control actions, lead the choice of the control layer towards the MPC.

It is important to remind though, that the choice of the control layer must be taken by the single cluster's operator. Implementing a MPC can require a deeper analysis of the system and a greater effort during its development, thus it is not necessarly the first choice of both the operators. A scenario in which one of the operators opt for a MPC and the other for a PI controller is not impossible, although the MPC can significantly improve the productivity and the profitability of the production system.

5.3 Distributed optimizers' validation

Once all the decisions related to the design of the simulation and to the choice of the control layer have been presented, it is possible to move further and to deal with the optimizers. Among the three optimizers developed (centralized, distributed based on primal deocmposition, distributed based on dual decomposition), the centralized one is used as reference for the validation of the distributed optimizers. The centralized optimizer has been validated in the work (Krishnamoorthy, B. A. Foss, *et al.*, 2018). In this paper it has been compared with optimizers based on SRTO and DRTO, see the same article for a complete discussion about HRTO advantages respect to traditional RTO techniques.

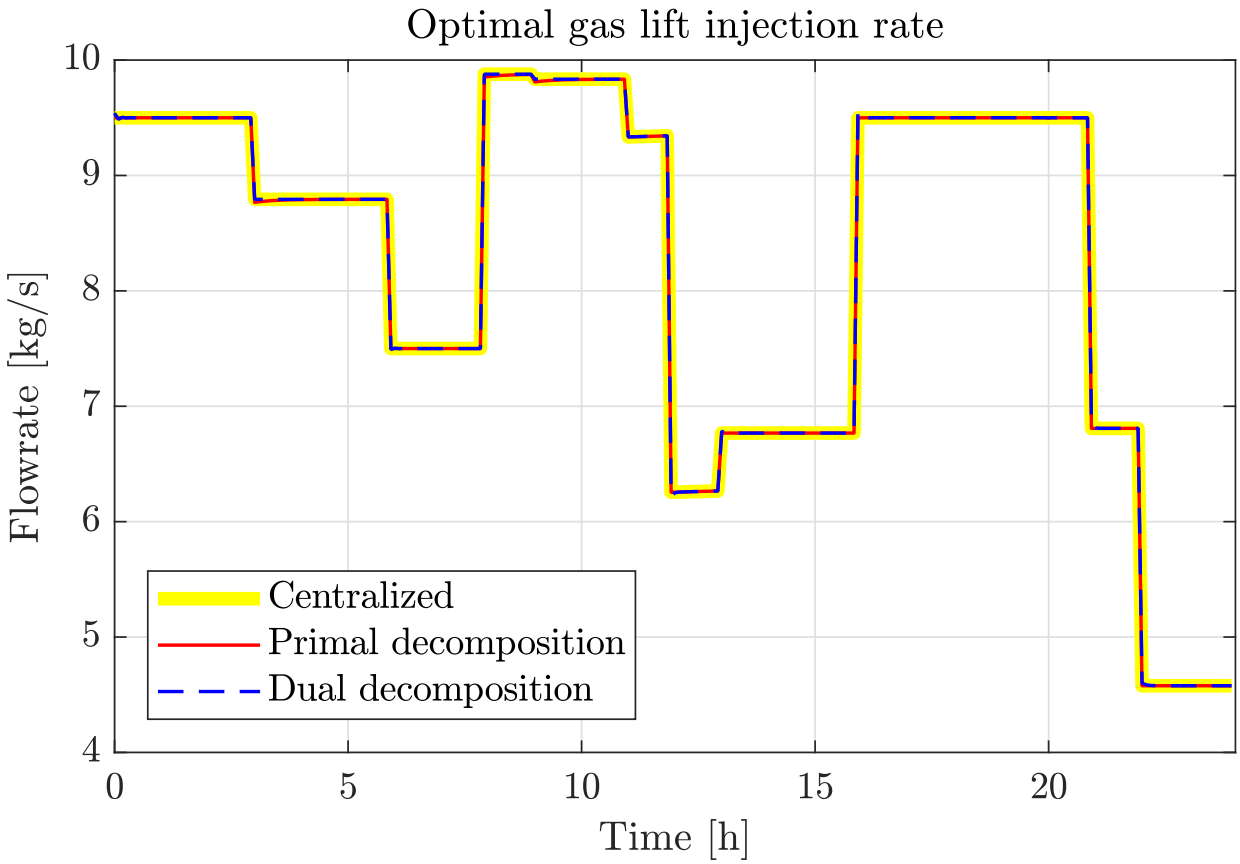

Figure 5.11: Optimal gas lift injection rate as calculated by the centralized and distributed optimizers.

The validation of the distributed optimizers is simply carried out comparing the optimal gas lift injection rate computed by the various optimizers in relation to the same disturbances. The results are presented in figure 5.11.

From this figure, it is evident that the distributed optimizers produce the same, exact solution of the optimization problem as the centralized optimizer. Only w_{gl} is considered in the validation, being it the manipulated variable and, thus, being these the effective set points used by the MPC. The same results would have been obtained if other variables were considered. The validation of optimizers has a great importance, because it guarantees that at every sample, the distributed optimizers converge to the real solution of the optimization problem.

5.4 Comparison between centralized and distributed optimizers

The fact that both centralized and distributed optimizers give the same optimal solution, means that if the two optimizers are coupled with the same control layer, the dynamic of the system would be identical, because the same controller with the same set points would actuate equal decisions. This would be the case if a PI controller was chosen to control

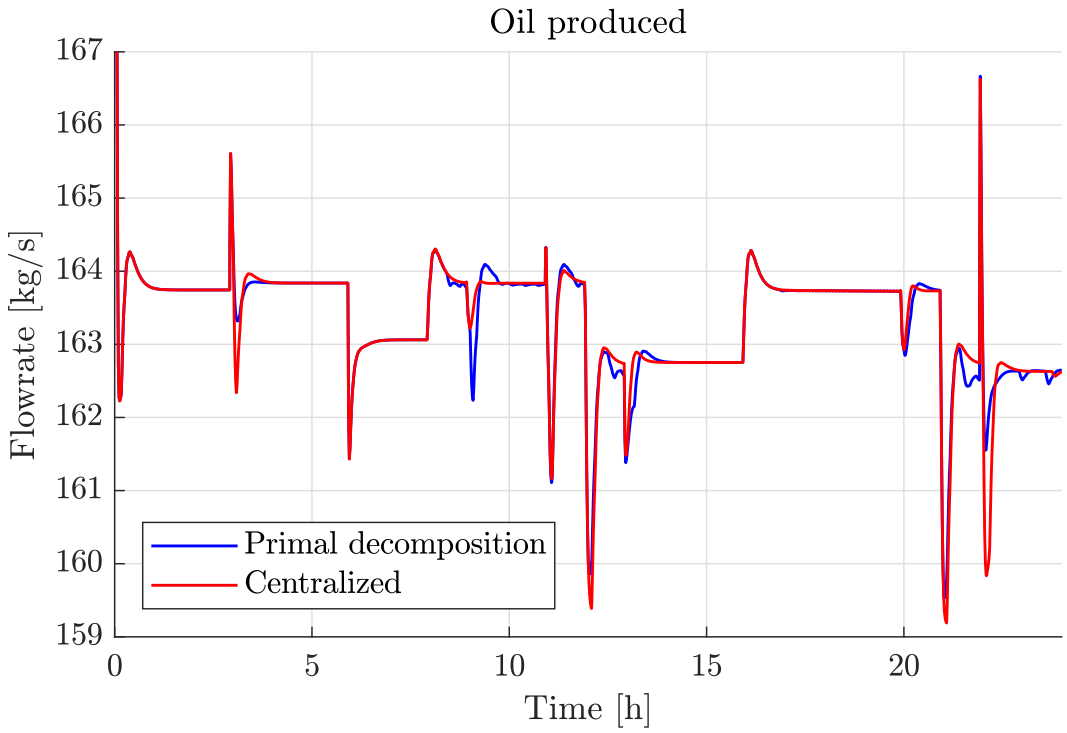

Figure 5.12: Comparison between centralized and distributed (based on primal decomposition) optimizers: produced oil.

and regulate the plant. However, the implementation of MPC is more complex and some discrepancies between the systems based on centralized or distributed optimizer can arise. The comparison between centralized optimizer and the distributed optimizer based on primal decomposition is illustrated in figures 5.12 - 5.13 - 5.14. In order to make the plots simpler and easier to understand, only the case based on primal decomposition is considered. However, as discusses in section 5.5, the differences between the dynamics of the primal decomposition based optimizer and dual decomposition based optimizer are almost negligible in this case study, thus, the comparison with the centralized optimizer can be carried out considering just one case.

Some interesting considerations can be made from these figures. The oil productivity is similar, some minor differences are present regarding mostly the istants in which a significant disturbance arises, as at 9 hours, while the steady state productivity is equal. Figures 5.13 - 5.14, that shows the trend of the two constrained variables w_{gl} and w_{pg}, are more interesting. The two systems deals in a very similar way both steady states and transients, but the centralized optimizer shows a better behaviour. In particular, the decomposed system have a strange response at the times 9 and 20 hours, where an unexpected reduction of w_{gl} occurs. Despite the fact that the constraint associated to w_{pg} is not violated, the distributed MPC choose, in both the cases, to cut the gas lift injection. This is a wrong behaviour, because it means producing less oil than the maximum allowed, which means a sub-optimal working conditions. This behaviour is very peculiar and can not be explained looking at the global

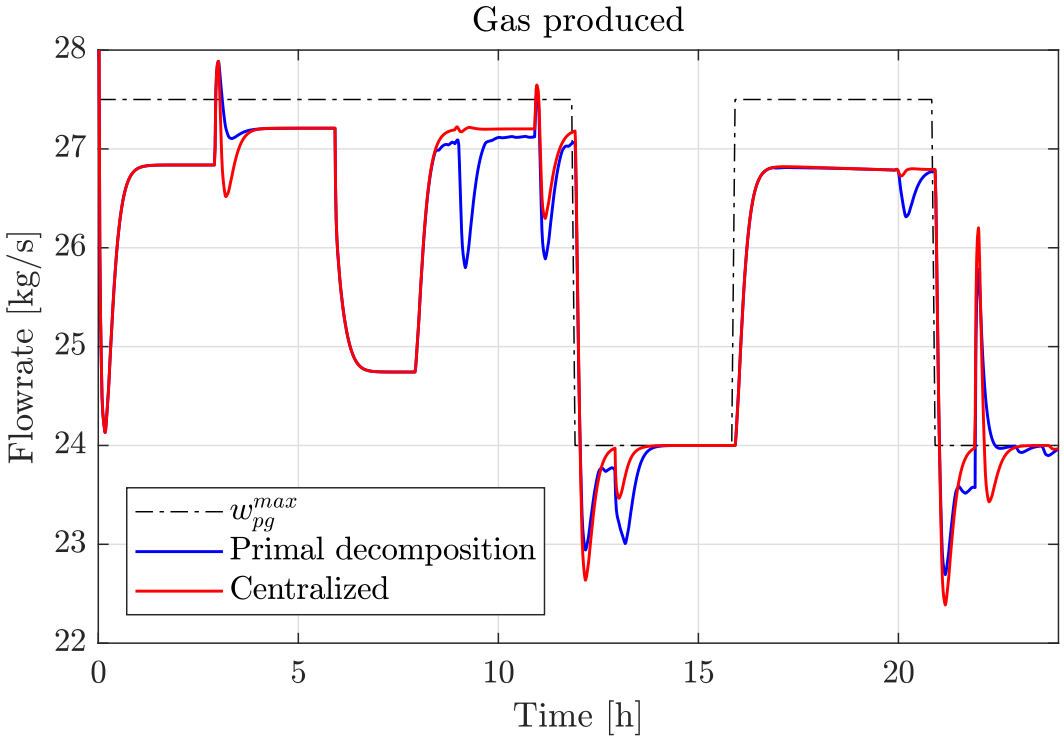

Figure 5.13: Comparison between centralized and distributed (based on primal decomposition) optimizers: gas lift injection rate.

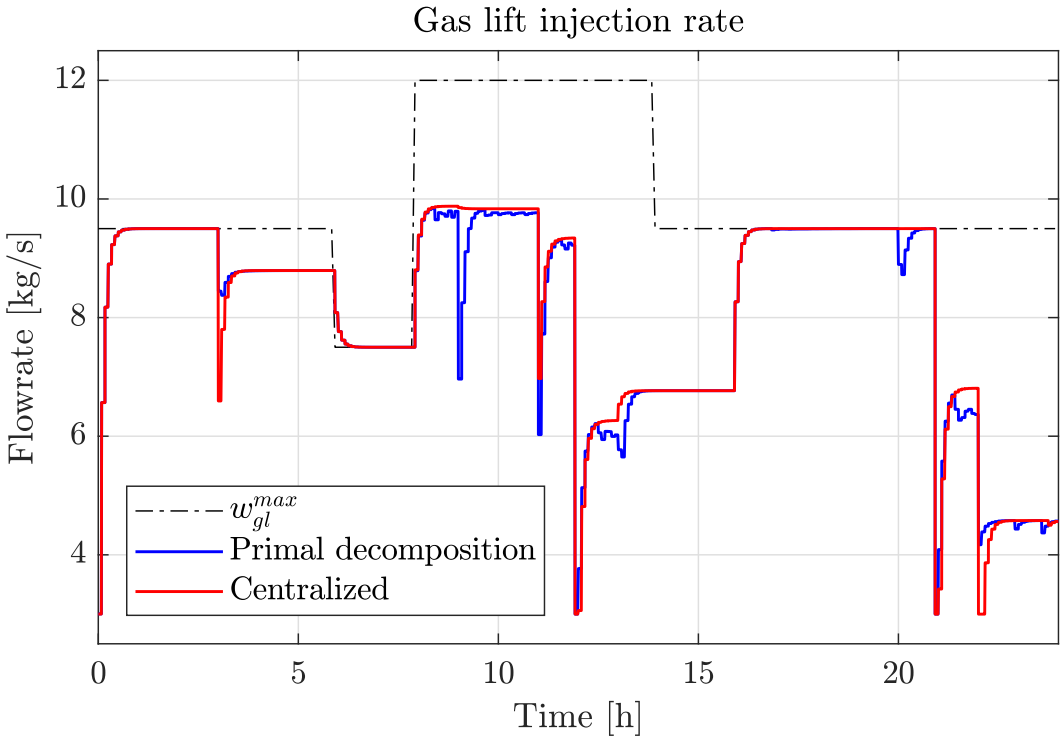

Figure 5.14: Comparison between centralized and distributed (based on primal decomposition) optimizers: produced gas.

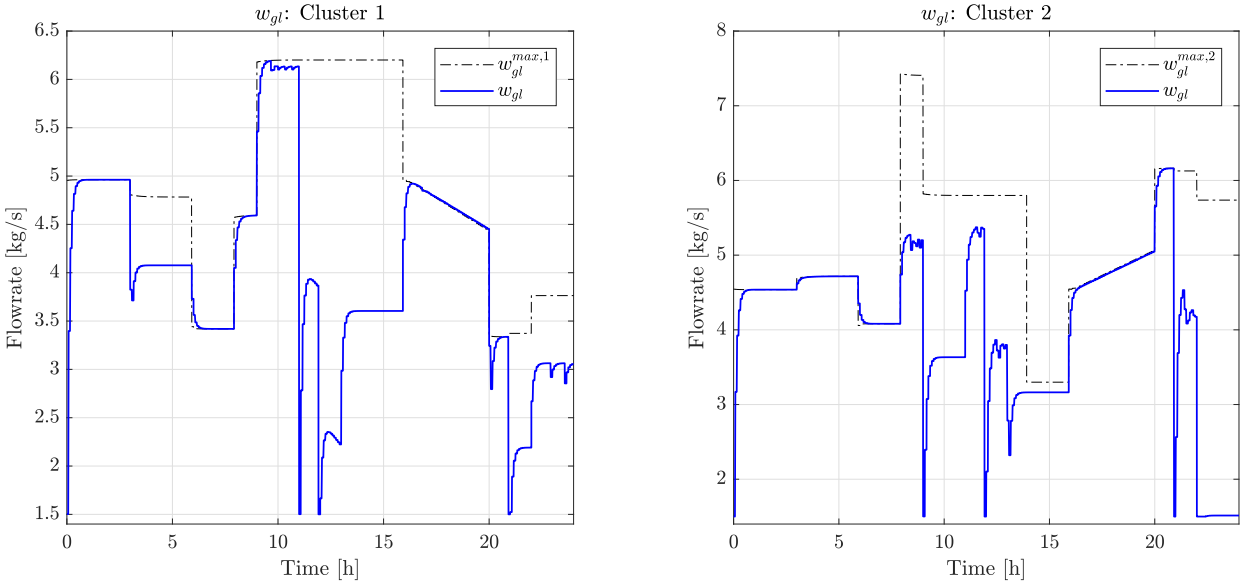

Figure 5.15: Total gas lift injection rate in the two clusters, optimizer based on primal decomposition.

productivities, it is thus necessary to focus on the single clusters. In figures 5.15 - 5.16, the values of w_{gl} and w_{pg} for the two clusters are plotted.

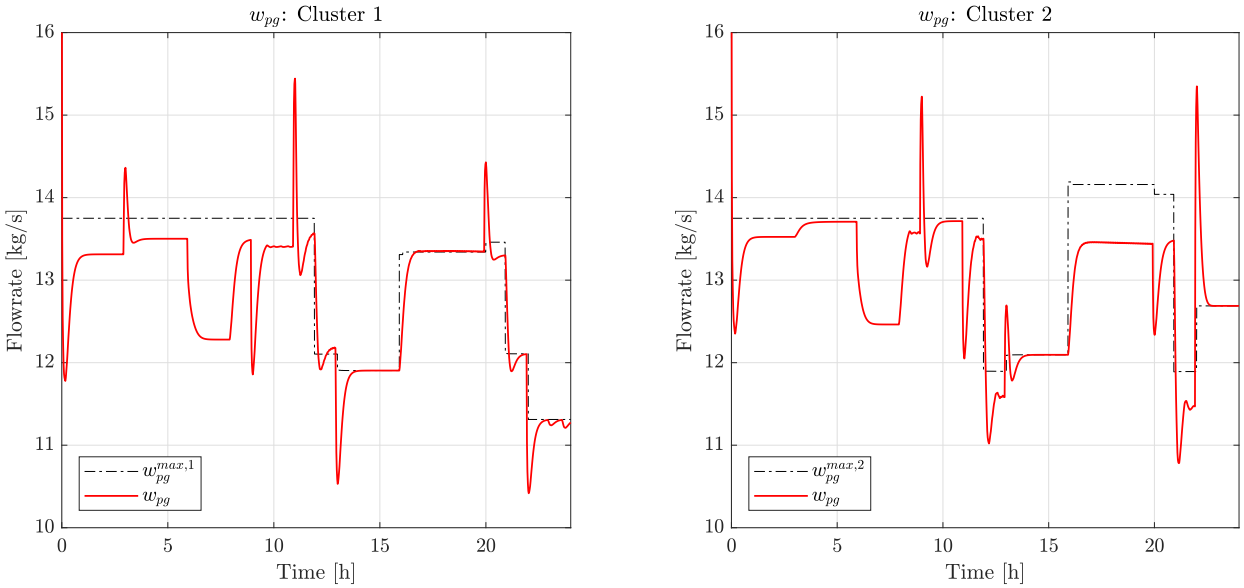

Figure 5.16: Total produced gas in the two clusters, optimizer based on primal decomposition.

If we focus our attention on the istant $t = 9\ hr$, it is possible to observe in figure 5.16 that in cluster 2 there is a peak in the production of gas, which is higher than the local constraint for that subsystem. The MPC reacts to this peak cutting the gas lift injection, as shown in the right picture of figure 5.15. However, the sudden increase in gas productivity in the second cluster is balanced by a reduction in gas production in the first cluster. In this way, although the local constraint imposed by the optimizer has been overcome in the

second cluster, the global constraint associated to w_{pg}^{max} is respected. This is the reason why, although the global constraint is respected, a sub optimal decision is taken by the distributed MPC. At 20 hours the same problem occurs, however the peak production occurs in the first cluster and the local constraint's violation is less pronounced.

These productivity peaks introduce an important problem in the usage of distributed optimizers: whenever a MPC, or a controller able to deal with constraints' violations, is implemented in the system, the optimizer must be able to compute, along with the optimal set points, local constraints. These local constraints are needed in order to allow the MPC or the advanced controller to take an action whenever one of the subsystems is, for example, consuming too much resources. However these local constraints are not real constraints, they have not a concrete and effective meaning. They are just an expedient to coordinate the global production. The computation and decision of local constraints is a delicate topic, because they should be defined in such a way that the global constraints are always satisfied, but without letting a local violation compromise the optimal process conditions.

In primal decomposition, it is the master coordinator itself that generates local constraints (the allocated resources) for the subsystems. For what concerns dual decomposition, no allocation or local constraints exist. In this work, the local constraints have been formulated normalizing the optimal value for the constrained variable respect to the global constraint itself. See chapter 4 for a deeper discussion about how local constraints are dealt by primal and dual decomposition in this work.

The problem related with the formulation of local constraints could be easily avoided adopting simpler control system, as the PI controlled described before. However, as shown in table 5.2, while a lower production of oil is almost negligible from a profits point of view, a longer dynamic constraint's violation is significant. Thus, the advantages of a decomposed MPC (d-MPC) are still consistent respect to a PI controller.

Table 5.2: Plant's income with PI controller and MPC controller.

Controller	Total oil produced [ton]	Total gas flared [ton]	Total income [10^7\$]
PI	14122	5.7134	1.3654
c-MPC	14117	2.8325	1.3710
d-MPC	14117	2.9615	1.3709

The differences in the behaviour of a centralized optimal control system and a distributed one are evident in the figures 5.13 - 5.14, however, the profits reported in table 5.2 are slightly affected. In this case study the suboptimal working periods introduced by the distributed optimizer is negligible, but the problem still exists, and it could acquire more importance in other kind of systems.

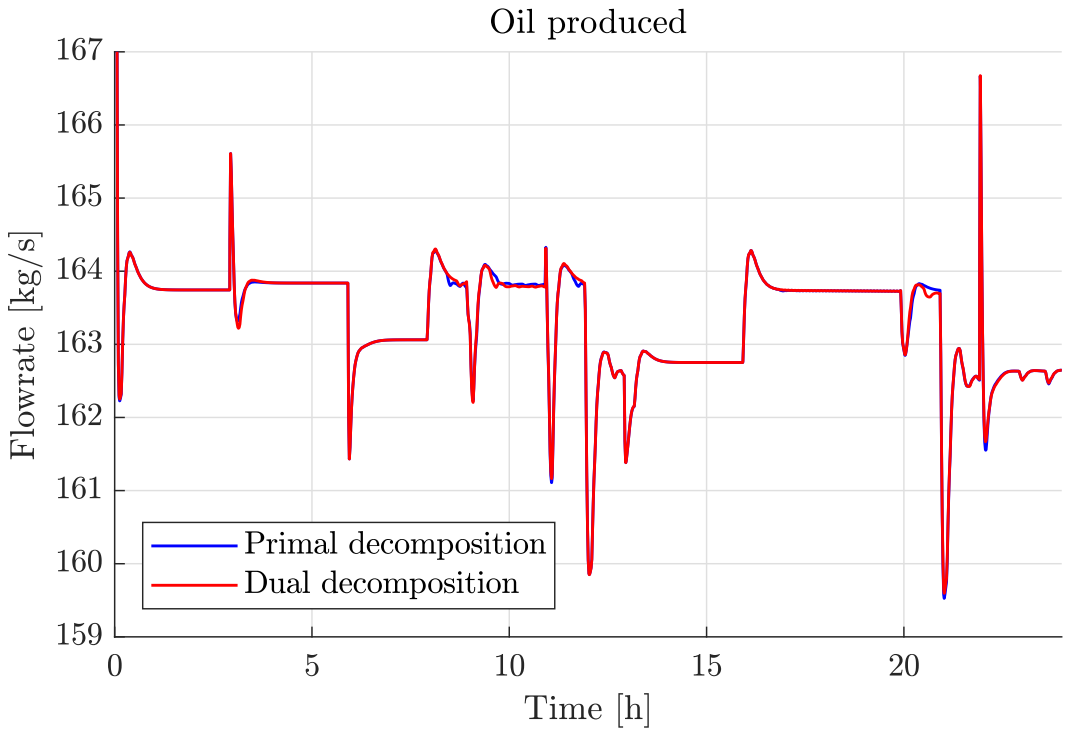

Figure 5.17: Comparison between distributed optimizers: produced oil.

5.5 Comparison between primal and dual decomposition

The last comparison that is carried out regards the two decomposed optimizer: the one based on primal decomposition (PD) and the one based on dual decomposition (DD). The respective trends of produced oil, gas lift injection rate and produced gas are shown in figures 5.17 - 5.18 - 5.19.

It is evident that the behaviour of the distributed optimizers is similar, very small differences occurs that can be considered negligible. In order to have an overview of the system dynamics inside the subsystems, it is possible to focus on the variables of the single clusters. This is shown in figures 5.20 - 5.21, in which the dashed lines are local constraints, the blue lines refer to the simulation with primal decomposition (PD) and the red lines to simulation with dual decomposition (DD).

Again, the variables' trends are practically identical. Two small differences between PD and DD are present in the first cluster at 10 hours and 20 hours. In particular some considerations can be made about the mismatch at 20 hours. In figure 5.20, in the first cluster at 20 hours, the primal decomposition based optimizer reduces w_{gl} more respect to the dual decomposition based optimizer. The reason of this small, different choice can be easily understood looking at the same point in figure 5.21. In both the cases, due to disturbances related to GOR, a sudden increase in gas production occurs. However, the two local constraints for PD (red dashed line) and DD (blue dashed line) are not exactly the same. Thus, the two MPC react differently to this constraint violation, in particular, in

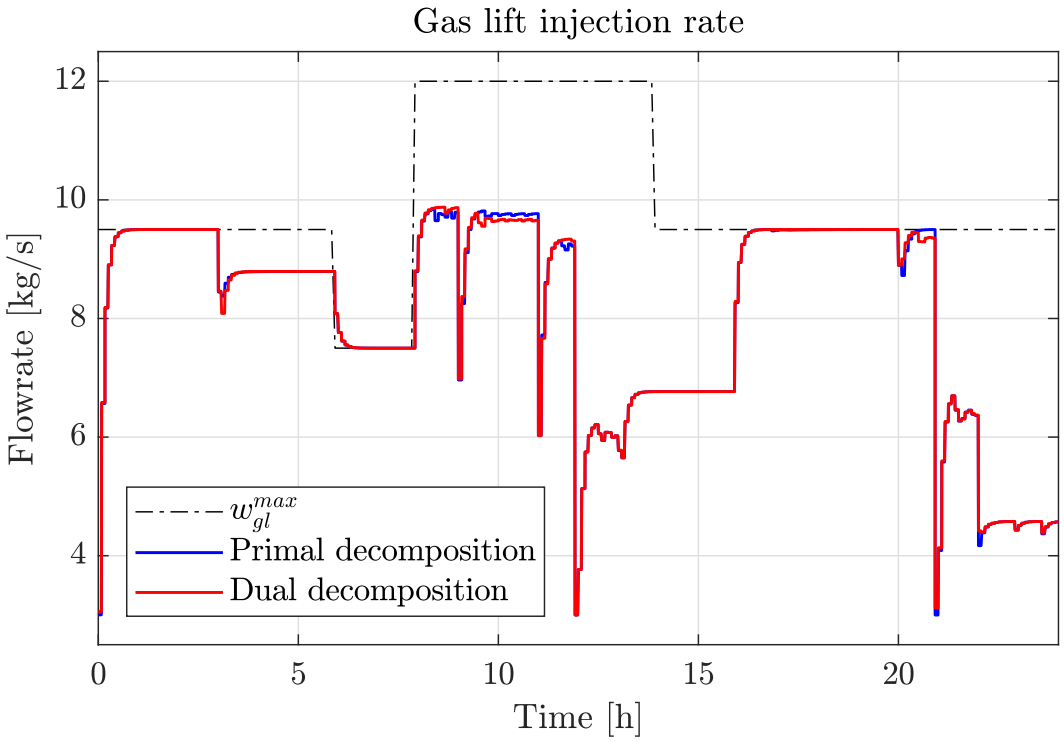

Figure 5.18: Comparison between distributed optimizers: gas lift injection rate.

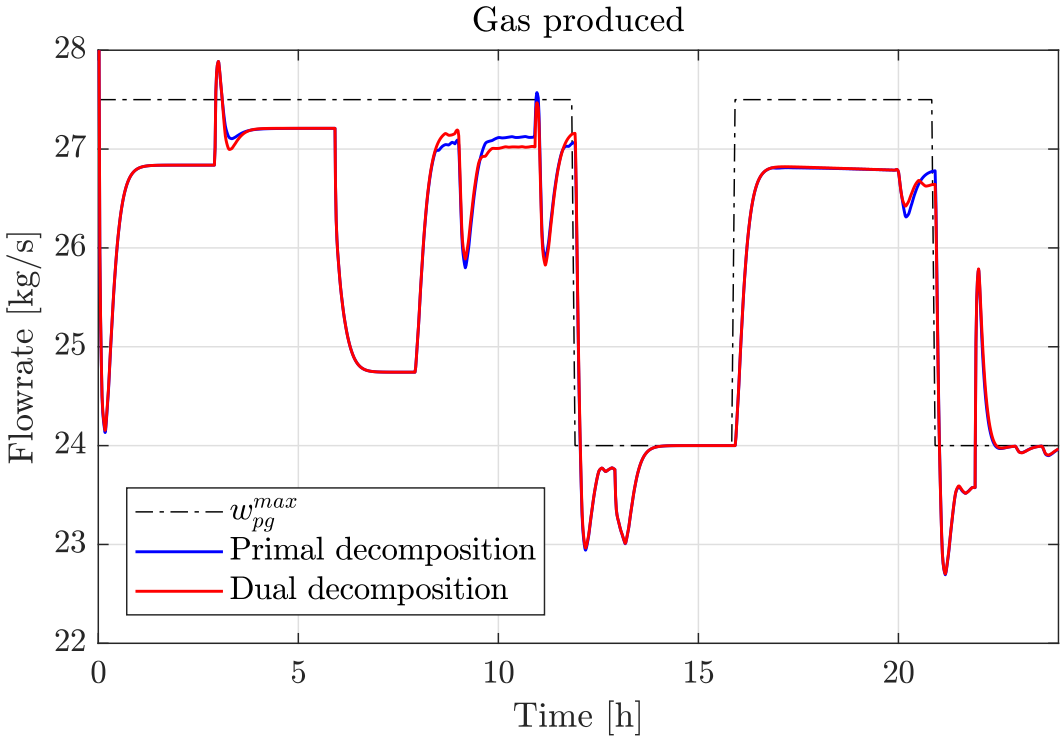

Figure 5.19: Comparison between distributed optimizers: produced gas.

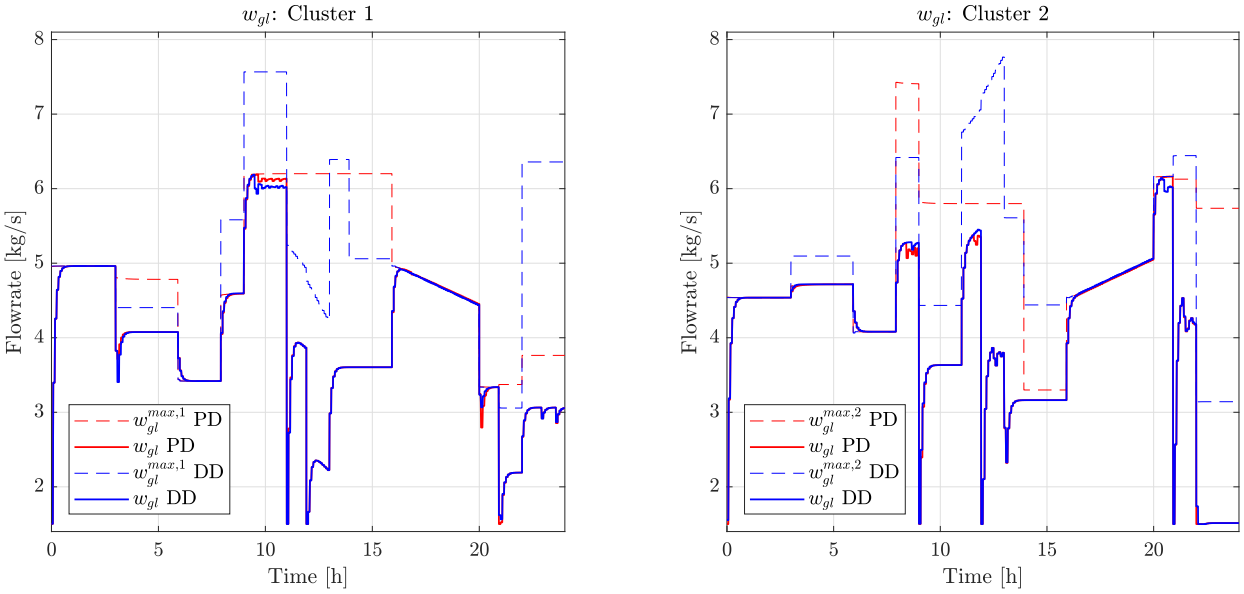

Figure 5.20: Total gas lift injection rate in the two clusters, distributed optimizers.

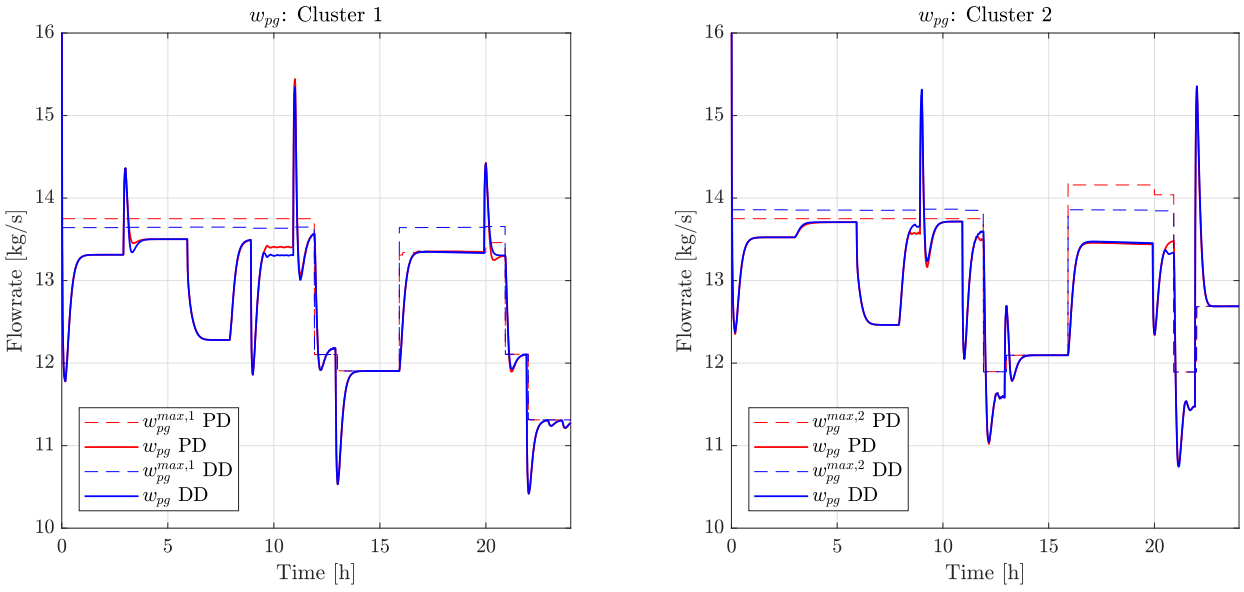

Figure 5.21: Total porduced gas in the two clusters, distributed optimizers.

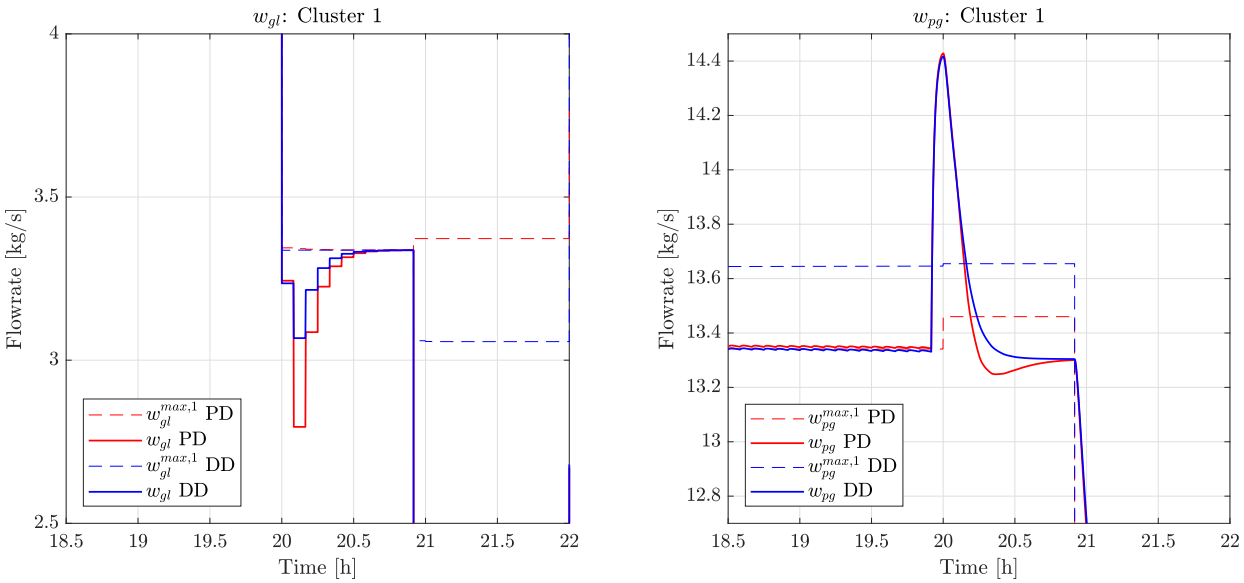

Figure 5.22: Particulars of figures 5.20 - 5.21, first cluster, $t = 20\ h$.

the primal decomposition case the local constraint is lower, making the constraint violation more significant and requiring a more decise action from the controller. A particular of this event is shown in figure 5.22.

This mismatch is almost insignificant in the case study considered, but it could become relevant in other kind of applications. The problem of an optimal choice of local constraints persist. From the simulations carried out in this work it is impossible to define which one of the two local constraints, the PD-based or the DD-based is better. One possibility could have been to apply the same approach used in dual decomposition for the definition of local constraints to primal decomposition. Instead of using the allocated resources, it could have been possible to formulate the local constraints as the optimal variables' values, normalized respect to the global constraints. However, to perform this kind of local constraint formulation, as shown in equations 4.6.12 - 4.6.15, it is necessary to collect informations about the optimal solutions from both the clusters. While these informations must be shared in dual decomposition to solve the master problem, in primal decomposition they are not required. Indeed, in order to solve the master problem, dual decomposition requires the solutions of all the optimization subproblems, while primal decomposition needs only a subgradient of those solutions. In a scenario as the one considered in this case study, the sharing of the clusters' optimal conditions could allow the two operators to have many informations about the other's productivity, profits and even strategies. Sharing only some subgradients would make impossible for the other operator to estimate precisely the conditions and profits of the other cluster's production. Primal decomposition is able to ensure a higher level of privacy between the two subsystems, thus, it is preferable to maintain this level of privacy and use, as local constraints, the resources allocated by the master coordinator.

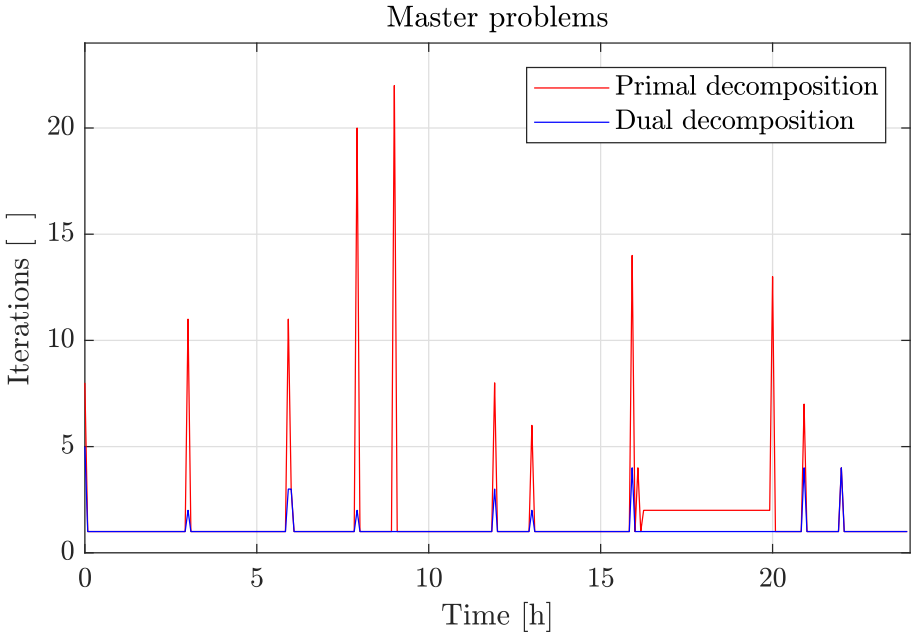

Figure 5.23: Iterations for master problem.

5.5.1 Convergence rate

One important aspect of the distributed optimizers is the solving of the master problem. Primal and dual decomposition follows two different approaches to achieve global convergence, thus, it is expected to find different convergence rates. The convergence rate depends deeply on the choice of the update parameter α. In this work, the value of α able to minimize the total number of iterations have been adopted, however this method can be used only if the disturbances are predictable or well known. When unknown disturbances or uncertainities are present, it is preferable to use a lower value of α, in order to ensure stability, which must be the first priority in the tuning of the optimizers.

In figure 5.23 the number of iterations required by the master problem in the two cases is represented.

For most of the samples, the master problems in both the decomposition methods requires just one iteration to achieve convergence. This is due to the "warm start" described in chapter 4: the solution of the optimization problem at the k time step is used as first guess for the next time step $k+1$. In this way, whenever no new disturbances occur, the optimal solution is equal to the one computed at the previous step, which is exactly the first guess inserted in the optimizer. However, when a step change disturbance occurs, the central coordinator require more than one iteration to solve the master problem, this is the reason for the presence of spikes in graph 5.23. In this cases, primal decomposition requires more iterations to solve the master problem. Moreover, between $t = 16\ hr$ and $t = 20\ hr$, when the ramp disturbance occurs, primal decomposition needs always 2 iterations to solve the master problem, while for dual decomposition one is sufficient. In table 5.3 the average and maximum number of iterations for both the decomposition methods are reported.

Table 5.3: Iterations for master problem.

Decomposition method	Average number of iterations	Maximum number of iterations
Primal Decomposition	1.5660	22
Dual Decomposition	1.0764	5

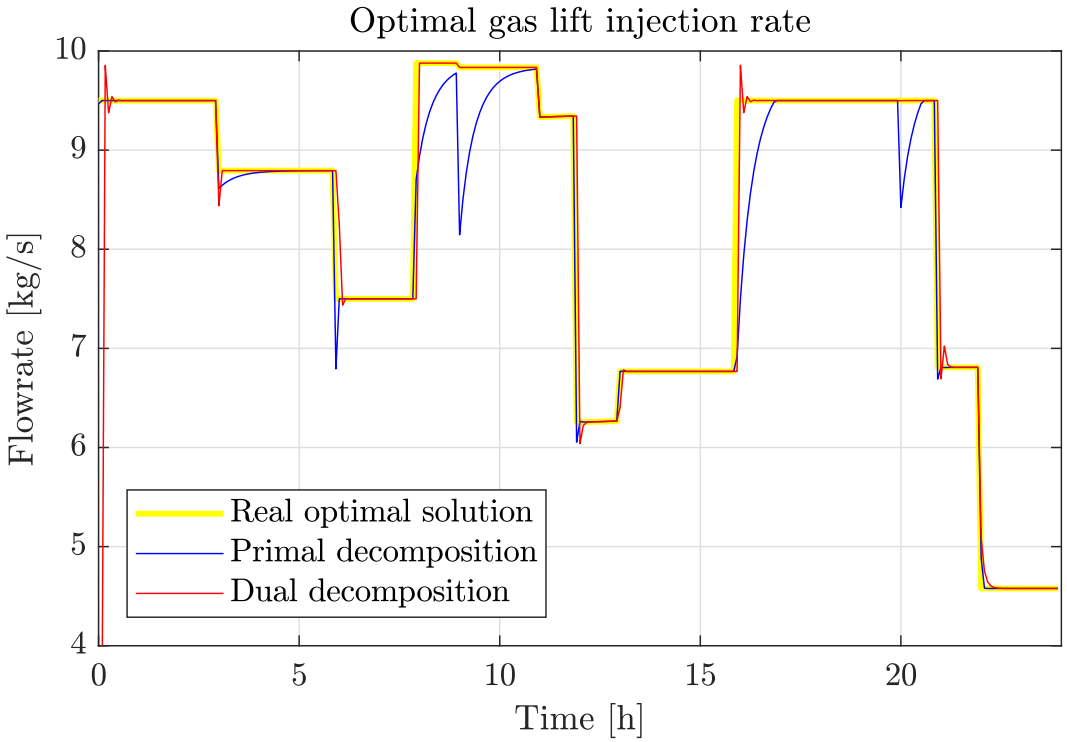

Figure 5.24: Optimal gas lift injection rate. The maximum number of iterations to solve the master problem is set equal to one.

Primal decomposition requires, on average, 50% more iterations than dual decomposition. Great differences are present in the maximum number of iterations, which is 22 for primal decomposition and only 5 for dual decomposition. It is evident that dual decomposition requires a significant lower amount of iterations to solve the master problem. The reduced number of maximum iterations can be relevant whenever the system is so complex that it is not easy to find a global optimum during one sample time. In this cases, the requirement of a low number of iterations can lead the choice between the two decomposition techniques towards the fastest one. This case study is too simple to present this problem, however it is possible to simulate it imposing in the distributed optimizers a maximum number of iterations equal to 1 to solve the master problem. In figure 5.24 one graph from this simulation is presented: the optimal injection rate computed by the distributed optimizers is compared to the same optimal value computed during normal simulations, with no limits on master problem's iterations.

It can be noted that the PD-based optimizer requires some samples to converge to the

real optimal solution. The system would work suboptimally during all these transients to find the real optimal process conditions, that can last even one hour (between $t = 7hr$ and $t = 8hr$). However, it must be noted that these transients are associated to the finding of the global optimum, not the local one. The production would be suboptimal from a global point of view, but the subsystems would be working optimally with that temporary allocation of resources. As expected, dual decomposition is faster and it can reach the global optimum in just a few samples. Moreover, looking at figure 5.24 it is possible to see the different behaviour of the two master problems: while in primal decomposition the solution gradually converge to the optimal one, in dual decomposition the first step solution is already very close to the optimal one. Their behaviours resemble somehow the behaviour of a second order overdamped system (in primal decomposition) or a second order underdamped system (dual decomposition). The peaks that can be seen in figure 5.24 at $t = 9hr$ and $t = 20hr$ in the primal decomposition line are due to the violation of local constraints and the consequent change in the resources allocation. This problem is not found in dual decomposition because the allocation of resources is not necessary for the solution of the master problem. In the other figures (5.25 - 5.26 - 5.27) the values of w_{pg}, w_{gl} and w_{po} for the these particular simulations are reported. While no particular differences can be found in dual decomposition's trends, primal decomposition shows slower dynamics, due to the longer time needed to converge to find the globally optimal process conditions. This delay is significant for the oil production: in figure 5.27 it is evident that dual decomposition permits a larger oil production.

5.5.2 Profits

As shown previously in this chapter, only minor differences are present between the two distributed optimizers. The profits of the plant in the two cases are presented in figure 5.28 and in table 5.4.

Table 5.4: Plant's income with distributed optimizers.

Decomposition method	Total oil produced [ton]	Total gas flared [ton]	Total income [10^7\$]
No (Centralized)	14117	2.8325	1.3710
Primal Decomposition	14117	2.9615	1.3709
Dual Decomposition	14117	3.0319	1.3708

While the oil production between PD and DD is exactly the same, small differences can be found for what concerns the flared gas, with consequences on the plant profits. However, both the distributed optimizers are able to manage wisely the constraints violations, which is the main factor that permits to compare the total incomes in this case study. In other kind of systems, where the costs related to constraint's violations are negligible if compared to the main source of income, the presence of suboptimal production periods, as the ones shown in figures 5.13 - 5.14, can make more relevant the differences between centralized and ditributed optimizers.

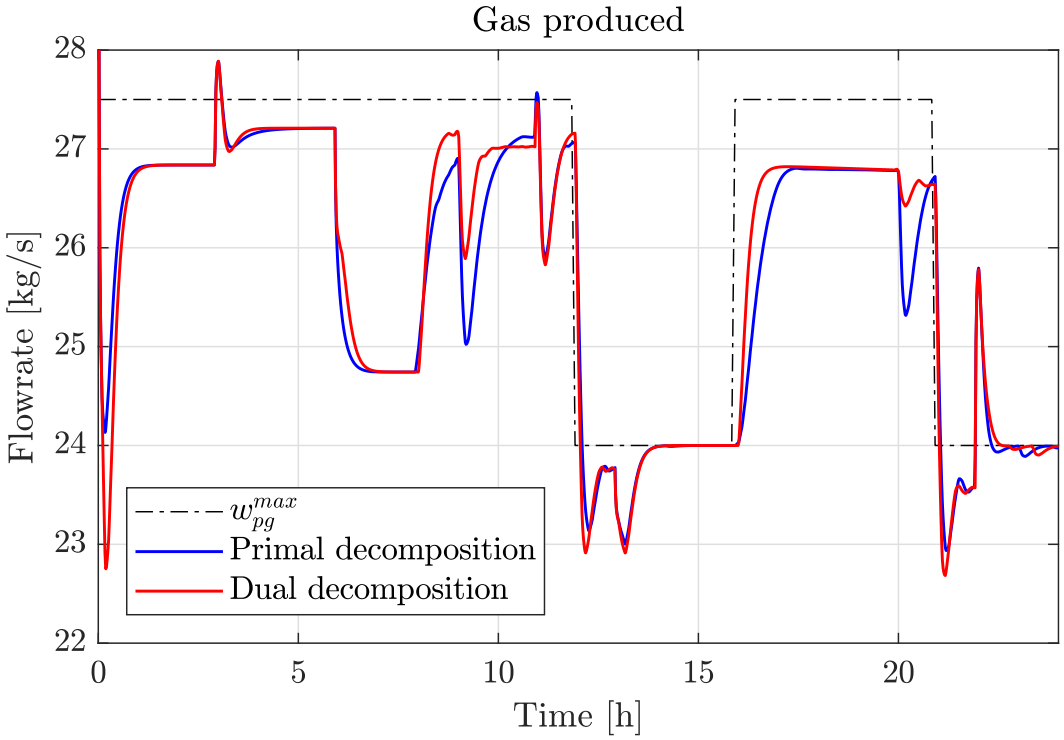

Figure 5.25: Produced gas in distributed optimizers. The maximum number of iterations to solve the master problem is set equal to one.

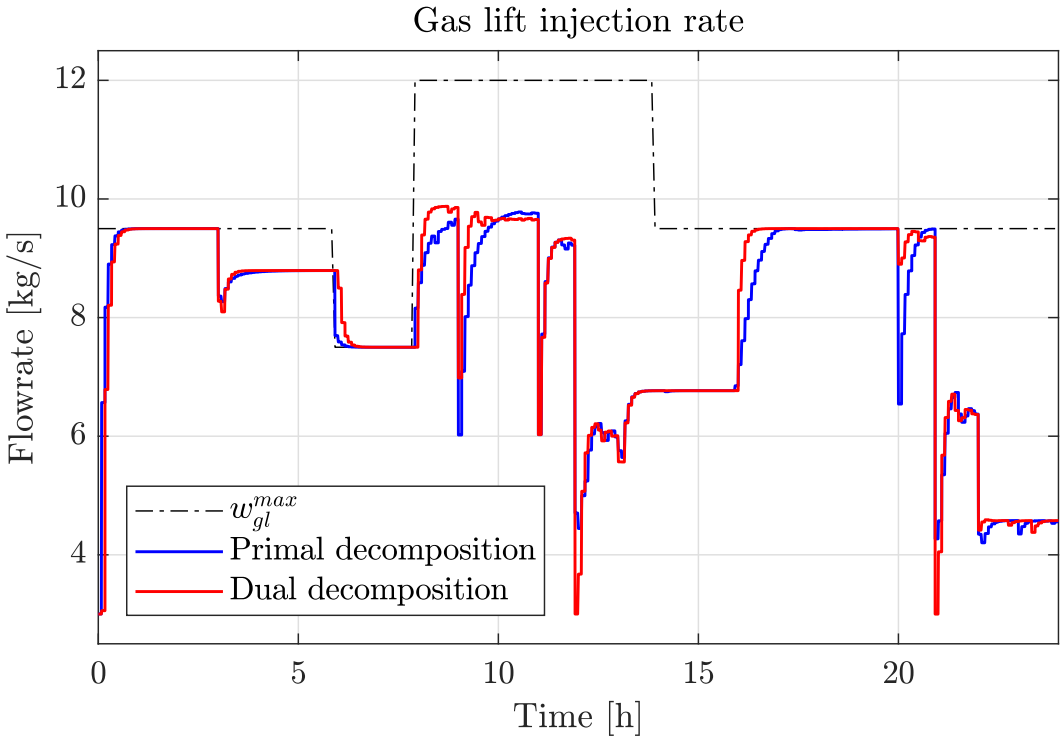

Figure 5.26: Gas lift injection rate in distributed optimizers. The maximum number of iterations to solve the master problem is set equal to one.

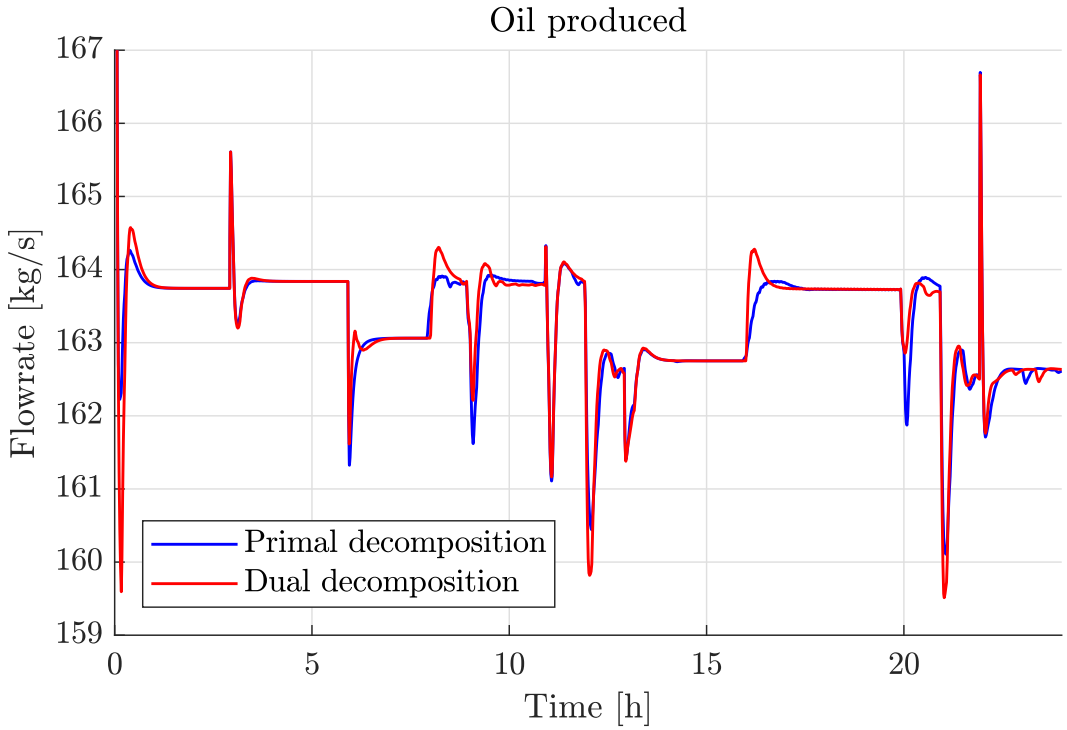

Figure 5.27: Produced oil in distributed optimizers. The maximum number of iterations to solve the master problem is set equal to one.

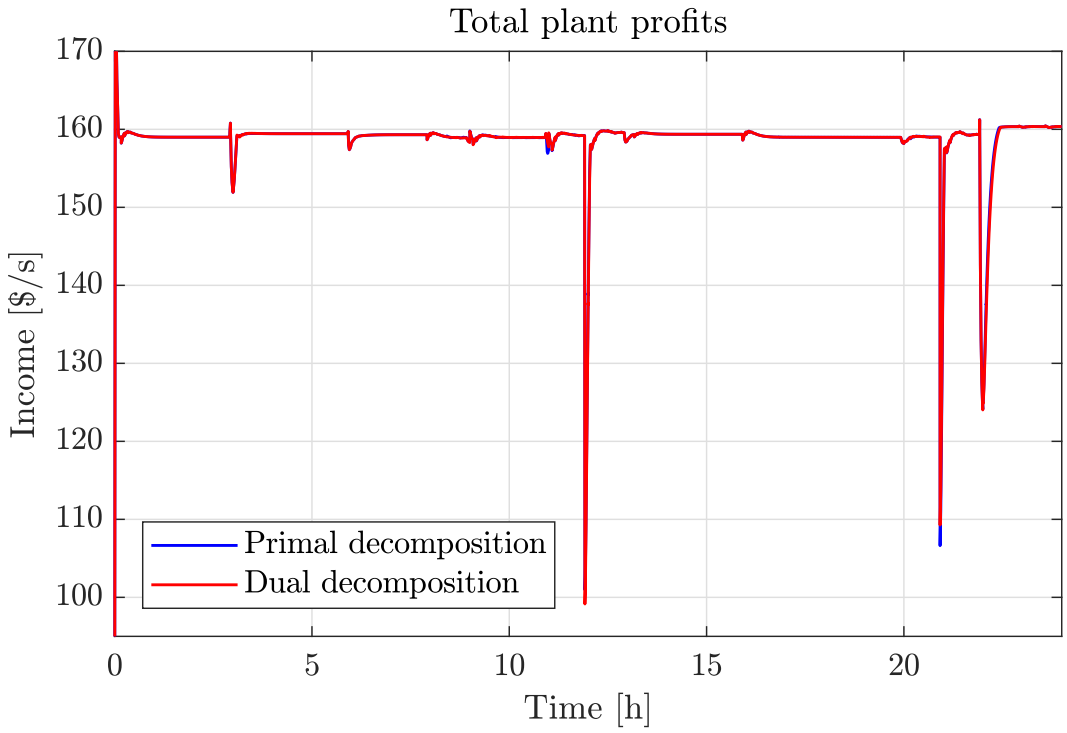

Figure 5.28: Comparison between distributed optimizers: profits.

Between the two distributed optimizers minor differences exist, and the choice between the two can be influenced by other factors, as the convergence rate, the computational time and the availability of informations.

5.6 Computational time

Lastly, ulterior considerations can be made about the computational effort associated to the three optimizers. In table 5.5 the average and maximum computational time for the solving of the optimal problem are reported. In the MPC's columns, there are two numbers for the distributed optimizers: they refer respectively to the computational time of the MPC in the first and second clusters.

Table 5.5: Computational times.

Optimizer	Avg. Opt. time [s]	Max. Opt. time [s]	Avg. MPC time [s]		Max MPC time [s]	
Centralized	0.0195	0.1792	0.9911		14.1090	
Primal Dec.	0.0554	0.7605	0.5260	0.5337	2.9361	5.7308
Dual Dec.	0.2050	2.5596	0.5085	0.5267	2.7382	7.2283

For what concerns the time required by the optimizers, it can be noted that the centralized optimizer is the fastest solver. The system is too simple to emphasize the improvements associated to system decomposition. The necessity of solving the master problem and repeting the optimization calculus for more iterations is more relevant, from a computational effort point of view, than the advantages introduced by system decomposition. While primal decomposition's computational times are of the same order of magnitude of centralized optimizer's ones, dual decomposition requires on average ten times more time. Looking to the number of iterations required by the two distributed optimizers, this result is unexpected: even though dual decomposition requires less iterations to solve the master problem, its computational effort is significantly higher than primal decomposition. The reason of this large difference could lie in the different formulation of the NLP problems: while in primal decomposition the optimal problem is constrained (as in the case of the centralized optimizer), in dual decomposition the optimization of the Lagrangian function is an unconstrained problem. Probably IPOPT (the NLP solver) needs a greater effort to solve these kinds of unconstrained problems. Another reason could be in the higher complexity of the Lagrangian function respect to a simpler cost function. It is not clear the exact reason of this difference in computational efforts, but it is possible to suppose that it is related to how IPOPT deals with the two different optimization problems.

For what concerns the comparison between centralized and decomposed MPC, the average computational times are similar: for both the controllers the average computatonal time is around one second. However, if we suppose that in the distributed MPC the computations are carried out in parallel, it is allowable to say that the distributed MPC requires half the time to perform the computations. The computational effort is exactly the half because the

two subsystems are composed by three wells each, thus, one subsystems is exactly one half of the original system. The advantages of the system decomposition can be noted looking at the maximum computational time required by the MPC: while in the centralized MPC the maximum time is around 14 seconds, in the distributed optimizers it is around 9 or 10 seconds. The decomposition advantages are more evident in the MPC because, even though the objective function is simpler, the computations that must be carried out are higher in number and complexity respect to the ones carried out during the static optimization.

Chapter 6

Concluding Remarks and future work

The purpose of this thesis was to investigate the possibility of developing and using a distributed optimizer based on the hybrid RTO framework. With the development and spreading of subsea technology, offshore oil wells networks have increased in dimensions and complexity. The possibility of performing a system decomposition can simplify the control and optimization of such production systems. An additional advantage of a decomposed optimization and control structure is the possibility of shutting down one subsystem for maintenance while still having an active controller optimizing the remaining subproblems.

In this work, the distributed optimizer based on HRTO has been developed and formulated in MATLAB R2017b programming environment. The case study refers to a gas-lifted oil wells network composed by two clusters of three wells each, on which the decomposition has been performed. The simulations show that the distributed optimizer converges to the same results as the centralized optimizer. For what concerns the control layer, a Nonlinear Model Predictive Controller has been chosen as a better alternative to a classical Proportional Integral controller, due to its ability in dealing actively with dynamic constraint violations.

During the comparison between centralized and distributed optimizers no significant differences have been found concerning the plants profits, but a delicate theme has emerged: the definition of local constraints. While in a centralized control system only the effective global constraints exist, in a distributed one it is necessary to formulate local constraints to coordinate the global production. Despite these constraints are not real and concrete limits, their violation could lead a distributed MPC to take suboptimal decisions.

Regarding the two decomposition methods adopted in this work, no appreciable differences in the profits have been found during the simulations. However, primal decomposition requires, on average, a lower computational time to perform the optimization, even though less iterations are necessary to dual decomposition in order to find the solution of the master problem.

In this case study a particular attention has been put on privacy issues and on information sharing between the two subsystems. While dual decomposition requires the sharing of the optimal process conditions themselves, primal decomposition needs only the subgradients of the subproblems' solution, from which no informations regarding productivities, profits and strategies can be deducted.

Even though in this work primal decomposition seems to be the better alternative, due to its lower computational effort and allowing a higher privacy between the subsystems, dual decomposition could be preferable whenever it is required to converge to the global optimum in a limited number of iterations. Thus, it is not possible to define in an univocal way the best decomposition method for this kind of optimizer, the choice must be taken considering the system to be optimized and the main objectives of the optimal control framework.

6.1 Further work

The case study considered in this work is still not enough complex to significantly appreciate the system's decomposition advantages from a computational effort point of view. It is necessary to try to apply the same distributed optimization framework to different and more complex systems. In particular it could be interesting to test the distributed optimizer's performances when the subsystems are organized in series, instead of in parallell as the wells clusters considered in this work. This could be the case of a large chemical plant, where the decomposition is performed on one single production line, following the succession of equipment through which raw materials must pass to be transormed in products. The subsystems would show different kinds of interactions, that are worth to be analyzed.

Further work is necessary also in the development of a rigorous method to formulate local constraints whenever advanced control layers are adopted. It is also possible to investigate other solvers or new techniques to reduce the number of iteration required for the solution of the master problem. One possibility could be to introduce a quadratic function for updating the master problem, as described in (Wenzel *et al.*, 2018).

Appendix A

Code Snippets

A.1 Simulator

```
function [F,x_var, z_var, p_var, alg, diff, L] =
    CentralizedSimulator(par)

import casadi.*

n_w = par.n_w;

L_w = par.L_w;
H_w = par.H_w;
D_w = par.D_w;

L_bh = par.L_bh;
H_bh = par.H_bh;
D_bh = par.D_bh;

L_a = par.L_a;
H_a = par.H_a;
D_a = par.D_a;

rho_o = par.rho_o;
C_iv = par.C_iv;
C_pc = par.C_pc;
rho_ro = sum(rho_o)/2;

mu_oil = 1*0.001; % 1cP oil viscosity

A_w = pi.*(D_w/2).^2;
```

```
A_bh = pi.*(D_bh/2).^2;
V_a = L_a.*(pi.*(D_a/2).^2 - pi.*(D_w/2).^2);

% differential states
m_ga = MX.sym('m_ga',n_w); % 1-6
m_gt = MX.sym('m_gt',n_w); % 7-12
m_ot = MX.sym('m_ot',n_w); % 13-18

% Algebraic states
p_ai = MX.sym('p_ai',n_w);          % 1-6
p_wh = MX.sym('p_wh',n_w);          % 7-12
p_wi = MX.sym('p_wi',n_w);          % 13-18
p_bh = MX.sym('p_bh',n_w);          % 19-24
rho_ai = MX.sym('rho_ai',n_w);      % 25-30
rho_m = MX.sym('rho_m',n_w);        % 31-36
w_iv = MX.sym('w_iv',n_w);          % 37-42
w_pc = MX.sym('w_pc',n_w);          % 43-48
w_pg = MX.sym('w_pg',n_w);          % 49-54
w_po = MX.sym('w_po',n_w);          % 55-60
w_ro = MX.sym('w_ro',n_w);          % 61-66
w_rg = MX.sym('w_rg',n_w);          % 67-72

% control input
w_gl = MX.sym('w_gl',n_w);

% parameters
p_res = MX.sym('p_res',n_w);
PI = MX.sym('PI',n_w);
GOR = MX.sym('GOR',n_w);        % the only time varying parameter
p_m = MX.sym('p_m',n_w);
T_a = MX.sym('T_a',n_w);
T_w = MX.sym('T_w',n_w);
R = par.R;
Mw = par.Mw;

% algebraic equations
f1 = -p_ai.*1e5 + ((R.*T_a./(V_a.*Mw) + 9.81.*H_a./V_a).*m_ga.*1
    e3) + (Mw./(R.*T_a).*((R.*T_a./(V_a.*Mw) + 9.81.*H_a./V_a).*
    m_ga.*1e3)).*9.81.*H_a;
f2 = -p_wh.*1e5 + ((R.*T_w./Mw).*(m_gt.*1e3./(L_w.*A_w + L_bh.*
    A_bh - m_ot.*1e3./rho_o))) - ((m_gt.*1e3+m_ot.*1e3 )./(L_w.*A_w
    )).*9.81.*H_w/2;
```

```
65 f3 = -p_wi.*1e5 + (p_wh.*1e5 + 9.81./(A_w.*L_w).*max(0,(m_ot.*1e3
       +m_gt.*1e3-rho_o.*L_bh.*A_bh)).*H_w + 128.*mu_oil.*L_w.*w_pc
       ./(3.14.*D_w.^4.*((m_gt.*1e3 + m_ot.*1e3).*p_wh.*1e5.*Mw.*rho_o
       )./(m_ot.*1e3.*p_wh.*1e5.*Mw + rho_o.*R.*T_w.*m_gt.*1e3)));
66 f4 = -p_bh.*1e5 + (p_wi.*1e5 + rho_o.*9.81.*H_bh + 128.*mu_oil.*
       L_bh.*w_ro./(3.14.*D_bh.^4.*rho_o));
67 f5 = -rho_ai.*1e2 +(Mw./(R.*T_a).*p_ai.*1e5);
68 f6 = -rho_m.*1e2 + ((m_gt.*1e3 + m_ot.*1e3).*p_wh.*1e5.*Mw.*rho_o
       )./(m_ot.*1e3.*p_wh.*1e5.*Mw + rho_o.*R.*T_w.*m_gt.*1e3);
69 f7 = -w_iv + C_iv.*sqrt(rho_ai.*1e2.*(p_ai.*1e5 - p_wi.*1e5));
70 f8 = -w_pc + 1.*C_pc.*sqrt(rho_m.*1e2.*(p_wh.*1e5 - p_m.*1e5));
71 f9 = -w_pg + (m_gt.*1e3./max(1e-3,(m_gt.*1e3+m_ot.*1e3))).*w_pc;
72 f10 = -w_po + (m_ot.*1e3./max(1e-3,(m_gt.*1e3+m_ot.*1e3))).*w_pc;
73 f11 = -w_ro + PI.*1e-6.*(p_res.*1e5 - p_bh.*1e5);
74 f12 = -w_rg.*1e-1 + GOR.*w_ro;
75
76 % differential equations
77 df1 = (w_gl - w_iv).*1e-3;
78 df2 = (w_iv + w_rg.*1e-1 - w_pg).*1e-3;
79 df3 = (w_ro - w_po).*1e-3;
80
81 % Define variables for combined systems (needed only for
       decomposition case)
82
83 % Form the DAE system
84 diff = vertcat(df1,df2,df3);
85 alg = vertcat(f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12);
86 x_var = vertcat(m_ga,m_gt,m_ot);
87 z_var = vertcat(p_ai,p_wh,p_wi,p_bh,rho_ai,rho_m,w_iv,w_pc,w_pg,
       w_po,w_ro,w_rg);
88 p_var = vertcat(w_gl,GOR);
89
90 L = -sum(w_po) + 0.25*sum(w_gl);
91
92 alg = substitute(alg,p_res,par.p_res);
93 alg = substitute(alg,PI,par.PI);
94 alg = substitute(alg,p_m,par.p_m);
95 alg = substitute(alg,T_a,par.T_a);
96 alg = substitute(alg,T_w,par.T_w);
97
98 dae = struct('x',x_var,'z',z_var,'p',p_var,'ode',diff,'alg',alg,'
       quad',L); ...
```

```
    % der(m_tot) = w_in - w_out;
opts = struct('tf',par.tf);

% create IDAS integrator
F = integrator('F','idas',dae,opts);
```

A.2 Dynamic estimator: EKF

```
function [f_EKF,JacFx,h_EKF,JacHx,z_EKF,yIndex] = EKF(par)
% Import CasADi
import casadi.*

n_w = par.n_w; % no. of wells;

% Modelling

L_w = par.L_w;
H_w = par.H_w;
D_w = par.D_w;

L_bh = par.L_bh;
H_bh = par.H_bh;
D_bh = par.D_bh;

L_a = par.L_a;
H_a = par.H_a;
D_a = par.D_a;

rho_o = par.rho_o;
C_iv = par.C_iv;
C_pc = par.C_pc;

mu_oil = 1*0.001; % 1cP oil viscosity

A_w = pi.*(D_w/2).^2;
A_bh = pi.*(D_bh/2).^2;
V_a = L_a.*(pi.*(D_a/2).^2 - pi.*(D_w/2).^2);

Mw = par.Mw;
R = par.R;

% differential states
```

```
35 m_ga = MX.sym('m_ga',n_w); % 1-2
36 m_gt = MX.sym('m_gt',n_w); % 3-4
37 m_ot = MX.sym('m_ot',n_w); % 5-6
38
39 % control input
40 w_gl = MX.sym('w_gl',n_w);
41 GOR = MX.sym('GOR',n_w);
42
43 % algebraic equations used for substitution in the ODE model
44 p_m = par.p_m;
45 p_ai = 1e-5.*(((R.*par.T_a./(V_a.*Mw) + 9.81.*H_a./V_a).*m_ga.*1
       e3) + (Mw./(R.*par.T_a).*((R.*par.T_a./(V_a.*Mw) + 9.81.*H_a./
       V_a).*m_ga.*1e3)).*9.81.*H_a);
46 p_wh = 1e-5.*(((R.*par.T_w./Mw).*(m_gt.*1e3./(L_w.*A_w + L_bh.*
       A_bh - m_ot.*1e3./rho_o))) - ((m_gt.*1e3+m_ot.*1e3 )./(L_w.*A_w
       )).*9.81.*H_w/2);
47 rho_ai = 1e-2.*(Mw./(R.*par.T_a).*p_ai.*1e5);
48 rho_m = 1e-2.*(((m_gt.*1e3 + m_ot.*1e3).*p_wh.*1e5.*Mw.*rho_o)./(
       m_ot.*1e3.*p_wh.*1e5.*Mw + rho_o.*R.*par.T_w.*m_gt.*1e3));
49 w_pc = C_pc.*sqrt(rho_m.*1e2.*(p_wh.*1e5 - p_m.*1e5));
50 w_pg = (m_gt.*1e3./(m_gt.*1e3+m_ot.*1e3)).*w_pc;
51 w_po = (m_ot.*1e3./(m_gt.*1e3+m_ot.*1e3)).*w_pc;
52 p_wi = 1e-5.*((p_wh.*1e5 + 9.81./(A_w.*L_w).*max(0,(m_ot.*1e3+
       m_gt.*1e3-rho_o.*L_bh.*A_bh)).*H_w + 128.*mu_oil.*L_w.*w_pc
       ./(3.14.*D_w.^4.*((m_gt.*1e3 + m_ot.*1e3).*p_wh.*1e5.*Mw.*rho_o
       )./(m_ot.*1e3.*p_wh.*1e5.*Mw + rho_o.*R.*par.T_w.*m_gt.*1e3))))
       ;
53 p_bh = 1e-5.*(p_wi.*1e5 + rho_o.*9.81.*H_bh + 128.*mu_oil.*L_bh.*
       w_po./(3.14.*D_bh.^4.*rho_o));
54 w_iv = C_iv.*sqrt(rho_ai.*1e2.*(p_ai.*1e5 - p_wi.*1e5));
55 w_ro = (par.PI).*1e-6.*(par.p_res.*1e5 - p_bh.*1e5);% w_ro = (-
       IPR.a + sqrt(IPR.a.^2+4*IPR.b.*(p_res-p_bh).*1e5))./(2.*IPR.b);
56 w_rg = 1e1.*GOR.*w_ro;
57
58
59 % differential equations
60 df1 = m_ga + par.tSim.*(w_gl - w_iv).*1e-3;
61 df2 = m_gt + par.tSim.*(w_iv + w_rg.*1e-1 - w_pg).*1e-3;
62 df3 = m_ot + par.tSim.*(w_ro - w_po).*1e-3;
63 df4 = GOR ;
64
65 % Concatenate the differential and algebraic equations
```

```
66 diff = vertcat(df1,df2,df3,df4);
67
68 % differential equations
69 df1C = (w_gl - w_iv).*1e-3;
70 df2C = (w_iv + w_rg.*1e-1 - w_pg).*1e-3;
71 df3C = (w_ro - w_po).*1e-3;
72
73 % Concatenate the differential and algebraic equations
74 diffC = vertcat(df1C,df2C,df3C);
75
76 % concatenate the differential and algebraic states
77 x_EKF = vertcat(m_ga,m_gt,m_ot,GOR);
78 p_EKF = vertcat(w_gl);
79
80
81 %%
82
83 f_EKF = Function('f_EKF',{x_EKF,p_EKF},{diff},{'x','p'},{'xdot'})
       ;
84 JacFx = Function('JacFx',{x_EKF,p_EKF},{jacobian(diff,x_EKF)});
85
86 y_model = vertcat(p_wh, p_bh, w_po, w_pg, w_ro, w_rg);
87
88 h_EKF = Function('h_EKF',{x_EKF,p_EKF},{y_model});
89 JacHx = Function('JacHx',{x_EKF,p_EKF},{jacobian(y_model,x_EKF)})
       ;
90
91
92 p_wh_Index = n_w+1:2*n_w;
93 p_bh_Index = 3*n_w+1:4*n_w;
94 w_po_Index = 9*n_w+1:10*n_w;
95 w_pg_Index = 8*n_w+1:9*n_w;
96 w_ro_Index = 10*n_w+1:11*n_w;
97 w_rg_Index = 11*n_w+1:12*n_w;
98
99 yIndex = [p_wh_Index, p_bh_Index,w_po_Index,w_pg_Index,w_ro_Index
       , w_rg_Index];
100
101 z_vec = vertcat(p_ai,p_wh,p_wi,p_bh,rho_ai,rho_m,w_iv,w_pc,w_pg,
       w_po,w_ro,w_rg);
102 z_EKF = Function('z_EKF',{x_EKF,p_EKF},{z_vec});
```

A.3 Nonlinear Model Predictive Controller

```
function [solver,MPC] = SetpointNMPC(par_x,MPCinit,Optimizer)
import casadi.*

[~,~,~,lbx,lbz,lbu,ubx,ubz,ubu] = Initialization_bounds(par_x);

dx0 = MPCinit.dx0;
z0 = MPCinit.z0;
u0 = MPCinit.u0;
u_in = MPCinit.u_in;

GOR_val = par_x.GOR;
PI_val = par_x.PI;

n_w = par_x.n_w;
R = par_x.R;
Mw = par_x.Mw;

%% Modelling

L_w = par_x.L_w;
H_w = par_x.H_w;
D_w = par_x.D_w;

L_bh = par_x.L_bh;
H_bh = par_x.H_bh;
D_bh = par_x.D_bh;

L_a = par_x.L_a;
H_a = par_x.H_a;
D_a = par_x.D_a;

rho_o = par_x.rho_o;
C_iv = par_x.C_iv;
C_pc = par_x.C_pc;
rho_ro = sum(rho_o)/2;

mu_oil = 1*0.001; % 1cP oil viscosity

A_w = pi.*(D_w/2).^2;
A_bh = pi.*(D_bh/2).^2;
```

```
41 V_a = L_a.*(pi.*(D_a/2).^2 - pi.*(D_w/2).^2);
42
43 % differential states
44 m_ga = MX.sym('m_ga',n_w); % 1-6
45 m_gt = MX.sym('m_gt',n_w); % 7-12
46 m_ot = MX.sym('m_ot',n_w); % 13-18
47
48 % Algebraic states
49 p_ai = MX.sym('p_ai',n_w);        % 1-6
50 p_wh = MX.sym('p_wh',n_w);        % 7-12
51 p_wi = MX.sym('p_wi',n_w);        % 13-18
52 p_bh = MX.sym('p_bh',n_w);        % 19-24
53 rho_ai = MX.sym('rho_ai',n_w);    % 25-30
54 rho_m = MX.sym('rho_m',n_w);      % 31-36
55 w_iv = MX.sym('w_iv',n_w);        % 37-42
56 w_pc = MX.sym('w_pc',n_w);        % 43-48
57 w_pg = MX.sym('w_pg',n_w);        % 49-54
58 w_po = MX.sym('w_po',n_w);        % 55-60
59 w_ro = MX.sym('w_ro',n_w);        % 61-66
60 w_rg = MX.sym('w_rg',n_w);        % 67-72
61
62 % control input
63 w_gl = MX.sym('w_gl',n_w);
64 w_gl_SP = MX.sym('w_gl_SP',n_w);
65
66 % parameters
67 p_res = MX.sym('p_res',n_w);
68 PI = MX.sym('PI',n_w);
69 GOR = MX.sym('GOR',n_w);        % the only time varying parameter
70 p_m = MX.sym('p_m',n_w);
71 T_a = MX.sym('T_a',n_w);
72 T_w = MX.sym('T_w',n_w);
73 R = par_x.R;
74 Mw = par_x.Mw;
75
76 % algebraic equations
77 f1 = -p_ai.*1e5 + ((R.*T_a./(V_a.*Mw) + 9.81.*H_a./V_a).*m_ga.*1
       e3) + (Mw./(R.*T_a).*((R.*T_a./(V_a.*Mw) + 9.81.*H_a./V_a).*
       m_ga.*1e3)).*9.81.*H_a;
78 f2 = -p_wh.*1e5 + ((R.*T_w./Mw).*(m_gt.*1e3./(L_w.*A_w + L_bh.*
       A_bh - m_ot.*1e3./rho_o))) - ((m_gt.*1e3+m_ot.*1e3 )./(L_w.*A_w
       )).*9.81.*H_w/2;
```

```
79 f3 = -p_wi.*1e5 + (p_wh.*1e5 + 9.81./(A_w.*L_w).*max(0,(m_ot.*1e3
       +m_gt.*1e3-rho_o.*L_bh.*A_bh)).*H_w + 128.*mu_oil.*L_w.*w_pc
       ./(3.14.*D_w.^4.*((m_gt.*1e3 + m_ot.*1e3).*p_wh.*1e5.*Mw.*rho_o
       )./(m_ot.*1e3.*p_wh.*1e5.*Mw + rho_o.*R.*T_w.*m_gt.*1e3)));
80 f4 = -p_bh.*1e5 + (p_wi.*1e5 + rho_o.*9.81.*H_bh + 128.*mu_oil.*
       L_bh.*w_ro./(3.14.*D_bh.^4.*rho_o));
81 f5 = -rho_ai.*1e2 +(Mw./(R.*T_a).*p_ai.*1e5);
82 f6 = -rho_m.*1e2 + ((m_gt.*1e3 + m_ot.*1e3).*p_wh.*1e5.*Mw.*rho_o
       )./(m_ot.*1e3.*p_wh.*1e5.*Mw + rho_o.*R.*T_w.*m_gt.*1e3);
83 f7 = -w_iv + C_iv.*sqrt(rho_ai.*1e2.*(p_ai.*1e5 - p_wi.*1e5));
84 f8 = -w_pc + 1.*C_pc.*sqrt(rho_m.*1e2.*(p_wh.*1e5 - p_m.*1e5));
85 f9 = -w_pg + (m_gt.*1e3./max(1e-3,(m_gt.*1e3+m_ot.*1e3))).*w_pc;
86 f10 = -w_po + (m_ot.*1e3./max(1e-3,(m_gt.*1e3+m_ot.*1e3))).*w_pc;
87 f11 = -w_ro + par_x.PI.*1e-6.*(p_res.*1e5 - p_bh.*1e5);
88 f12 = -w_rg.*1e-1 + GOR.*w_ro;
89
90 % differential equations
91 df1 = (w_gl - w_iv).*1e-3;
92 df2 = (w_iv + w_rg.*1e-1 - w_pg).*1e-3;
93 df3 = (w_ro - w_po).*1e-3;
94
95 % Define variables for combined systems (needed only for
       decomposition case)
96
97 % Form the DAE system
98 diff = vertcat(df1,df2,df3);
99 alg = vertcat(f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12);
100
101 % give parameter values
102 alg = substitute(alg,p_res,par_x.p_res);
103 alg = substitute(alg,T_a,par_x.T_a);
104 alg = substitute(alg,T_w,par_x.T_w);
105 alg = substitute(alg,p_m,par_x.p_m);
106
107 % concatenate the differential and algebraic states
108 x_var = vertcat(m_ga,m_gt,m_ot);
109 z_var = vertcat(p_ai,p_wh,p_wi,p_bh,rho_ai,rho_m,w_iv,w_pc,w_pg,
       w_po,w_ro,w_rg);
110 p_var = vertcat(w_gl,w_gl_SP,GOR);
111
112 L = sum((w_gl-w_gl_SP).^2); % Penalize setpoint deviation
113
```

```
114 f = Function('f',{x_var,z_var,p_var},{diff,alg,L},{'x','z','p'},{
    'xdot','zeval','qj'});
115
116 %% Direct Collocation
117
118 % Degree of interpolating polynomial
119 d = 3;
120
121 % Get collocation points
122 tau_root = [0, collocation_points(d, 'radau')];
123
124 % Coefficients of the collocation equation
125 C = zeros(d+1,d+1);
126
127 % Coefficients of the continuity equation
128 D = zeros(d+1, 1);
129
130 % Coefficients of the quadrature function
131 B = zeros(d+1, 1);
132
133 % Construct polynomial basis
134 for j=1:d+1
135     % Construct Lagrange polynomials to get the polynomial basis
        at the collocation point
136     coeff = 1;
137     for r=1:d+1
138         if r ~= j
139             coeff = conv(coeff, [1, -tau_root(r)]);
140             coeff = coeff / (tau_root(j)-tau_root(r));
141         end
142     end
143     % Evaluate the polynomial at the final time to get the
        coefficients of the continuity equation
144     D(j) = polyval(coeff, 1.0);
145
146     % Evaluate the time derivative of the polynomial at all
        collocation points to get the coefficients of the
        continuity equation
147     pder = polyder(coeff);
148     for r=1:d+1
149         C(j,r) = polyval(pder, tau_root(r));
150     end
```

```
151
152     % Evaluate the integral of the polynomial to get the
            coefficients of the quadrature function
153     pint = polyint(coeff);
154     B(j) = polyval(pint, 1.0);
155 end
156
157
158 nu = n_w;                   % Length(p_var);
159 nz = 12*n_w;                % Number of algebraic states
160 nx = 3*n_w;
161 np = nu;
162 nd = nx;
163
164 %% Build NLP solver
165
166 % empty nlp
167 w = {};      w0 = [];          lbw = [];          ubw = [];
168 J = 0;
169 g = {};      lbg = [];         ubg = [];
170
171 % initial conditions for each scenario
172 X0 = MX.sym('X0',nx);    Z0 = MX.sym('Z0',nz);
173 w = {w{:}, X0};          lbw = [lbw; lbx];          ubw = [ubw; ubx
        ];
174 w0 = [w0; dx0];
175
176 X0_par = MX.sym('X0_par',nx);
177
178 % initial conditions
179 g = {g{:},X0 - X0_par};  lbg = [lbg;zeros(nx,1)];  ubg = [ubg;
        zeros(nx,1)];
180
181 U0 = MX.sym('U0',nu);
182 GOR_est = MX.sym('GOR_est',nu);
183
184 % Formulate NLP
185 Xk = X0;          Xkj = {};          Zkj = {};          Uk_prev =
        U0;
186 js = 1;
187
188 for k = 0:par_x.Horizon_Samples-1
```

```
189
190     Uk = MX.sym(['U_' num2str(k) '_' num2str(js)],nu);
191
192     Upar = vertcat(Uk,w_gl_SP,GOR_est);
193
194     w = {w{:},Uk};
195     lbw = [lbw;lbu];
196     ubw = [ubw;ubu];
197     w0 = [w0;u0];
198
199     Xkj = {};
200     Zkj = {};
201
202     for j = 1:d
203         Xkj{j} = MX.sym(['X_' num2str(k) '_' num2str(j) '_'
                num2str(js)],nx);
204         Zkj{j} = MX.sym(['Z_' num2str(k) '_' num2str(j) '_'
                num2str(js)],nz);
205 %         if par.sv
206             s{j} = MX.sym(['s_' num2str(k) '_' num2str(j) '_'
                    num2str(js)],1);
207 %         end
208         w = {w{:},Xkj{j},Zkj{j}};
209         lbw = [lbw;lbx;lbz];
210         ubw = [ubw;ubx;ubz];
211         w0 = [w0; dx0;z0];
212     end
213
214     % Loop over collocation points
215     Xk_end = D(1)*Xk;
216
217     for j = 1:d
218         % Expression for the state derivative at the collocation
                point
219         xp = C(1,j+1)*Xk;  % helper state
220         for r = 1:d
221             xp = xp + C(r+1,j+1)*Xkj{r};
222         end
223         [fj,zj,qj] =  f(Xkj{j},Zkj{j},Upar);
224
225         g = {g{:},par_x.tf*fj-xp,zj};  % dynamics and algebraic
                constraints
```

```
226             lbg = [lbg;zeros(nx,1);zeros(nz,1)];
227             ubg = [ubg;zeros(nx,1);zeros(nz,1)];
228
229             % Gas capacity constraints on all the collocation points
230 %             if par.sv
231                 g = {g{:},sum(Zkj{j}(8*n_w+1:9*n_w))-s{j}};
232 %             end
233
234             lbg = [lbg;0];
235             ubg = [ubg;par_x.w_pg_max];
236
237             % Slack variables for gas capacity constraints
238 %             if par.sv
239                 w = {w{:},s{j}};
240                 lbw = [lbw;0];
241               if Optimizer.Centralized
242                 ubw = [ubw;2];
                                                                        %%%
243               else
244                 ubw = [ubw;1];
245               end
246                 w0 = [w0;0];
247 %             end
248
249             % Add contribution to the end states
250             Xk_end = Xk_end + D(j+1)*Xkj{j};
251
252             % Objective functions
253 %             if par.sv
254                 J = J + (B(j+1)*qj*par_x.tf ) + 40*s{j} + ...
255                     0.5*sum((Uk_prev - Uk).^2);
                                                              %%%
256
257 %             end
258         end
259
260         Uk_prev = MX.sym(['Uprev_' num2str(k+1)],nu);
261         Uk_prev = Uk;
262
263         % New NLP variable for state at end of interval
264         Xk = MX.sym(['X_' num2str(k+1) '_' num2str(js)], nx);
265         w = {w{:},Xk};
```

```
266         lbw = [lbw;lbx];
267         ubw = [ubw;ubx];
268         w0 = [w0; dx0];
269 
270         % Shooting Gap constraint
271         g = {g{:},Xk_end-Xk};
272         lbg = [lbg;zeros(nx,1)];
273         ubg = [ubg;zeros(nx,1)];
274 
275         g = {g{:},sum(Uk)};
276         lbg = [lbg;0];
277         ubg = [ubg;par_x.w_pg_max];
278 
279 end
280 
281 % create and solve NLP solver
282 opts = struct('warn_initial_bounds',false, ...
283                 'print_time',false, ...
284                 'ipopt',struct('print_level',1));
285 
286 nlp = struct('x',vertcat(w{:}),'p',vertcat(w_gl_SP,GOR_est,X0_par
        ,U0),...
287                 'f',J,'g',vertcat(g{:}));
288 
289 solver = nlpsol('solver','ipopt',nlp,opts);
290 
291 MPC.w0 = w0;
292 MPC.lbw = lbw;
293 MPC.ubw = ubw;
294 MPC.lbg = lbg;
295 MPC.ubg = ubg;
296 MPC.d = d;
```

Appendix B

EKF validation and plant's measurements

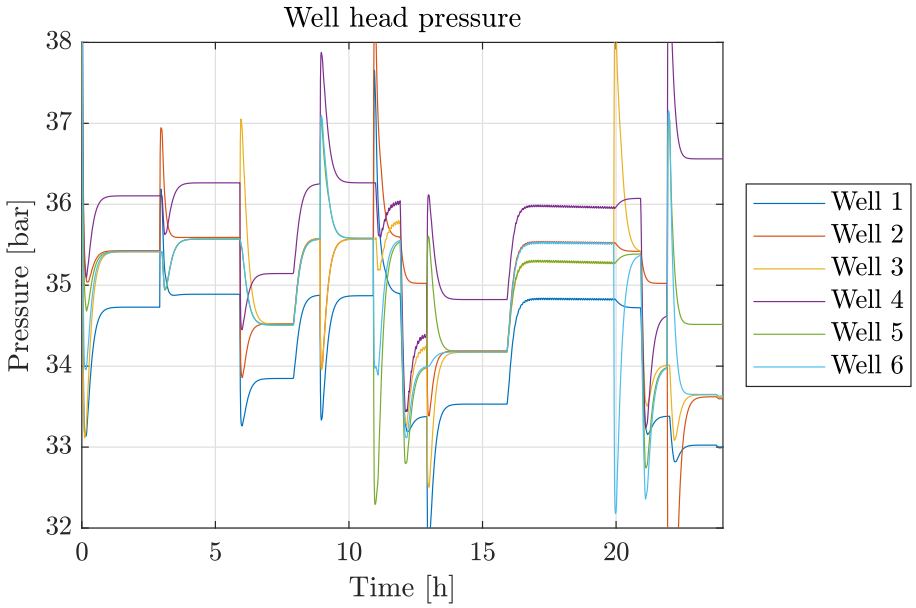

Figure B.1: Well head pressure during final simulation.

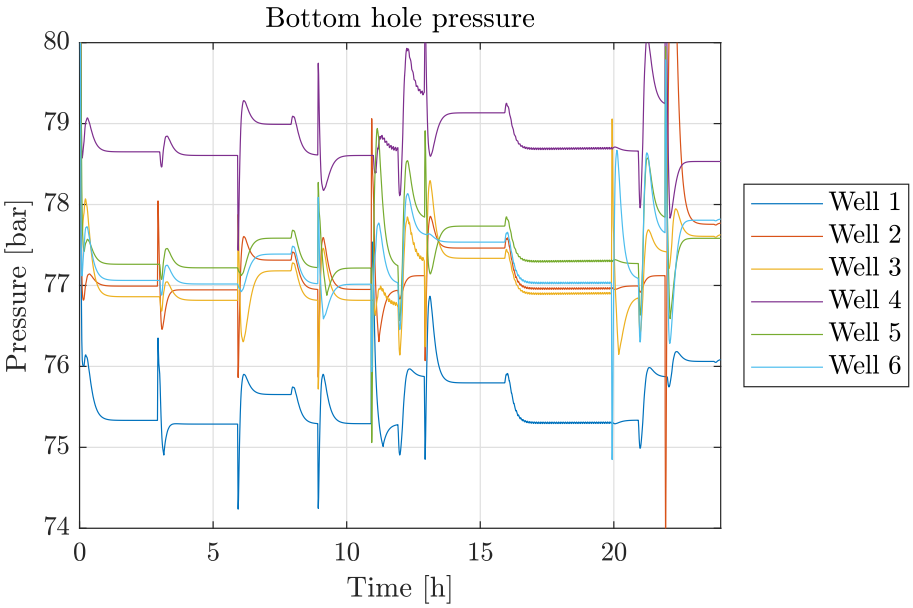

Figure B.2: Bottom hole pressure during final simulation.

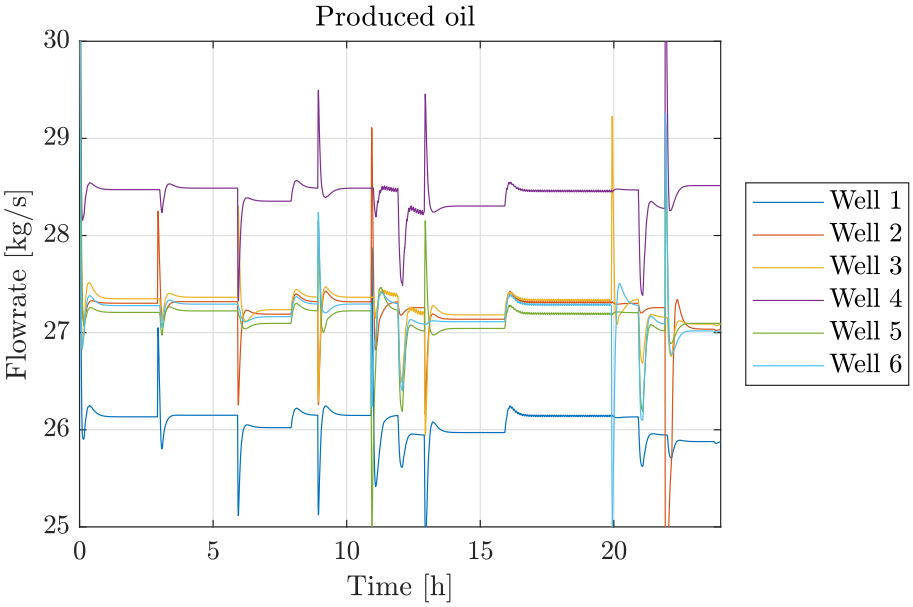

Figure B.3: Produced oil during final simulation.

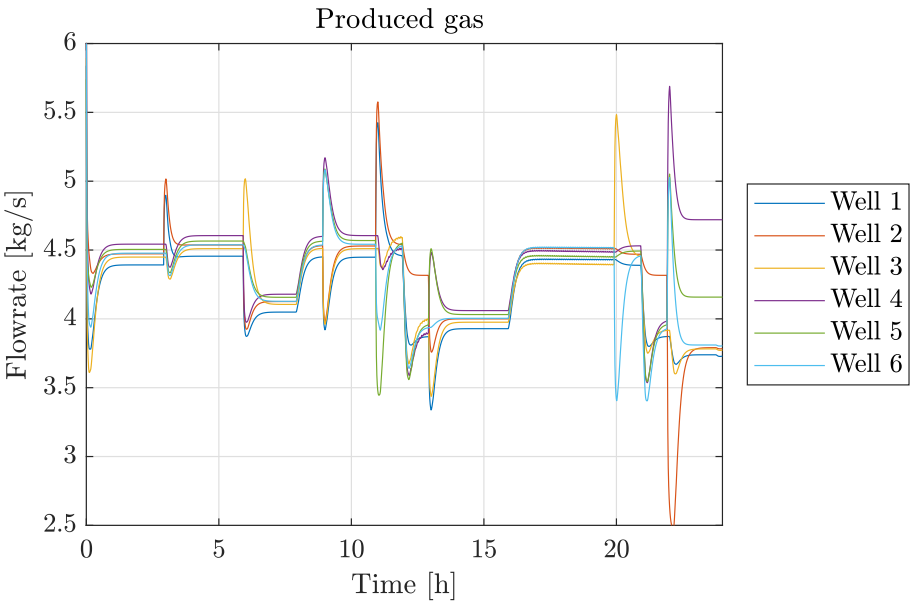

Figure B.4: Produced gas during final simulation.

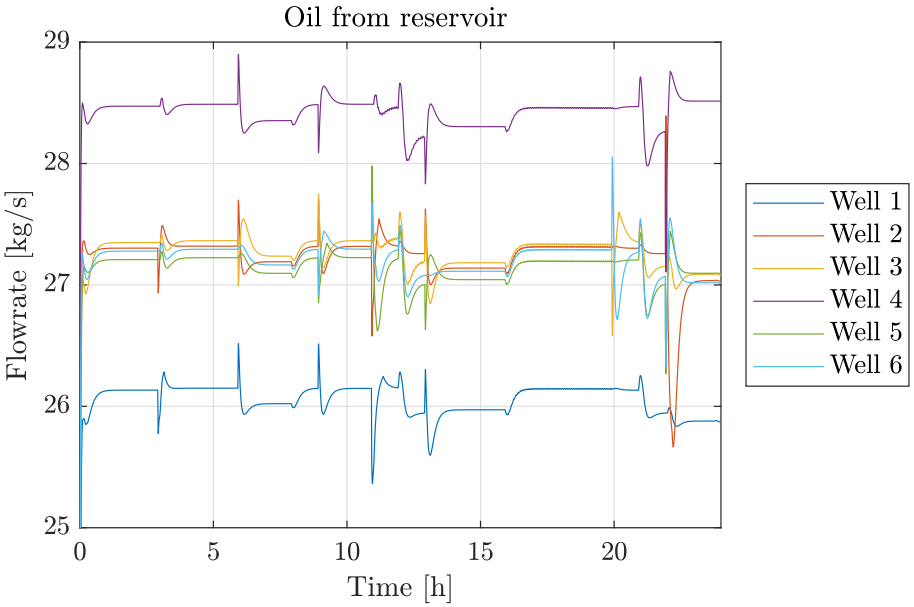

Figure B.5: Oil from the reservoir during final simulation.

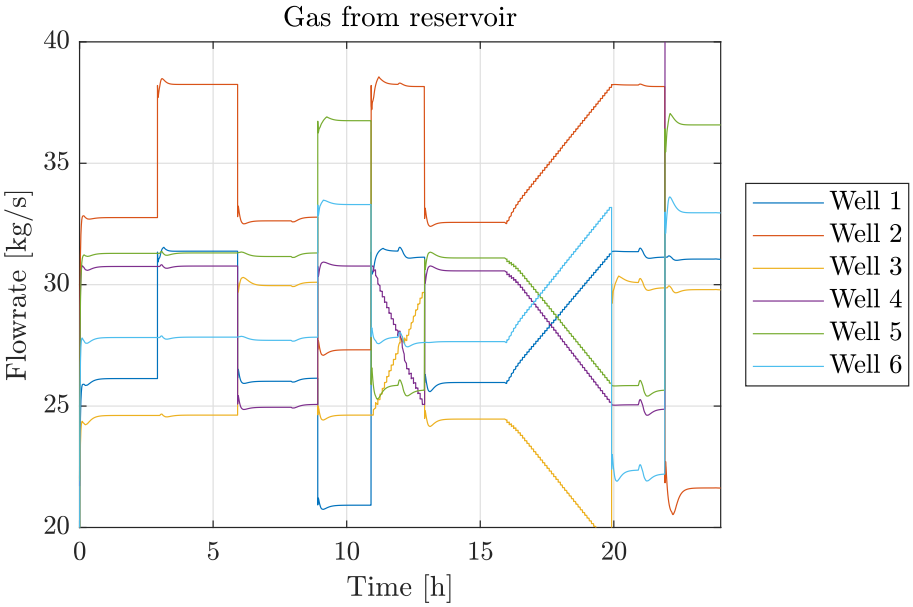

Figure B.6: Gas from the reservoir during final simulation.

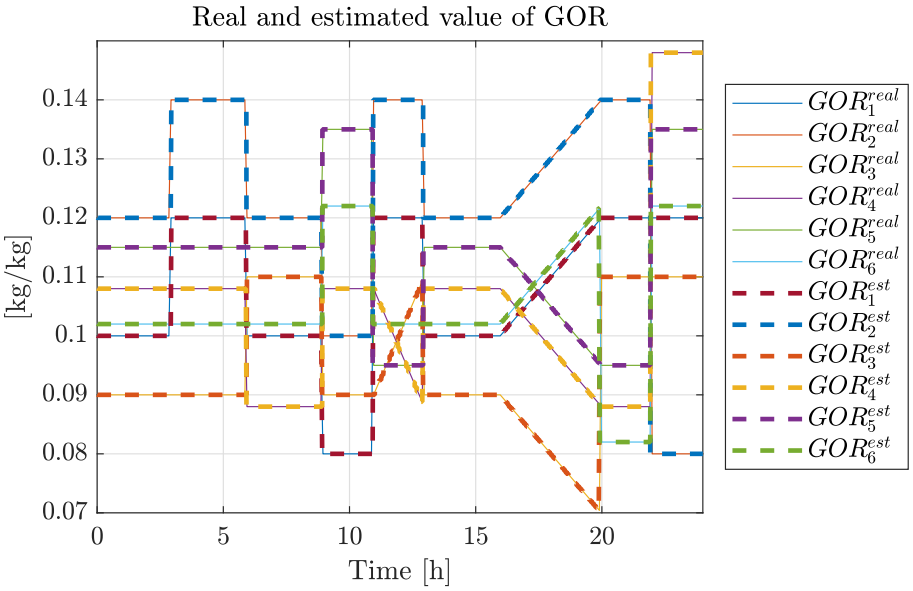

Figure B.7: Real and estimated GORs during final simulation.

Bibliography

Abdalsadig, M., A. Nourian, G. Nasr, and M. Babaie (2016). "Gas Lift Optimization to Improve Well Performance". *International Journal of Mechanical and Mechatronics Engineering* 10.3 (see p. 26).

Andersson, J. A. E., J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl (In Press, 2018). "CasADi – A software framework for nonlinear optimization and optimal control". *Mathematical Programming Computation* (see p. 36).

— (2018). *CasADi documentation*. https://web.casadi.org/docs/. [Online; accessed September-2019] (see p. 36).

Camara, M. M., A. D. Quelhas, and J. C. Pinto (2016). "Performance evaluation of real industrial RTO systems". *Processes* 4, p. 44 (see p. 7).

Dan, S. (2006). "Optimal state estimation: Kalman, H infinity and nonlinear approaches". In: *PROCEEDINGS OF THE IEEE*. John Wiley & Sons (see p. 38).

Darby, M. L., M. Nikolaou, J. Jones, and D. Nicholson (2011). "RTO: an overview and assessment of current practice". *Journal of Process Control* 21.6, pp. 874–84 (see p. 2).

Eikrem, G. O., L. Imsland, and B. Foss (2004). "Stabilization of Gas Lifted Wells Based in State Estimation". *IFAC Proceedings Volumes* 37.1, pp. 323–328 (see p. 26).

Farina, M., G. Ferrari, F. Manenti, and E. Pizzi (2016). "Assessment and comparison of distributed model predictive control schemes: Application to a natural gas refrigeration plant". *Computers & Chemical Engineering* 89 (see p. 56).

Foss, B. A. and J. P. Jensen (2011). "Performance analysis for closed-loop reservoir management". *SPE Journal* 16.01, pp. 183–190 (see p. 1).

Grune, L. and J. Pannek (2011). *Nonlinear Model Predictive Control: Theory and algorithm* (see p. 53).

Hauge, J. and T. Horn (2005). "The Challenge of Operating and Maintaining 115 Subsea Wells on the Troll Field". *Offshore Technology Conference, Houston, Texas*, pp. 1–4 (see p. 3).

Hindmarsh, A. C., P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward (2005a). *IDAS documentation*. https://computing.llnl.gov/projects/sundials/idas. [Online; accessed September-2019] (see p. 36).

— (2005b). "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers". *ACM Transactions on Mathematical Software (TOMS)* 31.3, pp. 363–396 (see p. 36).

Julier, S. J. and J. K. Uhlmann (2004). "Unscented Filtering and Nonlinear Estimation". In: *PROCEEDINGS OF THE IEEE*, pp. 401–422 (see p. 9).

Kalman, R. E. and R. S. Bucy (1961). "New results in linear filtering and prediction theory". *TRANS. ASME, SER. D, J. BASIC ENG*, p. 109 (see p. 38).

Kalman, R. (1960). *Contributions to the Theory of Optimal Control* (see p. 38).

Krishnamoorthy, D., M. A. Aguiar, B. A. Foss, and S. Skogestad (2018). "A distributed optimization strategy for large scale oil and gas production systems". *2018 IEEE Conference on Control Technology and Applications (CCTA)* (see p. 16).

Krishnamoorthy, D., B. A. Foss, and S. Skogestad (2018). "Steady-state real-time optimization using transient measurements". *Computers and Chemical Engineering* 115, pp. 34–45 (see pp. 2, 3, 6, 9–12, 28, 37, 42, 67).

Krishnamoorthy, D., B. Foss, and S. Skogestad (2016). "Real time optimization under uncertainity applied to gas lifted wells". *Processes* 4.4 (see pp. 28, 56).

Larsson, T. and S. Skogestad (2000). "Plantwide control - a review and new design procedure". *Model Identification and Control* 21.4, p. 209 (see p. 1).

Maciejowski, J. M. (2001). *Predictive control with constraints* (see p. 54).

Maciejowsky, J. M. (2002). *Predictive control: with constraints* (see p. 10).

MathWorks (2017). *MATLAB documentation*. https://it.mathworks.com/help/. [Online; accessed 10-September-2019] (see p. 36).

MATLAB (2017). *version R2017b*. Natick, Massachusetts: The MathWorks Inc. (see p. 36).

Moradzadeh, M., R. Boel, and L. Vandevelde (Feb. 2014). "Anticipating and Coordinating Voltage Control for Interconnected Power Systems". *Energies* 7, pp. 1027–1047. DOI: 10.3390/en7021027 (see p. 54).

Moreno, J. and T. Markeset (2002). "Identifying Challenges in the Maintenance of Subsea Petroleum Production Systems". *IFIP International Conference on Advances in Production Manegement Systems*, pp. 101–110 (see p. 1).

Rangaiah, G. P. and V. Kariwala (2012). "Plantwide control: Recent Developments and Applications". *John Wiley & Sons* (see p. 1).

Rawlings, J. B. and D. Q. Mayne (2009). *Model Predictive Control Theory and design* (see p. 53).

Skogestad, S. (1999). "Plantwide control: The search for the self-optimizing control structure". In: *In: Preprints 14th IFAC World*, pp. 325–330 (see p. 1).

— (2003). "Simple analytic rules for model reduction and PID controller tuning". *Journal of Process Control* 13.4, pp. 291–309 (see p. 51).

— (2004). *Control structure design for complete chemical plants* (see p. 1).

— (2017). *PID tuning using the SIMC rules*. http://folk.ntnu.no/skoge/prosessregulering/lectures/SiS6SIMC_tuning.pdf. [Online; accessed September-2019] (see p. 51).

Wan, E. (2016). "Sigma-Point Filters: An overview with applications to integrated navigation and vision assisted control". *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE* (see p. 9).

Wang, P. (2003). "Development and applications of production optimization techniques for petroleum fields". *PhD thesis* (see p. 26).

Watcher, A. and L. Biegler (2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". *Mathematical Programming* (see p. 36).

— (2018). *IPOPT documentation*. https://www.coin-or.org/Ipopt/documentation/. [Online; accessed September-2019] (see p. 36).

Wenzel, S., R. Paulen, S. Kramer, B. Beisheim, and S. Engell (2018). "Shared Resource Allocation in an Integrated Petrochemical Site by Price-based Coordination using Quadratic Approximation". *2016 European Control Conference (ECC)* (see pp. 21, 86).

Acknowledgements

First, I would like to thank Professor Flavio Manenti, for accepting to be my supervisor and for all the opportunities he gave me.

Thanks to all NTNU's PSE group, for welcoming me and giving me the possibility to work on this topic. In particular, I would like to thank my supervisors: Professor Sigurd Skogestad, for his guidance and constant encouragement, and PhD candidate Dinesh Krishnamoorthy, for constantly helping me with great attention and patience.

My exchange period in Trondheim has been amazing, mostly because of all the people I met. A huge "thank you" to all of them, honorable mentions to the people of HK31 and the guys of the Easter trip. The Erasmus was the cherry on top of all the incredible years I spent at Politecnico. Thanks to all my friends from university, to the people from GranoPuro and from Pol.Or. and, of course, to all the friends from Stezzano.

A huge thanks also to my family, for always supporting and encouraging me during these years of study.

Last, but not least, thanks to Marina, for checking the grammar of all this work (except for this page, I hope there are no blunders) and for all the rest.

Carlo Valli, Milan
September, 2019

Colophon

This document was created using LaTeX 2$_\varepsilon$ and edited within the TeXWorks editor, with the help of arara (by Paulo Cereda) typesetting directives. The text body is set in 11 pt Latin Modern Roman, a typeface derived from the Computer Modern fonts designed by Donald E. Knuth. The bibliography was typeset using BibLaTeX.

Copyright Notice